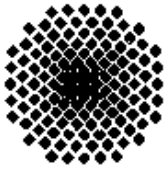

Universität Stuttgart
Fakultät Informatik

Examiner:	A/Prof. Dr. Thomas Bräunl
Supervisor:	A/Prof. Dr. Thomas Bräunl
Started:	1/10/98
Finished:	31/3/99
CR Classification:	G.1.1, I.2.9, I.2.10, I.2.11, I.4.8, I.5.4

Diploma Thesis No. 1700

Robot Soccer

Birgit Graf

In Cooperation with

The University of Western Australia

Department of Electrical and Electronic Engineering
Centre for Intelligent Information Processing Systems

Abstract

This thesis describes the necessary hardware and software developments to prepare a group of mobile robots called *EyeBot vehicles* [Bräunl98/2] to participate at *RoboCup* - the robot soccer World Cup.

The vehicles used were designed around the *EyeBot* platform - a system developed especially to be used with small mobile robot systems. Thereby all processing is done on a Motorola M68332 microcontroller located directly on the platform. No processing is done off-board.

The *CIIPS Glory* robot soccer team consists of five players, one a goal keeper. The four field players have different roles linked to specific areas of competence on the soccer field.

Each robot is equipped with several infrared proximity sensors to measure distances to obstacles. Further a virtual bumper system is used to detect collisions. A 24 bit colour camera is directly attached to the *EyeBot* platform. It is the main sensor used to navigate the robot. As long as all the objects on the soccer field at *RoboCup,* e.g. like the ball, walls and goals have different colours, the overall approach in my program is to detect objects by an analysis of colours and react respectively.

For ball detection and path planning several algorithms have been developed and tested. Due to a relatively low processing speed, my approach was biased on developing simple but effective methods. Analysing its environment quickly and still getting precise results is the most important task for each player. The robots must always act fast and correctly considering continuous changes of their environment.

Specific colours for the ball (orange) and both goals (blue, yellow) are taught to the robot before a game is started. A specific starting position is further given for each of the players.

After a game is started images are constantly read from the colour camera and analysed. If the ball has been detected its global coordinates are calculated out of the position in the image. In additional threads, the updated position of the robot as well as infrared sensor readings are acquired.

A behavioural system has been provided to allow each robot to react to the sensor readings in a proper way. Depending on the positions of the ball and other robots on the field, a player selects a specific driving operation and executes it. If possible it drives directly to the ball and kicks it towards the opponents' half. Facing the wrong direction it must calculate a path how to get behind the ball in order not to shoot towards its own goal. All driving operations include obstacle avoidance - if an obstacle is detected in a robot's way it will not proceed but back up in order to get around the obstacle on a different path.

A first test in a real contest situation was given by the *RoboCup* competition in November '98 in Singapore. Comparing ours to other teams' performances as well as analysing other teams' algorithms gave me knowledge about different playing technics and team strategies. Some of them might be applied in order to improve my team's performance for future competitions.

Table of contents

List of Figures

List of Tables

List of Examples

Chapter 1
Introduction

1.1 Motivation

“The ultimate goal in Artificial Intelligence and probably in robotics is to build intelligent systems capable of displaying complex behaviours to accomplish the given tasks through interactions with a dynamically changing physical world.” [Asada97]

Traditional Artificial Intelligence (AI) research has been mainly pursuing clearly defined tasks in static environments. However, far more interesting problems arise and are eventually solved dealing with complex and changing environments. A strong emphasis in robotics still lies on problems concerning robotic manipulators and static robots. However, in the last few years more and more research was done dealing with mobile robots in dynamic environments.

Another point is that “engineering aspects” such as designing and building hardware systems and their controls have always played a far more important role than research concerning multiagent collaboration, strategy acquisition, real-time reasoning and planning. These topics were mainly dealt with by Computer Science or AI researchers using software agents on the computer. The important factor is though that research of multiple groups is united to get new results that go beyond the ones each group could acquire working by themselves.

RoboCup, the robot soccer World Cup, was created as a proposal of a new standard AI problem. Concerning several research areas it poses a problem of very high complexity. The fact that one is dealing with physical bodies instead of software agents alone makes it also a lot more similar to the real world. To solve real world problems, observing specific aspects from only one point of view might be useful in the beginning, but to understand a problem in its whole complexity different fields of science must be involved.

With the goal to get an optimal performance playing soccer against other teams of real robots, *RoboCup* requires the united knowledge of engineers to design and built the robots, computer scientists to program them and even sports scientists with the task of inventing complex sports playing strategies close to actual human behaviour. The design of the robots has to be adequate for this problem, although not so specific as to prevent them from solving other tasks similar in complexity to the soccer game. The programming involves fields like image processing, data acquisition and real time reasoning. While results from other research projects may be used, the diversity of this task compared to traditional ones makes it necessary to create new algorithms that on the first look may seem specific for this problem. Nevertheless they may be useful in future research and after further development lead to important new understandings.

1.2 Goals

With my work I want to create a basis for future robotics research dealing with simple systems in complex environments. I will describe a group of small autonomous mobile robots called *EyeBot vehicles*, which are used in robot soccer, but at the same time are built flexible enough to solve other complicated tasks.

In contrary to all other teams participating in the *RoboCup* small size league, in my approach all processing is done on-board. Although this method is clearly at a disadvantage with respect to performing well at a *RoboCup* competition, an approach of truly autonomous and locally intelligent systems makes more sense to further research in mobile robot applications in general.

The greatest bias of my work is laid on developing efficient image processing routines. On the one hand they have to be simple in order to be fast enough. On the other hand they have to be powerful enough to reliably detect objects in low resolution images. Due to the named factors, in my program, image processing is based only on colour detection. The used pictures have a resolution of 80 x 60 pixels and images are acquired with a frame rate of about 2.5 pictures per second.

As a second goal several driving routines have to be invented. They must enable the robot to reach its target without too big deviations, however, as for image processing, time restrictions make it necessary to simplify calculations as far as possible.

The robots were programmed in order to perform at the *RoboCup* competition which took place in Singapore in November '98. This competition represented a first test for my program. In comparison to other teams its effectiveness could be evaluated.

1.3 Thesis Outline

After the introduction to my work in this section, Chapter 2 describes the history and goals of *RoboCup*: The Robot World Cup Initiative. *RoboCup* has lately become one of the major AI research areas. *RoboCup Challenge* hereby describes subgoals which can be met within the next few years.

Chapter 3 provides a description of the robots I was working with in order to participate at the *RoboCup Pacific Rim Series* competition in Singapore:

- *EyeBot vehicle I*: First wheeled robot designed around the *EyeBot* platform
- *EyeBot vehicle II*: Improved version, created in order to fulfil such complex tasks as playing soccer

These robots are equipped with several different actualtors, sensors and *EyeBot vehicle II* even with a communication system which will also be described.

The *EyeBot* mobile robot platform was developed specifically for the use in mobile robotics. Its architecture and use is discussed in Chapter 4.

Programming the *EyeBot* platform and thereby the corresponding robots is done in the programming language C. Special libraries for handling mobile robots are hereby available and will be described in Chapter 5.

The program developed to solve my task is outlined in Chapter 6. A main program enables the user to set selected factors for image analysis and drive control. These factors are described in Chapter 6.1.

The *CIIPS Glory* robot soccer team consists of five players - one goal keeper and four field players. To be successful a general team structure is required which will be outlined in Chapter 6.2.

Starting the soccer playing program initializes several threads which enable a player to perceive the situation on the soccer field and react correspondingly. The structure of the most important threads for obstacle avoidance, position tracking, object detection, ball approach and scoring as well as an overview of the way these behaviours work together is therefore described in Chapter 6.3.

For detecting an object, first of all the centre of this object must be found in the picture. Then its global coordinates can be calculated. The necessary algorithms are described in Chapter 6.3.3.

Different ways of approaching the soccer ball must be selected depending on the situation. Chapter 6.3.4 describes the criteria for this selection as well as the algorithms developed to solve this task.

The first match at the *RoboCup* competition in Singapore gave several ideas of how my algorithms could further be improved. Therefore several changes were made directly at the competition site in reaction to circumstances detected during this match. A list of these changes is contained in Chapter 6.3.7.

Different playing strategies could be observed by other teams. A review of interesting approaches and possible ways of adoption for my team is given in Chapter 7.

Finally, Chapter 8 contains additional ideas for future research and improvements of my current program. Chapter 9 closes with a summary.

The appendices provide additional information and give detailed insight into further aspects of my work.

Appendix A lists some important mathematical functions mainly used for path planning algorithms while Appendix B contains formulas for controllers used to ensure that a robot drives on the assigned path.

In order to show the complexity of the written program and to give an overview of the applied functions and threads, Appendix C lists the header files used for my robot soccer program.

Appendix D gives a report about the *RoboCup Pacific Rim Series* competition together with a listing of the participating teams and the match results.

Appendix E contains the web page addresses of several *RoboCup* teams which can be useful sources for further information on *RoboCup* and Appendix F lists the references used in the creation of this document.

Finally, in Appendix G acknowledgments to all people supporting my work on this project are made, whereas Appendix H contains a declaration of myself composing this document without any further help.

Chapter 2
RoboCup: The Robot World Cup Initiative

“The Robot World Cup Initiative (*RoboCup*) is an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined. For this purpose, *RoboCup* chose to use soccer game, and organize *RoboCup*: The Robot World Cup Soccer Games and Conferences. In order for a robot team to actually perform a soccer game, various technologies must be incorporated including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. *RoboCup* is a task for a team of multiple fast-moving robots under a dynamic environment. *RoboCup* also offers a software platform for research on the software aspects of *RoboCup*.“ [RoboCup98/1]

2.1 History

“In the history of artificial intelligence and robotics, the year 1997 will be remembered as a turning point. In May 1997, IBM Deep Blue defeated the human world champion in chess. Forty years of challenge in the AI community came to a successful conclusion. On July 4 1997, NASA’s pathfinder mission made a successful landing and the first autonomous robotics system, Sojourner, was deployed on the surface of Mars. Together with these accomplishments, *RoboCup* made its first steps toward the development of robotic soccer players.” [RoboCup98/2]

As a new standard problem for AI and robotics research, *RoboCup* was founded by Dr. Hiroaki Kitano of the Sony Computer Science Lab in Tokyo in 1997. Kitano describes the long term goal of *RoboCup* as to, within the middle of the next century, “develop a robot soccer team which beats the World Champion” [Kitano98]. The accomplishment of this target sure does not seem very likely at the moment. With current technologies it is rather impossible. However, this final goal may inspire a series of reachable subgoals.

The first *RoboCup* competition was held 1997 in Nagoya, Japan and the subgoal “built a real robot soccer team which can play reasonably well with modified rules” [Kitano98] was already successfully achieved. Further improvement could be observed the next year at two additional *RoboCup* tournaments in Paris and Singapore.

2.2 Goals

At first glance, robot soccer does not look like a very scientific approach. The idea is, however, to see the achievement of the final target - develop robots that can play soccer like humans - as a landmark for the achievement of which various new technologies are developed that might be useful in future AI research.

“The important issue for a landmark project is to have goals high enough so that a series of technological breakthroughs are necessary to achieve this goal” [Kitano98].

The first project developed in the area of creating artificial intelligence comparable to humans was the development of a powerful chess computer. With the victory of Deep Blue over Garry Kasparov, a human

chess player, the final target of this project was accomplished and a new, more complicated and complex challenge had to be invented. With computer chess being a rather standard and clearly defined problem, the new task had to be chosen in order to involve new technologies for industries of the next generation.

Office navigation and tennis ball collection were other tasks to be solved at the AAAI Mobile Robot Competitions [AAAI99]. However, being already far more complex than the chess problem the demand for a task with higher goals comparable with the complexity of the real world was still there.

"*RoboCup* was designed to meet the need of handling real world complexities, though in a limited world, while maintaining a reasonable problem size and research cost." [Kitano98]

Robot soccer meets real world complexities in several ways. Compared to computer chess, it involves a dynamic instead of a static environment. Every calculation has to be done in real time instead of taking turns in chess. The information the robot works on is incomplete and without the use of a global vision system no central control is available to obtain missing data.

2.3 RoboCup Challenge

RoboCup offers significant long term challenges which will take a few decades to meet. The long term research issues to meet the final goal are therefore too many to be listed up at this place. *RoboCup Challenge* describes the mid term technical challenges which are the targets for the next 10 years. They will finally lead to the accomplishment of the long term goals.

RoboCup Challenge is organised in three major classes which are also reflected in the division of the *RoboCup* competition into several different leagues:

2.3.1 The RoboCup Synthetic Challenge

The *RoboCup Synthetic Challenge* [Kitano97/2] mainly refers to the simulator league. The fundamental issue of this challenge is to deal with designing a multiagent system that behaves in real time. It is further important not only to act but also to react to specific situations on the soccer field in without much hesitation.

Especially in the simulator league, where soccer teams of 11 virtual players compete against each other, the game strategy plays an important factor. Therefore the *RoboCup Synthetic Challenge* for *RoboCup-97* consisted of the following categories:

Learning

How can I teach a group of software agents to behave according to certain goals in a collaborative and adversarial environment? This is the first issue of *RoboCup Synthetic Challenge*. The possible robot learning types are: off-line skill learning by individual agents, off-line collaboration learning by a team of agents, on-line skill and collaborative learning, and on-line adversarial learning.

Teamwork

In a centrally controlled group of robots, teamwork has to be planned and executed in real time. It is particularly important to be flexible towards any kind of uncertainties and unforeseen changes. This includes dynamic changes in the team's goals, team members' unexpected inabilities to fulfil their responsibilities or unexpected discovery of opportunities. It is important that the major team strategy does not rely on pre-planned coordination that fails to provide such flexibility.

The *RoboCup Teamwork Challenge* therefore consists of the following research tasks: Contingency planning for multiagent adversarial game playing (define possible strategic tasks and objectives to achieve), plan decomposition and merge, and execution of team plans.

Programs can be evaluated by analysing the basic performance of the team, robustness towards changes, general performance, real-time operation, generality as well as conformity and learning.

Opponent Modelling

A key issue of multiagent interaction is to model and reason about other agents' goals, plans, knowledge, and capabilities. This issue can be divided into the following parts: on-line tracking, on-line strategy recognition, and off-line review.

The evaluation of a program takes place by having teams play other teams that use specifically selected programs, for example a team with easy to foresee actions. Another method analyses the teams' soccer playing abilities with their tracking functionality switched off.

2.3.2 The RoboCup Physical Agent Challenge

The *RoboCup Physical Agent Challenge* [Asada97] deals with research using real robots. The first task therefore is to design a robot platform and its control strategy. There are two ways to build robot soccer players: by designing each component needed to perform a special task or behaviour separately and then assemble them all, or design components that can perform multiple tasks. The decision which of this ways is followed by building the robots, extremely influences the abilities the robot has apart from playing soccer. In the *RoboCup* small size league, for example, many teams consist of robots that were designed only to play soccer and cannot be used for anything else. The idea, though, should be to have autonomous robots with on-board sensors and processing which are not only remote controlled vehicles but can be used for several other applications in AI research.

Real robots have "physical bodies" - now what specific challenges arouse out of this factor? First of all, perception plays the most important role for the robots. A robot soccer player should observe in real-time the behaviour of other objects on the field which might be the ball, opposing or own team's players. As long as the visual information is only given in 2-D, an appropriate, which in this case mainly means not too time-consuming, way to convert the received data into 3-D space has to be found.

Another important issue are the actions a robot can perform. Hereby, special interest lies in the ability to control a ball in a multiagent system. In most traditional robotics systems objects are mainly stationary or follow predictable trajectories. However, robot soccer may one day involve controlling a 3-D (bouncing or flying) motion of the ball, playing with limbed robots (control walking motion) and generating "faint" actions assuming that the opponent has a capability of action observation and prediction.

Different behaviours must be selected depending on the current situation. Since the target is a soccer playing team, simple reflexive behaviours are not enough. Only by understanding a situation in its whole complexity (which is rather impossible at the moment but can be obtained to a certain degree) an appropriate action can be selected considering the positions of all the robots on the field. Due to the variety of situations an approach of robot learning might be more appropriate than trying to foresee every possibility. Asada [Asada97] proposes teaching robots several simple tasks (like shooting the ball) and letting the robots learn to combine them in order to create complex behaviour.

Real-time reaction is another essential factor. There is no time to analyse a situation carefully (as in chess, for example), therefore the player should always react without hesitation. It should particularly not try to wait for a better situation until getting involved in the game. A perfect situation will not occur.

1997 three specific challenges were given for the *Physical Agent Challenge Phase I*:

Moving the Ball

The ability to move a soccer ball over the field to a specified area while avoiding stationary or moving objects is one of the main tasks a robot soccer player should be able to perform. This task involves having a vision system which is able to detect and track moving objects, preferably several objects at the same time (e.g. ball and goal). Other ways of perception might be necessary, for example additional sensors used to detect obstacles on the field. Once the several objects on the field can be recognised, what kind of actions does the robot perform? The range goes from simply pushing the ball forward to complicated kicking and in case of the goal keeper ball holding mechanisms. In addition obstacle avoidance and ball-carrying actions can be combined to get optimal results. Time restrictions make it very important that the mapping from perception to action takes place fast enough. A lookup table would be optimal but seems to be difficult with the variety of possible situations. The other extreme is to teach the system to take decisions with almost no prior knowledge. Every team has to find the optimal variation between these two extremes.

To evaluate the abilities of robot soccer players concerning this problem, their performance in dribbling the ball to certain positions with or without avoiding obstacles on their way can be analysed.

Catching the Ball

In this challenge the skills of catching a ball under several circumstances such as pass receiving, goal keeping or intercepting are tested. It involves the ability to analyse the movement of the ball correctly. The current and future ball position must then be compared with positions of the players on the own team, or in case of intercepting opponents' passes, with the other teams' players.

Cooperative Behaviour (Passing the Ball)

In contrast to the two previously described challenges, this one focuses not only on the skills of one robot but on the cooperative behaviour of several players of one team. Hereby communication and the correct timing while passing and receiving the ball plays an important role.

As can be seen in the next chapters many of the named aspects arise in the work I have been doing. Unfortunately no cooperative behaviour could be achieved but the robots learned to track a soccer ball on the field and kick it, or in case this is not possible, catch it in front of them and carry it towards the goal.

2.3.3 The RoboCup Infrastructure Challenge

The *RoboCup Infrastructure Challenge* is an attempt to make it easier for research groups to establish infrastructure aspects of *RoboCup*. Such challenges include special education programs or robot platform and components standards which facilitate comparing the abilities of the robots.

2.4 RoboCup - the Competition

The main goal for the *RoboCup* real robot competition is “real worldness” [Kitano95], i.e. to make the game as similar to the human soccer game as possible. Due to the uncertainties and uncontrollability, this is mainly given in the real robot competition which is a lot more complex in terms of what situations can arise on the soccer field than for example in the Simulation League. To support playing under these circumstances *RoboCup* rules are made as liberal as possible, “so that they do not obstruct surprise and creativity” [Kitano95].

2.4.1 RoboCup Rules

Two *RoboCup* competitions took place in 1998, the first and official one in Paris, during 2-9 July 1998. *RoboCup-98* had coincided with The World Cup France 98 and was held in conjunction with ICMAS-98[1] and Agent World 98 [RoboCup98/1]. The second one, *RoboCup Pacific Rim Series* was held in Singapore from 22-27 November 1998 together with PRICAI-98[2] [RoboCupS98]. Several changes were made in the rules for *RoboCup Singapore*, which will be outlined in the following listing.

The changes done for the Singapore competition will not be taken over for *RoboCup-99* in Stockholm [RoboCup99]. New rules planned for this event will be stated as well. Possible changes in my program due to these new rules are discussed in Chapter 8.8.

As long as my work involved programming a team for the small size robot league I will not consider rules for the simulation league but only for the real robot competitions. The following is a summary of the most important rules for the small size robot league [RoboCup98/3], to get data for the middle size league all size values are multiplied by 3 and a global vision system is prohibited.

Robots

The maximum floor area of a robot is 180 square centimetres. The height is restricted to 15 cm without or 21.5 cm with local vision system. Robots must be marked by a coloured ping-pong ball on top of them, so that “friend and enemy” can be distinguished.

For *RoboCup-99* in Stockholm every participating robot has to fit into a 18 cm diameter cylinder. For rectangular robots this restricts their diagonal length to 18 cm.

1. International Conference on Multiagent System
2. Pacific Rim International Conference on Artificial Intelligence

Team

A team consists of no more than 5 robots. One of them can be a goal keeper which is allowed to hold the ball for a time period of 15 seconds.

Field

The field has the size of a ping pong table. It is painted in green with white walls and a number of white lines on it (defense areas, centre line, centre circle, see Figure 1).

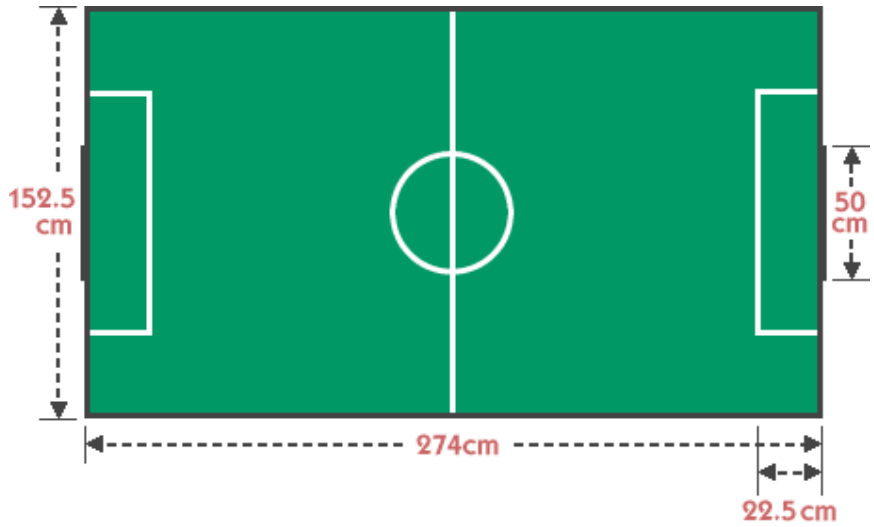

Figure 1. Size values for the small size league soccer field [RoboCup96]

Corners are designed in a special way to prevent the ball or robots from getting stuck. (Figure 2).

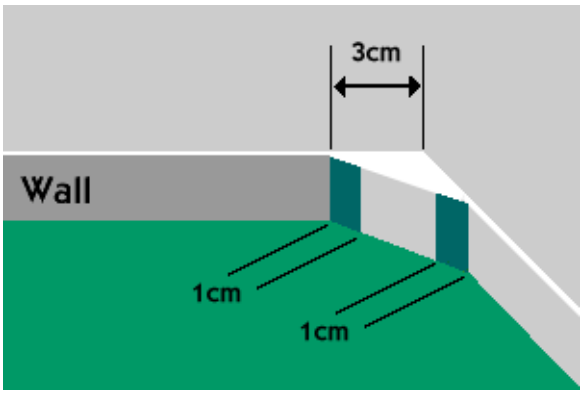

Figure 2. Special design of the corners [RoboCup99]

Goals

The goals are 50 cm wide, each goal and the area of the field inside the goal is painted in a special colour (one blue, the other yellow).

Ball

An orange golf ball is used.

Length of the Game

The game consists of first half, break and second half, each of them taking 10 minutes. The time is stopped during interruptions of the game.

Due to the lack of time, at the competition in Singapore the duration of each half was reduced to 7 minutes, the breaks between halves were reduced even further. Numerous game interruptions still made most of the games last over one hour.

Communication System

The use of a wireless communication system which connects robots and a computer located outside the field is permitted.

Global Vision System

The use of a global camera on top of the field is permitted. Current discussions range around whether each team shall be allowed its own camera system or one central camera will be installed. My team did not make use of a global camera system but only relied on the robots' local on-board camera.

Special Situations During the Game:

(I) Kick-off

After rather chaotic kick-offs at the tournament in Paris where, just like in human soccer, the rules specified that kick-offs had to be made towards the opponents' half, the *RoboCup Singapore* committee decided that the robots had to start playing towards their own team's half. This helped to prevent robots from running into each other as soon as the game was started. This change is not being taken over into the rules for *RoboCup-99* in Stockholm.

(II) Lack of progress / Free ball

For the competition in Singapore the following rule was applied: If the ball is not touched by any robot for 10 seconds a free ball is called. The ball will be put at one of the free ball positions in the field (Figure 3) which is next to where the ball lies. Robots of each team can be placed at a distance of 15 cm from the ball. All robots are free to move as soon as the umpire blows his whistle to restart the game.

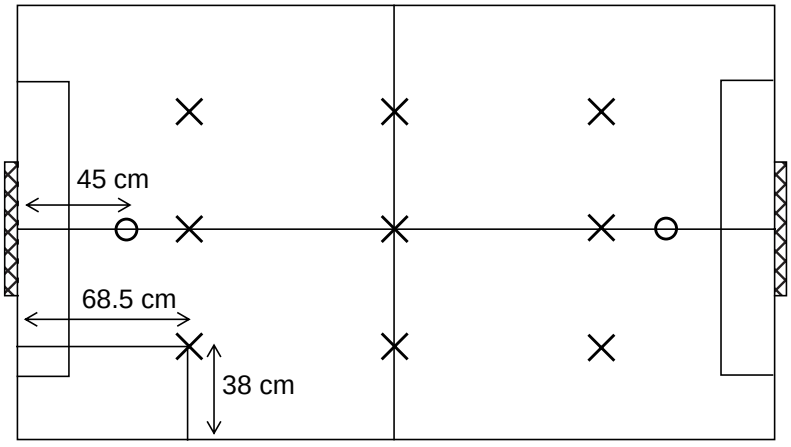

Figure 3. Free kick/ free ball (x) and penalty (o) positions [RoboCupS98]

According to the rules for the *RoboCup-98* in Paris, if the ball had not been touched by a robot after a period of 30 seconds, a free kick was awarded to the team which last touched the ball. The same regulations will be valid 1999 in Stockholm but with a restricted time period of 20 seconds.

(III) Fouls

Fouls are called when a robot intentionally attacks another robot, when more than one defense robot enters the defense zone intentionally or when a robot tries to hold the ball or another robot. They result in a free kick or penalty.

In Singapore the decision whether a robot was actually attacked was left to the umpire. Pushing other teams' players around on the field and running into them at high speeds was not regarded as a foul.

Clear definitions of fouls are given for *RoboCup-99* in Stockholm: "It is unacceptable for multiple robots to charge a single robot and it is also unacceptable to hit the back of a robot even if it has the ball and it is unacceptable to push another player along the table." [RoboCup99]

(IV) Free kick

The ball will be put at the closest of the free kick positions (equivalent to free ball positions, Figure 3), a robot of the team against which the foul was committed is allowed to stand directly behind the ball and play it.

(V) Penalty kick

The ball will be put at the penalty kick position which is located at a distance of 45 cm from the goal (Figure 3). A robot of the team against which the foul was committed is allowed to stand directly behind the ball and play it. All other robots except the robot which will play the ball and the opposing goal keeper are not allowed to move until the penalty has been played.

(VI) Throw-in

A throw in takes place if the ball is played over the walls of the field. This situation never occurred during the competition, no team was able to lift the ball above the height of the walls.

Chapter 3
The CIIPS Glory Robot Soccer Team

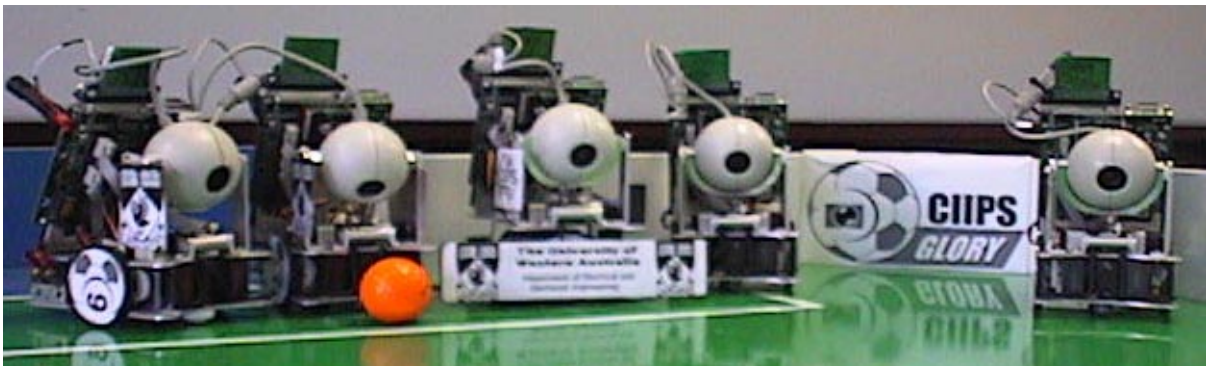

Figure 4. *CIIPS Glory* robot soccer team

My team (Figure 4) was named after the successful *Perth Glory* soccer team. It consists of five robot soccer players, one of them a goal keeper.

3.1 Mechanics

All robots are of the type *EyeBot vehicle II* and were specifically developed to fulfil complex tasks as in my example playing soccer. Due to missing parts, all *EyeBot II* robots except one could be built only in October ‘98, therefore my initial work had to be done on a different robot, the *EyeBot vehicle I*.

EyeBot Vehicle I

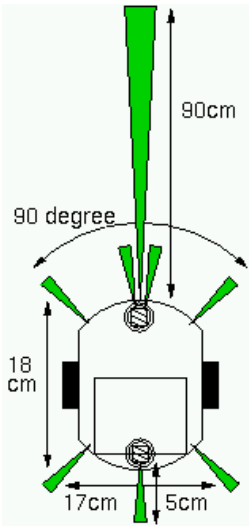

Figure 5. *EyeBot vehicle I*

The size of the *EyeBot vehicle I* (Figure 5) is 17 x 18 cm. It is the first robot being developed around the *EyeBot* platform (Chapter 4).

The robot is equipped with two driven wheels on each side of the robot. To prevent the robot from tipping over there are two freely moveable supporting wheels at the front and back of it.

Seven binary infrared proximity sensors as well as a bumper system in shape of a silicon tube are placed all around the robot. In the beginning only one PSD[3] sensor was attached at the front of the robot. Later two more PSD sensors were added at each side of the robot. The robot is further equipped with a digital camera. It is attached above the front PSD sensor, facing forward.

EyeBot Vehicle II (Soccerbot)

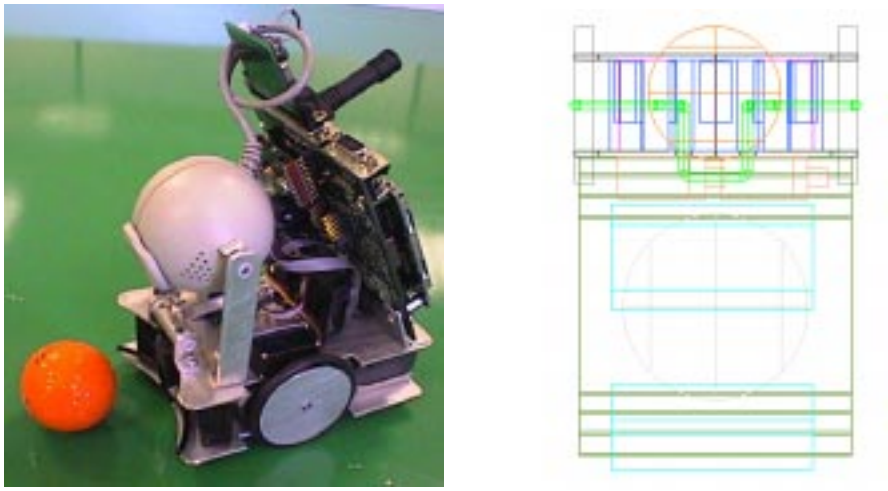

Figure 6. *EyeBot vehicle II*

EyeBot vehicle II (Figure 6) was built in order to improve the soccer playing performance of the robot and create a platform to fulfil more complex tasks than *EyeBot vehicle I*. To fulfil the *RoboCup* regulations for the small size robot league the robot has been reduced to a size of 10 x 15 cm. To fulfil height restrictions the *EyeBot* board is mounted onto the robot at an angle.

To catch the soccer ball, the front of this robot is designed in a curve. The size of the curved area had to be calculated out of the rule that at least two thirds of the ball's floor area must always be outside the convex hull around the robot. With the ball having a diameter of approximately 4.5 cm the depth of this curve is restricted to 11 mm.

EyeBot vehicle II is using the same wheels as its predecessor. Two servos are additionally used for camera tilt and a ball kicker.

The bumper of the *EyeBot vehicle I* is not used for this robot, neither are the binary infrared sensors. Instead it is equipped with 5 PSD sensors all around the robot.

3. Position sensitive device

The Goal Keeper

Figure 7. The goal keeper

One *EyeBot vehicle II* is built different to the others. It plays the position of the goal keeper. To enable it to defend the goal, it has to drive sideways, but still be able to look and kick forward. For this purpose the top plate of the robot is mounted at a 90 degree angle to the bottom plate. The kicking device is enlarged to the maximally allowed size of 18 cm to enable optimal performance at the competition.

3.2 Actuators

3.2.1 DC Motors

All *EyeBot vehicles* are equipped with an equivalent driving system. Thereby each wheel is controlled separately by a Faulhaber [Faulhaber97] DC motor (Figure 8).

Figure 8. DC motor for right wheel

This arrangement enables a robot to move forward and backwards as well as drive turns or spin on the spot. No special braking mechanism has been developed, the speed of the robot is simply set to 0 in order to stop it.

3.2.2 Servos

To kick away the soccer ball, *EyeBot vehicle II* is equipped with a kicking device (Figure 9), controlled by a HiTec servo [HiTec98].

Figure 9. Curved front with “kicker”

To improve vision of the robot another servo is added to the camera (Figure 10) which now makes it possible to tilt the camera up and down and thus track objects like the soccer ball more precisely. Especially with objects getting very close to the robot, the “old” *EyeBot vehicle* had great difficulties, because it lost them out of sight once they were lying right in front of the robot.

Three different camera positions are used in my program: `UP`,`MIDDLE` and `DOWN`. An appropriate angle is selected depending on the distance of the ball. If this value gets smaller than the threshold of 0.15 m the camera is tilted to the `MIDDLE` position. If the distance becomes smaller than 0.06 m, the ball is lying directly in front of the robot and the camera position is set to `DOWN`.

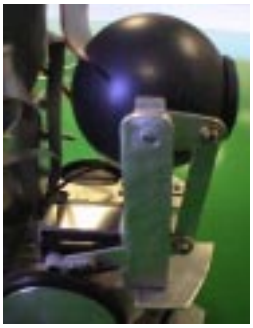

Figure 10. Tiltable camera - here in position `UP`

3.3 Sensors

3.3.1 Bumper

The *EyeBot vehicle I* is equipped with an acoustic bumper system. The bumper is designed as a silicon tube, laid out in a circle all around the robot, with two microphones at the rear end. When collision occurs, the time delay between the two microphones' interrupts can be used to determine the point of collision all around the robot. Unfortunately the bumper did not work at the expected accuracy and therefore was not used in my program.

3.3.2 Wheel Encoders

For both *EyeBot vehicles* the driven wheels are equipped with optic shaft encoders to measure distances the robot has moved. Both wheels are controlled separately, therefore even when setting the same speed the current used to drive the wheels might be slightly different and the robot leaves the straight line. Observing the output of the shaft encoders can help to verify that the robot actually executes the assigned driving operation.

To detect when the robot runs into obstacles a virtual bumper system is used for *EyeBot vehicle II*. Thereby the data of the shaft encoders is constantly compared to the assigned driving operations. If the difference gets too big, the robot suspects that one of its wheels has stalled because it ran into an obstacle.

3.3.3 Binary infrared Sensors

The infrared proximity sensors on *EyeBot vehicle I* are able to detect obstacles within a range of 5 cm. They do not announce the distance of the obstacle.

3.3.4 PSD Sensors

The SHARP GP2D02 PSD sensors [Sharp98] detect the distance of obstacles in their way by sending infrared signals at a frequency of about 1.3 shots per second. Then the time until the signal is reflected at an obstacle and returned is measured. By only using three PSD sensors *EyeBot vehicle I* successfully navigated in and analysed the structure of a maze, comparably to the *Midi Mouse Competition*, another mobile robot challenge held several years ago.

The PSD sensors work considerably well at distances between 10 and 20 cm, nevertheless more accurate measurements would be helpful. One of the major problems with the sensors used for the *EyeBot vehicle* occurs when getting very close to objects. The signal of the sensor bounces more than once until it gets from sender to receiver. Therefore the received distances end up being higher than they really are. Another problem is the difference between sensors. The sensor only returns the time it took to receive a previously sent signal. Unfortunately even standing at the same distance to an object variations in these time intervals are observed for different sensors.

To get optimal results a specific table must be set up for each single sensor. Table 1 shows an excerpt of these tables for four selected PSD sensors, whereas a graphical representation is given in Figure 11. Due to the lack of possibilities to accurately measure distances beyond these barriers, values are restricted to the area between 60 and 999 mm. Unfortunately, the necessary procedure to set up these tables is extremely time consuming and rather impracticable for large numbers of sensors.

Reply time (ms)	20	40	60	80	100	120	140	160	180	200	220	240	256
PSD1	60	98	122	158	187	231	273	327	403	637	999	999	999
PSD2	60	60	119	158	208	268	328	412	586	999	999	999	999
PSD3	71	90	105	123	150	189	254	387	999	999	999	999	999
PSD4	88	106	131	169	220	268	325	413	600	999	999	999	999

Table 1. Assigned distances (in mm) to reply times (in ms) for selected PSD sensors

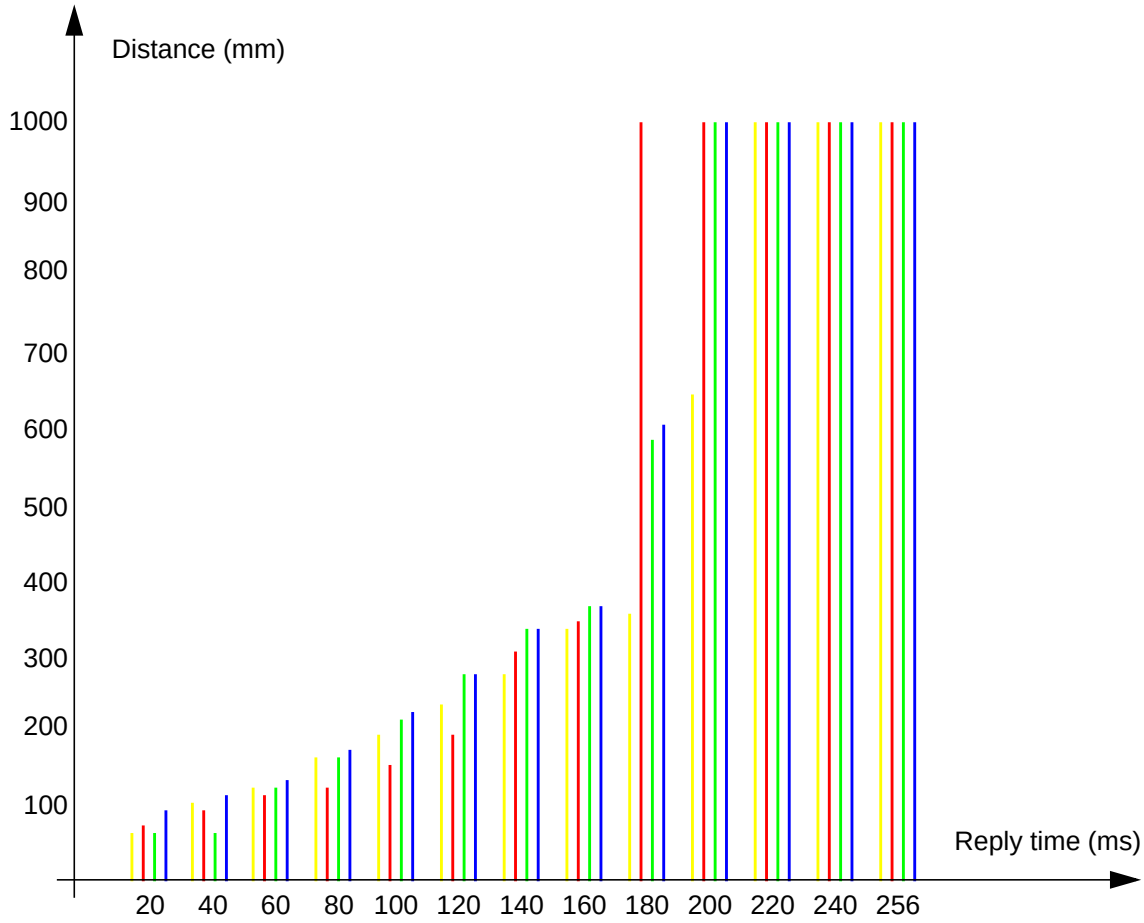

Figure 11. Relation between reply times and distance of an obstacle for selected PSD sensors

3.3.5 Camera

The most important sensor for solving difficult tasks is the Connectix QuickCam camera [Connectix98] attached at the front of all *EyeBot vehicles*. In the beginning *EyeBot vehicle I* was only used for very simple tasks, therefore it was only equipped with a greyscale camera. Because image processing and especially colour recognition is the key requirement of my soccer program on one hand and the abilities of the robot could be extremely increased on the other hand, this camera was soon replaced by a 24-Bit colour camera. Three different operation modes can be selected for the camera angle: WIDE, NORMAL and ZOOM. Images with a resolution of 80 x 60 pixels are used.

Unfortunately the selected colour camera is not very fast and important information about a recently released improved model could not be acquired. Therefore another camera module is currently being tested and will soon be fitted to the *EyeBot* platform.

To improve the view on the soccer field, different camera lenses were tested for the camera. The goal keeper is equipped with a specially wide lens in order to be able to oversee the largest possible part of the soccer field.

3.4 Communication System

To enable the robots to “talk to each other”, distribute data and actually perform as a team, a communication module [RFSolutions98] is added to each *EyeBot II vehicle* (Figure 12). The system used is an FM Radiometrix Transceiver, a “miniature UHF module capable of data transmissions at speeds of up to 40kB/s over distances of around 30 m” [Storey98].

Figure 12. Wireless communication module

Chapter 4
Control Hardware: The EyeBot Platform

Figure 13. *EyeBot* platform

The extendable *EyeBot* base platform [Bräunl98/1] [Bräunl98/2] was developed around the key requirements of image processing. It therefore features a digital camera and an LCD graphics display (Figure 13). It also provides a sufficient number of I/O ports for the connection of various sensors and actuators or any future extensions. While most robot vision systems are either tethered or remote-controlled, on-board real-time vision is feasible for large and expensive mobile platforms. It seems to be rather difficult to implement real-time vision on a small and inexpensive system. *EyeBot* is a platform that accomplishes this goal. It is an 8.6 cm $\times$ 8.6 cm board built around the Motorola M68332 microcontroller [Motorola99] with a number of interface options and real-time vision - ideal for mobile robot systems.

EyeBot has been successfully used in the construction of several wheel-driven vehicles, a 6-legged walking machine, and two biped walkers (Figure 14). It is further being considered for the project of a flying robot.

Although the controller runs at moderate speed (25 MHz), it is fast enough to compute basic image operations on a low resolution image in real time. Image acquisition, Sobel edge detection and display on the LCD for a 80 $\times$ 60 image, for example, can be performed at a rate of about 7 frames per second using a QuickCam greyscale camera. Unfortunately this speed is reduced radically when using a colour camera - not because of the processor speed but because it takes extremely long to read the picture of the camera.

EyeBot's graphics LCD is essential for interaction between the robot and the programmer. He needs to see the robot's view in order to set camera parameters or orientation. Although the camera provides colour images at medium resolution, the display can only show low resolution black/white images. This is sufficient as a feedback to the programmer when running the robot, but not for program development. Therefore a remote control system has been developed [Storey98]. It enables the user to set up wireless communication between the robot and a workstation in order to see the view of the robot on the computer screen.

Figure 14. The *EyeBot* mobile robot family

For more complex image processing operations the interactive image processing tool *Improv*[4] can be used (Figure 15). *Improv* has been implemented on a PC under Linux/X windows using a QuickCam digital greyscale camera and in the latest version (2.0) even the 24-Bit Colour QuickCam. For the greyscale version a Pentium-PC (90 MHz) can deliver up to 10 frames per second at 160 x 240 pixels resolution with 4 bit greyscale per pixel. The image rate drops depending on the operations performed on the image. In five additional subwindows, image processing functions can be applied in a step-wise fashion. Each of the five subwindows takes the previous window as input.

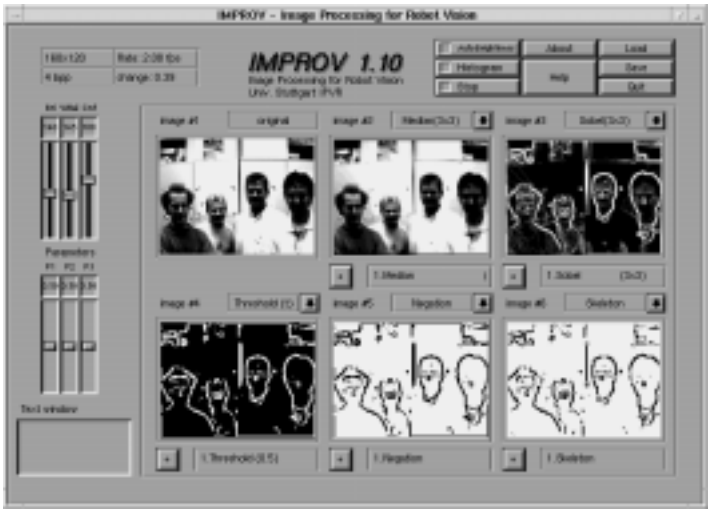

Figure 15. *Improv* tool for complex image processing in real time

4. Image processing for Robot Vision

Chapter 5
Programming Environment

A version of the *gnu* C-compiler and library has been adapted for *EyeBot*, so program development can be made in a high level language using assembly routines for time-critical passages. The microcontroller's timing processor unit (TPU) is being used for servo control with pulse width modulation (PWM), for sound synthesis and sound playback as well as the control of infrared distance sensors.

The operating system *RoBIOS*[5] was written in C plus *m68k* assembly language using the *gnu* C compiler and assembler tools. *RoBIOS* comprises a small real time system with multithreading, libraries for various I/O functions and a number of demonstration applications. The C low level text input and output routines have been adapted for *EyeBot*. This enables the use of the standard C I/O library *clib* together with the *EyeBot* system for user application programs. For example, a user can call `getchar()` in order to read a key input and use `printf()` in order to write text onto the screen. A list of the available function groups and some exemplary commands can be seen in Table 2.

Function Group	Description	Example	Description for example
OS	Operating System		
MT	Multi-Tasking	`OSReady(thread);`	Set status of given thread to READY.
SEM	Semaphores	`OSSemP(sem);`	Do P operation.
TIME	Timer	`OSGetCount();`	Get the time since last reset (in 1/100 s).
RS232	Download and RS-232		
LATCH	Latches: other in-/output		
PAR	Parallel Port		
AD	Analog-Digital Converter		
AU	Audio	`AUTone(freq,ms);`	Play tone with given frequency for the given time (nonblocking).

LCD	LCD Output	`LCDClear();`
		
		`LCDSetString (row,column,string);`
		
		`LCDMenu(string1, string2,string3, string4);`
			Clear LCD.
			
			Write string to the given display position.
			
			Specify description menu for keys.

Table 2. *RoBIOS* function groups

5. Robot Basic Input Output System

Function Group	Description	Example	Description for example
KEY	KEY Input	`KEYWait(key);`	Wait until key is pressed.
CAM	Camera	`CAMGetColFrame(buf, convert);`	Get colour picture.
IP	Image Processing	`IPColor2Grey (src,dest);`	Convert colour to greyscale picture.
BUMP	Bumper	`BUMPCheck(handle, timestamp);`	Check occurrence of a single bump.
IR	Infrared sensors	`IRRead(handle);`	Read actual state of the selected IR-sensor.
PSD	PSD distance sensor	`PSDGet(handle);`	Get distance measured by selected PSD.
SERVO	Servo positioning	`SERVOSet(handle, angle);`	Set selected servos to the same given angle.
MOTOR	Motor control	`MOTORDrive (handle, speed);`	Set selected motors to the same given speed.

VW	V-Omega Driving Interface	`VWSetSpeed(handle, v,w);`
		
		`VWDriveStraight (handle,delta,v);`
			Set new linear speed: v(m/s), rotational speed: w(rad/s).
			
			Drive distance `delta` with speed v straight ahead (using integrated PI-controller).

Table 2. *RoBIOS* function groups

Special care has been taken to keep the *RoBIOS* operating system flexible among several different hardware configurations, because the same system is to be used for wheeled robots and for legged robots. Therefore, a hardware description table (HDT) has been included in the system design. It is used by the operating system to find out which hardware components are currently connected to the system and to provide device drivers. These hardware components can be sensors or actuators (motors or servos), whose control routines are already available in the general *RoBIOS* system. HDT allows easy detection, initialization, and use of hardware components.

Entries in the HDT table define the current hardware configuration of the system. *RoBIOS* contains access routines to find and use the device drivers corresponding to the table entries. Example 1 shows the HDT entry for one of the robot's servos. The parameters specify (in this order): minimum driver version required, tpu channel the servo is attached to, tpu timer that must be used, pwm period, minimal and maximum hightime of the pwm period. The last two values must be found in order to set the required maximal and minimal extension of the servo.

Example 1: HDT entry for servo 11

```
servo_type servo11 = {2, 12, TIMER2, 20000, 1650, 2250};
```

Special setups also must be made for the shaft encoders. To get the correct values, the radius of the wheels and the number of clicks returned for a certain distance are required. Other specific values must be described for the PSD sensors. A table matching response time to the corresponding distance value is required for each of the sensors.

Example 2 is an excerpt of the HDT file for *EyeBot vehicle II*. The given entry list specifies all components attached to the robot, together with their semantics. The first line, for example, specifies the motor driving the right wheel of the robot as an object of the group `MOTOR` called `MOTOR_RIGHT` and a pointer to the `motor0` data structure.

A second motor is used on the left side, both motors are equipped with shaft encoders. Other components are servos for camera and kicker and PSD sensors. Bumper and binary infrared sensors are not used, the respective entry is therefore commented out.

Example 2: HDT structure for *EyeBot vehicle II*

```
{
    MOTOR,MOTOR_RIGHT,"RIGHT",(void *)&motor0,
    MOTOR,MOTOR_LEFT,"LEFT",(void *)&motor1,
    QUAD, QUAD_RIGHT,"RIGHT",(void *)&decoder1,
    QUAD, QUAD_LEFT,"LEFT",(void *)&decoder0,
    VW,VW,"Drive",(void *)&drive,

    /* BUMP, BUMP_LEFT,"LEFT",(void *)&bumper0, */
    /* IR, IR_LF,"LF",(void *)&ir0,             */

    SERVO,SERVO11,"S11-C",(void *)&servo11,
    SERVO,SERVO12,"S12-K",(void *)&servo12,

    PSD,PSD_LEFT,"P0-L",(void *)&psd0,
    PSD,PSD_FRONT,"P1-F",(void *)&psd1,
    PSD,PSD_RIGHT,"P2-R",(void *)&psd2,

    WAIT,WAIT,"WAIT",(void *)&waitstates,
    INFO,INFO,"INFO",(void *)&roboinfo,

    END_OF_HDT,UNKNOWN_SEMANTICS,"END",(void *)0
};
```

Any program which needs to access a sensor or actuator object defined in the HDT can now do so by defining a handle as displayed in Example 3. Here, `GROUP` stands for any actuator group or sensor group, e.g. `MOTOR`, `PSD`, etc. Each group entry provides the handle data type and an initialization function. There are usually some additional routines depending on the functionality of the particular object group (Table 2).

Example 3: Addressing actuators or sensors in a program

```
GROUPHandle myhandle;
myhandle = GROUPInit(myobject);
```

After compilation on the computer a program can be downloaded into the RAM of the *EyeBot* and started. It can further be saved into the ROM, which makes it available even after the robot has been switched off.

Chapter 6
Software Architecture

In the *CIIPS Glory* robot soccer team each field player is equipped with the same program. Differences are only given through the robot IDs which indicate a player's position in the team. Due to its individual design the goal keeper is equipped with a specific motion algorithm.

For all players once the program has been started the possibility to change settings according to the robot's environment must be given. It must further be possible to initiate the soccer playing program. The latter consists of two major operations - find the soccer ball on the field and kick it towards the opponents' goal. Position tracking and obstacle avoidance are important features which should be integrated in these tasks.

6.1 Main Menu

The main menu is displayed when the robot soccer program is started on an *EyeBot Soccerbot*. In the latest version the options `SET`, `GO`, `aOff` and `END` are displayed. The current camera picture and robot settings as machine ID and player position are also shown on the screen. The user can press the `SET` button to change the standard settings for ball and goal colour values, starting position, camera values, drive control and image processing parameters. For a complete list see Table 3.

`COL`	`BALL`		set ball colour
	`GOAL`		set goal colour
	`INFO`		information
`POS`	`LBC`		set position to left back (corner of defense zone)
	`RBC`		set position to right back
	`...`	`LFC`	set position to left forward
		`RFC`	set position to right forward
		`MID`	set position to centre point
`...`	`CAM`	`Brigh.`	change brightness
		`Hue`	change hue
		`Satur.`	change saturation
	`IMG`	`Thresh ball`	change threshold for ball detection
		`Thresh goal`	change threshold for goal detection
	`DRV`	`Lin speed`	change linear speed
		`Rot speed`	change rotational speed
		`Player pos.`	change player position

Table 3. Main Menu setting options

GO starts the different threads of the soccer program which will be described in the following sections. It is important that a large enough stack is provided for each process. Especially the use of complex image processing routines could otherwise lead to severe and hard to detect mistakes during runtime.

The third parameter was introduced specifically for the competition in Singapore. It is used to switch on an attack function for kick-offs and free kicks. For a more detailed description see Chapter 6.3.7.

6.2 Team Structure

Each of the five players in my soccer team has a specific role. There are one goal keeper, two defenders (left, right) and two attackers (left, right). The goal keeper always drives along in front of the goal and does not leave the defense area. Defenders usually stay in their own half of the field, attackers in the opponents' half. Each of the field players' roles is linked to one special quarter of the field. In case a robot does not have to follow the ball, it drives on a triangular path to look for the ball in its area (Figure 16).

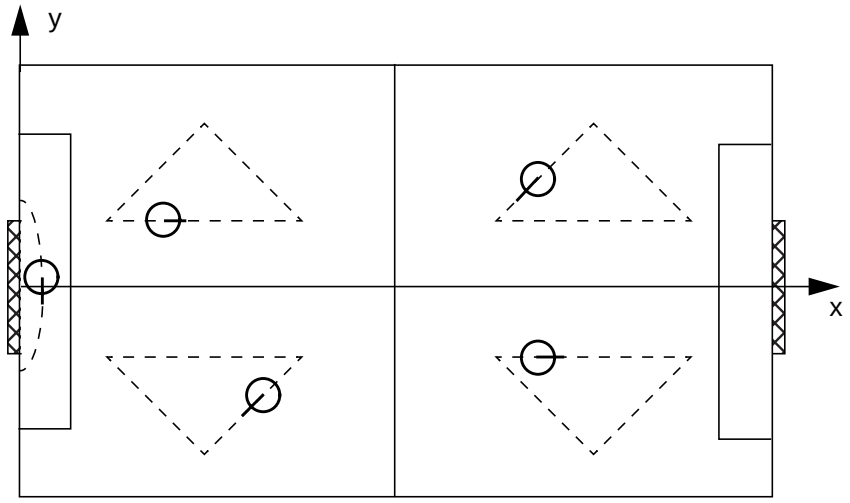

Figure 16. Robot patrolling motion according to different roles

In an earlier approach the robot just returned to a home position in the centre of its area, however, this showed to be insufficient. In case it is relatively far away the robots are not able to detect the ball even within their own area.

Example 4 displays the positions the left defender drives to, in case it does not see the ball. It only stays in the left back quarter of the field. Equivalent positions are given for the other three field players of the team.

Example 4: "home"-positions of left defender

```
PositionType home[4][3] =
{{{LENGTH/4, 0.2, 0.0},{LENGTH/2 - 0.3, WIDTH/2 - 0.3, 0.0},
{0.3, WIDTH/2 - 0.3, 0.0}},
[...]
```

In the current version of my program a robot only drives to the ball if it is detected within the half of the field connected to the position of the player. By that a defender never enters the opponents' half of the field and

an attacker never comes back into its own team's half. If the ball is detected at a point where the robot is not allowed to go, it ignores it and continues the search in its region. This must be done because, especially with objects being far away from the robot, what has been detected might not be the ball but some other similar coloured object, like a spectator's shoes.

During the competition in Singapore this method worked very well for my team. At least one robot always had its location close to the ball and, given enough time, managed to detect and approach it.

Having a reliable image processing system and a working communication system the robots could communicate the position of the ball to the other players of their team and decide which robot is in the best position to approach the ball. It would further be possible to change roles dynamically during play. If, for example, a defender robot was following or dribbling the ball and thereby leaving its position, it would communicates this status to one of the attackers, which could then change home-position and role with this robot.

This approach still enables the robots to act individually and independent of any global sensing system. In case the communication system breaks down or cannot be used, every robot can easily remain in the restricted area of its current role without restraining the abilities of the whole team too much.

6.3 Behaviour Modules

In order to regularly gain data from each sensor as well as to be able to react accordingly , several concurrent threads are started at the beginning of the game. Position tracking, search for the ball and obstacle detection are the most important ones. The driving of the robots is controlled by another process, `drive_control`, which decides the next action by using a subsumption logic approach [Bräunl96/2]. According to several flags set in the threads and a hierarchical order between the different sensor readings (Figure 17), an appropriate motion command is selected.

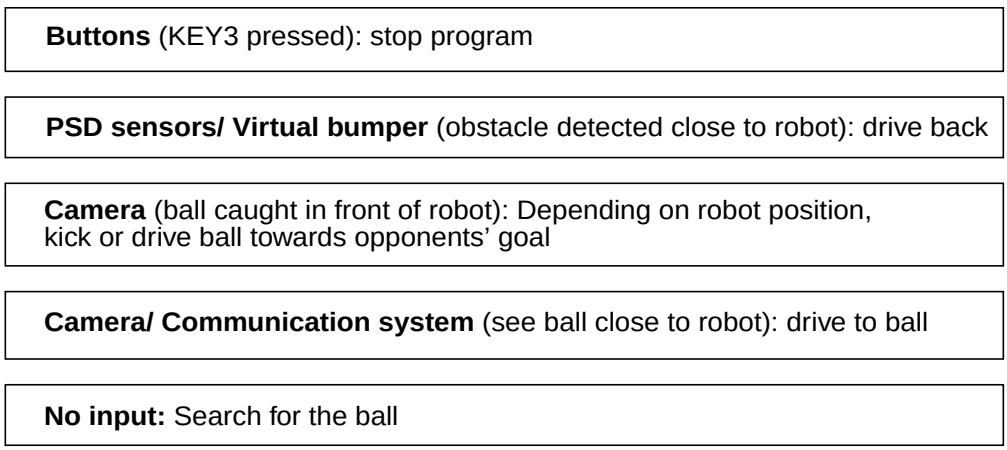

Figure 17. Hierarchy of sensor readings and resulting robot behaviours

The lowest hierarchical item is the search for the ball, which makes a robot drive around in its area of competence and check for ball coloured regions.

This procedure is interrupted as soon as the ball is found at a point close to the robot. With the goal to transfer its position in the field to all other robots in the team as well as check the reliability of the information the coordinates of the ball are first transferred into global ones on the soccer field.

In case it is in the best position among its team mates a robot starts driving towards the ball. It must thereby try not to kick the ball to the wrong direction, thus, eventually drive around it in order to approach it from behind. If the ball is caught in front of the robot, another function takes over which makes the robot start dribbling or kick the ball towards the opponents' goal.

Second highest priority is given to the obstacle detection process. As soon as an obstacle is detected close to the robot, the obstacle avoidance routine gets full control of the driving actions to prevent the robot from running into walls or opposing players. Another thread makes it possible to interrupt the running program by pressing the KEY3 button.

6.3.1 Obstacle Avoidance

"Fair Play" has always been considered an important issue in human soccer. Therefore in my robot soccer team the same importance has been laid on this factor. The robots constantly check for obstacles in their way and, if this is the case, try to avoid hitting them.

A robot's PSD sensors and shaft encoders are used to detect these obstacles. If one of the PSD sensors returns a distance value below a certain threshold an obstacle has been detected in front of the robot. If comparing the shaft encoders' return values to the desired motion of the robot indicates that the robot's wheels have stalled the robot has probably hit an obstacle. In both cases the `avoide_obstacle` routine is called which makes the robot drive backwards for a certain distance until the obstacle is out of reach. If the robot has been dribbling the ball to the goal, it turns quickly towards the opponents' goal to kick the ball away from the obstacle which could be a wall or an opposing player. After that the robot restarts its search for the ball.

Difficulties come up in cases where the PSD sensors fail and the robot does run into a wall. Often the wheels slip on the ground of the soccer field and the robot does not detect it is not driving. The contrary case appears when the robot drives in parallel to the wall and one of the wheels stalls through touching the wall. The robot then keeps driving backwards without proper reason.

6.3.2 Position Tracking

A soccer player who does not know where its own team's and the opponents' goal are located cannot be of much help for its team. Therefore another important task for each robot is to constantly know its position on the soccer field.

For path calculations a global coordinate system has been laid on the soccer field. The zero-position is located in the centre of my team's goal, the x-axis leads to the centre of the opponents' goal, the y-axis along the field line to the left. In the same way all angles are positive to the left and negative to the right, a robot's orientation is 0 facing along the x-axis (Figure 18).

Position tracking is part of the robot's operating system *RoBIOS*. Given the starting position on the field (which must be set before the game is started) the updated robot coordinates (x-,y-coordinates and head-

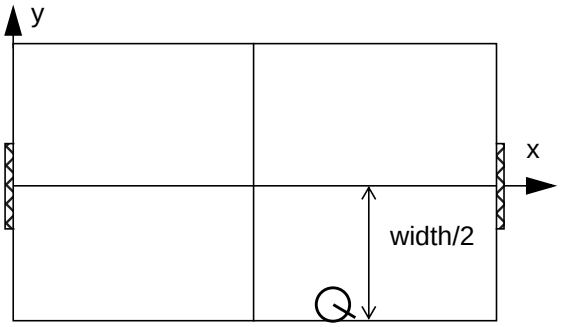

Figure 18. Global coordinate system

ing) are calculated through the robot's odometric data (shaft encoder readings) and can be requested at any point of the program. Errors due to inaccuracies of the shaft encoders are corrected by resetting the position in `avoide_obstacle`, e.g. set y-value to half the field width if running into right side wall (Figure 18). Position data is given in meters whereas PSD sensor readings are returned by the according function in millimetres. It is important not to forget converting these values before resetting the robot position.

Unfortunately this correction method can only work in cases where the position of the robot is relatively close to the wall. The value cannot be reset in any other case. Doing that would lead to bad errors in case the robot detects one of the opposing or its own team's players.

6.3.3 Object Detection

Eyesight is one of the most important abilities of a human soccer player. It is even more important for a soccer robot which is not additionally equipped with tactile, acoustic and proprioceptive sensors as humans. The robot is also not able to detect an opponent's intentions. Analysing the situation it might be possible to anticipate opponents' actions to a certain degree. However, it is impossible to copy human intuition or a person's "feeling" that the someone might act in a specific way.

In this sense the "heart" of my soccer robot program is the thread which analyses the visual input from the on-board digital colour camera in order to detect the important objects and their positions on the soccer field. The ball or the opponents' goal are detected using their colour. These colours are taught to the robot before the game is started. The robot must be placed in front of an object, then the mean colour of the centre region on the screen is calculated and saved for the object's future identification. More complicated algorithms as edge detection or shape recognition were considered but in the end not used in order to speed up this task and detect objects as fast as possible. Due to a clear definition the easiest way is a simple analysis of colours.

6.3.3.1 Centre of an Object

Before analysing an image in order to detect objects, the desired size of this object depending on its position in the picture is calculated. A fraction of this value is then used as a minimum. At the same time the maximum size for the object is given by multiplying the desired value with a factor. Checking sizes is important in order to distinguish the ball and the yellow goal which have very similar colour patterns.

The input picture is continuously searched for areas whose mean colour value compared to the trained colour value of the ball or goal is below a certain threshold and which has the desired size.

Initial Simulation

EyeBot vehicle I was initially not equipped with a colour camera. Therefore it was necessary first to run image processing routines on a different environment. It appeared helpful to use a system where the analysed colour image could be seen in the same way as by the camera. Therefore my first picture analysing approaches were tested on a Linux PC using the *EyeBot* simulator *EyeSim* and especially generated colour pictures (Figure 19).

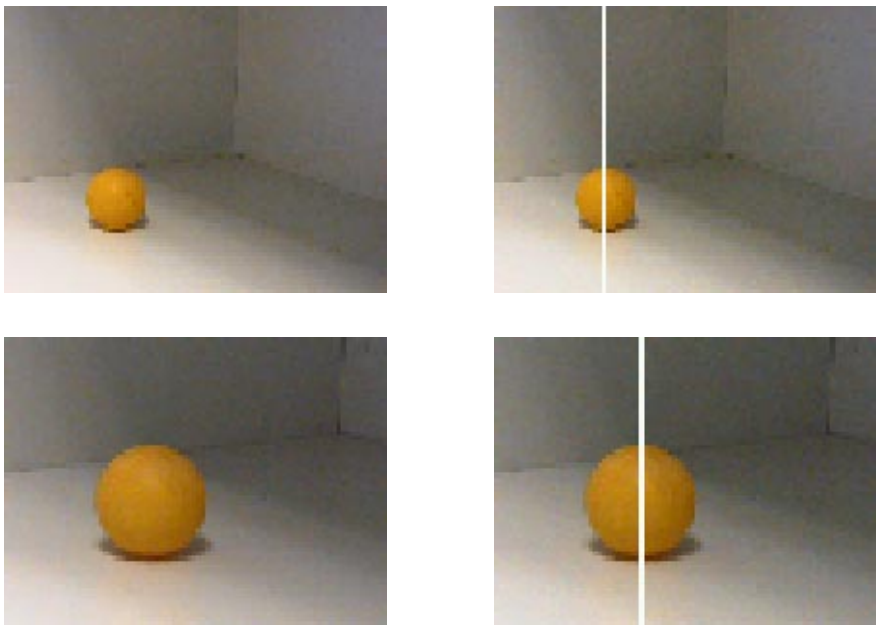

Figure 19. Test (left) and output (right) pictures with white line marking presumed ball position

In this approach the picture data (ppm format) is read and stored into an array. The picture is analysed and a resulting picture, which includes a white line marking the position of the searched object (in this case the ball) is written to an output file, also in ppm-format. The pictures have the size of 80 x 60 pixels which is the same size as of pictures from the colour camera attached to the *EyeBot* platform. This procedure is equivalent to the *RoBIOS* routine `CamGetColFrame(img)` which reads a picture from the camera and stores it in an array. Therefore, the created algorithms could be easily adapted for the *EyeBot* system as soon as it was equipped with its colour QuickCam. To compare results of different methods such as analysing randomly generated pictures, ppm pictures, or real camera pictures, a first test version of my program was equipped with the option to select between these different input variations.

Size of an Object

Before trying to detect an object in the picture its approximated size must be defined. One of my initial algorithms calculates this size using a minimal value for objects detected in the highest lines of the picture. For objects closer to the robot, the pixel value added to this size is calculated as a fraction of the line's number in which the object is detected. This approach was not used for the final version of my program because of difficulties in calculating the perfect object size values. The size of objects in the picture does not increase linearly, therefore the used linear function showed to be incorrect for numerous distances. Besides, calculating the desired object size out of the line number every time the picture was analysed showed to be an unnecessary step that only cost precious operating time and by that slowed down the overall speed of the program.

In my final program one of two tables (Example 4, second line) is used which, for any of the three available

camera positions and every line of the image, contains the factor of how to transfer a size value in pixels into the corresponding one in meters and vice versa. In this way the desired size of an object in the picture can be calculated precisely out of its clearly defined “real” size (e.g. size of the golf ball: 4.5 cm, goal: 50 cm).

Example 5: Tables to convert pixel into meter values and vice versa

```
float yfact[3][imagerows];
float x2m[3][imagerows];
```

Both tables are further used to calculate the position of a detected ball out of its pixel coordinates in the picture. For further information see Chapter .

Image Analysis

We now know the desired dimension and colour of an object. In order to find this object in the robot’s camera picture two different algorithms were developed and tested. In a first approach the detected size factor was used for the object detection algorithm. Only areas of the given dimension are tested. In case an object is slightly larger than expected neighbouring pixels are also analysed and eventually added to a detected object region.

Figure 20 gives an example for this algorithm. The picture is searched pixel by pixel (from bottom to top, left to right) until one being alike to the previously defined object colour is found. Alike hereby means that the difference to the object colour must be below a certain threshold. The values to be compared are the RGB colour values. In the example, object pixels are displayed in grey, others in white. In case the search is successful (in the bottom line (a) pixel 4 is of ball colour), the found pixel’s surroundings (unknown region: to the right) of the desired object size are tested in a similar way. For the example the desired size of an object is set to the test value of 10 pixels which leads to testing the area from pixel 4 to pixel 13. To deal with the uneven surface of the golf ball or reflections of light on the soccer field, not all of the pixels have to be of the desired colour. Another threshold is used to specify the number of pixels that must fulfil the requirements. Being successful with the analysis of the surroundings, the desired object has been found. In case the object is larger than expected and to get its width, the following pixels on the right side of the found region are analysed as well. The size of the desired object is determined the moment a pixel of non-object colour is found. In the example the line is not examined any further to the right because pixel 14 is not of object colour. The finally detected object region is displayed by a black line and goes from first = 4 to last = 13.

There showed to be one major problem with this algorithm. It deals with uneven surfaces in a region of the desired dimension of the object, but does not consider inconsistencies beyond that region. Therefore the returned size of an object is often found as too small and the object itself located too far to the left. It is further necessary to tune two different thresholds for each object - a rather time consuming and delicate procedure which should be avoided if possible.

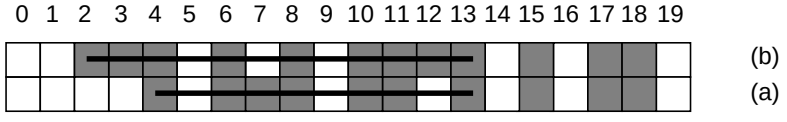

Figure 20. Looking for objects coloured regions in an image

To detect the centre of an object concerning its height the line above the one where the object has been found is searched in an equivalent way. In the example a larger region can be found in the top pixel row (b) going from pixel 2 to pixel 13 and is displayed through another black line.This procedure is repeated as long as the width of a found object is larger than the one detected in the line below. The largest found region is defined as the centre of the object.

This algorithm was not applied in later versions of my program because it showed to be too inaccurate due to the described general image processing problems.In the final version of my program, whole pixel lines are searched for regions of the desired colour. Figure 21 displays another simplified line of pixels. The implemented algorithm initially searches for ball coloured pixels at each end of a line (see region (a): first = 0, last = 18), then the mean colour value for the region inbetween is calculated. If it is below the threshold the ball has been found, otherwise the region will be narrowed, attempting to find a better match (see region (b): first = 4, last = 18). The algorithm stops as soon as the size of the analysed region becomes smaller than the desired dimension of the object. In the shown example line, a ball with a size of 15 pixels is found after two iterations.

Figure 21. Analysing a whole image line

The chosen algorithm appeared to be faster and more accurate than the previously described method. Most of the time the middle of the ball was detected correctly. Only few times the position was moved slightly to the right or left, depending on shadows lying on one side of the ball.

Texture

One of the main problems in detecting the orange soccer ball was its resemblance to the yellow goal. Figure 22 shows 10 by 10 pixel excerpts of pictures taken of the yellow goal and the ball. The colours look very alike, the only difference is the smoothness of the colour structure of the goal compared to a rather unsteady one of the golf ball. In another approach these different textures were used in order to get better results in differing the ball and goal. When teaching the ball and goal colours to the robot for their future identification, not only the mean colour values are saved but also the deviation of neighbouring colour values of the selected region. Unfortunately even with this approach I did not get any better results, presumably because depending on the distance and therefore on the size of the ball its texture varies too much.

Figure 22. Texture of the golf ball (left) and a corner of the yellow goal (right)

RGB versus HSV

The initial use of RGB colour values was changed to an approach where these three components are translated into HSV (Hue, Saturation, Value) parameters. Only the H-value is then used for the comparison of colours. This approach showed to be more reliable due to steadiness in different lighting conditions. By comparing only one value instead of three this function further helped to be more time efficient.

6.3.3.2 Global Ball Position

Once the ball has been found in the picture the next step is to translate these values into global coordinates on the soccer field. This is done as follows. First, the position of the ball as seen from the robot is calculated out of the given pixel values. This calculation must depend on the *height* of the centre of the ball in the picture. The higher the position in the image, the further an object's distance from the robot. In a second step, given the robot position and orientation, these local coordinates are transferred into a global position on the field.

In a first approach a simple quadratic function was used to do these calculations. Through several simple tests matching distance and width (in cm) to detected pixel values were gained. Then a function to represent the found relations was needed. For distance and width of an object the following calculations were selected:

$$distance\ (cm) = 0.01 \cdot (height\ (pixles)^2 + 11).$$

$$width\ (cm) = 0.0075 \cdot (height\ (pixels)^2 + 15).$$

Figure 23 shows the found relations and selected approach functions between pixel coordinates and position in meters depending on the line in the picture where the ball is found. The real distance of the ball is calculated directly through this function whereas the values for the width display the size of an object which extends over a whole pixel line. The width of an object in meters is therefore given by dividing a pixel value through the number of pixel columns in the picture. The resulting value is then multiplied by the pixel width of the object.

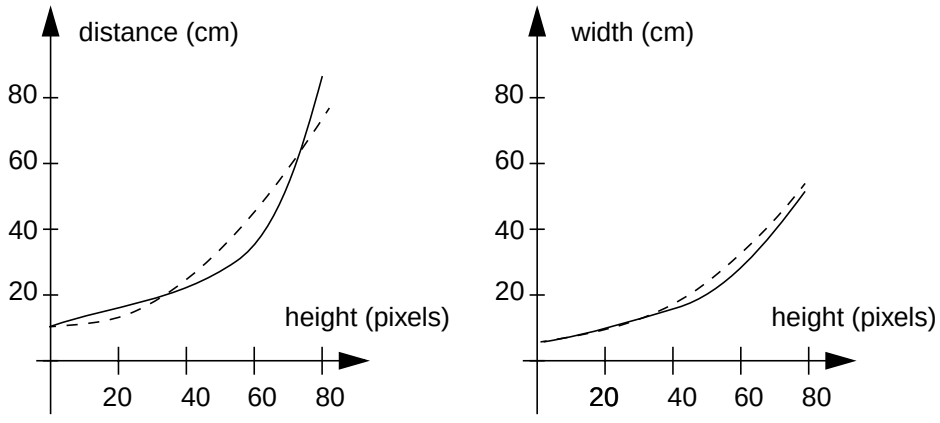

Figure 23. Found relations (solid lines) and approach functions (dashed lines) to convert camera values from pixels into cm

In the current version of my program the tables described in the previous section are used (Example 4). They allows it to calculate the exact ball position in meters out of the pixel coordinates on the screen and depending on the camera position. By this more precise results can be acquired and at the same time

complicated calculations can be avoided.

This table was created manually - by measuring distance and width of objects detected in each pixel line. In order to simplify these measurements, a white sheet of paper put on the green soccer field was used. The gained values describe conversion factors for pixel coordinates of objects lying directly on the ground. This must be considered when detecting the position of the ball in the picture, only its bottom position can be used to calculate the exact position on the field. This method was chosen to be able to use the given table for both types of calculations, ball and goal position.

As long as the robot looks at the ball at an angle from the top, the given assignment is unique (Figure 24a). Problems would arise if the robot was looking straight towards the ball. Getting closer to the robot the latter would then only change its size in the picture, however, the position of its centre would remain the same (Figure 24b).

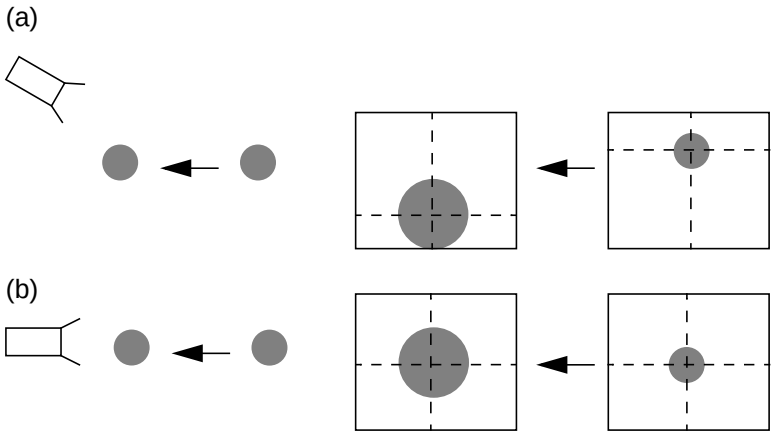

Figure 24. Images of an approaching ball using a sideways (a) and straight (b) looking camera

An example output picture of the robot LCD is displayed in Figure 25. The lines indicate the position of the detected ball in the picture while its global position on the field is written down next to the camera picture (value in centimetres).

Figure 25. LCD output after ball detection

6.3.4 Approaching the Ball (Field Players)

After detecting the global position of the ball on the field, the robot converts these values back into local coordinates seen from its current position. This step must be done in order to compensate robot motion

taken place after the image analysing thread has detected the ball and before the drive control thread could reacted on this change. In an initial version of my program the robot tried to detected a ball and drive directly to its computed local position. However, the time until these coordinates could be achieved and analysed was relatively long and the robot had already turned on the spot approximately 30 degrees before starting to approach the ball. With my final algorithm, after converting the detected global position of the ball into local coordinates, the robot drives to it without any further hesitation.

If more than one robot detects the ball, this conflict shall be solved through communication between the robots, so only one robot approaches the ball. The other robots try to find favourable locations on the field. Since the robots have a rather limited field of vision with their local cameras it is important that they are spread around the whole field. Only by that is there a chance that one of them actually finds the ball again after it has been lost out of sight.

The decision which of the robots starts driving depends on the positions of the robots on the field. Each robot communicates its position to the others, the one in the best position tries to approach the ball. As it is always easier to drive straight to the ball than around it, priority for the driving operation will always be given to a robot which is in the position to drive directly to the ball. Only if none of the robots is in this position, one of them will start turning and approach the ball from behind. Without the use of a communication system a robot which has seen the ball will always drive to it.

6.3.4.1 Selecting the Correct Motion Command

Two different motion patterns must be found. Depending on the position of a player on the field and the heading of the ball the robot can either drive directly towards the ball or, in order to avoid kicking to the wrong direction, must drive behind it. Figure 26 shows the necessary decision factors.

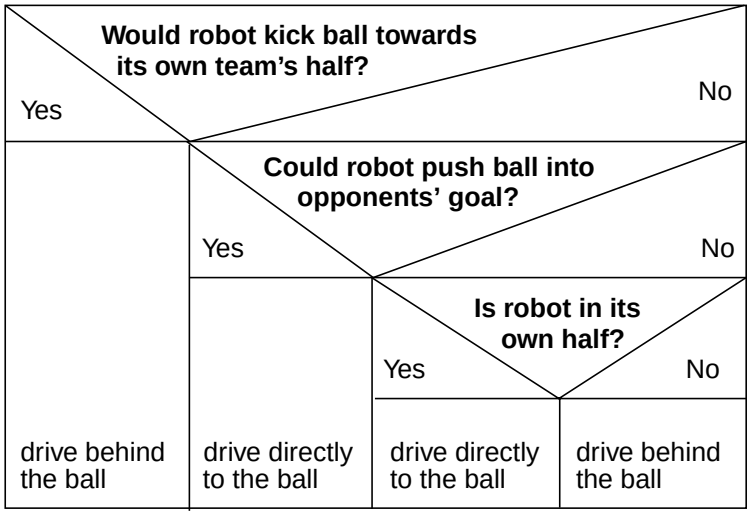

Figure 26. Ball approach strategy

Would robot kick ball towards its own team's half?

With φ being robot orientation and β being the angle between robot orientation and ball position the absolute value of the robot orientation after driving a curve to the ball is $\varphi' = \varphi - 2\beta$ (Figure 27). The absolute value of this angle must not be more than 90 degrees.

Could robot push ball into opponents' goal?

With *ballx* and *bally* being the coordinates of the ball on the field, *length* being the length of the field and φ' being the above described heading of the robot after driving directly a curve to the ball, $ydist = (length - ballx) \cdot tan(\varphi') + bally$ is calculated as the distance where the robot would hit the goal line if it drove a curve to the ball and then towards the goal in a straight line (Figure 27).

If the absolute value of *ydist* is smaller than half the width of the goal, the robot is able to push the ball directly into the goal.

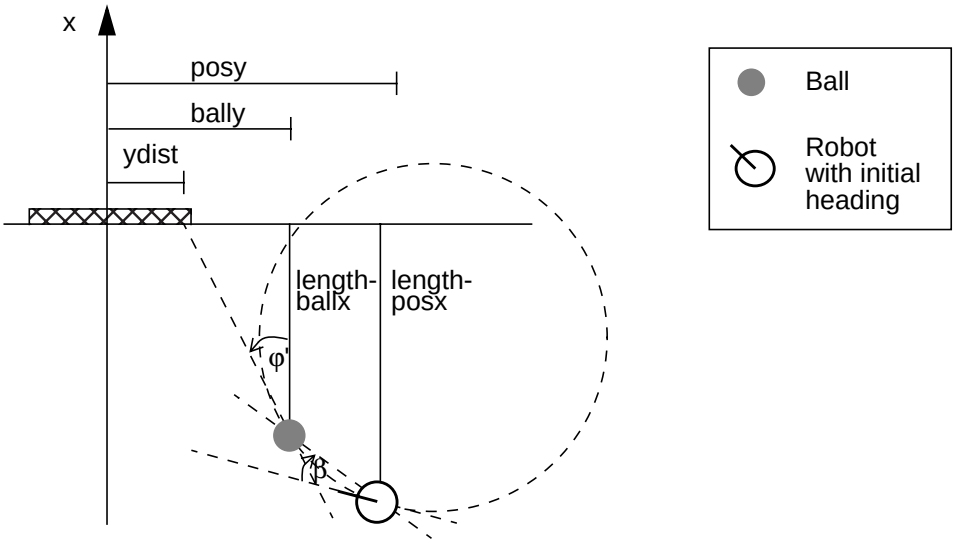

Figure 27. Calculating the possibilities to score a goal

Is robot in it's own half?

If the x value of the robot position is smaller than half the length of the field the robot is still in its own team's half.

6.3.4.2 Path Planning

Direction of the ball

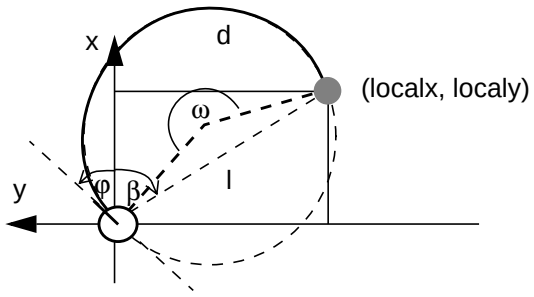

Figure 28. Calculating heading and path to drive to the ball

In order to approach the ball, the angle between robot orientation and a line from the robot to the ball must be known. With *posx, posy* being the global coordinates of the robot and *ballx, bally* being the global position of the ball, the local coordinates of the ball as seen from the robot are computed as
localx = *posx* - *ballx*.
localy = *posy* - *bally*.

The local angle of the ball is then given as $\beta = DiscAtan(localy, localx)$ (Figure 28). *DiscAtan* hereby computes the principal value of the arc tangent of $\frac{localy}{localx}$ in a simplified way. The signs of both arguments are further used to determine the quadrant of the return value. A detailed description of the `DiscAtan()` function can be found in Appendix A.

In case the robot does not face directly towards the x-axis, its global orientation φ must be considered. The resulting angle of the ball is $\delta = \beta - \varphi$ (seen from x-axis, “move back” robot orientation, “move forward” ball orientation). The value of this angle must be adjusted to the range between π and $-\pi$. Example 6 shows the function used for these calculations.

Example 6: Get angle between robot and ball

```
/*********************************************************************/
/** Get angle.
    Get angle between robot orientation phi and point (diffx, diffy).

    @param diffx, diffy difference in coordinates between point to
    current robot position
    @param phi robot heading
    @return diffphi angle between robot heading and point (x, y) */
/*********************************************************************/

float get_angle(float diffx, float diffy, float phi)
{
  float diffphi;

  /* angle to desired position */
  if (fabs(diffx) < epsilon && fabs(diffy) < epsilon)
    diffphi = phi;             /* point reached -> no angle difference */
  else
    diffphi = DiscAtan((int)(1000.0 * diffy), (int) (1000.0 * diffx));

  /* difference to current heading */
  diffphi -= phi;

  /* angle always between 0 and 2*PI */
  if (diffphi >= 2.0 * PI)
    diffphi -= 2.0 * PI;
  if (diffphi <= 0)
    diffphi += 2.0 * PI;

  /* angle always between -PI and PI*/
  if (diffphi >= PI)
    diffphi -= 2.0 * PI;

  return diffphi;
}
```

Several basic motion patterns to approach the ball have been created and implemented. In one of my first attempts only the following simple driving actions were used: drive directly to the ball (if robot could kick it directly in the goal), drive onto a circular path (in order to get the right heading for kicking it in the opponents' goal) and in case the robot faces towards its own goal, drive on a straight line beneath, then behind the ball.

Drive directly to the ball

With δ being the previously computed angle between robot orientation and ball, the angle to turn while driving on a circular path to the ball is $\omega = 2\delta$ (Figure 28). With *l* being the distance between the robot and the ball, the distance to drive in a curve is given by $d = l \cdot \omega \cdot sin(\omega)$ (Appendix A).

Turn and drive circular path to the ball

The radius of this circle is chosen in a specific way. By driving on its path the robot can hit the ball directly towards the opponents' goal. The robot first turns on the spot to get the right heading, then drives to the ball.

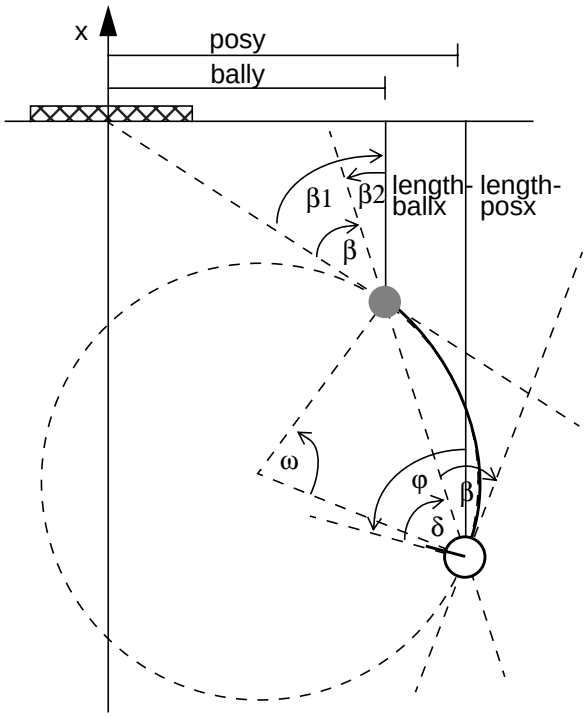

Figure 29. Calculating a circular path to the ball

The angle for turning on the spot is given through the sum $\beta + \delta$ (Figure 29). The calculation of these two angles is done as follows:

The circle angle β describes the angle of the robot to the ball after it has turned towards it. It consists of the following two angles: $\beta = \beta 1 + \beta 2$.

The angle ω the robot has to turn by driving on a circular path is given directly as $\omega = 2\beta$.

Angle $\beta 1$ describes the goal heading from ball to x-axis. *length* is the value for the length of the soccer field. x is 0 in the centre of the own team's goal. $\beta 1$ = *DiscAtan(bally, length - ballx)*.

Angle β2 is equivalent to the previously defined angle β. It describes the heading of the ball as seen from the x-axis. β2 = *DiscAtan (bally - posy, ballx - posx)*.

Computing angle δ, the orientation of the ball seen from the robot, is then done the standard way.
$\delta = \beta 2 - \varphi$.

Alternative: drive only straight lines

In this alternative approach complicated driving operations are avoided. The robot does not drive curves. Using an additional intermediate target point beneath the ball can further be used to enable a player to get behind the ball without kicking it to the wrong direction. From the intermediate target, the robot turns on the spot and drives a straight line to a point at a certain distance behind the ball. Behind means in this case, that from the selected position the robot will be able to push the ball into the goal by again turning on the spot and driving another straight line.

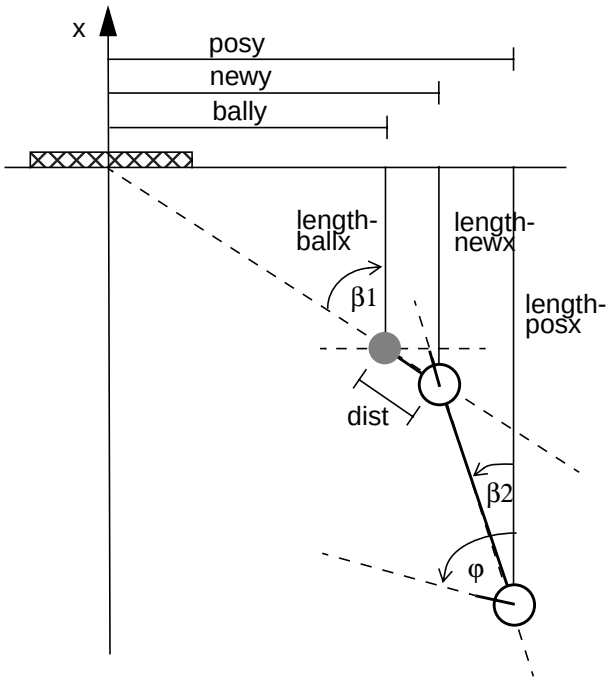

Figure 30. Driving behind the ball in straight lines

The factors for the robot motion are calculated as follows (Figure 30).

Equivalent to the last subsection, β1 is defined as the angle between a line from the ball position to the centre of the goal and the x-axis on field, *ballx* and *bally* are the global coordinates of the ball on the field.
β1 = *DiscAtan(bally, length - ballx)*.

Now given the heading of the goal, the desired new robot position behind ball can be computed, *dist* thereby is the desired distance of the robot behind the ball.

$newx = ballx - (dist \cdot cos(\beta 1))$.

$newy = bally - (dist \cdot sin(\beta 1))$.

β2 is computed in a similar way as β1, however, it defines the angle between old and new position of the robot and is given through *posx* and *posy* - the global coordinates of the robot on the field.
β2 = *DiscAtan(newy - posy, newx - posx)*.

First the robot must turn on the spot about angle $\alpha 2 = -\varphi + \beta 2$, φ is the heading of the robot on the field. Then comes a straight line of length $\sqrt{(newy - posy)^2 + (newx - posx)^2}$.

Next turning operation towards the ball is about the angle $\alpha 1 = -\beta 2 - \beta 1$. To hit the ball towards the opponents' goal the robot increases its speed and drives a last straight line of length *dist* to the ball.

The trouble with the described approach was that in order to approach the ball from the right direction the robot often drove on too big circles. Driving to several intermediate positions beneath and behind the ball the robot had to stop and turn on the spot too often as to be fast enough to reach the ball in time. Therefore another method to drive behind the ball had to be found.

In the current version of my program, the robot either drives directly to the ball (Figure 31a,c) or onto a circular path with predefined radius to reach the ball with the right heading facing directly towards the opponents' goal (Figure 31c,d). Only in very rare cases where the distance between robot and ball is smaller than the diameter of this circle, the robot turns away until it does not see the ball any more, then drives behind it and turns 180 degrees to face it from the other side (Figure 31e).

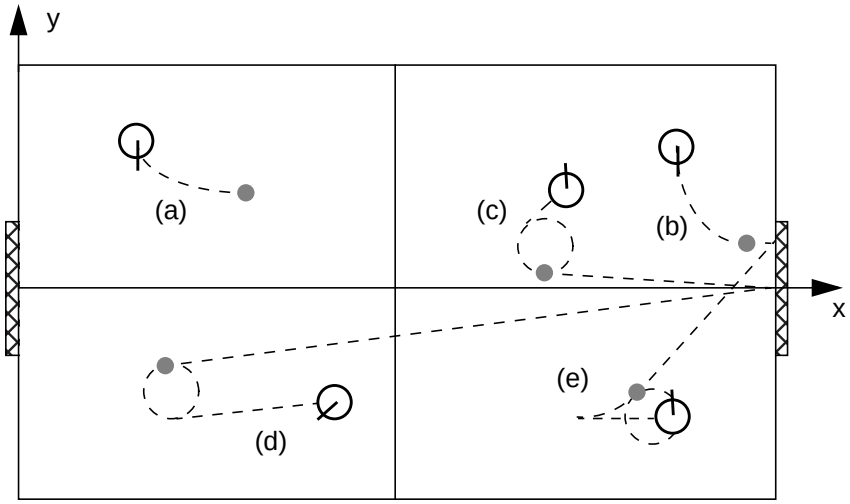

Figure 31. Ball approach cases

Drive around the ball

If a robot is looking towards the ball but at the same time facing its own goal, it can drive to a circle with a fixed radius that goes through the ball. The radius of this circle is chosen arbitrarily and was defined to be 5 cm. The circle is placed in a way that the tangent at the position of the ball also goes through the opponents' goal. The robot turns on the spot until it faces this circle, drives to it in a straight line and behind the ball on the circular path.

First of all the side of the ball where the robot drives around it must be found. It is given through the *y* values of the ball and robot position. In order to push the ball towards the opponents' goal and away from walls, the robot always drives on that side of the ball which is closer to the border of the field.

Next, the centre of the circle must be computed out of the now known *side* (`RIGHT=1,LEFT=-1`) and the specified *radius* (Figure 32). As in the last two subsections, $\beta 1$ is calculated as the angle between a line from the ball position to the centre of the goal and the x-axis on field. $\beta 1$ = *DiscAtan(bally, length - ballx)*.

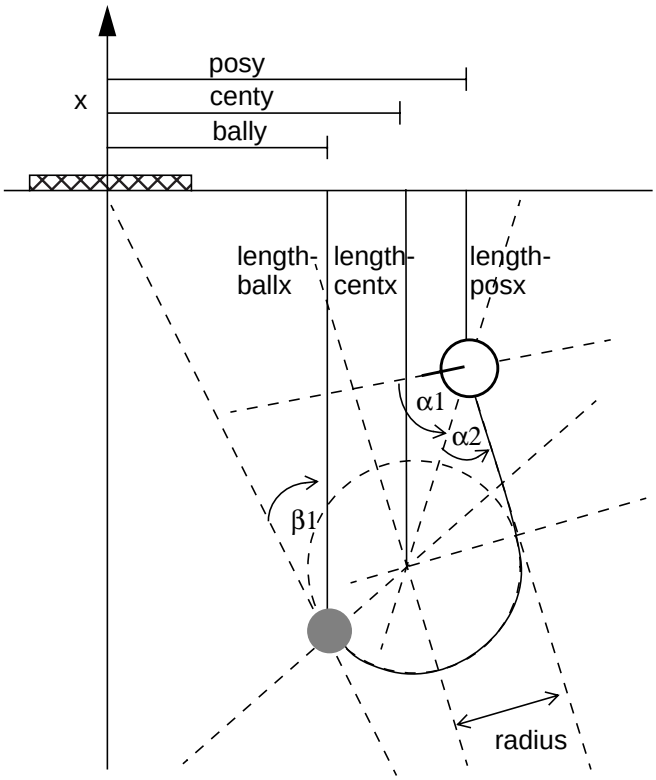

Figure 32. Driving behind the ball

With the line from the ball to the goal being a tangent to the circle, its centre can be specified as

$centx = ballx - (side \cdot radius \cdot sin(\beta 1))$.

$centy = bally - (side \cdot radius \cdot cos(\beta 1))$.

The distance between robot position and centre of the circle is given by

$dist = \sqrt{(centx - posx)^2 + (centy - posy)^2}$.

Using the standard function to compute the angle from the robot to the centre of the circle, gives us $\alpha 1$ = *DiscAtan(posy - bally, posx - ballx).*

$\alpha 2$ is located between a line from the centre of the circle to the robot and the tangent from the robot onto the circle. It is calculated through $\alpha 2 = side \cdot asin\left(\frac{radius}{dist}\right)$.

These are all the values we need. The robot turns on the spot about angle $\delta = \alpha 1 + \alpha 2$, then drives a straight line with length $\sqrt{radius^2 - dist^2}$ which brings it on the circular path. To reach the ball, the previously described function to drive directly to the ball is called.

A PI controller is included in every standard motion command for the robot (`VWDriveStraight()`, `VWDriveTurn()`, `VWDriveCurve()`). The selected driving operations are therefore executed with high accuracy. Errors in approaching the ball only occur if incorrect ball coordinates have been calculated.

6.3.4.3 Splines

Except for the case that the robot drives directly to the ball, all of the previously described driving operations start with the robot turning on the spot. To prevent from losing precious time with this operation another way of driving to the ball has been tested.

In this approach the rotational and linear speed of the robot are constantly updated depending on deviations from the desired course. In opposition to all previous motion patterns this course does not consist of simple circle segments and straight lines but is given through a curve which describes the best path from the robot's current position to the ball.

Given the robot position P_k and its heading DP_k as well as the ball position P_{k+1} and heading DP_{k+1} (once it arrives at the position of the ball the robot must be facing the opponents' goal) it is possible to calculate a spline which for every fraction *u* of the way to the ball describes the desired position of the robot. In case the ball lies towards the robot's own goal, two different splines are calculated, leading from robot position to an intermediate point beneath the ball and in the next step from this point to the desired position behind the ball.

The "Hermite blending functions" $H_0 \ldots H_3$ with parameter *u* are defined as follows [Bräunl98/5]:

$H_0 = 2u^3 - 3u^2 + 1.$

$H_1 = -2u^3 + 3u^2.$

$H_2 = u^3 - 3u^2 + u.$

$H_3 = u^3 - u^2.$

The desired current robot position is then defined by:

$P(u) = p_k H_0(u) + p_{k+1} H_1(u) + Dp_k H_2(u) + DP_{k+1} H_3(u).$

Similar to other driving functions provided by the standard library as `VWDriveStraight` or `VWDriveCurve`, a thread `drive_spline_IRQ` to control the motion of the robot while driving on this spline has been created. The function `OSAttachTimer()` is used to set the frequency how often this thread is called. This frequency must be chosen carefully. It should be high enough to prevent the robot from losing track, but at the same time it must not be called too often in order not to slow down other threads as image processing and obstacle detection. An optimal value was found at a frequency of about 6 Hz.

`drive_spline_IRQ` is started once at the beginning of the soccer program. A global variable `spline_flag` is used to determine whether the robot will drive on a spline and the necessary variables are valid and can therefore be used.

`start_spline` finally sets the factor *ufact* to increase the fraction of the spline in each step. This factor changes depending on the distance the robot has to drive. It is calculated as $ufact = \frac{1}{numberloops}$.

The overall number of loops depends on the distance to drive and the distance driven during each loop:

$numberloops = \frac{dist}{distperloop}.$

The distance to drive in each sector of the way (*distperloop)* is finally computed by multiplying the desired speed of the robot and the length of the interval in which the thread `drive_spline_IRQ` is called.

A P-controller (Appendix B) is used to calculate the linear and rotational speed of the robot at every point of its way to the ball trying to get it as close to the spline as possible. If the ball cannot be detected any more (e.g. the robot had to drive around it and lost it out of sight), the robot keeps driving to the end of the original curve. A new driving command is issued as soon as the (moving) ball is recognised at a different position.

This strategy was first designed and tested on the *EyeBot* simulator *EyeSim* (Figure 33) before running on the actual robot. Running it with goal coordinates input by hand, the program worked fine. Unfortunately this was not the case when used together with image processing and other threads. Calling the function with the desired frequency slowed down the overall processing speed too much. If the robot drove very well on the spline curve, image processing speed was cut down to less than one picture per second which is insufficient. Due to this problem the driving approach described in the previous chapter was chosen for the final version of my program.

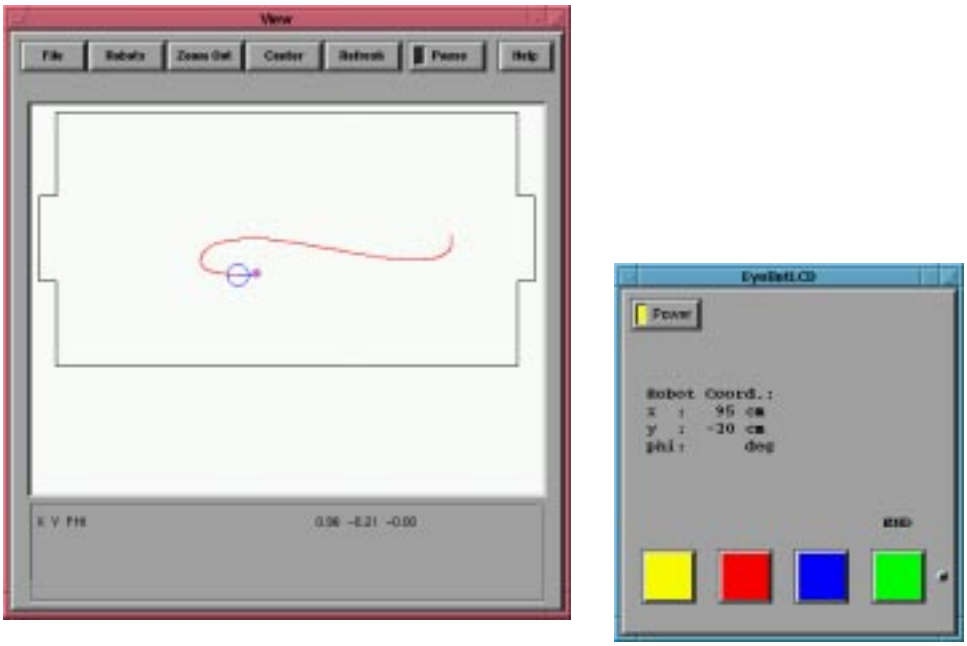

Figure 33. *EyeSim* screen display

6.3.5 Scoring (Field Players)

After describing several possibilities to approach the ball I now want to explain considered robot movements after a player has driven towards the ball. In case the ball has been successfully captured, the robot can then dribble or kick it towards the opponents' goal.

In an initial approach where *EyeBot vehicle II* was not yet built, the position of the ball in the picture was used to detect whether the ball had been caught in front of the robot. If the ball was detected very close to the robot, then disappeared out of sight, the robot acted as if it had caught the ball in front of it and started dribbling it towards the goal.

The *EyeBot vehicle II*, however, is built in a special way not to lose track of the ball even when it is lying directly in front of the robot. To detect whether the ball is caught, the camera can be tilted up and down depending on the ball's position. If the ball is detected close enough to the robot, the camera is first tilted to a `MIDDLE` position, then `DOWN` to keep sight of the ball. The robot does not drive towards the goal any more. This appeared to be too hard, even with its curved front the robot frequently lost the ball. Therefore as soon as the robot gets close enough to the ball and is facing towards the opponents' half, it activates its kicker to shoot the ball towards the goal.

In case the robot accidentally faces the wrong direction while having caught the ball in front of it (e.g. if the ball rolls in front of the robot), it drives a curve of specified radius (not too narrow, or the ball could be lost) and dribbles the ball until the correct heading is reached. During this motion the robot keeps moving its camera up and down, on one hand to make sure the ball is still there and on the other to look out for the opponents' goal. If the robot sees the opponents' goal, it also kicks the ball.

To determine whether the robot shoots the ball or drives towards the goal, the angle difference between the robot's heading φ and the direction of the middle of the opponents' goal
β = *DiscAtan(posy, length - posx)* is calculated. Being in its own team's half the robot does not have to face directly towards the opponents' goal, it kicks the ball away as soon as the absolute value of the difference between these angles becomes smaller than 90 degrees. A different threshold is used if a robot is already in the opposing team's half - it is then reduced to 45 degrees.

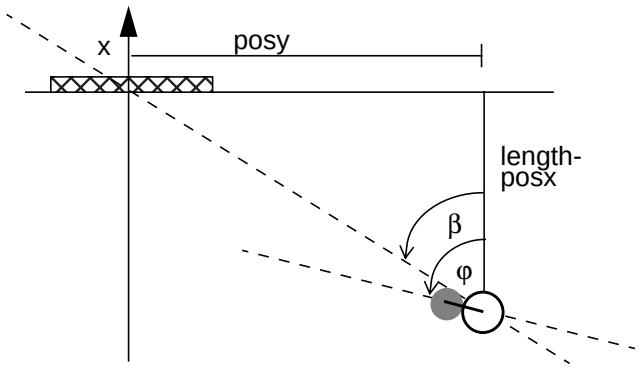

Figure 34. Deciding whether to dribble or shoot the ball

6.3.6 Blocking the Ball (Goal Keeper)

The driving algorithm for the goal keeper is rather simple: The robot is started at a position of about 10 cm in front of the goal. As soon as the ball is detected, it drives between ball and goal on a circular path within the defense area. The robot follows the movement of the ball by tilting its camera up and down just as the field players. The difference is that only two camera positions are used. If the ball is in a position where the robot would have to look down a long way in order to see the ball, it already uses its kicker, to shoot the ball away.

The circle the robot drives on has its centre behind the goal, a radius of about 50 cm appeared to be optimal. With *cent* being the distance between goal line and centre of the given circle, which can be computed out of the used radius and the distance the robot has been placed in front of the goal, the angle from the x-axis to the robot's current position is given through α = *DiscAtan(posy, cent + posx)*.

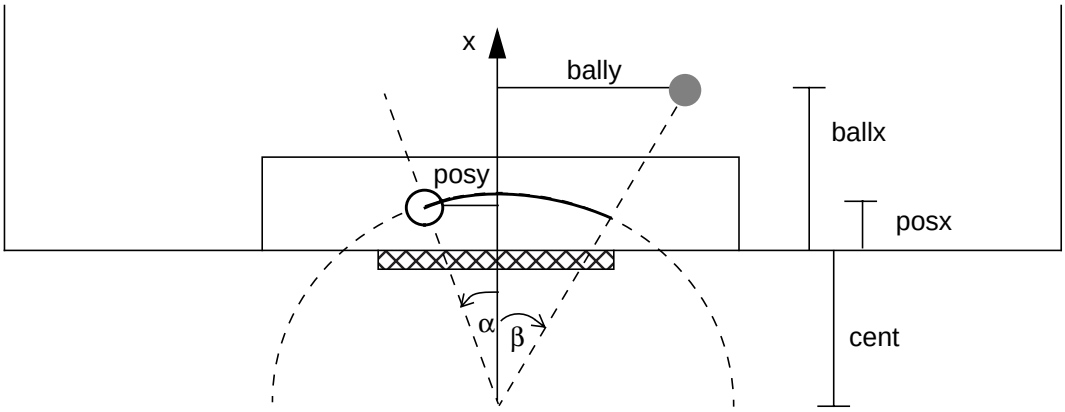

Figure 35. Goal keeper driving on a circular path in front of the goal

If the ball is seen in the current picture and was also seen in the previous one, a prediction of its future position is used for calculating the motion of the robot. This position is gained by simply subtracting the current and previous position values.

With *ballx, bally* either being the current or, if possible, presumed future ball position, the angle to the ball is then given by β = *DiscAtan (bally, cent + ballx)*.

Having seen the ball, the robot calculates the difference $\delta = \alpha - \beta$. It then drives a curve with this angle. The distance to be driven is given by multiplying δ with the radius of the circle the robot must drive on (Appendix A).

The goal keeper uses different speeds depending on the angle difference and the distance of the ball. Driving faster is better in order to catch the ball before it rolls into the goal, however, the desired driving operation might not be executed at optimal accuracy. Trying to slow down the robot abruptly after driving at high speeds results in the robot spinning on the spot or rolling along another few cm before coming to a complete stop. To cope with this, the robot always first compares desired (α) and actual heading. It turns on the spot until facing the correct direction, then it starts driving.

If the robot reaches the corner of its goal, which is tested by either analysing the robot position or using the virtual bumper, it does not drive any further. It remains on its position and turns on the spot to keep track of the ball. If the ball is not seen in a predefined number of pictures, the robot suspects that the ball has changed position and therefore turns on the spot until facing the correct direction. It then drives back to the middle of the goal to restart its search for the ball.

With help of the described possibility to correct its position, my goal keeper nearly always drives to the correct position and intercepts approaching balls. It is further able to drive back to its “home”-position in front of the goal in case the ball cannot be detected.

Problems occur just like for every other player when the robot is pushed away by opposing players. Only then it might lose the possibility to track its position and is therefore unable to return to the desired coordinates in front of the goal.

6.3.7 Special Developments for RoboCup Singapore

According to special circumstances only discovered after the first competition match in Singapore the following program modifications were made:

Rename program to `startup.hex`. A program of this name stored in an *EyeBot's* ROM is automatically executed with every start or reset of the robot. With this modification the robots are restarted a lot faster after game interruptions which took place very often. At the same time the thread checking for press of the button KEY3 (Figure 17) was eliminated to speed up image processing and drive control.

Different default starting positions were defined according to the robots' playing position. Starting positions are the corners of the own or the opponents' penalty area for defenders or attackers respectively. The robots always face towards the centre of the field. Each robot has a default starting position but can also be started at each of the other given positions. For kick-offs or action taking place around the centre of the field the centre point is used as an additional starting position.

Special program code was developed for kick-off and free kick / free ball situations. With one of the buttons in the starting menu the option “attack” can be switched on or off. Having been switched on, the attack can take place directly after button press or with a delay of one second. It results in the attackers driving 20 cm at maximum speed and defenders kick the ball and drive back 10 cm. No image processing or obstacle avoidance is done during execution of these actions.

Kick-offs are played with one defender and one attacker. The ball is kicked by the defender towards the own team’s half to the attacker. Then the robot drives out of the way. The attacker drives to the ball and kicks it towards the opponents' goal.

Only attackers are used for free kicks or free balls. They are placed in front of the ball and driving towards it as fast as possible. Figure 36 shows two attackers passing the ball at a free kick.

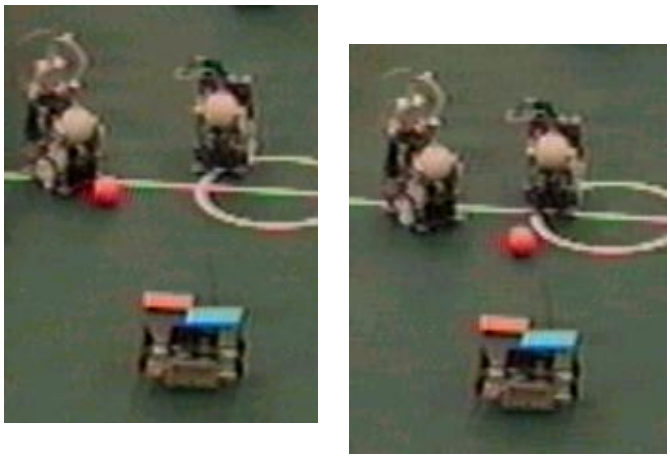

Figure 36. Passing the ball

This action appeared to be most effective during play. Due to correct positioning in front of the ball the robots always managed to hit the ball and to kick it further towards the opposing team's half.

Use of the communication system was cancelled. Wireless communication did not seem to work on two of the robots. Due to the late completion of this system the program code for transferring robot and ball position was not finished in time anyway. Communication would only have been used to synchronise starting the robots.

Chapter 7
Related Research

Figure 37. Collection of robots participating at *RoboCup-97* in Nagoya

In this chapter hardware configuration, perception, motion, and general team tactics of several exemplary other teams participating at *RoboCup* shall be described and compared. The necessary information was mainly gained out of team descriptions submitted in preparation for the *RoboCup* competitions in Nagoya and Paris. Additional information could be found on the teams' www-homepages. Web addresses of several small and middle size league teams are listed in Appendix E.

As far as I know, none of the small size league teams except mine is using on-board vision systems. Algorithms involving object detection with the overhead camera are a too different approach as to be of any help for a local vision system team. Therefore the work of most middle size league teams but only of three small size league teams has been reviewed.

In the middle size league, information about nearly all the participating teams could be gathered. Unfortunately, no detailed algorithm descriptions were available about Tübingen's runner up team *Attempo!* Instead additional information about the teams participating at *RoboCup-98* in Paris and descriptions of several matches could be found on Tübingen's homepage. The successful teams in detail:

CS Freiburg

The winning team of the Paris competition came from Germany. It mainly drew attention the usage of laser range-finders which helped the robots to determine the exact positions of own and opposing players. [Gutmann98/1] [Gutmann98/2]

Attempo!

This German team and runner up of the Paris competition uses a special self-developed image processing system to detect the ball on the field.

Trackies

This Japanese team won the 3rd place in Paris, it had further won *RoboCup-97* in Nagoya together with the *Dreamteam*. [Asada96] [Asada97] [Uchibe96]

Uttori United

The second Japanese team used three heavy robots (50 kg each) feared by other teams for their ability to simply push opposing players away. The team from Tokyo came in 4th in Paris. In Nagoya the *RoboCup Engineering Challenge Award* was presented to this team for the design of its omnidirectional driving mechanism. [Yokota98]

USC/Dreamteam

The Californians won the competition in Nagoya. In Paris they did not qualify for the semifinals. [Shen97] [Shen98]

RMIT Raiders

Together with *Uttori United*, this Australian team won the Engineering Challenge award in Nagoya for developing a specialized omnidirectional driving mechanism. In Paris they did not use this mechanism anymore. However, even with their new system they had large problems in controlling their players and did not qualify for the semifinals. [Price97] [VanBui98]

The Spirit of Bolivia

The players of this team are also known as *Ullanta performance robots*. These robots are not only used for playing soccer but are also part of a robot theatre group. They did not qualify for the semifinals in Paris but can still praise their defense for never having received a goal. All matches in Paris ended with a 0:0 draw. [Werger98]

Further information of the following teams is used: *ART* team from Italy [Nardi98], *ISocRob* from Portugal [Aparicio98], *GMD* from Berlin [Kuth98], *Agilo RoboCuppers* from Munich [Klupsch98], and *Real MagiCol* (“Realismo Magico Colombiano”) developed in cooperation of universities in Columbia and France [Moreno98].

The following successful teams were chosen as representatives for the small size league.

CMUnited

The team from Carnegie Mellon University, Pittsburgh attracted attention by winning the *RoboCup* small size league competition two consecutive times, '97 in Nagoya and '98 in Paris. [Veloso97] [Veloso98].

The *RoboRoos* [Wyeth98] from the University of Queensland, Australia arrived second, the Cambridge University Robot Football Team *CURF* [Rowstron98/1] [Rowstron98/2] fourth at the competition in Paris.

7.1 Robots and Basic Mechanics

Five of the considered middle size league teams use modified Pioneer robots [ActiveMedia98]. The others built their robots either from scratch or based on model cars. All small size league teams use self made robots. An interesting fact is that both middle size league teams entering the finals of the Paris competition

built their robots based on the Pioneer platform. It seems that a fully functional commercial platform has one big advantage: all research can be concentrated on the essentials of teaching the robots to play soccer without being concerned too much about basic skills as driving and reading sensor data.

The *Freiburg* and *Tübingen* team both have their players equipped with flexible flippers to catch the ball in front of the robot. Additionally, both teams use a kicking device to push the ball away. The *ART* and *Uttori United* teams have developed a kicking device based on air pressure. Constituted by two independent devices it is possible for the Italian players to shoot the ball to the left, right or straight ahead. *Uttori's* system is strong enough to shoot the ball at a distance of about 1 m to 1.5 m from the robot.

7.2 Vision System

For all the reviewed teams vision plays the most important role. Therefore several teams use moveable cameras in order to enlarge the vision range of their players. Two teams further use goal keepers with omnidirectional view. This enables them to keep track of an approaching ball and at the same time observe the goal line to keep their position in front of the goal.

Two of the selected middle size teams outline that they detect movement of the ball by comparing consecutive pictures. This makes it possible to predict future environment features and thereby calculate an intersection point between the path of the ball and the robot. Improved image processing is possible for teams participating in the small size robot league and using global vision. I also want to refer to algorithms developed by these teams both to point out similarities and to point out differences to local vision systems.

CS Freiburg

As in most of the other teams the ball is detected by searching for specifically coloured areas, so called "blobs" in the picture. The width, height and area size of a detected object is then calculated. Out of these pixel coordinates the relative position of the ball is computed with respect to the actual position of the robot. "This mapping is learned by training the correspondence between pixel coordinates and angles and distances for a set of well-chosen real-world positions and using interpolation for other pixels." [Gutmann98/2]

Treckies

To detect the ball, two of this team's players are equipped with rotating cameras. The robots can "look" to the side in order to detect the position of the ball as quickly as possible. To observe ball and goal line at the same time, the goal keeper is equipped with an omnidirectional viewing system.

Asada [Asada97] does not try to calculate the exact position for his *Treckies.* The position of an object is only determined by its size, approximated position in the picture and orientation (only for the goal, Figure 38). The same approach is followed by the *Dreamteam* which only differentiates three main distances and directions.

USC/Dreamteam

To analyse the situation on the soccer field the same colour QuickCams as in my team are used. However, not only one but two different cameras are attached at the front and back of each robot. They detect objects according to their colour and shape. Thereby a sample-based method is used, focusing the attention during image processing on selected areas depending on the searched object. For example, the ball

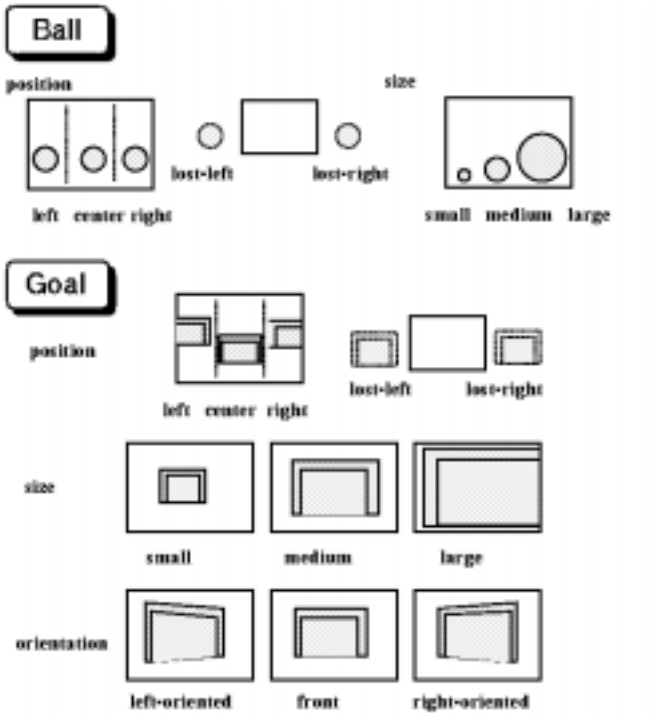

Figure 38. Simple distinction of ball and goal position for the *Treckies* [Asada96]

is only searched for in lower pixel lines whereas the opponents' goal is more likely to be detected further away, therefore in higher pixel lines. Only whole picture regions' colours are compared with the desired object's colour, not single pixels.

“The vision software checks each pixel colour to see if it may belong to one of the key objects on the field. The set of pixels of particular colours are mapped to identifiable objects and their size and position on the screen are used to calculate direction and distance from the actual object on the field.” [Shen97] The distinction between different objects on the field is thereby made using a decision tree method. The approximated direction of an object is calculated out of its position on the screen whereas to get its distance the size of a found object is analysed and compared to the known real object size.

An important perception of this team is the fact that object recognition is more stable using HSV colour space. Several tests also lead to the understanding that pictures taken by moving robots are often not good enough, therefore pictures are only taken by standing players.

To automatically adjust parameters of the vision system, pictures of the several objects are taken and analysed before the game is started. Special importance is thereby laid on taking pictures at a different angle and shadow. The gained data must be merged in a way to find a set of rules that enables the robots to recognize each object under different circumstances.

RMIT Raiders

In addition to using omnidirectional moving robots at the competition in Nagoya, this team also had omnidirectional view: their camera was facing upwards into a mirror, therefore all the surroundings of the robot could be observed. In addition, a global camera mounted 7 meters above the field was used. However, global vision is now prohibited for the middle size league and this approach could not be followed any further. Apart from that, the ominidirectional vision system appeared to be insufficient with a viewing radius of only 1 m. The team now also uses forward looking QuickCam cameras.

A similar approach as in other teams is used to detect objects on the soccer field. The algorithm searches through all pixels in the picture for “interesting” ones - where the RGB values lie below the maximal and above the minimal value identifying a selected object. After thresholding, adjacent index pixels with the same value are grouped together as one “blob”. The size and the centre of the blob can then be used to identify the object and get further information, such as its distance. Difficulties appear if overlapping blobs are observed. For instance, it can lead to the mistake of seeing only one robot instead of two in case they are standing close to each other.

The Spirit of Bolivia

In order not to lose sight of both the goal line and an approaching ball, the goal keeper of this team is equipped with two cameras. One is facing the goal line, the other looking sideways (Figure 39). To detect an approaching ball, the goal keeper is positioned at the corner of its goal.

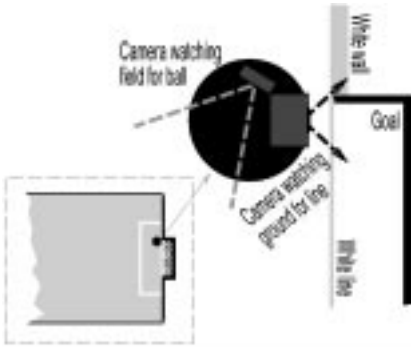

Figure 39. Goal keeper with two cameras [Werger98]

GMD

In addition to their colour camera, this team’s players are equipped with 4 colour detectors used to analyse the immediate surroundings of a robot. To further measure the light intensity, 16 gray scale sensors are applied. The used camera can be moved on a vertical axis about approximately 180 degrees.

As in several other teams, the ball is searched for on account of its colour and the centre of gravity calculated. A detected goal is represented by its bounding box.

Agilo RoboCuppers

In Munich's team, colour and shape of detectable objects are stored in a database and compared to the input image. Hereby not only the ball is searched for, but also opposing and own team’s players. By analysing the direction of the white lines in the field, the robot is able to localize its position.

CMUnited

This small size robot team uses the global vision system to detect ball and all players in the field by their colour and size. Due to the clearly defined distance of the camera to the field, the size of each object is known and can therefore be used to distinguish “real” from “fake” objects.

UQ RoboRoos

“Each image is segmented by colour. [...] Colour classification is performed by observing the distance between the RGB vector of each pixel from a specified prototype vector.” [Wyeth98] If an interesting pixel is found, the algorithm searches through its surroundings in order to find the largest region matching the template set for a certain object. “If the match level exceeds a predefined threshold the object is identified

and its location reported.” [Wyeth98] This system can be used to identify own and opposing robots as well as the orange golf ball on the soccer field.

CURF

The Cambridge University team translates each frame from its vision system into HSV space. Colour adjustments take place in a calibration phase before the start of a game. “Once the initial locations of objects being traced have been found the system software is capable of processing at full speed.” [Rowstron98/1] To enable this speed of 50 frames per second, the process of relocating objects in the soccer field is restricted to search areas where the mobile platforms or ball are likely to be. For robots this means searching up to 20 pixels from their old location, the ball might move faster and is therefore searched in surroundings 30 pixels from its previous position.

Team name	System	Image resolution	Speed	Vision range	Colour representation
CF Freiburg	Cognachrome/ Newtown Lab system	200 x 250 pixels	depending on laser scan: 5-10 global updates/s	3-4 m	RGB
Trackies	Sony camera	?	?	2 field players with moveable cameras, goal keeper: omnidirectional view	HSV
Uttori United	CCD camera	512 x 512 pixels	?	?	?
USC/ Dreamteam	2 QuickCams	658 x 496 pixels	10 frames/s	?	RGB
RMIT Raiders	QuickCam	?	8 frames/s	3 m	RGB
The Spirit of Bolivia	Cognachrome/ Newtown Lab system	?	60 frames/s	60 degree lens, goal keeper: 2 cameras	?
ART	CCD camera	380 lines	10 frames/s	goal keeper: omnidirectional view	HSV
ISocRob	?	640 x 480 pixels	50 frames/s	?	
GMD	Cognachrome/ Newtown Lab system	?	60 frames/s	View: 3 m, used: 1.5 m, moveable cameras: 180 degrees	
Real MagiCol	WAT205 PAL	284 x 288 pixels	10 frames/s	360 degrees	RGB
RoboRoos	CCD camera	?	60 frames/s	whole field	RGB
CURF	Pulnix CCD	768 x 576 pixels	50 frames/s	whole field	HSV
CIIPS Glory	QuickCam	80 x 60 pixels	2.5 frames/s	0.5 m	HSV

Table 4. Vision system specifications

7.3 Additional Sensors

Shaft encoders on the wheels are used by most of the teams to identify their players' positions. Price [Price97] outlines that this method is always combined with trouble. E.g. when the wheels slip on the floor a player soon loses track of its position. Therefore, a digital compass is used by several teams to identify the orientation of their robots.

The included sonar sensoring system is used by all of the teams working on Pioneer robots. Detecting obstacles is the task for all of the teams except the *Freiburg* team, which uses this data to double check the distance of a previously identified ball.

In order to detect collisions of their players with walls or other robots, *Uttori United* and *GMD* describe the use of tactile sensors applied all around a robot. These sensors are activated as soon as the robot hits an obstacle.

CS Freiburg

This team is additionally equipped with a laser sensoring system (Figure 40). Due to precise and frequent measurements (8 scans per second, accuracy of 5 cm up to a distance of 30 m) this system enables a robot to find and correct its position on the field. Distances of walls around a robot are thereby gained by matching sensor data against an apriori model of the soccer field. Laser scan data differing from the estimated position of the walls can further be used to identify other objects on the field, such as own and opposing team’s players.

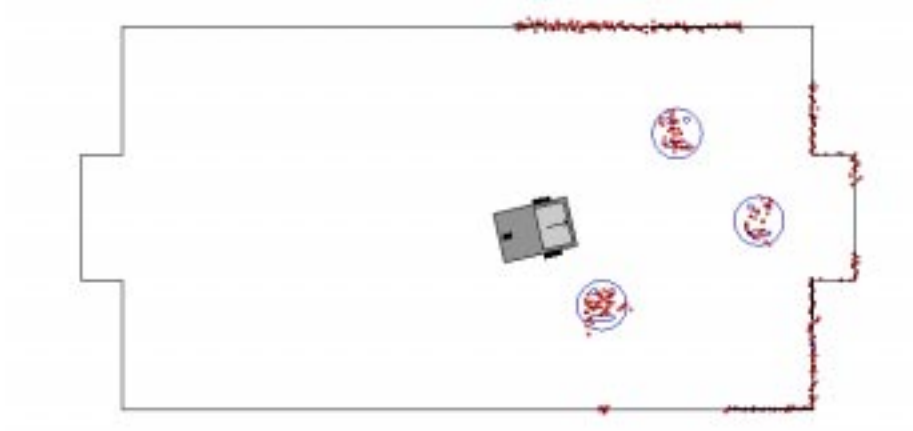

Figure 40. Detecting walls and robots through laser scans [Gutmann98/2]

“The position information obtained from odometry and laser scan matching is fused by using a Kalman filter.” [Gutmann98/2] This gives the most plausible coordinates for the robot's position. The distance the robot has covered since the laser scan was taken and matched is also taken into consideration. If an object is lost out of sight it is not immediately deleted from the world model. Last position, estimated heading and velocity are used to calculate the given ball's or robot's most likely position for another few seconds.

ART

This team is equipped with a digital compass and an infrared system. Together with information gained from the vision system, static (goals, walls, field) and dynamic (ball, players) objects are identified. There-

by the reliability of the various input devices, frequency of information acquisition and specific features of the objects are taken into account. The global position of a robot is adjusted due to information about static objects. Information about dynamic objects changes these objects' positions towards the player.

ISocRob

The Portuguese team does not only use sensors directly attached to its robots. Seven infrared sensors are placed along the border of the field (Figure 41) and constantly send out signals. To detect them, a rotating IR receiver is placed on each robot. As long as each sender uses a different frequency, angles of the currently viewable senders relative to the current robot position can be calculated. Due to this information, the exact location of the robot on the soccer field can be obtained.

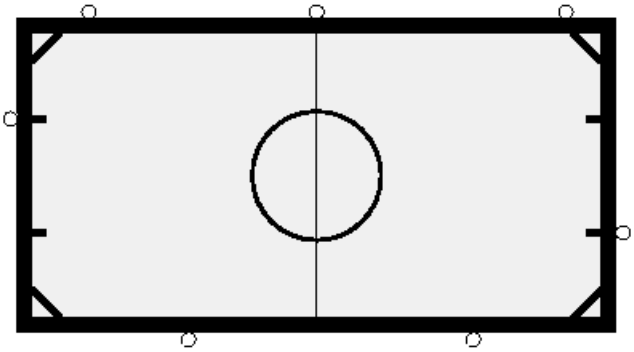

Figure 41. Infrared sensors placed at the border of the field [Aparicio98]

On-board IR sensors together with 16 bumping sensors placed all around the robot are used to provide additional information about a player's immediate vicinity.

In addition to the described features, the robots also use sound sensors which make it possible for them to detect the umpire's whistle blow (at a specified frequency) and react respectively.

RoboRoos

It is planned to add infrared sensors to the players of this team. They can be used to track the ball and opponents and to secure self localization. However, these sensors are currently not housed in the robots.

CURF

Each robot is equipped with several infrared sensors. They are used to detect opposing players which is not applied by the vision system. Sensors on each side of the robot are used to detect obstacles in its path. It stops its current motion and continues its way only when the opposing player has passed by. The front infrared sensors are further used to detect whether the ball has been captured in front of the robot and can therefore be dribbled along.

7.4 Basic Motion Skills

CS Freiburg

Due to the given identification of all objects in the field, the *Freiburg* team can use an A* algorithm to calculate the shortest collision-free path from a starting position to a goal position (Figure 42).

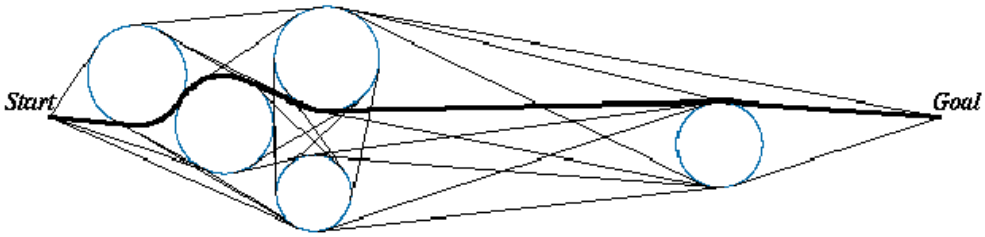

Figure 42. Path planning of a *CS Freiburg* player [Gutmann98/2]

The following different driving behaviours are distinguished:

- approach: approach a target position carefully
- go-to-position: plan a collision-free path to a given target position
- observe-ball: turn to look at the ball but do not approach it
- search-ball: turn the robot in order to find the ball
- move-ball: determine a straight line to the goal which has the largest distance to any other player on the field and dribble the ball on this line
- shoot-ball: accelerate the ball by either quickly turning the robot or activating the kicking device (depending on position)

Treckies

These robots are provided with only the primitive motion commands forward, backwards, and stop. Applied on its two wheels the robot has the possibilities to drive straight, drive curves or turn on the spot. Q-learning, a method of reinforcement learning is used to teach the robots which motion pattern to apply depending on the given state-space perception and the special task for this player. One motion pattern is thereby only altered when the state-space changes. This must be done because several state-spaces do not change very quickly (e.g. when the robot is dribbling the ball and the opponents' goal is far away). The robot can then proceed to drive forward for a relatively long time. Using the named algorithm the robots must learn to what degree actions in the distant future affect the total value of an action and act respectively (e.g. drive around an opposing player in order not to lose the ball).

Correct behaviour is initially taught to the robots by running the program on a simulator. Only if the correct motion patterns have been learned, the program is transferred onto the real robot.

USC/Dreamteam

Depending on approximated ball and robot positions, the following different motion patterns are distinguished:

- kick: kick the ball (drive directly to it)
- line-up: move to line up the ball and the goal
- intercept: calculate interception path between robot and moving ball
- homing: go back to home position
- detour: drive around the ball to approach it from behind

A situation is defined depending on the visual information given. Each detected object is identified by a vector containing the following four elements: distance between object and observer, direction of the object, changes in distance and changes in direction. For each possible set of observed location vectors, exactly one of the described motion commands is selected (out of a "policy table" for the selected player's

role, Figure 43). Remember that in order not to get too large a variety of possibilities distances are only defined as far, medium and near, directions differed between right, middle and left.

Situation	Action
Ball =<near, left>, Goal0=<near,center>	Move-left
Ball =<near, right>, Goal0=<near,center>	Move-right
Ball =<far, _>	Homing
Ball =<?,_>, Goal0=<near,left>	Turn-right
Ball =<?,_>, Goal0=<near,right>	Turn-left
.....	.....

Figure 43. Policy table for a goal keeper [Shen97]

RMIT Raiders

Information gained by the vision system is used for several features in this team. If a robot is supposed to move but the camera picture does not show large changes, the assumption that the robot has got stuck is made and the robot will try to free itself. If another robot is detected in the way of a player it will pass this robot on its left or right side.

The *Raiders* team ‘99 does not use omnidirectional motion any more. Due to inaccuracies in the robot’s odomentry, position tracking and exact driving was not possible.

Based on his experiences with omnidirectional as well as “normal” robots, Van Bui has come to the understanding that “a simple system with extensive debugging facilities is preferable to a complex system that may have theoretical advantages.” [VanBui98]

The Spirit of Bolivia

The *Ullanta* team bases its approach on the idea to “fully exploit minimal systems” [Werger98]. Therefore only three basic motion commands are available: Orbit-CW, Orbit-CCW and Kick-Ahead (Figure 44).

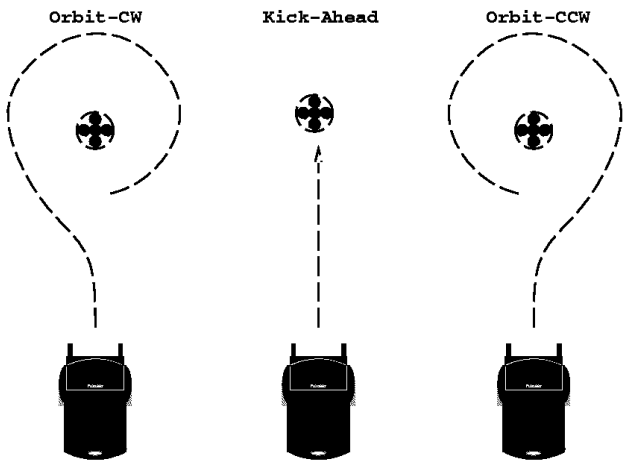

Figure 44. Basic behaviours of *Spirit of Bolivia* players [Werger98]

Robot motion behaviours are activated by uninterpreted sensor input. Once the ball is detected in a picture, this information is not translated into coordinates on the field. Instead, depending on the position of the ball and the two goals in the picture, the appropriate motion command is selected (Figure 45).

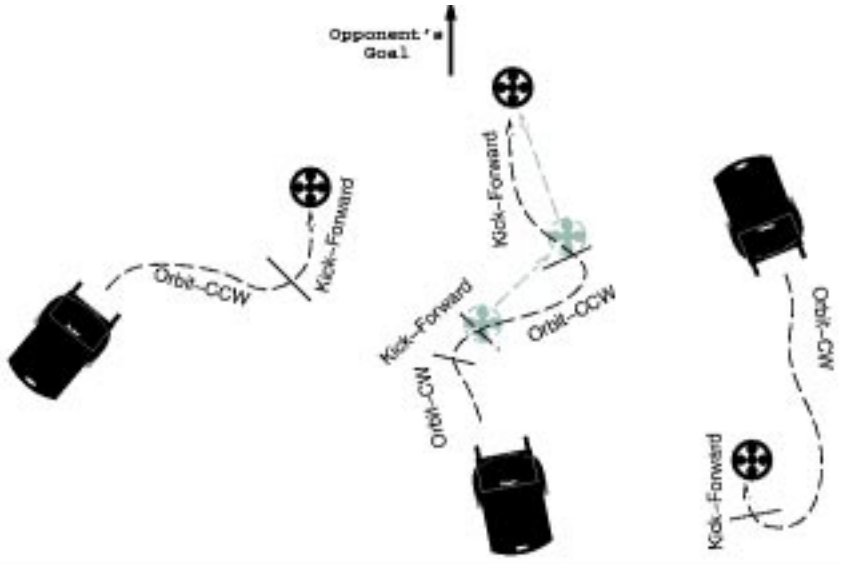

Figure 45. Changing motion behaviours depending on ball position [Werger98]

The driving speed of the robot is then set with help of a table containing rotational speed factors depending on the selected motion behaviour and the position where the ball has been seen in the picture.

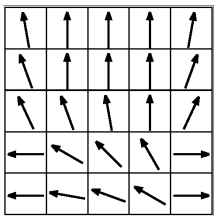

Figure 46. Rotational speed factors for Orbit-CW [Werger98]

In case the ball is lost out of sight, its position is stored in the short term memory of the robot. It therefore e.g. keeps driving around the ball, even without seeing it, to approach and kick it to the correct direction.

Three different actions are further distinguished depending on the relation between robot, ball and wall positions. The robot can dribble, which means push the ball along if it is located in front of the player. It kicks the ball with its rear in case it drove too fast so that it overtook the ball which is now lying beneath the robot. In case the ball gets stuck on a wall, the robot bats against it trying to free the ball and continue play.

CMUnited

An extended Kalman filter is used to track and predict the motion of the ball. With this filter it is possible to calculate the system state at the next step and therefore compute possible interception points between robot and ball.

The velocity of the robot is calculated with a modified version of Braitenberg's reactive control strategy:

Depending on angle φ to the target and the predefined translational and rotational base speeds α and β, the robot's new translational and rotational speed factors are set to

$v_{trans} = \alpha \sin \varphi$.

$v_{rot} = \beta \cos \varphi$.

These values are constantly updated. The following motion patterns are given for the robots:

- Ball Collection (Figure 47): A trajectory between the ball and the final target for the ball is calculated. The robot drives to a position on this line behind the ball. It can then dribble the ball towards the opponents' half or even directly into the goal.

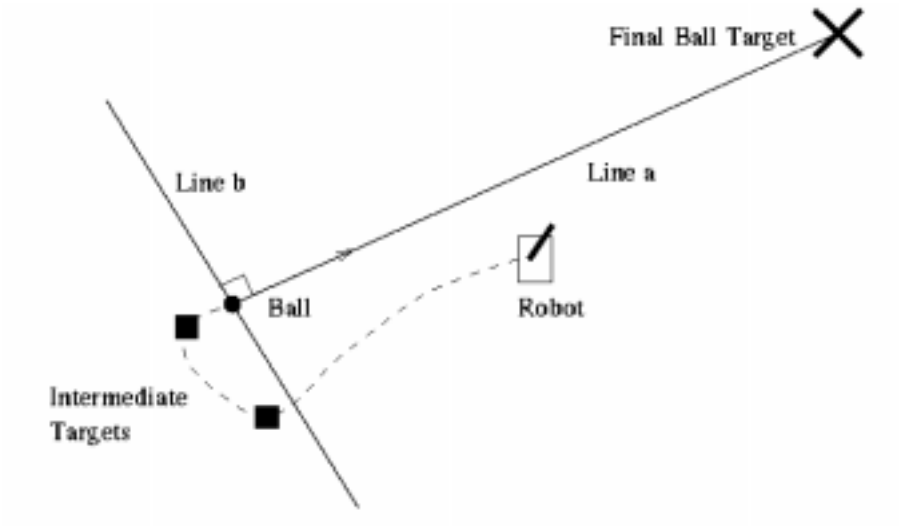

Figure 47. Ball collection [Veloso97]

- Ball Interception (Figure 48): For this motion pattern a trajectory from the target to the robot is computed. The robot then waits for the right moment when the ball passes by, intercepts it and shoots it towards the target. This action is usually favoured against the ball collection. Only if no ball interception is possible a robot will drive behind the ball.

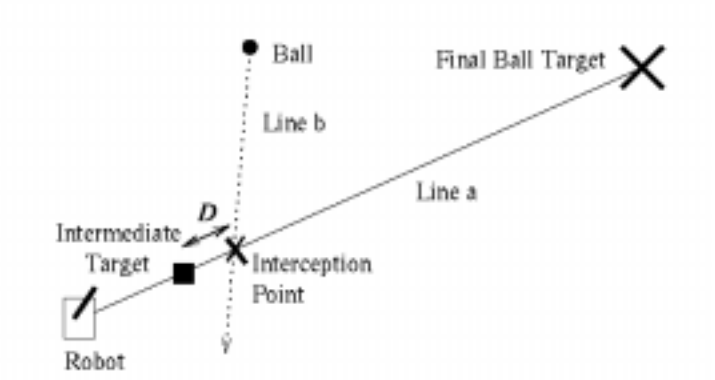

Figure 48. Ball interception [Veloso97]

- Obstacle avoidance (Figure 49): Both previously described motion patterns include obstacle avoidance. The path to the target is calculated in a way that it leads to several intermediate positions far enough from opposing players on the field.

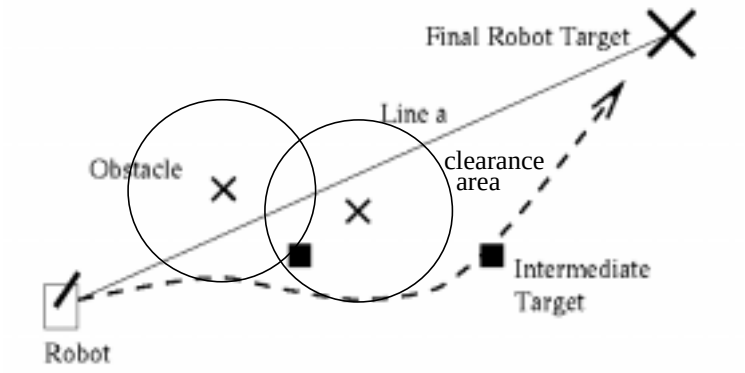

Figure 49. Obstacle avoidance [Veloso97], modified after [Veloso98]

The team for *RoboCup-98* applied several major improvements. “The target configuration for the motion planner has been exceeded. [It] includes not only a cartesian position but also a direction that the robot is required to be facing when arriving at the target position.” [Veloso98]

7.5 Communication System

CS Freiburg

Wireless communication is used to set up a global world model including positions of all the objects (robots and ball) on the field. Opposing or own players can be identified by comparing robots detected through laser scans and mapping these coordinates to the known positions of the own team’s players. In case the communication system cannot be used, it is still possible to run the program. The robots then only use their own local world model.

If, for example, a robot directly sees the ball and tries to approach it, it also only uses its local information. “Due to the inherent delay between sensing an object and receiving back a message from the global sensor integration the information from the global world model is always 100-400 milliseconds old.” [Gutmann98/2] This period is too long to use global data to control the immediate robot behaviour.

Communication is also used between robots, however, only in a very simple way. If, for example, one robot has captured the ball, it sends a “clear out” message, asking all other players to get out of the way so that the ball holding robot is able to shoot towards the opponents' goal.

Treckies

Asada [Asada97] describes his team as “controlled by a remote computer through radio link.” Communication is therefore used to transfer camera readings to several Pentium machines standing on the border of the field. All image processing is done off-board. After having analysed the current situation, the necessary motion commands are sent to the players. Each robot has its own master computer. Different frequencies are used for each communication link.

Uttori United

The developers of the Uttori United team believe “communication is [an] essential tool for cooperative actions of autonomous robots. [...] In order for a team to score a goal, its players must cooperate in bringing the ball forwards, manoeuvring between four players.” [Yokota98] Therefore, special bias in this team

has been laid on developing several communication patterns to be applied during soccer play (Figure 50). Each robot has the same importance, they are not guided by a “master” standing on the border of the field but act in a completely autonomous way.

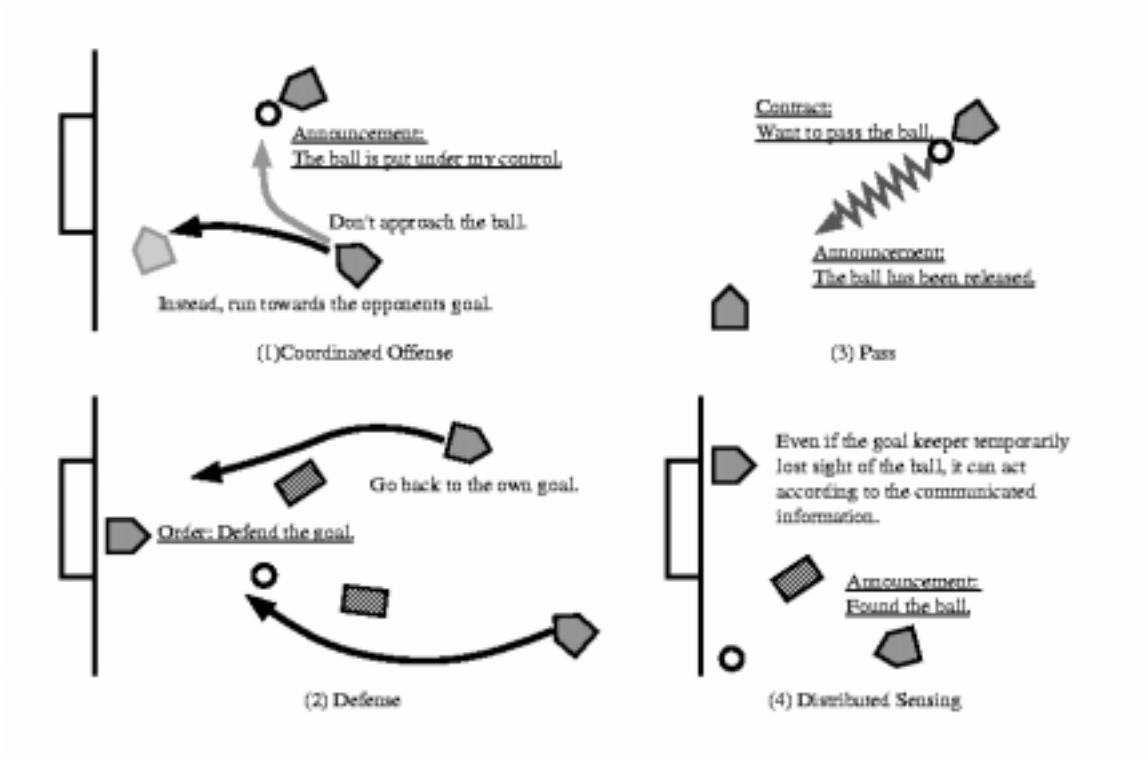

Figure 50. Communication patterns used during play [Yokota98]

The several communication patterns are specified as follows:

- Announcement: A broadcast from one robot to all other players of the team. E.g.: “I've got the ball!”
- Order: One robots tells another robot to act in a specified way. This does not contain any guarantee that the required action will really be executed. E.g.: “Back up!” (If the goalie sees the ball near it).
- Request: Retrieve information from one particular agent. E.g.: “Where are you?”
- Contract: Two or more agents try to agree on a specific action. E.g. pass the ball among each other.
- Collective decision: “voting mechanism”.
- Synchronization (follows collective decision): Start intended action when receiving this message.

USC/Dreamteam

California's *Dreamteam* outlines the intentional renouncement of any off-board resource basing their research on “totally autonomous and integrated robots” [Shen98].

ART

This team uses communication to transfer information about the status of the system among its agents. Gathered local information is broadcast to all players.

Real MagiCol

In this team consecutive global maps are additionally compared in order to detect motion of objects on the field.

Team name	System	Usage
Freiburg	WaveLan radio ethernet	master with global map
Tübingen	ARtem radio ethernet	map
Uttori United	Wireless LAN	communication among autonomous robots
Trackies	UHF transmitter	individual image processing off-board
USC/Dreamteam	-	-
RMIT Raiders	?	‘97: analyse global vision signal
The Spirit of Bolivia	-	-
ART	High bandwidth connection	master with global structure
ISocRob	WaveCell RF ethernet	communication between robots
GMD	Wireless LAN	server PC
Agilo RoboCuppers	Wireless ethernet	master
Real MagiCol	+	“coach”
CMUnited	RF system	master
RoboRoos	RF modules	master
CURF	“Piconet”	master
CIIPS Glory	FM Radiometrix Transceiver	planned: master with global map

Table 5. Communication system specifications

7.6 Multi Agent Cooperation

Most of the teams described use special roles for their robots: one goal keeper, two defenders and two attackers. Thereby the goalie always stays on the goal line trying to get between ball and goal. The defenders also position themselves close to their own team’s goal, trying to find the best position to keep the ball away from the goal. Forwards mainly stay in the opponents’ half of the field trying to shoot goals.

CF Freiburg

Figure 51 shows the standard home positions of the field players in the *Freiburg* team with overlapping areas of competence for the different roles. If these areas were non-overlapping interference between the robots would not be possible. A different strategy is used without communication system: non overlapping regions are selected for the different roles to prevent players from “fighting” for the ball with their own team mates.

Treckies

Within this team, different tasks are assigned to different types of robots. The two standard attackers have their main aim in shooting the ball into the goal, whereas finding the ball by turning their cameras is the major task of the players with active vision system.

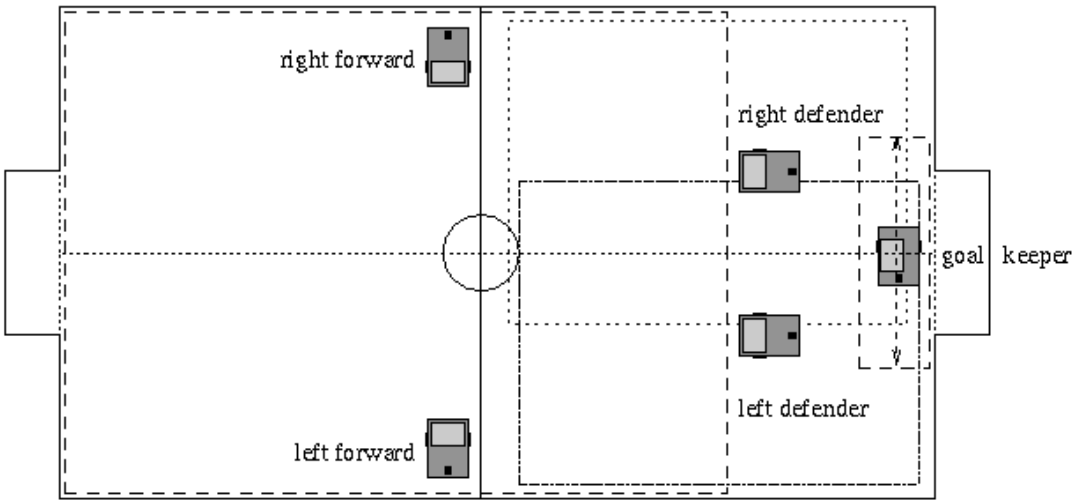

Figure 51. Home positions and areas of competence for *CF Freiburg* players [Gutmann98/2]

USC/Dreamteam

Players here are split into the standard three categories: goal keeper, defenders and attackers. Different policy tables and therefore behaviours are set up for each of these roles. If a field player does not detect the ball within its territory it drives back to a preselected home position to restart its search.

The Spirit of Bolivia

This team does not have any special roles for its four field players. With one player being in ball possession as many robots as possible try to follow this robot. If the leading robot loses the ball it is very likely that one of the other robots can recover it. In order to build this attack formation, the robots use their sonar sensors to measure the distance to the leading robot and follow it. To defend the ball against opposing players, the *Bolivia* robots group themselves around the ball trying not to let the opposing players pass.

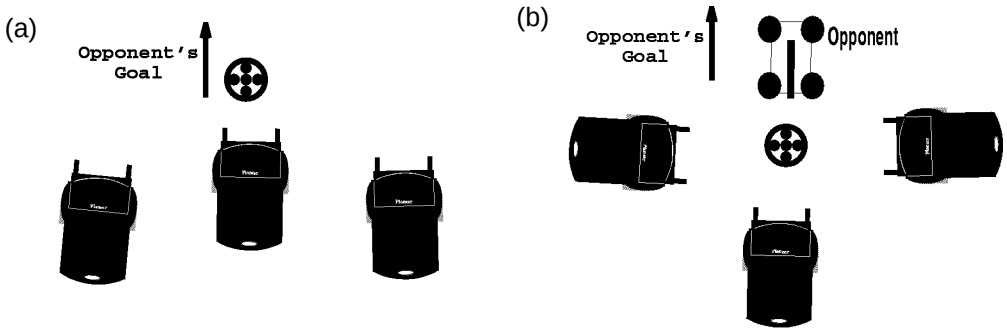

Figure 52. Offence (a) and defense (b) team behaviours of *Spirit of Bolivia* players [Werger98]

Agilo RoboCuppers

The *Agilo RoboCuppers* do not use preselected roles for their robots. Instead, through communication with a master computer staying on the boarder of the field, the best role of each robot is selected depending on the current situation. Thus, it is possible that robots change their roles during play.

Real MagiCol

France and Columbia's team does not only use different positions for its players. Different behaviours depending on a player's position and the general situation in the game can be applied. A defender may be prudent and always stay in a defensive position in front of the goal or aggressive and try to dribble or kick the ball towards the opponents' half. The position of a “coach” standing on the border of the field is added to the team which “performs global robot localization, role and formation distribution, supervision [of robot] activities and manages external information.” [Moreno98]

CMUnited

Two different formations (offensive, defensive) can be selected depending on the current score or the general team tactics (Figure 53).

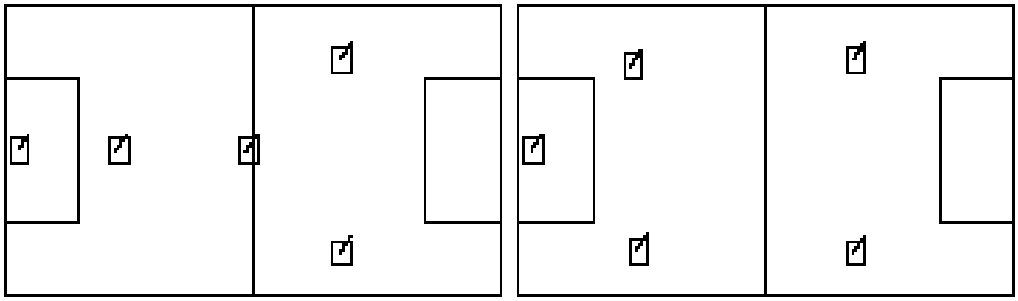

Figure 53. Offensive and defensive formation of the robots [Veloso97]

Each robot has a special role. However, it is possible for the players to change positions if necessary. If, for example, one robot is following the ball into another player’s region, these two robots switch roles. Always the robot being closest to the ball is set into an active state. It then tries to approach the ball. A robot which could receive a pass from this robot is set to an auxiliary state. The robot close to the ball then evaluates the situation and, depending on the scoring possibilities (obstruction free path?), plays the ball to its team mate or directly towards the opponents’ goal.

Three defensive behaviours are specified for the team entering the competition in Paris (Figure 54). Main goals are to stop the opponents from scoring and not endanger the own team’s goal. If the ball is in front of a player and it can shoot it away with the given clearance towards opposing players, it will do so. Otherwise, it just stays in front of the ball in order to block the opponent. If a defender is located between opponent and ball it will stay there in order to annoy the opponent.

Figure 54. Defensive behaviours: shooting the ball, blocking and annoying an opponent [Veloso98]

Attackers which do not have the ball try to find the best strategic position so that they do not block another player's shot. Apart from that it is important that a player is placed in such a position that team mates can actually pass the ball towards it (e.g. not behind opposing players). Figure 55 shows an example for a good strategic position.

Figure 55. Good strategic position for an attacker [Veloso98]

RoboRoos

Queensland's team always plays with only one defender, one centre and two wings. Thus, as soon as one robot is in possession of the ball, the other players position themselves at their assigned positions in order to clear the way and be in position to receive a pass.

In order to specify the desired position of the ball a potential field method is used. The basic field plan contains more desirable regions the closer you get to the opponents' goal. Desirable potentials are then added for the own team's players and undesirable regions for opponents (Figure 56).

(a)

6	6	6	5	4	3	2	3
9	8	7	5	3	2	1	0
9	8	7	5	3	2	1	0
9	8	7	5	3	2	1	0
9	8	7	5	3	2	1	0
6	6	6	5	4	3	2	3

(b)

6	6	6	5	5	4	3	3
9	8	7	5	4	4	2	0
9	8	7	5	4	2	2	0
9	8	7	5	3	1	0	-1
9	8	7	5	3	1	-1	-1
6	6	6	5	4	2	1	2

Figure 56. (a) Base potential field for determining the desirability of the ball. (b) Potential field with a robot of each team superimposed [Wyeth98]

Two types of commands can be sent to the players, depending on the current situation on the field, GOTO and KICK. In a defensive situation a GOTO command is sent to all players which are currently not going for the ball. "The parameters for the GOTO command are designed to occupy the space between each opposition robot and the goal." [Wyeth98]

If the ball has been captured by a player of the own team it will follow a KICK command in order to dribble the ball to a better position. The ball holding robot will further send a GOTO command to each of its team

mates not involved with manipulating the ball. The target positions for these players are specified by another potential field concerning position of the ball, opposing and own team's players.

CURF

A similar strategy is also used in this team. A terrain map representing the current state of play is designed. Opposing agents are thereby represented as hills, own agents as depressions. "Tracing a path through this terrain which attempts to remain in the lowest cumulative altitude will produce a path that avoids opposition and favours locations near our own players." [Rowstron98/1] Once the map is formed, the best path is found by applying Dijkstra's least distance path finding algorithm for a connected network. The A* pathfinding algorithm is used for speed.

"Once the optimal path for the ball has been determined, the physical agents must be set in action to guide the ball along that path." [Rowstron98/1] This is done by assigning different roles to the players on the field. A cooperation planner is used to set up passes between players or provide blocking and intercepting behaviours.

7.7 Multi Agent Learning

I further want to outline a special learning strategy of one team playing in the simulator league and winning the Scientific Challenge Award '97. This team used a genetic evolutionary approach to gradually teach 11 software agents the game of soccer. The only information given to the agents were the specification of the rules. The abilities of the individual players were assessed and depending on that, only the "fittest" agents were chosen to "breed" agents of the next generation. The gradual improvement of these agents' soccer playing abilities appeared similar to the ones of little children: first all players just chased the ball, then gradually a symmetric coverance of the field and later the specialisation of each player could be observed. [Hedberg97]

Chapter 8

Evaluation of the Used Algorithms and Possible Future Improvements

The application in a real contest situation showed various difficulties of the existing algorithms and playing strategies. Improvements can and should be made in order to successfully compete in future *RoboCup* competitions.

Mainly the image processing and therefore robot driving speed must be increased. The current version of my program runs at a frame rate of about 2.5 images (60 x 80 pixels) per second which is not very fast. Other small size league teams using global vision systems or even middle size league teams using local vision get higher resolution pictures at a frequency between 10 and 60 Hz.

A detailed description of possible changes concerning the several behaviours of the robots is given in the following subsections.

8.1 Team Structure

Further research concerning the structure and general tactics of the *CIIPS Glory* team should mainly be based on developing and improving communication between the robots. Only by that can the advantages of other teams using global vision systems be compensated.

At least one of the robots must be able to see the ball at each point of the game and communicate this location together with its own position to its team mates. These can then react to the given information by comparing their positions to the ball and among each other. Using that information, optimal positions on the field can be obtained in order to support the robot following the ball. Eventually, even passes between the robots can be made possible. This, however, depends on the accuracy of the acquired position data. It is further important that once having it in sight, the robot must by all means try not to lose the ball any more.

Using the communication system new problems will have to be solved as e.g. synchronizing the communication data and deciding how to react on contradictory data (different positions of the ball). It is further important to notice that using wireless communication will be rather time consuming. Only the most important data shall be transmitted and difficult operations will have to be made on an external server.

8.2 Obstacle Avoidance

There is no definite way to detect whether a robot has run into an obstacle. The PSD sensors fail too often and so does the virtual bumper. Using the shaft encoders, the robot can never know where the obstacle that it ran into really is. It is important to find a reliable method to detect when and on which side of it the robot has touched an obstacle. Only like that can it drive away to the correct direction and does not get trapped, for example, by driving backwards into walls or other robots. Equipping the robots with several

binary infrared sensors as used for *EyeBot vehicle I* might be the best solution.

If an obstacle is detected far enough from the robot, the player should not stop its motion. In case another robot has been found (check with coordinates) the player should find a way to drive around the opponent and then continue on its way to the target position. This is particularly important in case the target is the ball and the player might lose it out of sight by driving away from it. Motion predictions should further be included in these analyses in order to find the best side to pass an opposing player.

8.3 Position Tracking

If a robot does not see the ball for a long time, it will drive around several times in its assigned area of the field. If this occurs or other robots push a player sideways, such that the movement cannot be detected by the shaft encoders, it loses track of its position. Therefore a recalibration method, which means a way to constantly check and eventually correct the robot's position during play must be found.

One idea might be trying to detect the white lines on the soccer field and reset the robot's orientation respectively. As long as image processing is very time consuming, this method probably slows the program down too much. Apart from that, getting reliable results in line detection might be just as complicated as correct ball or goal detection.

Another method could make the robot drive along a wall, measure the distance, and then correct the angle towards it. However, during this period the robot will not be able to take part in the game which is absolutely necessary. Especially with all the other teams using global vision systems, searching for the ball should have absolute priority among other tasks. In the small size league other teams' players know the correct position of the ball at each moment of the game and can react according to that very quickly. To be able to compete with other teams, no time must be wasted.

It might be possible to compare distances to the wall if a robot does not see the ball and therefore drives around on the triangular path in its area. While driving on the path leading along the side wall the robot could read data from the PSD sensors on its side and correct its heading respectively. However, there might be a problem if other robots stand between the wall and the robot. It must be sure that the robot drives close enough to the wall, but not too close, in order to avoid getting false results.

Several middle size league teams use a digital compass to control the position of their robots. If possible, this compass should be acquired and implemented in the *EyeBot* soccer robot system. It might not give very precise results, but sometimes it is already enough to know whether a robot is facing towards its own or the opponents' goal and by that prevent it from severe mistakes such as shooting the ball in the wrong direction.

An approach as used in the *ISocRob* team might also be helpful. If the robot is not able to track its position by itself, additional infrared sensors on the border of the field might help to localize the position of a player. This, however, can only be done if the necessary receivers are small enough to be fitted on the mobile platforms. Besides, the calculations should not be too time consuming - the major goal must still be to quicken up the processing speed of the program.

8.4 Object Detection

Unfortunately, even the finally used image processing functions have certain difficulties in detecting the ball depending on the lighting conditions. This problem, however, is reported by nearly all other teams. It occurs due to shadows, spotlights and reflections on the soccer field. The fact that every robot of the my team is equipped with the same program and therefore has the same initial camera settings might cause further trouble. It is necessary to set the camera brightness and the colour values for ball and goal individually for each robot and depending on the environment. However, with the goal to create flexible programs that work under different conditions there should be a way that the robots independently and individually adjust themselves to their environment.

Another problem is that several times a robot still defines regions as “ball”-regions when the ball is actually in a completely different area of the field. Some areas seem to have similar H-values as the desired ball colour. It might further be possible that the mean of several completely different values unintentionally results in a value close to the desired one. It might therefore still be necessary to examine each pixel separately. It might further be necessary to combine both ways of representation, HSV and RGB, to get the best control. Further evaluation concerning this factor is necessary.

An approach similar to the *CF Freiburg* or *Dreamteam* might be of use. In order to identify their colours objects are examined at different angles and brightness. Minimal and maximum values instead of a mean value are further used to describe these regions. It is important to examine whole objects and not only selected regions within them. A graphical representation of the current situation on the soccer field should be implemented to be used prior to a game. By mouse clicking on one of the displayed robots the detected camera picture could be constantly communicated to the computer and displayed on the screen. Circular (for the ball) or rectangular (for the goal) regions might then be chosen to select colour patterns representing these objects.

There is one major problem in getting the exact position of the ball. The image processing algorithm described in Chapter 6.3.3.1 looks for the bottom of the ball, nevertheless due to shadows, lighting reflections or other inaccuracies the ball might not give enough contrast. It is therefore not detected at the bottom line but only further above. Calculating the position of the ball out of these pixel coordinates gives a greater distance of the ball than there really is. It is definitely a good idea trying to find the centre of the ball. To get its bottom, still before converting the pixel position into meters on the field, could be done by subtracting half of the ball’s size in pixels (Figure 57). This, however, is still rather inaccurate because of distortions of the picture - horizontal and vertical dimensions are not alike and the top region of the ball is displayed in less pixel lines than its bottom. Apart from that the bottom of the ball is often covered by shadows - the colour might not be detected as ball colour at all.

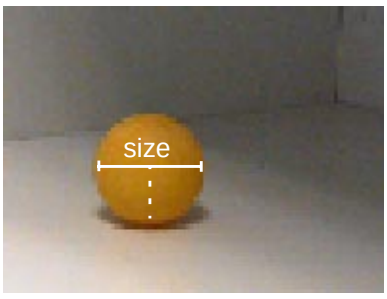

Figure 57. Finding the bottom of the ball

In order to find the middle of the vertical extent of the ball and by that its bottom position, it might be possible to run the same algorithm in a vertical and horizontal dimension and thereby examine complete pixel lines and columns.

In case the centre of the ball is used instead of its bottom position, the algorithm might be improved by not searching through the picture from bottom to top but in random order. Once a line containing object coloured pixels is found, in a first step both lines, above and below, must be examined in order to find out whether the centre of the ball lies blow or above this line. This means one more step than in the previously describe method, however, searching through the lines in random order leads to an earlier detection of lines containing object coloured pixels in general.

Future approaches will have to find a better way to match detected ball position in the picture and its actual distance from the robot. In case it appears too difficult to get the exact position of the ball's bottom, the table matching pixel and meter values might be modified in a way similar to the method used by the *CF Freiburg* team. Distances are thereby selected to match pixel values representing the centre instead of the bottom of the ball. However, due to differences in shape and size different tables would have to be set up for ball and goal detection.

Detecting distances of objects only depending on their size is used in several other teams. However, major problems occur using small resolution pictures, e.g. discriminating the ball and the yellow goal would be extremely hard. It is therefore of no use to consider applying this method for my team.

Distances measured by the front PSD sensor could be used in addition to image processing results. The exact distance to the ball could be acquired enabling higher accuracy in approaching the ball. This, however, is only possible after a method to precisely calibrate the PSD sensors has been found.

8.5 Approaching the Ball

Still no perfect algorithm for approaching the ball has been found. Distinguishing the two cases of either driving directly towards the ball or around it was the best possible strategy, however, it can still be improved. Driving directly to the ball, the robot most of the time does not face the correct direction towards the opponents' goal whereas for driving around the ball, the robot always has to turn on the spot before driving which should be avoided. A way to drive to or around the ball directly (without prior turning) and after which the robot is facing the desired way must be found. Calculations for this motion must not be too time consuming. In order to be able to compete successfully at future *RoboCup* competitions it is on the contrary desirable to speed up the program and thereby the motion of the robot.

Using splines to describe the desired driving path of the robot still seems to be a good idea, but problems in the processing speed made the use of this approach impossible. However, the discrete arc tangent function (Appendix A) was never applied to this approach. Together with the already set up tables for sine and cosine, using this function might prevent from the use of complicated trigonometric functions and therefore speed up drive control to an appropriate value. An additional library function `VWDriveSpline` should be created, using the same PI-controller (Appendix B) as other driving operations. This would simplify the use of the described spline function and make it available for everybody.

The approximation of possible future ball positions used for the goal keeper (Chapter 7) might be adapted for other players of my team. This position is calculated in a very simple way out of the current and previous ball position, which gives presumptions for where the ball might be seen in the next picture. More

accurate approaches as e.g. used by *CMUnited* could be implemented.

Several successful teams as *The Sprit of Bolivia* are using very simple reactive methods to control motion of their players. They do not completely analyse the situation but just set the rotational and linear speed depending on several predefined factors. *The Spirit of Bolivia* team does not use complicated calculations to analyse the global position of a detected ball on the field. This can just as well be done by transmitting the local ball position together with the local robot position to a more powerful computer aside the field. The latter can then calculate the global position of the ball and communicate it to the other players.

The *Ullanta* team does not even specify a target position for its player but only reacts on the detected ball position on the screen. “Complex behaviour through minimal control” [Werger98] is the motto of this team - due to the simple design of the robots this might just as well be taken over for the *CIIPS Glory* team. Experiments involving Wergers driving methods should be made in comparison to the currently used motion algorithms.

8.6 Scoring

Improvements in kicking the ball towards the opponents' goal at the right time mainly depend on correct position tracking as described in Chapter 8.3. Detecting the goal in the picture did not work very well, probably due to its large size. Improvements could be made, but priority should be laid on other factors such as the mentioned position tracking, ball detection and path planning.

8.7 The Goal Keeper

The goal keeper definitely works better than all the other players of the team. This might be due to the factor that it only performs very small movements and the danger of losing track of its position is therefore rather small. Apart from that it can always correct its heading which is defined according to the current position (right angle towards the centre of the circle).

Anyway there could be improvements. Sometimes the movement of the goal keeper - even driving at rather high speeds - does not seem to be fast enough. It is necessary to find a method to speed up the robot including the ability to stop in time before running into side walls. A way to stop the robot at exactly the desired position must further be found. This could be done by measuring the stopping distance and creating a routine to gradually slow down the robot speed.

Improved image processing is absolutely necessary for the goal keeper as well. Too often during play it thought it had detected the ball, but instead it had only seen the defenders of its own team approaching the goal on their way along the defined paths on their quarter of the field. This might also be avoided by colouring all the robots completely in black - a tactic several other teams were already using.

As in the *RoboRoos* or *CURF* team a specifically designed goal keeper might be quite useful as well. Being able to catch the ball and kick it away at a position where no opposing players can interfere is definitely a big advantage. Especially in the small size league where the ball can usually be seen by the opposing team at all times, it can be an advantage being able to hide the ball from their view and select a region to pass it without being blocked.

8.8 RoboCup - Reacting on Changes for Future Competitions

The rules for *RoboCup Stockholm* allow human interference during matches only for especially selected situations. It is not allowed to reposition each robot by hand, in order to block the way against opposing teams in free kick situations. Robots that have got stuck in corners or at the border of the field can be moved, but only enough to free them.

It will be necessary that all robots are started remotely using the wireless communication system. According to the situation, they can then drive to the desired positions on the field. This was already done by *Freiburg's* team at the competition in Paris.

To minimize interference a way should to be found that each robot repositions itself in case of a kick-off or free kick. This includes necessary corrections of the robots' positions, which during the last visited *RoboCup* competition in Singapore was done manually after each interruption of the game.

In future *RoboCup* competitions human interference will probably be restricted to the minimum of pressing a button after the umpire blew his whistle. The robots then have to react autonomously to the given information and reposition themselves. Once the robots have positioned themselves, another button press restarts the game.

Having a graphical interface displaying the positions of the robots together with an initially simplified approach of positioning the robots through mouse clicks (“drag” robot to desired position) could also be of use.

Chapter 9
Summary

9.1 Facts

I have outlined the main aim of *RoboCup* - the robot soccer World Cup - as to build a humanoid robot soccer team capable to beat the World Champion.

The *CIIPS Glory* robot soccer team has been developed to reach a first subgoal - play soccer under simplified circumstances. Thereby the bias was not laid on making these robots as human as possible but to use a simple system and try to bring it to the border of its capabilities. Much like in an ant colony, cooperation between numerous agents shall then be the major means to compensate individual weaknesses and produce an overall positive result. The basis for this behaviour - simple but still effective abilities of single agents - has been laid in my work.

I have described the necessity to detect objects as the ball, the two goals and eventually the opposing players on the soccer field and calculate their global position. Several behaviour modules have been developed to solve the task of image processing and others. Detecting the ball and the goals is done by a simple analysis of colours. Therefore an algorithm for fast image processing has been developed. It works on 24-Bit colour pictures using a resolution of 80 x 60 pixels. A mean speed of 2.5 frames per second could be achieved on the used 35 MHz microcontroller.

Opposing players are recognized by the robot's PSD sensors in case they get too close to it. Walls are further detected by a virtual bumper system.

Depending on the position of the detected objects a decision is made. The robot either drives away from an obstacle, searches for the ball or, in case it has been detected, approaches the ball. The latter can be done in two different ways, the robot either drives directly to the ball or, if necessary, around the ball to approach it from behind. Several driving methods have been developed and tested in order to solve this task.

Direct interaction between multiple mobile robots could not be achieved due to the late completion of the communication system. However, a simple team tactic was given by assigning different positions and corresponding areas on the field to each player of the team.

The selected algorithms proved their abilities at the *RoboCup* competition in Singapore. However, their application in a real contest situation lead to numerous ideas for improvements which could initially not be detected. Therefore a description of interesting approaches used by other teams participating in *RoboCup* competitions and proposals for adoption have been given. Improvements in image processing speed and driving accuracy are the major requirements in order to compete successfully in future *RoboCup* tournaments. Future research should further improve cooperative behaviour between players. Developing effective methods for using the communication system shall be the major means towards this aim.

9.2 Outlook on Future Robotics Research

Only when no further improvements can be made using the *EyeBot* robots, larger robots shall be considered for solving equivalent tasks. This approach can help to produce simple but effective programs which on more powerful machines can be run extremely fast and therefore give them very good abilities. This process can also be compared to the human process of growing up: simple tasks are initially learned and applied on simple platforms such as a child's body. Later the human body gets more complex and accordingly more complex problems can be solved. To simulate this process of growth through genetic algorithms or generally use learning strategies for the robots might also be an interesting approach.

Many middle size league teams already use very complex platforms and show quite effective performances. It is, however, important that even successful teams try to improve and further develop their systems towards the final goal.

Two years out of the predicted fifty to reach the final goal of *RoboCup* are only a small fraction. The interesting thing is, how far science will actually be after this period. Will it ever be possible to design humanoid robots that think and look like humans? *RoboCup-99* already provides a legged robot league and we can be curious how many teams will actually participate in this competition. It is, however, a lot easier to design and program four or six legged walking machines than bipedal robots which tend to tip over whenever they try to start walking. Only few humanoid robots have been developed yet. In order to equip them with all necessary sensors and actuators they get very complex and are not affordable for most research institutions.

Having a human like body still does not give a robot all the abilities humans have. The incredibly complex design of the human brain and neural system still gives hundreds of questions to today's scientists. A human being is not only the sum of its parts, not only a machine that reacts to certain signals. It also contains emotions, feelings, things that cannot be simulated because it is unknown where they actually come from.

Stories about robots ruling the world, being designed as children’ s playfellows or living among humans as equals must surely still be regarded as Science Fiction. However, only a few centuries ago, nobody would have thought it possible that humans ever set foot on the moon - and still it happened. Everything seams to be possible with a little imagination and enough time. We can therefore be curious about future development of robotics and AI research. Upcoming *RoboCup* competitions will probably serve as the major platform to demonstrate these developments.

Appendix A
Mathematical Functions

Trigonometric Functions

I now want to give a quick overview over the standard trigonometric functions [Sieber88]:

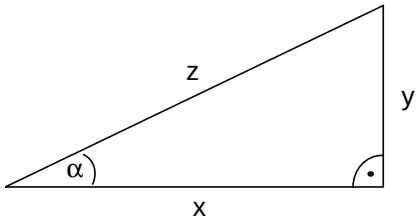

Figure 58. Trigonometric functions

Given a triangle with one side standing perpendicular to another and angle α at one of the other corners, the following equations are defined (Figure 58): $sin\alpha = \frac{y}{z}$, $cos\alpha = \frac{x}{z}$ and $tan\alpha = \frac{y}{x}$.

Circles

Paths driven by a soccer playing robot nearly always have the shape of circle segments. Some of the most important formulas concerning calculations on circles and circle segments shall therefore be described (Figure 59).

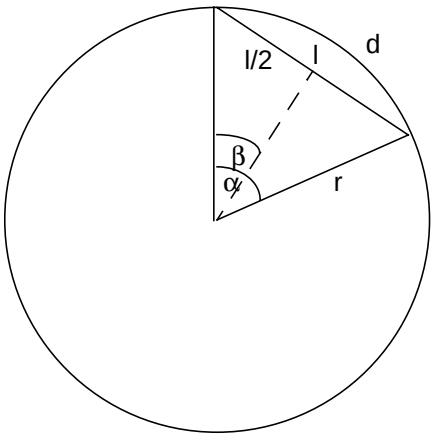

Figure 59. Calculating circle segments

The scope of a circle, depending on its radius r is given through $s = 2\pi r$.

The equivalent distance d can be calculated on circle segments, given the size of this segment through angle α. $d = r\alpha$.

If, for example, a robot must drive on a circular path and only the length *l* of the distance between start and target position is known, with $\beta = \frac{\alpha}{2}$ and $r = \frac{l}{2\sin\beta}$ the distance to drive is $d = \frac{l\beta}{\sin\beta}$.

Discrete Arc Tangent (atan)

Calculating the atan appeared to be necessary for several functions, especially to compute the angle of the ball seen as from the robot - the basis for every driving action. Using the standard `atan()` function which works on floating point numbers, appeared to be very time consuming. The responsible factors for this delay are accuracies, not necessarily needed for path calculations. Therefore a method only using integer operations as described by Bässmann [Bässmann98] has been implemented in my program.

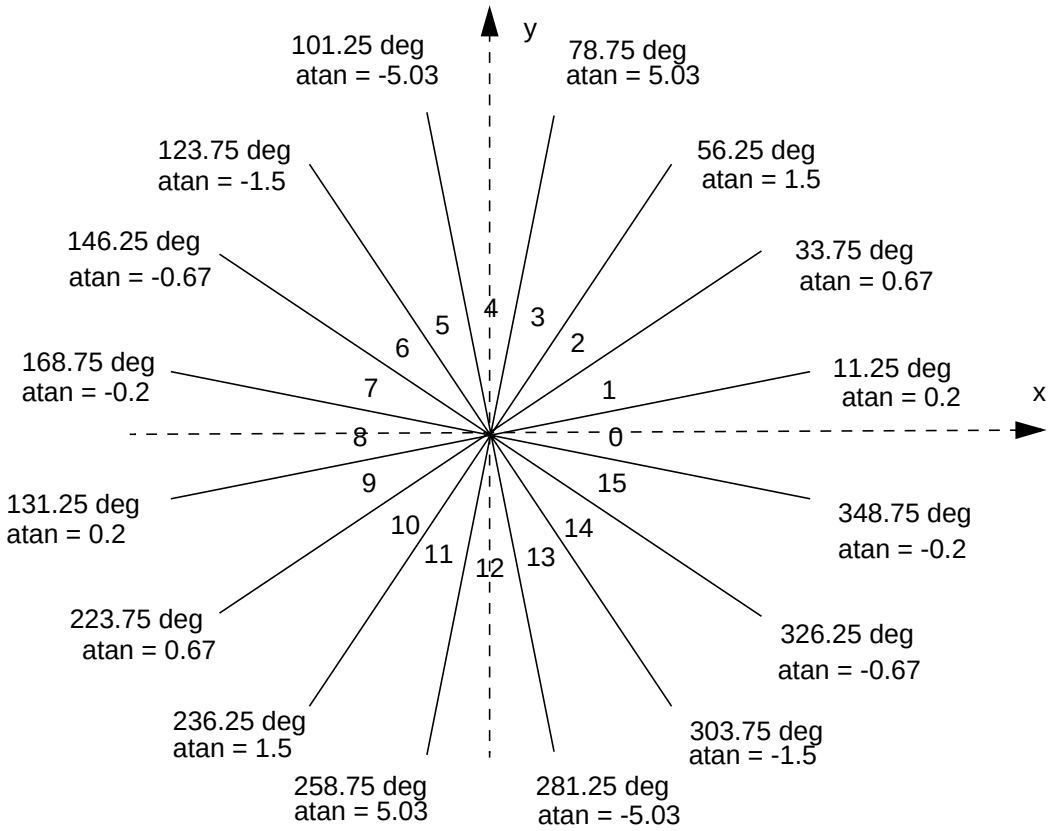

Figure 60. Considered angles and respective atan values

A circle is divided in 16 segments/angles and only for them the atan is computed (Figure 60). These values can be calculated once, then saved in a table before starting the program. Symmetry plays an important role. The sign of a resulting angle is calculated separately from its actual value (4 possible values).

Example 7 shows the procedure used to calculate the desired angle out of given coordinates `dy` and `dx` in 16 steps. The absolute value of the atan must be computed only for one of the quadrants. The procedure therefore starts with building the absolute values of both given coordinates. Then follows a test whether one of the given values equals zero. In that case no further calculations are required. If none of the coordinates equals zero, the quotient `Adx/Ady` must be built. To avoid further floating point operations, the given values are multiplied by 100. This factor results of the following consideration: The accuracy of the quo-

tient quo is high enough and the range of values for Ady (due to using a long variable) sufficient for the given application.

The actual computation of the atan value bases on a comparison of the quotient quo and the in selected borders (also multiplied with 100). Corresponding to this comparison the angle in the first quadrant is set to a value between zero and four. To adapt this value to the current quadrant, a correction of the angle values due to the sign of coordinates dy and dx is necessary.

Example 7: Function to calculate the discrete arc tangent

```
/*****************************************************************/
/** Discrete arc tangent.
    Function to calculate arc tangent only using integers.

    @param dx, dy coordinates (multiplied by 1000 -> integers)
    @return atan of dy / dx
*/
/*****************************************************************/
float DiscAtan(int dy, int dx)
{
  int phi;
  long quo, Adx, Ady;
  int return_value;

  Adx = (long) abs(dx);
  Ady = (long) abs(dy);

  if (Adx == 0 || Ady == 0)
    return_value = ((Adx == 0 && Ady == 0) ? 0:
((Adx == 0) ?
             ((dy < 0) ? -4 : 4):   /* 12:4 */
             ((dx < 0) ? 8 : 0)));
  else
  {
    quo = (100 * Ady) / Adx;

    phi = ((quo < 20) ? 0:
           ((quo < 67) ? 1:
            ((quo < 150) ? 2:
             ((quo < 503) ? 3 : 4))));

    return_value = ((dy > 0) ?
            ((dx > 0) ? phi : 8-phi):    /* phi:8-phi */
            ((dx < 0) ? -8+phi:          /* 8+phi */
             ((phi == 0) ? 0 : -phi)));  /* 16-phi */

  }
  return (float)return_value * PI/8.0;
}
```

This procedure can be extended to each desired resolution. Only the comparing algorithm must be changed. An approximation using 16 different angle values showed to be sufficient to calculate path lengths for driving operations as used in my program. It appeared to be a lot less time consuming and therefore helped to increase the overall processing speed.

Appendix B
Control Theory

“The objective of PID (proportional - integral - derivative, or three-term) controllers is the regulation of a process output around a constant setpoint.” [Singh87] It is used in feedback configuration where *u(t)* is the process input. This value is influenced by the controller in order to get outputs as close to the setpoint as possible.

$$u(t) = u_p(t) + u_d(t) + u_i(t)$$

Applied on robot motion input and output of the controller are the current and changed controlled speed of the robot. The setpoint is described as a path the robot has to drive on which leads to $\varepsilon(t)$ as the deviation between actual and desired robot position. The following terms are used:

Proportional Action:

$$u_p(t) = K \cdot \varepsilon(t)$$

The deviation $\varepsilon(t)$ is influenced by a large control amplitude for large values and small control amplitudes for small values. This system alone, however, tends to “overshoot”. Changes are applied only a certain time after derivations have been noticed, the respective control action might therefore be applied too late.

This factor influences the speed of the robot directly depending on its position. If e.g. the robot has driven too far to the left, the rotational speed factor will be set in order to make the robot turn to the right. Factor *K* must be chosen appropriately in order not to have the robot spin on the spot or drive in circles when the deviation gets very large.

Derivative Action:

$$u_d(t) = (K T_d) \cdot \frac{d(\varepsilon(t))}{dt}$$

The control system is stabilized through this parameter by anticipatory control. Adding $u_d(t)$ to the control law will adapt the control action to the natural velocity of $\varepsilon(t)$.

Changes of the error are considered through this factor. If e.g. the robot has driven too far to the left, however, the next time measured it is already closer to the desired curve, the control action must be decreased in order to prevent the robot from leaving its path to the other side.

Integral Action:

$$u_i(t) = \left(u(t_o) + \left(\frac{K}{T_i}\right)\right) \cdot \int_{t_0}^{t} \varepsilon(\tau)\, d\tau$$

This factor is generally necessary to fulfil the objective of constant setpoint regulation. P regulation works well for high derivations, in order to adjust the system even to very small derivations and thereby get very close to the setpoint this additional factor must be considered.

This factor is important to control not only the position but also the heading of a robot. If e.g. the robot has driven too far to the left it first turns right to approach the desired path. However getting closer to the path it starts turning to the other side in order to move towards the path with the right heading.

All terms can also be used without the others. However, the best result is received through combination of all three factors. The time constants T_i, T_d and the static gain K must initially be tuned in order to get optimal results.

Appendix C
Header Files

The header files show the signature of the functions building my robot soccer program. The following listing might give an idea about the complexity and interoperation of the whole program.

Example 8: General operations

```
/*********************************************************************/
/** @name general.h
    @author Birgit Graf, UWA, 1998 */
/*********************************************************************/

/** Thread to check whether KEY3 is pressed. Set end_flag
    respectively. */
void PPend_test();

/** Print error message and end program. */
void error(char *str);

/** Return -1 for input<0, 1 for input > 0 and 0 for input = 0. */
float fsign(float number);

/** Say whether player is goal keeper. */
int I_am_goalie();

/** Set Parameter (float) over keyboard. */
float set_fparam(char text[], float minp, float start, float maxp,
float inc);

/** Set Parameter (int) over keyboard. */
int set_iparam(char text[], int minp, int start, int maxp, int inc);

/** Function to calculate arc tangent only using integers. */
float DiscAtan(int dy, int dx);
```

Example 9: Initializing and reading sensors

```
/*********************************************************************/
/** @name sensors.h
    @author Birgit Graf, UWA, 1998 */
/*********************************************************************/

/** Init PSD senors that are used in the program. */
int InitPSD();

/** Thread to check front-psd sensor and virtual bumper.
    Set obstacle_flag respectively. */
void PPobstacle_test();
```

Example 10: Reading and analysing image data

```
/*********************************************************************/
/** @name image.h
    @author Birgit Graf, UWA, 1998 */
/*********************************************************************/

/** Init camera. */
void InitCam();

/** Change brightness, hue and saturation in case default values
    are not good enough (e.g. different light conditions), works only for
    colour-camera. */
void set_cam_parameters();

/** Change thresholds which are used to select ball/goal coloured
    pixels from others and size of ball/goal. */
void set_img_parameters();

/** Set ball colour to colour detected in centre of picture
    (mean value of 5x5 area around centre of picture). */
void set_ball_colour(colimage img);

/** Set goal colour to colour detected in centre of picture
    (mean value of 5x5 area around centre of picture). */
void set_goal_colour(colimage img);

/** Convert colour picture into greyscale (for LCD output),
    and mark ball position. */
void mark_object(image greyimg, int x_middle, int y_middle,
int object_size);

/** Search for regions in rows, whose mean colour value fits the
    colour of the ball, return coordinates of its centre. */
int find_ball(colimage img, int *x, int *y, int *size);

/** Search for regions in rows, whose mean colour value fits the
    colour of the goal, return coordinates of its centre. */
int find_goal(int *x, int *y, int *size);

/** Thread to analyse picture by calling find_ball and checking
   camera position, set see_ball_flag and got_ball_flag respectively. */
void PPball_test();

/** Get position of ball on field as found out in image processing
    routines and orientation of ball towards goal. */
void get_ball_coord(PositionType *ball);

/** Get current values for ball colour. */
int get_ball_col();

/** Get current values for goal colour. */
int get_goal_col(int *blue_goal);

/** Change RBG to HSV -- use hue only. */
int RGBtoHue(BYTE r, BYTE g, BYTE b);
```

Example 11: Driving

```
/*****************************************************************/
/** @name drive.h
    @author Birgit Graf, UWA, 1998 */
/*****************************************************************/

/** Init VW interface. */
int InitDrive();

/** Change parameters for driving. */
void set_drv_parameters();

/** Print x- and y-coordinates in cm and orientation in deg on LCD. */
void print_pos();

/** Set robots x and y positions. (0,0) = centre of own goal,
    looking towards opponents goal, y goes positive to left,
    x positive forward, angle is positive to right, negative to left. */
void set_coordinates();

/** Thread to move GOAL KEEPER robot and output text (drive status)
    on LCD. Activate robot movement according to flags end_flag,
    obstacle_flag, got_ball_flag and see_ball_flag, concerning
    different priorities. */
void PPdrive_goal();

/** Thread to move FIELD PLAYER robot and output text (drive status)
    on LCD. Activate robot movement according to flags end_flag,
    obstacle_flag, got_ball_flag and see_ball_flag, concerning
    different priorities. */
void PPdrive_field();
```

Example 12: Initializing and setting servos

```
/*****************************************************************/
/** @name servos.h
    @author Birgit Graf, UWA, 1998 */
/*****************************************************************/

/** Init servos. */
void InitServos();

/** Release servos. */
void ReleaseServos();

/** Move camera up or down and set cam_pos (= position of camera). */
void move_camera(int new_pos);

/** Kick ball (which should be in front of robot). */
void kick_ball();
```

Appendix D
RoboCup in Singapore

Small size league

As you will see, all the robots of the other teams which played in this league have one major thing in common - they only use global vision systems and therefore do not have any sensors attached directly to the robot. By talking with some of the other participants I gained the knowledge that most of them only use a very simple algorithm to control their robots. The only thing they try to find in their camera picture is the ball and the players of their own team. Then they send their robots to the ball. They do not use any additional sensors to detect other robots or walls of the soccer field.

Unfortunately these factors appeared as a huge problem for my robots during play. Very often they were “overrun” and pushed around by other teams’ players which did not take care whether there were other robots in their way to the ball. Due to the differences in their driving speed the *CIIPS Glory* robots rarely got a chance to catch the ball and unfortunately did not shoot any goal.

Group A

CIIPS Glory - Department of Electrical & Electronic Engineering, University of Western Australia, Australia (see previous chapters)

Lucky Star - Ngee Ann Polytechnic, Singapore (Figure 61)

Both teams from Ngee Ann Polytechnic not only used the same robots, even their programs showed several similarities. The robots were always driving very fast, looking for nothing but the ball. Not looking for any obstacles or other robots on the field, the players of these teams frequently ran into other players and pushed them around on the field. However, these actions were not defined as fouls and due to well functioning ball recognition the teams were very successful and both reached the final.

Figure 61. Rectangular robot used by Lucky Star and IC^3

Lilliputs - Electronics & Communication Engineering Department, Singapore Polytechnic (Figure 62).

Figure 62. Lilliput robot

These robots mainly drew attention by running into walls and spinning on the spot quite often. The team seamed to have huge difficulties in identifying their robots by analysing the coloured tops.

Group B

IC^3 - Ngee Ann Polytechnic, Singapore (Figure 61 and corresponding description)

All Botz - Auckland University, New Zealand (Figure 63)

This team joined the competition only at the last minute. Their robots were designed not only for *RoboCup* but also to participate in other robot competitions allowing global vision. They used model cars instead of specifically built robots. Additionally, that they did not use a camera with a special holding construction on top of the field but had their camera placed beneath the field looking down on it from the side.

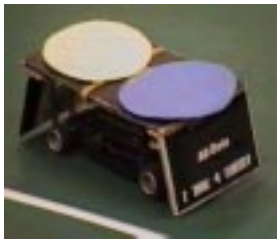

Figure 63. All Bot

KU-boxes - Kinki University, Japan (Figure 64)

The developers from Kinky University had very impressing tools to train their robots before starting a match. They used a large colour TV screen to display the picture of the top camera. It was then possible to enlarge the picture to any desired degree and draw with the mouse to select the colour pattern identifying certain areas as for example the ball and the players of the own or opposing team. Unfortunately the calculating capacity of the used computer seamed to be insufficient for the complicated processes running during play. As a result the robots only drove at very low speeds or not at all.

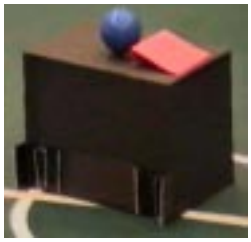

Figure 64. KU-box

Group C

Field Rangers - Electronics & Communication Engineering Department, Singapore Polytechnic, Singapore (Figure 65)

Another team with lots of difficulties in defining their robots' positions. The goal keeper of this team was designed in a special way as well - its width was extended to the maximal size.

Temasek Polytechnic - Temasek Polytechnic, Singapore (Figure 66)

Figure 65. Field Rangers: Goal keeper and field player

The robots from this team showed a similar shape as Pioneer robots, used in several middle size league teams all over the world. Their design shall enable the robots to kick the ball by driving beneath it and swinging the back of the robot around its front wheels. Apart from some nice rear end kicks, the team unfortunately did not see the ball very often during play which resulted in the robots standing around on the soccer field and doing nothing most of the time.

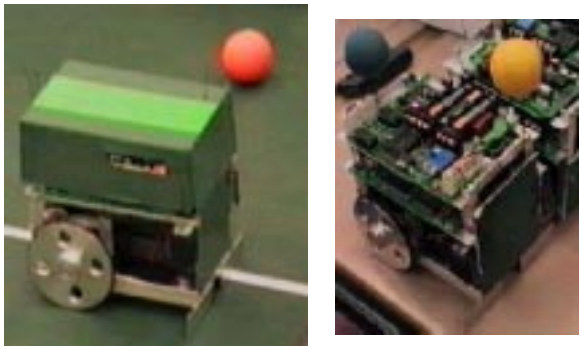

Figure 66. Pioneer - like robots of Temasek Polytechnic

UQ RoboRoos - Department of Computer Science and Electrical Engineering, University of Queensland, Australia (Figure 67)

This other Australian team became second in the last *RoboCup* competition in Paris. It uses very specialized robots. Through the little ping pong balls on the robots the own team's players and their orientation are identified. The goal keeper is designed in a special way according to *RoboCup* rules. It is capable to catch the ball inside it, then it looks for a free way on the field and kicks the ball towards the opponents' goal. One of the attackers actually shot a very nice goal after receiving a pass from its goal keeper. On the computer screen, all components on the field can be identified by mouseclick. Special moves of the robots can be programmed during interruptions of the game. An example for these actions is the capability to exactly specify where to shoot penalties - depending on the position of the opponents' goal keeper.

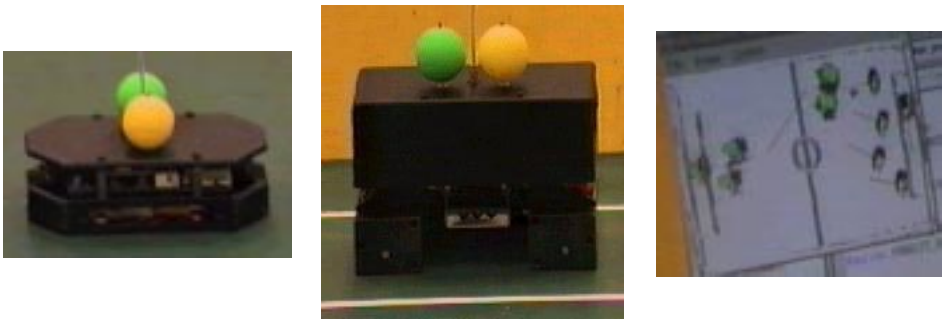

Figure 67. RoboRoos: Field player, goal keeper and computer control screen

Match results

CIIPS Glory - Lucky Star	0:9
All Botz - IC^3	1:2
Field Rangers - Temasek Polytechnic	2:0
Lucky Star - Lilliputs	0:0
All Botz - KU-Boxes	2:0
Temasek Polytechnic - UQ RoboRoos	0:2
CIIPS Glory - Lilliputs	0:2
IC^3 - KU-Boxes	2:0
Field Rangers - UQ RoboRoos	0:2

Table 6. Preliminaries

Lucky Star - IC^3	0:1
IC^3 - UQ RoboRoos	2:0
Lucky Star - UQ RoboRoos	1:0

Table 7. Semifinals

Lucky Star - IC^3	0:1

Table 8. Final

Pictures of my team's matches

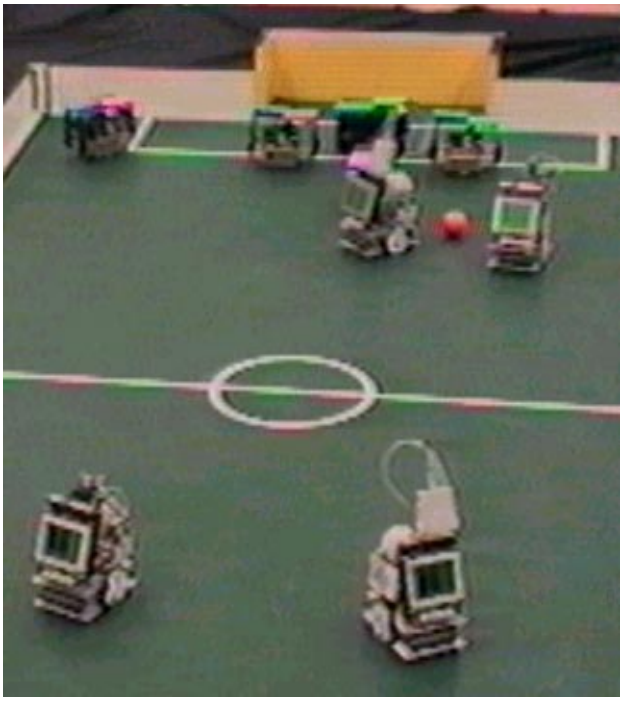

Figure 68. Go for goal! An unfortunately unsuccessful attack against the *Lilliput* team

Figure 69. Setting up for the first match [RoboCupS98]

Figure 70. Better be quick! Our goalie against an attacker of the *Lucky Star*

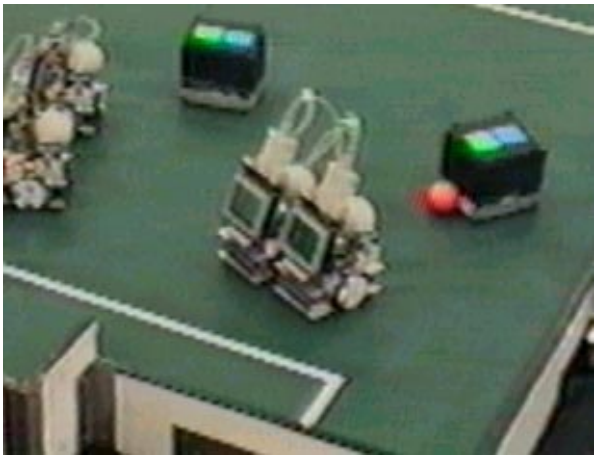

Figure 71. Blocking the way against the *Lucky Star*

Figure 72. Goalie on the watch - giving a hard time to the striker of the *Lucky Star.* Meanwhile the field players look for the ball in their assigned areas of the field.

Middle Size League

I also want to demonstrate teams participating in this real robot league, especially because during play several similarities to the robots of my team, concerning programming approaches and corresponding difficulties could be observed.

In the middle size league only two teams participated. They used very different approaches, one team with self constructed soccer robots, the other with commercial Mach5 robots [Newton98]. The rules specified that robots had to be started remotely, however, only one of the teams managed to follow this rule. During play none of the teams seemed to use communication between the robots, only the player which could see the ball actually approached it. If no robot could see the ball all players of the team started spinning on the spot slowly, trying to find the soccer ball.

Participating Teams

WISLEY - Singapore Polytechnic

These robots were built specifically for the *RoboCup* competition. Unfortunately they appeared to be rather delicate and several robots failed during play. Concerning programming strategies they seemed to have similar problems as the robots I was working with. Quite often the robots did not see the ball, even if it was lying directly in front of them and with the ball being a few meters away from the robot they were not able to see it at all.

Using a digital compass, the robots were always able to track their positions. They either drove directly to the ball or, in case they were facing towards their own goal, backed up, then drove around the ball to hit it from the other side.

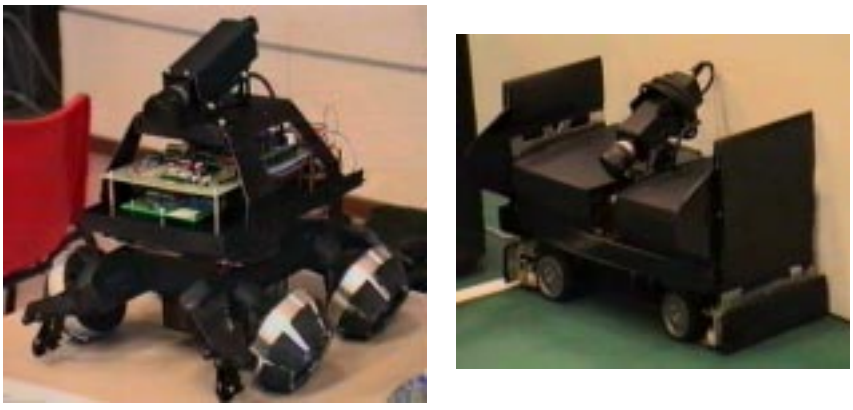

Figure 73. *WISLEY's* field player and goal keeper

Alpha plus Alpha - Ngee Ann Polytechnic, Singapore

In this team the only robot designed in a special way was the goal keeper. It used two different cameras, one facing towards the field, looking for the ball, the other following the goal line in order to track and thereby keep its position in front of the goal. The field players showed several good dribblings with the ball and shot some spectacular goals.

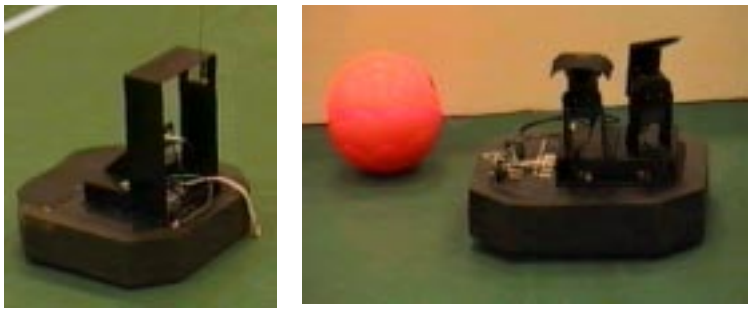

Figure 74. *Alpha plus Alpha* - common field player and modified goal keeper robot with two cameras

Match results

Alpha plus Alpha - WISLEY	2:0
Alpha plus Alpha - WISLEY	2:0

Table 9. Middle size league matches

??? League

In contrary to other small size league teams, players of the *CIIPS Glory* team are able to solve the task of following an object of a certain colour everywhere - not only on a specially designed soccer field. However, playing the big soccer ball on the middle size league field was not very successful due to the rough surface which made it hard for the robot to move without its wheels stalling. Due to the previously specified size of the golf ball the robot did further not detect the soccer ball as such because it was too big (Figure 75).

Figure 75. Small robot with big goals...

Appendix E
Homepages of RoboCup Teams

Team Name	Institution	Homepage
AGILO RoboCuppers	TU München, München, Germany	`http://www9.in.tum.de/research/mobile_robots/robocup/`
Alpha ++	Ngee Ann Polytechnic, Singapore	`http://www.np.edu.sg`
ART - Azzurra Robot Team	Universita degli Studi di Roma “la Sapienza”, Rome, Italy	`http://www.dis.uniroma1.it/~nardi/ART.html`
Attempo!	Universität Tübingen, Tübingen, Germany	`http://www-ra.informatik.uni-tuebingen.de`
CS Freiburg	Albert-Ludwigs-Universität, Freiburg, Germany	`http://www.informatik.uni-freiburg.de/~nebel/`
Deakin Black Knights	Deakin University, Geelong, Australia	`http://www.et.deakin.edu.au/blackknight/`
Dutch Team	4 collaborating Dutch Universities: UVA,VU,TUD,UU, Holland	`http://www.wins.uva.nl/~mielko/Soccer/Dutch-committee.html`
GMD Robots	German National Research Center for Information Technology , Sankt Augustin, Germany	`http://set.gmd.de/SET/standard/RoboCup_e.htm`
ISocRob	Instituto de Sistemas e Robotica, Lisboa, Portugual	`http://lci.isr.ist.utl.pt/projects/mrob/socrob`
Real MagiCol	Universidad del Valle (Colombia) - Université d’Evry Val d’Essonne (France)	`http://www.univ-evry.fr/infos/rencontres/realmagicol.html`
RMIT Raiders	Royal Melbourne Institute of Technology, Melbourne, Australia	`http://www.robocup.rmit.edu.au`
Stuttgart CoPS	Universität Stuttgart, Stuttgart, Germany	`http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/comros/`
The Spirit of Bolivia	Brandeis University, Los Angeles, USA	`http://www-robotics.usc.edu/~barry/ullanta/UPRsoccer.html`
Trackies	Osaka University, Osaka, Japan	`http://www.er.ams.eng.osaka-u.ac.jp/robocup/trackies`
Ulm Sparrows	Universität Ulm, Ulm, Germany	`smart.informatik.uni-ulm.de/SPARROWS`
USC/Dreamteam	University of Southern California, Marina del Rey, USA	`http://www.isi.edu/isd/dreamteam/`
Uttori United	Utsunomiya University, Tokio University, RIKEN, Tokio, Japan	`http://robot.eng.toyo.ac.jp/robolab/robocup/UTTORI/index-e.html`

Table 10. **Middle size league** teams in alphabetical order

Team Name	Institution	Homepage
All Botz	University of Auckland, Auckland, New Zealand	`http://www.tcs.auckland.ac.nz/~jacky`
Big Red	Cornell University, New York, USA	`http://www.mae.cornell.edu/Robocup/`
CIIPS Glory	University of Western Australia, Perth, Australia	`http://www.ee.uwa.edu.au/~braunl/eyebot/soccer/`
CMUnited	Carnegie Mellon University, Pittsburgh, USA	`http://www.cs.cmu.edu/~robosoccer/`
FU-Fighters	Freie Universität Berlin, Berlin, Germany	`ttp://www.inf.fu-berlin.de/lehre/WS98/RoboCup/index.html`
Luck Star / IC^3	Ngee Ann Polytechnic, Singapore	`http://www.np.edu.sg`
Lusitanos	University of Minho, Guimaraes, Portugal	`http://www.robotica.dei.uminho.pt/indexI.html`
Rogi 2 Team	University of Girona & LEA-SICA, Girona, Spain	`http://rogiteam.udg.es/rogiangles/`
TPots	Temasek Polytechnic, Singapore	`http://www.tp.ac.sg`
UQ RoboRoos	University of Queensland, Brisbane, Australia	`http://www.elec.uq.edu.au/~wyeth/socce`
Usp99	Sao Paulo University, Sao Paulo, Brazil	`http://www.pcs.usp.br/~anna`
VUB AI-lab Team	Vrije Universiteit Brussel, Belgium	`http://arti.vub.ac.be/robotic_agents.html`

Table 11. **Small size league** teams in alphabetical order

Appendix F
References

[AAAI99] American Association for Artificial Intelligence, 1999.
`http://aaai.org/`

[ActiveMedia98] Active Media Incorporated: Pioneer Mobile Robots, 1998.
`http://www.activmedia.com/robots/`

[Aparicio98] Pedro Aparicio, Rodrigo Ventura, Pedro Lima, Carlos Pinto-Ferreira: “ISocRob - Team Description”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 423-429, Paris, 1998.

[Asada96] Minoru Asada: “An Agent and an Environment: A View on “Having Bodies” - A Case Study on Behaviour Learning for Vision-Based Mobile Robot”. In *Proceedings of IROS-96 Workshop: Towards Real Autonomy*, pp. 19-24, Osaka, 1996.

[Asada97] Minoru Asada, Peter Stone, Hiroaki Kitano, Barry Werger, Yasuo Kunioshi, Alexis Drogul, Dominique Duhaut, Manuela Veloso, Hajime Asama and Sho’ji Suzuki: “The RoboCup Physical Agent Challenge: Phase I”. In *Applied Artificial Intelligence (AAI) Journal, Vol. 12, Number 2-3*, 1997.

[Bässmann98] Henning Bässmann, Jutta Kreyss: “Bildverarbeitung Ad Oculos”, Berlin, 1998 (in German).

[Bolton99] W. Bolton: “Mechatronics”, Addison Wesley Longman, 1999.

[Bräunl96/1] Thomas Bräunl: “Skriptum Robotik II - Mobile Roboter”, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, Stuttgart, 1996 (in German).

[Bräunl96/2] Thomas Bräunl: “Der Teppich-Krieger - Mobilen Mini-Roboter selber bauen und steuern”. In *c’t*, Heft 1, pp. 290 - 296, 1996 (in German).

[Bräunl97] Thomas Bräunl: “Improv and EyeBot - Real-Time Vision on-board Mobile Robots”. In *Proceedings of IEEE Mechatronics and Vision in Practice (M2VIP)*, pp. 131-135, Toowoomba, Australia, 1997.

[Bräunl98/1] Thomas Bräunl: “EyeBot - Online Documentation”, 1998.
`http://www.ee.uwa.edu.au/~braunl/eyebot/doc/`

[Bräunl98/2] Thomas Bräunl: “EyeBot - Robot Systems”, 1998.
`http://www.ee.uwa.edu.au/~braunl/eyebot/robots.html`

[Bräunl98/3] Thomas Bräunl: “CIIPS Glory - Autonomous Robot Soccer Players with Local Intelligence”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 497-502, Paris, 1998.

[Bräunl98/4] Thomas Bräunl, Birgit Graf: “Robot Soccer with Local Vision”. In *Proceedings of PRICAI-98 Workshop on RoboCup*, pp. 14-23, Singapore 1998.

[Bräunl98/5] Thomas Bräunl: “Lecture notes for IPS205 (Computer Graphics)”, The University of Western Australia, Department of Electrical & Electronic Engineering, Perth, 1998.

[Connectix98] Homepage of Connectix Corporation, 1998.
`http://www.connectix.com/`

[Faulhaber97] Dr. Fritz Faulhaber GmbH & Co. KG.
`http://www.faulhaber.de/`

[Gutmann98/1] J-S. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, T. Weigel, B. Welsch: “The CS Freiburg Team”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 451-457, Paris, 1998.

[Gutmann98/2] Jens-Steffen Gutmann, Wolfgang Hatzack, Immanuel Herrmann, Bernhard Nebel, Frank Rittinger, Augustinus Topor, Thilo Weigel, Bruno Welsch: “The CS Freiburg Robotic Soccer Team: Reliable Self-Localization, Multirobot Sensor Integration and Basic Soccer Skills.” To appear in: M. Asada (ed.): *RoboCup-98: Robot Soccer World Cup II*, Springer-Verlag, Berlin, Heidelberg, New York, 1998.

[Han98] Kwun Han and Manuela Veloso: “Reactive Visual Control of Multiple Non-Holonomic Robotic Agents”. Submitted to *IEEE International Conference on Robotics and Automation (ICRA)*, Leuven, 1998.

[Hedberg97] Sara Reese Hedberg: “Robots playing soccer? RoboCup poses a new set of challenges in intelligent distributed computing”. In *IEEE Concurrency*, October-December ‘97, pp. 13-17, 1997.

[HiTec98] Assistive Communications Technology from HITECH Group international.
`http://www.hitec.com`

[Kitano95] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda and Eiichi Osawa: “RoboCup: The Robot World Cup Initiative”. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife*, pp. 19-24, Montreal, 1995.

[Kitano97/1] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda and Eiichi Osawa: “RoboCup: The Robot World Cup Initiative”. In *Proceedings of the First International Conference on Autonomous Agents (Agent-97)*, pp. 340 - 347, Marina del Rey, 1997.

[Kitano97/2] Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda and Minoru Asada: “The RoboCup Synthetic Agent Challenge 97”. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Nagoya, 1997.

[Kitano98] Hiroaki Kitano, Minoru Asada, Itsuki Noda, Hitoshi Matsubara: “RoboCup: Robot World Cup”. In *Crossroads - the ACM’s First Electronic Publication*, Spring 1998 - 4.3, 1998.
`http://www.acm.org/crossroads/xrds4-3/xrds4-3.html`

[Klupsch98] Michael Klupsch, Thorsten Bandlow, Marc Grimme, Ignaz Kellerer, Maximilian Lückenhaus, Fabian Schwarzer, Christoph Zierl: “Agilo RoboCuppers: RoboCup Team Description”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 431-437, Paris, 1998.

[Kuth98] A. Kuth, A. Bredenfeld, H. Günther, H.U. Kobialka, B. Klaaßen, U. Licht, K.L. Paap, P.G. Plöger, H. Streich, J. Vollmer, J. Wilberg, R. Worst, T. Cristaller: “Team Description of the GMD RoboCup-Team”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 439-

449, Paris, 1998.

[Lampart98] Thomas Lampart: “Visuelle Programmierung mobiler autonomer Roboter”. Diploma Thesis No. 1581, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, Stuttgart, 1998 (in German).

[Moreno98] C. Moreno, H. Loaiza, A. Suarez, E. Gonzalez, S. Lelandais: “Real MagiCol - Complex Behaviour through simple Behaviour Oriented Commands”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 475-482, Paris, 1998.

[Motorola99] Motorola Homepage.
`http://www.mot.com/`

[Nardi98] Daniele Nardi, Giorgio Clemente, Enrico Pagello: “ART - Azzura Robot Team.” In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 467-473, Paris, 1998.

[Newton98] Newton Research Labs.
`http://www.newtonlabs.com/`

[Price97] Andrew Price, Andrew Jennings and John Kneen: “Omnidirectional Robots: New Challenges in Mobility, Vision and Tactics for Robot Soccer”, 1997.

[RFSolutions98] RF Solutions: “Jack Daniel Company commercial radio wireless equipment and technical information”, 1998.
`http://rfsolutions.com/`

[RoboCup96] The RoboCup Federation: “1996 version of the RoboCup Regulations”.
`http://www.robocup.org/regulations/45.html`

[RoboCup98/1] The RoboCup Federation: “RoboCup Official Site”, 1998.
`http://www.robocup.org`

[RoboCup98/2] The RoboCup Federation: “A Brief History of Robot World Cup”, 1998.
`http://www.robocup.v.kinotrope.co.jp/overview/23.html`

[RoboCup98/3] The RoboCup Federation: “Small-Size Robot League regulations”.
`http://www.robocup.v.kinotrope.co.jp/regulations/42.html`

[RoboCup99] The RoboCup Federation: “RoboCup F-180 League”.
`http://www.cs.cmu.edu/~trb/robocup-small-rules`

[RoboCupS98] RoboCup Singapore National Committee: “RoboCup Pacific Rim Series”.
`http://jsaic.krdl.org.sg/robocup/`

[Rowstron98/1] A. Rowstron, B. Bradshaw, D. Cosby, T. Edmonds, S. Hodges, A. Hopper, S. Lloyd, J. Wang, S. Wray: “CURF: Cambridge University Robot Football Team”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 503-510, 1998.

[Rowstron98/2] A. Rowstron, B. Bradshaw, D. Cosby, T. Edmonds, S. Hodges, A. Hopper, S. Lloyd, J. Wang, S. Wray: “CURF: Experience in Paris”, 1999.

[Sharp98] Sharp Electronics corporation.
`http://www.sharp.de`

[Shen97] Wei-Min Shen, Jafar Adibi, Rogelio Adobbati, Ali Erdem, Hadi Moradi, Behem Salemi, Sheila Tejada: “Autonomous Soccer Robots”. Presentation at the *First International Workshop on RoboCup*, Nagoya, 1997.

[Shen98] Wei-Min Shen, Jafar Adibi, Rogelio Adobbati, Bongham Cho, Ali Erdem, Hadi Moradi, Behem Salemi, Sheila Tejada: “Building Integrated Mobile Robots for Soccer Competition”. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Leuven, 1998.

[Singh87] Madan G. Singh: “System & Control Encyclopedia”, Pergamon Press, Oxford,1987.

[Sieber88] Helmut Sieber: “Mathematische Begriffe und Formeln”, Klett-Verlag, 1988 (in German).

[Storey98] Daniel Storey: “Wireless Communication for Intelligent Robotic Agents“, Honours Thesis at the University of Western Australia, Department of Electrical and Electronic Engineering, Perth, 1998.

[Uchibe96] Eiji Uchibe, Minoru Asada, Shoichi Noda, Yasutake Takahashi, Koh Hosoda: “Vision-Based Reinforcement Learning for RoboCup: Towards Real Robot Competition”. In *Proceedings of IROS-96 Workshop on RoboCup*, Osaka, 1996.

[Vishujit98] Nalwwa S. Vishujit: “A Guided Tour of Computer Vision”, Addison Wesley Publishing Company, 1998.

[VanBui98] Anh Van Bui, Jonas Gregersen, Andrew Jennings, John Kneen, Lin Padham, Paul Parisi, Michael Wirth: “RMIT Raiders”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 483-489, Paris, 1998.

[Veloso97] Manuela Veloso, Peter Stone, Kwun Han, Sorin Achim: “CMUnited: A Team of Robotic Soccer Agents Collaborating in an Adversarial Environment”. In *Proceedings of the First International Workshop on RoboCup*, Nagoya, 1997.

[Veloso98] Manuela Veloso, Peter Stone and Kwun Han: “The CMUnited-97 Robotic Soccer Team: Perception and Multiagent Control”. In *Proceedings of the Second International Conference on Autonomous Agents (Agent-98)*, Minneapolis, 1998.

[Werger98] Barry Werger, Pablo Funes, Miguel Schneider-Fontan, Randy Sargent, Carl Witty and Tim Witty: “The Spirit of Bolivia: Complex Behaviour Through Minimal Control”. In H. Kitano (ed.): *RoboCup-97: Robot Soccer World Cup I*, pp. 348-356, Springer-Verlag, Berlin, Heidelberg, New York, 1998.

[Willms96] Martin Willms, Susanne Gerl: “Folien zum Fachpraktikum Digitale Bildverarbeitung”, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, 1996 (in German).

[Wyeth98] Gordon Wyeth, Brett Browning, Ashley Tews: “UQ RoboRoos: Preliminary Results for a Robot Soccer Team”, 1998.

[Yokota98] K. Yokota, K. Ozaki, N. Watanabe, A. Matsumoto, D. Koyama, T. Ishikawa, K. Kawabata, H. Kaetsu, H. Asama: “Cooperative Team Play Based on Communication”. In *RoboCup-98: Robot Soccer World Cup II. Proceedings of the Second RoboCup Workshop*, pp. 491-496, Paris, 1998.

Appendix G
Acknowledgements

I would like to thank all the staff at the Centre for intelligent Information Systems at the University of Western Australia for their assistance throughout the year.

In particular I would like to thank Associate Professor Thomas Bräunl for his cooperation during the time I stayed in Perth.

I further want to outline the work of Richard Meager and Ivan Neubronner of the mechanical and electronic workshop at UWA who did an excellent job in re-designing and building *EyeBot vehicle II* and the new *EyeBot* platform.

In addition my thanks go to the following people who helped me pass an excellent time during my stay in Perth: Cristina Angel and friends, Astrid Reul, all members of the UWA Outdoor Club and Squash Club.

For their encouragement and support during the creation of this document I finally want to thank Sabine Dettling, Gudrun Graf, Jens Höfflinger, Hartmut Keller, and Antje Raap.

Appendix H
Declaration

I hereby declare that I composed this document all by myself and only used the referenced aids.

Birgit Graf