Simulated RoboCup
- Creating a generic API

M. Bergkvist
pt99mbe@student.bth.se

T. Olandersson
pt99tol@student.bth.se

J. Sturesson
pt99jst@student.bth.se

2003-01-16

Abstract

The field of simulated RoboCup is still in its infancy, the lack of a stable, well-documented and public API is apparent. This report describes the effort to provide such an API to the RoboCup community. The ambition have been to keep the API agent architecture independent to allow for all possible types of decision makers to be built on top of it.

Due to the inherent noise in the domain, we set out designing the positioning functionality with the use of fuzzy logic. The complexity of this approach, and its inability to keep reasonable execution times when scaling up from the field size of the legged league to the field size of the simulated league, forced us to discard this approach in favour of the simplified positioner now present in the API.

To enable users to immediately start using the API, we have created an example player called Tiro. This player uses the common behaviour model, and can easily be extended to fit a particular purpose.

Contents

Chapter 1

Introduction

The first part of the report describes the RoboCup domain to give a frame of reference to the rest of the report. A short history of RoboCup is presented, as well as the goals of RoboCup and the different domains it offers. The second part describes the practical work we have performed. In this part we describe the main parts of the API for simulated RoboCup that we have developed. The discussion includes the problems that we have encountered and the corresponding solutions that we have chosen.

The report is concluded with a discussion on future work in the area.

1.1 Educational purpose

According to the course plan [6], the course aims to provide the students with:

- A deeper knowledge and understanding of an advanced topic within the field of Computer and Information Science.
- Knowledge of current research results within the selected topic.
- Knowledge of current state-of-the practice in the selected topic.

1.2 Project goal

Our first and foremost goal was to gain experience in developing an architecture for a self-coordinating agent system. For this purpose we chose the domain of simulated RoboCup, which provides an excellent test-bed for the development of autonomous agents. Introducing the concept of competition to RoboCup, stimulates a deeper commitment, since most of us are competitive by nature.

A secondary goal was to create a generic API for the development of RoboCup agents, which we could offer to the community. Our impression is that a stable API that is publicly available is much coveted. Hence, we assume that it could benefit not only our own research, but anyone interested in creating his or her own RoboCup team, without having to start from scratch.

Chapter 2

What is RoboCup

2.1 Background

Mackworth introduced the idea of using soccer-playing robots in research. Unfortunately, the idea did not get the proper response until the idea was further developed and adapted by Kitano, Asada, and Kuniyoshi, when proposing a Japanese research program, called Robot J-League, a professional soccer league in Japan.

During the autumn of 1993, several American researchers took interest in the Robot J-League, and it thereafter changed name to the Robot World Cup Initiative or RoboCup for short. RoboCup is sometimes referred to as the RoboCup challenge or the RoboCup domain. In 1995, Kitano et al. proposed the first Robot World Cup Soccer Games and Conferences to take place in 1997.

The aim of RoboCup was to present a new standard problem for AI and robotics, somewhat jokingly described as the life of AI after Deep Blue. RoboCup differs from previous research in AI by focusing on a distributed solution instead of a centralized solution, and by challenging researchers from not only traditionally AI-related fields, but also researchers in the areas of robotics, sociology, real-time mission critical systems, etc.

To co-ordinate the efforts of all researchers, the RoboCup Federation was formed. The goal of RoboCup Federation is to promote RoboCup, for example by annually arranging the world cup tournament. Members of the RoboCup Federation are all active researchers in the field, and represent a number of universities and major companies. As the body of researchers is quite large and widespread, local committees are formed to promote RoboCup-related events in their geographical area.

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. [2]

2.2 The goals of RoboCup

The RoboCup Federation has set goals and a timetable for the research. Setting goals and a timetable are means of pushing the state-of-the-art further, in conjunction with formalized test-beds. The most important goal of RoboCup is to advance the overall technological level of society, and as a more pragmatic goal to achieve the following: By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, in compliance with the official rule of the FIFA, against the winner of the most recent World Cup. [2]

There will be several technological advancements, even if the goal of the robotic soccer team is not reached, starting with Team-Partitioned, Opaque-Transition Reinforcement Learning (TPOT-RL) which has found application in the domain of packet routing in computer networks. TPOT-RL is a distributed learning method in domains where agents have limited information about environmental state transitions. [2]

2.3 What is the SoccerServer?

Soccer Server is a system that enables autonomous agents to play a match of soccer (association football) against each other. Soccermonitor is a program that displays the virtual field from the soccerserver on the monitor (see Figure 2.1).

A match is carried out in a client/server style: A server, soccerserver, provides a virtual field and simulates all movements of a ball and players. Each client controls movements of one player. Communication between the server and each client is done via UDP/IP sockets. Therefore users can use any kind of programming systems that have UDP/IP facilities.

The client sends commands to control a player of the client and receives information from sensors of the player. In other words, a client program is a brain of the player: The client receives visual and aural sensor information from the server, and sends control-commands to the server. Each client can control only one player. So a team consists of the same number of clients as players. Communications between the clients must be done via soccerserver using say and hear protocols. [2]

2.4 Active domains

Current RoboCup activities are divided in three areas; RoboCupSoccer, RoboCupJunior and RoboCupRescue.

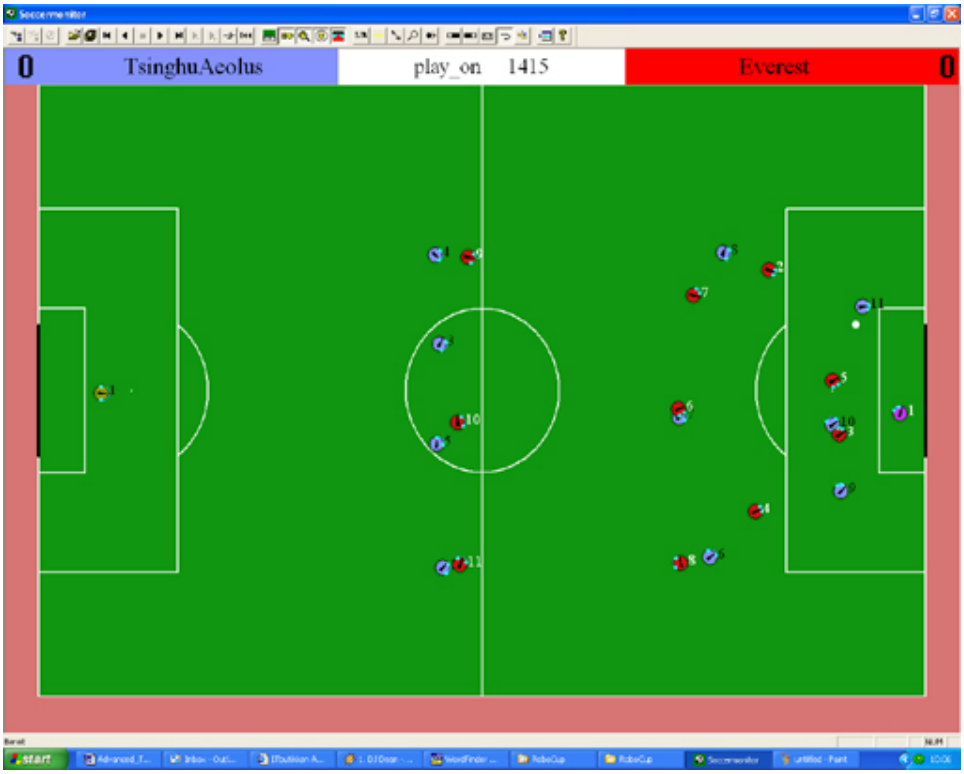

Figure 2.1: This picture is a snapshot from the RoboCup World Cup 2002, where the winning team TsinghuAeolus is about to score.

2.4.1 RoboCup Soccer

The competitive soccer has the main focus of the three areas of the RoboCup organization. The researchers use the games to gain and exchange technical information from each other. Another important purpose is that the games also serve as a great chance to educate and entertain the public. [7]

RoboCupSoccer is divided into the following leagues:

- Simulation league
- Small-size robot league
- Middle-size robot league
- Four-legged robot league
- Humanoid league

2.4.2 RoboCup Rescue

Disaster rescue is one of the most serious social issues, involving very large numbers of heterogeneous agents in a hostile environment. The intention of the RoboCupRescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work coordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, a standard simulator and decision support systems, evaluation benchmarks for rescue strategies and robotic systems that are all integrated into a comprehensive systems in future [7].

RoboCupRescue is divided into the following leagues:

- RoboCupRescue simulation league
- RoboCupRescue robot league

2.4.3 RoboCup Junior

RoboCupJunior is a project-oriented educational initiative that sponsors local, regional and international robotic events for young students. It is designed to introduce RoboCup to primary and secondary school children, as well as undergraduates who do not have the resources to get involved in the senior leagues. The focus in the Junior league is on education. [7]

2.5 Simulated league

The RoboCup simulator league is based on the RoboCup simulator, the soccerserver. All games are visualized by displaying the field of the simulator by the soccer monitor on a computer screen. The soccer server is written to support competition among multiple virtual soccer players in an uncertain multi-agent environment, with real-time demands as well as semi-structured conditions.

One of the advantages of the soccer server is the abstraction made, which relieves the researchers from having to handle robot problems such as object recognition, communications, and hardware issues, e.g., how to make a robot move. The abstraction enables researchers to focus on higher level concepts such as co-operation and learning. Since the soccer server provides a challenging environment, i.e., the intentions of the players cannot be mechanically deduced, there is a need for a referee when playing a match. The included artificial referee is only partially implemented and can detect trivial situations, e.g., when a team scores. However, there are several hard to detect situations in the soccer server, e.g., deadlocks, which brings the need for a human referee. There have been four world cups and one pre-world cup event. [2]

Chapter 3

The need for an API

Experience tells that the APIs that are available to the public are either to bound to a specific solution, or have poor perception and a weak world model. Another major drawback that has to be taken in to consideration is the low level of usability. This is because they are often badly documented and therefore it is difficult to gain a deep knowledge about them.

With a functional, well documented, stable and open source API, the RoboCup community can be broadened. Developers can concentrate on agent specific problems that the RoboCup domain provides and implement different problem solving theories. Time and effort don't have to be spent on low level issues that are common to all RoboCup agent implementations.

Chapter 4

Practical work

The type of environment in RoboCup is one of the hardest to deal with according to the definition of environments in "Artificial Intelligence, A modern approach [8].

- It is inaccessible since the field of view is limited to the view angle and it is only possible to see a part of the soccer field, i.e. the agent has to maintain an internal state of the soccer field.
- It is nondeterministic because the next state of the environment cannot be determined by its current state and the actions selected by the agent. For instance, there are 21 other players the agent can only guess what actions they will select, and it is not possible to calculate the exact trajectory of the ball.
- It is nonepisodic because the agent has to plan its actions several cycles ahead, and every action the agent does has impact on the subsequent cycle.
- It is semidynamic because the environment will not change while the agent is deliberating if the agent comes to a decision and acts within passage of the cycle time (currently 100ms). If the agent doesn't send a command to the server before the ending of the cycle, the server calculates the next state without any action from the agent.
- It is discrete in the sense that the agent know what flags and number of players he can see, and what actions the agent can do. But it is continuous in the sense that the possible perceptions of the flags and the players is neither limited nor clearly defined.

4.1 Fuzzy logic

Fuzzy logic is a superset of conventional Boolean logic, which has been extended to handle the concept of partial truths. It was introduced by Dr.

Lotfi Zadeh of UC/Berkeley in the 1960's as a means to model the uncertainty of natural language, by allowing partial set membership rather than crisp set membership or non-membership. This approach to set theory was not applied to control systems until the 70's due to insufficient small-computer capability prior to that time. Professor Zadeh reasoned that people do not require precise, numerical information input, and yet they are capable of highly adaptive control. If feedback controllers could be programmed to accept noisy, imprecise input, they would be much more effective and perhaps easier to implement. [10]

4.1.1 Description

The major difference between fuzzy logic and Boolean (standard) logic is that possible values range from $0.0 \rightarrow 1.0$ (inclusive), not just 0 and 1. For example, you could say that the fuzzy truth value (FTV) of the statement "Graham is tall" is 0.75 if Graham is 2 metres tall. To write this more formally:

$$m(TALL(Graham)) = 0.75$$

m is a membership function and is the function that would map 2 metres to an FTV of 0.75. Membership functions can be incredibly simple, or incredibly complex. For example, a relatively simple membership function could be:

$$m(TALL(x)) = \left\{ \begin{array}{ccc} 0 & ; & x < 5 \\ \frac{x-5}{2} & ; & 5 \leq x \leq 7 \\ 1 & ; & x > 7 \end{array} \right\}$$

A formal definition of a membership function can be stated as a function that maps each point of fuzzy set A to the real interval $[0.0, 1.0]$ such that as $m(A(x))$ approaches the grade of membership for x in A increases. [10]

Logic operators

In Boolean logic there are $\vee$ (union), $\wedge$ (intersection) and $\neg$ (not) operators. These operators exist in fuzzy logic too, but are defined differently:

- $A \wedge B = MAX(m(A(x)), m(B(x)))$
- $A \vee B = MIN(m(A(x)), m(B(x)))$
- $A\neg = 1 - mA(x)$

The intersection and union operators are the obvious departure from both Boolean logic and standard probability functions. To see why they are defined like this, let us look at an example. Take the statements "Graham is

very tall" and "Graham is very clever", if you were to combine these using probability, you would get:

$$m(TALL(Graham)) * m(CLEVER(Graham)) = 0.90 * 0.90 = 0.81$$

Note: This is obviously making the assumption that 'very' equates to 0.9 in both the membership functions $TALL$ and $CLEVER$. Let us also make the assumption that the range $[0.8 \rightarrow 0.9]$ equates to 'quite'. The resulting statement reads "Graham is quite smart and clever" this is obviously not correct. Using fuzzy logic:

$$MIN(m(TALL(Graham))*m(CLEVER(Graham))) = MIN(0.90, 0.90) = 0.90$$

Which yields the correct statement "Graham is very smart and clever". The result of this gets more noticeable as more variables are taken into consideration, for example 6 variables with values of 0.8 would yield 0.262 using probability, far from the fuzzy value of 0.8. [10]

Hedges

Hedges are operators that are independently created to modify the fuzzy values. Like a lot of fuzzy logic, they can be equated to English words. For example, the hedge $VERY$ could be equated to $m(A(x))2$, or $SOMEWHAT$ to $m(A(x))0.5$. [10]

Note that equating words to fuzzy values and performing operations upon them (e.g., $LOWERTHAN$) requires complicated algorithmic manipulation. It has been done though, most notably by F. Wenstop who equated words to 7-valued fuzzy vectors.

4.1.2 Domains

Fuzzy logic can be most readily applied to expert systems whose information is inherently fuzzy. Doctors, lawyers, engineers can diagnose problems a lot quicker if the expert system they use to diagnose the problem lists a few fuzzy solutions that they can use to augment their own findings.

Another area the fuzzy logic is used is handwriting recognition, e.g. in Japan, complicated Kanji strokes can be detected as they're written using fuzzy methods. Applications of fuzzy logic have also been seen in areas such as cement kiln control and financial prediction/control.

An area of more direct interest to us is robotics navigation in noisy environments. Team Sweden in RoboCup uses a landmark based navigation of Sony legged aibo robots using fuzzy logic. It is used to account for errors and imprecision's in visual recognition and the poor odometry as a result of using legged robots. [1]

4.1.3 Fuzzy logic in the simulated league

According to the RoboCup server manual the distance to a flag could be distorted with approximately +-10%. Since this would give us a possible error of 10 m on a 100 m distance we decided that our positioning should to be able to handle this uncertainty. The choice fell on a positioning model incorporating fuzzy logic, which have been used successively for navigation in real life robots.

The first couple of weeks were spent on trying to understand the model and to customize it to fit in our domain. When we realized that we were running behind on the time schedule, we decided to visualize the data from the server in order to establish if it had this high level of noise (see section 4.2.1).

With a graphical representation of the server data at hand, we immediately realized that the data were far more accurate than it said in the manual. Thus the need of a time consuming and implementation-difficult fuzzy positioning model was clearly overrated. At this point we agreed upon a simplified positioning model, that was a great deal easier to implement as well as much less time consuming. Implementing this simplified model had the benefit of a higher confidence in the stability of the positioner since the possibilities of errors were drastically reduced, compared to if we were to implement a fuzzy positioner with our novice skills in fuzzy logic.

4.2 CRaPI

CRaPI - Correct Robust and Perfect API for simulated RoboCup, is our attempt to provide an API that meets the needs described in section 3.

4.2.1 Architectural overview

An overview of the architecture of CRaPI is shown in Figure 4.1. The central unit is the Player, which communicates with the server via ServerConnection. Yaffa, to the extreme left, is shown as an example of a decision maker that utilizes CRaPI.

4.2.2 Worldmodel

A world model is an agent's perception of its environment. The world model keeps track of all objects in the environment that an agent can perceive with any of its sensors. Consequently, all that an agent knows about its environment is kept in its world model. The main tasks of the world model is therefore to constantly update its view of the position and direction of all mobile objects in its surrounding.

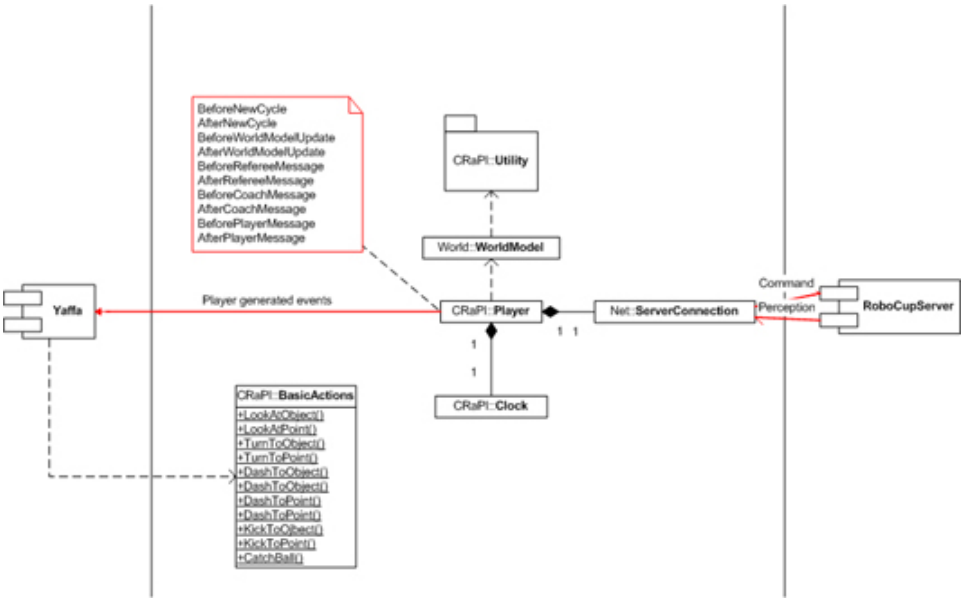

Figure 4.1: Architectural overview

Visualization tool

As we described in section 4.1.3, we developed a tool to visualize the data from the server. The main reason was to simplify the interpretation of the server data, to be able to ascertain the consistency between the player's world model and the view shown in the soccer monitor.

The visualization tool is developed for our needs only, it is not released to the public. If we feel that there exist a demand in the community of such a tool we might reconsider.

It offers the following functionality;

- Which players world model to display
- Toggle display of visible flags of the selected player
- Toggle display of the intersections where the selected player can be located
- Toggle display of the selected player
- Toggle display of the team mates of the selected player
- Toggle display of the opponents of the selected player
- Toggle display of the ball

- Display the position, body direction and face direction of the selected player

Figure 4.2: Visualization of the world model where the position of the selected player, the distance to the seen flags, the team mates and the ball is displayed.

Positioning

The task of positioning can be divided into two major sub task, i.e. calculating the position of oneself, and calculating the position of all other mobile objects. Mobile objects are all objects that have the possibility to change location on the field, e.g. players and ball.

The calculation of the own location is based on the fact that the positions of all flags are known, and that the agent receives the distance to the flags that are currently within its view cone. The seen flags are drawn as black solid dots in Figure 4.2.

The main parts of the positioning algorithm:

1. Sort flags according to distance.

2. A predefined number of flags are chosen for the calculation of intersections.

3. For each intersection, calculate which square in the field grid it is located and increase the counter for this square with one

4. Calculate the score of each square in the field grid by multiplying the counter value with the squares inverted distance from the player.

5. The center of the square with the highest score is chosen as the location of the player.

A special case occurs when no flags are seen, hence the location based on intersections cannot be calculated. The CRaPI solution to this problem is to use what is called dead reckoning, which basically is to calculate ones position based on previous position, amount of speed and direction of speed and the believed impact on previous action.

The next step is to calculate the location of all other mobile objects. This is quite straightforward since the visual sensor gives information about distance and direction to all seen mobile objects. The result can be seen in Figure 4.2, where blue solid dots represent seen teammates and the white solid dot represents the ball.

4.2.3 Sensors

The world model of a RoboCup client is updated with data received by its array of sensors. The types of sensors that a RoboCup agent can utilize are; visual sensor, physical sensor and aural sensor. The visual sensor receives data for all objects that are visible to the agent, i.e. the objects that are within the agents view cone and not too far away. The input is on the form: *(see Time ObjInfo+)*. A list of possible *ObjInfo* for the *see* message is listed in Table 4.1.

The physical sensor receives data concerning the state of the own body. The input is on the form: *(sense_body Time ObjInfo+)*. A list of possible *ObjInfo* for the *sense_body* message is listed in Table 4.2.

The aural sensor receives data of what the agents hear. The sender can either be another player, the referee or a coach. The input is on the form: *(hear Time Sender "Message")* or *(hear Time Online_Coach Coach_Language_Message)*. A description of the different data in the *say* message is shown in Table 4.3.

4.2.4 Timing

All actions received by the server at the end of the simulator cycle will be used when updating the state of the simulation. The Figure 4.3 shows an example of the agent sending commands back to the sever, but due to the

ObjInfo	Description
Distance	The distance to the object.
Direction	The direction to the object.
DistChange	The change in distance to the object.
DirChange	The change in direction to the object.
BodyDir	The body direction relative the facing direction of the agent, only if the object seen is another player.
HeadDir	The head direction relative the facing direction of the agent, only if the object seen is another player.
TeamName	The team name of the seen player, only if the object seen is another player.
UniformNumber	The uniform number of the seen player, only if the object seen is another player.

Table 4.1: Visual sensor data

delays in the network and the fact that the server takes some time before reading the received message, they are not always executed in the same simulation step as the agent sent them. Therefore the need of an internal clock to keep track of when it is time to send a command to the server is a vital part of every RoboCup agent.[4]

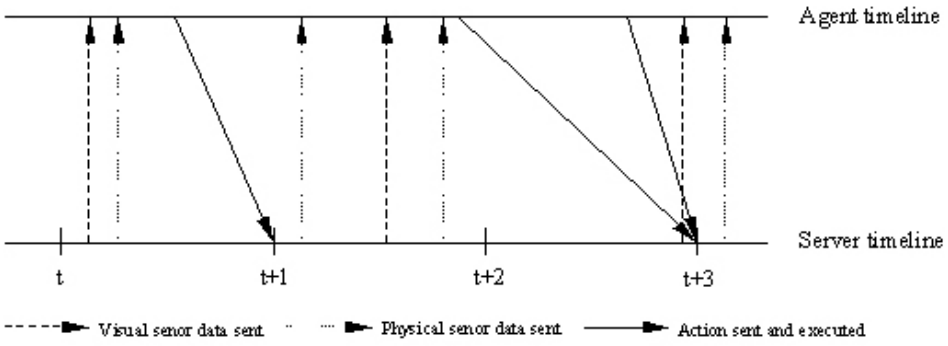

Figure 4.3: An illustration of what data the agent receives, when it is received and when commands sent to the server will be received and executed.

The event driven, internal clock in CRaPI takes care of the timing of the agent. When the client receives a sensor input from the server, CRaPI supposes that a new cycle has begun and the internal clock is synchronized with the server clock. The generated events of the internal clock are shown in Table 4.4. It isn't mandatory to receive all the events, its up to the user to decide which events to be notified about.

ObjInfo	Description
ViewQuality	The current setting of the agent's ViewQuality, affects the amount and quality of the visual data sent to the agent.
ViewWidth	The current setting of the agent's ViewWidth, affects the amount and quality of the visual data sent to the agent.
Stamina	The current stamina of the agent.
Effort	The current effort of the agent.
AmountOfSpeed	The amount of the agent's current speed vector.
HeadDirection	The relative direction of the agent's head.
DashCount	The number of dashes made by the agent so far.
KickCount	The number of kicks made by the agent so far.
SayCount	The number of says made by the agent so far.
TurnCount	The number of turns made by the agent so far.
TurnNeckCount	The number of turn necks made by the agent so far.

Table 4.2: Physical sensor data

4.2.5 Agent architecture independent

For the reasons explained in chapter 3, we aimed to make CRaPI agent architecture independent. This is solved by restricting CRaPI to only take care of the communication with the server, build the worldmodel from the perception and applying an event driven model.

The player using CRaPI is informed of changes with events, therefore any part of the player can be notified about the events that are of interest. It is possible to use CRaPI and build a new player just by registering for the event *AfterNewCycle*, but it is also possible for a more advanced player to register for all of the events in CRaPI. We have tried to publish events for any state change in CRaPI. The core events of CRaPI is listed in Table 4.5

4.3 Tiro

Tiro is a RoboCup player that has been developed to allow users to quickly start using the CRaPI framework. We developed it to provide a tutorial on how to use CRaPI. There are three layers in Tiro that abstracts the decision making; DecisionMaker, GameState, Behaviour. Tiro is parameterized and is set up by XML.

Data	Description
Sender	The sender of the message.
Direction	The relative direction to the sender. Replaces the Sender data when the sender is unknown.
Message	The actual message.
Online_Coach	A coach that can communicate directly to the players and get noise free information about movable objects.
Coach_Language_Message	A standard language for online coaches.

Table 4.3: Aural sensor data

Event	Description
TickPlayOn	Indicating how much time there is left on the current cycle.
TickPlayOff	Indicating how much time has passed since the match clock was stopped.
TickOnCycleSoonEnding	Indicating that the cycle will soon end.
TickOverTimeOnCycle	Indicating that a new cycle should have been received from the server.

Table 4.4: Events generated by the internal clock of CRaPI

4.3.1 DecisionMaker

The DecisionMaker is an abstract base class that defines the interface for decision makers. StateAnalyzer is our implementation of a DecisionMaker, which evaluates the current state of the game.

4.3.2 GameState

GameState is a class that when chosen by the StateAnalyzer evaluates the set of associated Behaviours. Examples of GameStates are AttackGameState and DefenseGameState, which have different Behaviours associated.

4.3.3 Behaviour

A Behaviour has two functions. The first function is to evaluate how beneficial it is, in the current situation, for itself to be fired. The second function is to produce a list of Commands that is to be sent to the server.

Event	Description
BeforeNewCycle	Indicating that a new cycle is coming once the clock has been synchronized.
AfterNewCycle	Indicating that a new cycle has started.
BeforeWorldModelUpdate	Indicating that the WorldModel is to be updated.
AfterWorldModelUpdate	Indicating that the WorldModel is updated.
BeforeRefereeMessage	Indicating that a RefereeMessage is to be handled.
AfterRefereeMessage	Indicating that a RefereeMessage has been handled.
BeforeCoachMessage	Indicating that a CoachMessage is to be handled
AfterCoachMessage	Indicating that a CoachMessage has been handled
BeforePlayerMessage	Indicating that a PlayerMessage is to be handled
AfterPlayerMessage	Indicating that a PlayerMessage has been handled

Table 4.5: Core events in CRaPI

4.3.4 XML

XML is used extensively in Tiro to define parameters such as; team name, server ip, server port and time out. XML is also used to define each Player in the team. For each Player, the XML defines start location, which DecisionMaker to use and which Behaviours to use for each GameState.

4.4 Open source

Both CRaPI and Tiro are open source. CRaPI is released under the Lesser GPL license and Tiro is released under the GPL license. We have used the services at SourceForge [9] for development and release, since it provides the possibility of other developers to help improve CRaPI and Tiro. It also provides us with a free homepage [3] where we e.g. provide information on current development and documentation. By this we believe that it is easier for future development of other RoboCup agent teams that wishes to test and evaluate different reasoning methods.

Chapter 5

Future work

5.1 Implement look-ahead

A possible improvement for the decision maker would be to implement look-ahead functions for all mobile objects and to calculate the impact on the environment of the agent's own actions. This forecasted information could for example be utilized when developing goal-based agents that require a number of actions to be taken to fulfil its goal.

5.2 Generic parser

All received messages from the soccerserver are currently parsed in an ad hoc manner. A major improvement would be if the factory and expert patterns were applied to detach the parsing from e.g. the world model, i.e. a generic parser that is responsible for parsing server messages. This change would improve the maintenance of CRaPI.

5.3 Refine the visualization tool

The visualization tool could for example easily be extended with functionality to display the charges of an electric field, used in the EFA - Electric Field Approach. This would be useful when tuning the placement and strengths of the charges. More information on EFA can be found in 'Using the Electric Field Approach in the RoboCup Domain' [5].

Bibliography

[1] P. Buschka, A. Saffiotti, and Z. Wasik, *Fuzzy landmark-based localization for a legged robot*, in Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Takamatsu, Japan, 2000, pp. 1205–1210. Online at http://www.aass.oru.se/~asaffio/.

[2] M. Chen, E. Foroughi, F. Heinz, Z. Huang, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wang, and X. Yin, *User Manual, RoboCup Soccer Server*, 2002.

[3] CRaPI, *http://crapi.sourceforge.net*.

[4] F. Heinz, *Robocup a system for developing robocup agents for educational use*, master's thesis, Linköpings universitet, 2000.

[5] S. Johansson and A. Saffiotti, *Using the electric field approach in the RoboCup domain*, in RoboCup 2001: Robot Soccer World Cup V, A. Birk, S. Coradeschi, and S. Tadokoro, eds., no. 2377 in LNAI, Springer-Verlag, Berlin, DE, 2002, pp. 399–404. Online at http://www.aass.oru.se/~asaffio/.

[6] PAD004, *http://idenet.bth.se/*, 4 November 2003.

[7] RoboCup, *http://www.robocup.org*.

[8] S. Russel and P. Norvig, *Artificial Intelligence - A Modern Approach*, Prentice Hall, 1995.

[9] SourceForge, *http://sourceforge.net*.

[10] K. Tomsovic and M. Chow, *Tutorial on fuzzy logic applications in power systems*, IEEE-PES Meeting in Singapore, (2000), p. 87.