

ROBOT SOCCER: GAME OR SCIENCE

Frans Groen, Matthijs Spaan, Nikos Vlassis

*University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam*

Abstract: Robot soccer is representative for the application of Multi Agent Systems in real-world dynamic situations. Robot soccer contains most of the important elements that are present in real world multi-agent applications, such as perception of dynamic environments, reactive behaviour, team coordination, communication and fusion of information, decision making in the presence of adversary objects. In this paper we will describe the different RoboCup leagues and illustrate the different scientific challenges by the approach taken in the Dutch middle size soccer team: Clockwork Orange.

Keywords: mobile robot, robot soccer, Multi Agent System.

1. INTRODUCTION

Games have always played an important role in the development of AI technologies. After Deep Blue won the match against Garry Kasparov, the question arose: What after chess? The RoboCup challenge to have in 2050 a team of humanoid robots playing a soccer match against a human team has the potential to become such a new standard problem. The RoboCup Federation [1] organizes yearly world championships in different leagues: the simulation league, the small size league, the middle size league, the legged league and recently the humanoid league. Competitions in multiple leagues offer the possibility to focus research on different aspects of this challenge.

Robot soccer is representative for the application of Multi Agent Systems in real-world dynamic situations. There are different robots that have to work together toward a common goal, the domain is continuous and dynamic, there are opponents whose behaviour will not be fully predictable and because of the competitive element of the game it is necessary to act sensible and fast. This together with the fact that the game offers a constricted controllable domain and is entertaining and challenging makes it an ideal test-bed for multi-agent collaborating robotics research. To keep the game as close as possible to the real game of soccer, most rules used in human soccer apply. There are some important differences mostly due to the robots physical

limitations, such as the ability to catch the ball, or kick sideways, or even just move deftly across the playing field.

Applications of Real-World Multi-Agents are in service robots, transportation, exploration of hazardous environments and public safety to mention a few. These agents will be “intelligent on-line embedded systems” which are able to operate in human habited dynamic environments. Local intelligence and mutual communication make systems robust to erroneous perception or malfunctioning of one or more robots. To realize such a challenge the integration of many technologies is needed such as mechatronics, control theory, computer vision, self-learning systems and cooperative autonomous systems.

What should the robots be able to do? *Perception of the real world* to find and track the ball and the players from color image sequences. *Communication and fusion of information* obtained at different moments by different robots to create a dynamic model of the world. *Reactive behavior*, to create the basic actions of the robot. *Strategy development* to decide what actions are optimal in a given state. Scientific challenges are in the localization and prediction of the position of the robot itself and the dynamic objects around it. In the sensor-based behaviors (dribble) and how to maintain shared world models. In the creation of adaptive team behavior and opponent modeling.

In the year 1997 the Robot World Cup Initiative was started as an attempt to improve AI and robotics research by providing a standard problem in which a wide range of technologies can be integrated and examined. The first RoboCup championship games were held in 1997 in Nagoya, Japan. The second international robot soccer games ran side-by-side with the real soccer world championship games in Paris in 1998. The third RoboCup championships were held in Stockholm 1999. The Stockholm games are co-scheduled the biennial International Joint Conference on Artificial Intelligence. Since then this yearly event was held in Melbourne in 2000, Seattle in 2001 and Fukuoka in 2002. Beside these world championships there are several regional events such as the German Open in Paderborn.

The number of teams attending in the different leagues of RoboCup has increased dramatically since the first World Cup (from 40 teams in 1997 to about 100 teams from about 20 different countries in 2001). Over the years the games also started getting more attention. The number of spectators has increased from 5000 in Nagayo 1997 to 20,000 in Seattle 2001.

2. ROBOCUP LEAGUES

There are different leagues within the RoboCup competitions, each with their own challenges. New developments are the *Humanoid league* en the *Rescue league*. The roboCup Federation sees as it next challenge Robot Rescue: The Search and Rescue for large Scale Disasters, e.g. searching for survivors. It started as simulation project but involves now also a real environment developed by NIST.

2.1 Small-Size Robot League

The small-size league is played on a table tennis-sized field. Each team consists of five small robots of about 15 centimeters. The Playing time is 2 x 10 minutes with a 20 minutes break for technical service in between. A camera above the field is used to get a complete view of the game, which is send to the computers of the teams on the side of the field. From this image using the color coding of the ball and the different robots a world model is constructed. Using this world model the actions of the different robots are determined and send to the robots by way of wireless communication. Since the world model is complete and quite accurate research here focuses on robot coordination, team behavior and real time control. The games in this league are typically very fast.

2.2 Middle size league

The robots in the Middle size league are ca 50 centimeters (1.5 feet) in diameter. They compete on a field of about 10 meters long and 5 meters wide. The

objects are colour coded. The ball is orange, the goals are yellow and dark blue the robots are black with a light blue or pink hat, the field is green and the lines are white The main difference with the small-size league is that there is no global vision of the field. Visual information is received from a camera on board of each robot. The Robots can communicate with each other by way of wireless communication. The number of players is maximum 4 per team. The playing time is 2 x 10 minutes with a 20 minutes break for technical service. To create cooperative team behaviour robots have to know where they are on the field and self-localization is a key issue. Currently often omni-directional vision systems are used that give a 360 degree view of the field, which makes the self-localization easier. This is the league in which the Dutch robot soccer team Clockwork Orange, which will be described in the next section, participates.

2.3 Sony Legged robot League.

On a field, slightly larger than the small-size league, teams of four Sony AIBO's (the well-known robotic toy dog) compete. These robots walk on four legs, and are thus the first 'step' toward a league of biped humanoid robots. Since every team uses the same robots, the only difference between the teams is in the software.



(a)



(b)

Fig. 1 Small size (a) and legged (b) league

2.4 Simulator League.

This looks like a standard computer game, but the essential difference is that each player is its own simulated robot, driven by its own program, each able to decide their own next move. Because this simulation frees the researchers from such inherent physical limitations of 3-dimensional robots as mechanical parts, wheel control and other functions, these screen players are able to perform on a far more advanced level. They are able to interact and cooperate, even changing play strategies from defense to offence, or from forward to back.

The number of players is 11 per team. The playing time is 2 x 5 minutes. The players in these teams can be either homogeneous or heterogeneous and have various properties such as dexterity and pace. Since there is only a limited amount of uncertainty in the information the teams have about the game, compared to the other leagues it is relatively easy to generate a complete and accurate world model, although communication is limited. This enables the teams to concentrate on cooperative team behaviour and tactics.

The entire match is played on screen, If you like you can run log files of the simulation competitions on your own PC. You can find the log-files on the simulation site of RoboCup

2.5 RoboCup Jr.

To attract students from high schools for technical studies, there is also an event for high schools: RoboCup Jr. These robot soccer players can be build

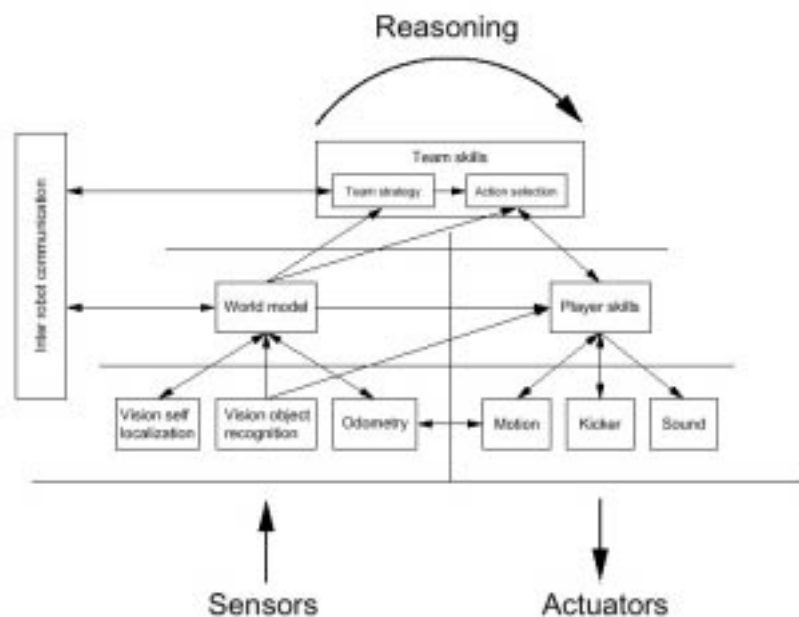
with for example LEGO Mindstorms. A special infra light-emitting ball is used to play the competitions of RoboCup Jr. From one goal to the other the gray value of the field changes from black to white, so the direction of the field can be obtained with simple sensing.

3. THE DUTCH MIDDLE SIZE TEAM

The Dutch team is collaboration between the Universities of Delft, Amsterdam and Utrecht. The robot players of the team in the competitions have been NOMAD Scout robots: a mobile platform from Nomadic technologies. The robots have a differential drive and a pneumatic kick device as actuators. The sensors used for RoboCup are the wheel encoders for odometry sensing and a camera for localization and the detection of the other objects in the field.

A low-level motor board controls the original hardware from Nomadic technologies, while a high-level computer is connected to the custom hardware, like the camera, wireless Ethernet and the kick device.

In figure 2b the functional software architecture is sketched. At the lowest level we have the interface between the hardware and the software. The virtual sensors at this level are the odometry sensor, the vision self-localization and the vision object recognition. These virtual sensors send their information to the world module, which fuses these measurements with previous information and the information from the other robots, to come up with the best local estimate of its position and that of the other robots and of the ball. The position of the ball



(a) (b)
Fig. 2 Nomad Scouts (a) and their functional software architecture (b)

is also directly send the player skill module to enable a fast feedback loop for dribbling with the ball. At the highest level the team strategy is defined based upon information from the world model and the most appropriate action is selected, given the current situation of the world and the team strategy. The basic actions are realized by the payer skills module, with forms the interface with the actuators and motion control. For communication between the robots a BreezeCom wireless Ethernet system is used.

3.1 Virtual sensors and actuators

The virtual sensors and actuators usually do not communicate directly with the hardware but use device drivers from the operating system as intermediates. The *Odometry module* is a virtual sensor, which keeps track of the motion of the robot. It gets it data from the motor hoard, which forms the interface to the wheel encoders. Do to slip and collisions the error in the odometry increases with the traveled distance. An University of Michigan Benchmark test as described in [2] has been run to estimate the systematic error. For a typical Nomad this error turned out to be 20cm after driving a 3m square (1.7%).

As knowing your own position is crucial for constructing a global world model we also use the camera for self-localization. The *Vision self-localization* module [3] uses the lines on the field and the goals. The self-localization mechanism involves a global and a local method. The global method first splits the screen into multiple regions of interest and finds straight lines in each of these. These are matched to the world model giving an estimate of the position. Because of the symmetry of the field multiple candidates are found. We then use Multiple Hypothesis Tracking to follow all the candidates over time, updating them for our own movement.

The local method is used to verify these candidates and to correct for changes. We check all the candidates, verifying their heading and distance to the goals and the overall result given to us by the local method. The loop is repeated until one candidate remains, which is used to update our position.

This self-localization mechanism requires the robot to move around otherwise candidates cannot be discarded. For this reason the Team skills module is notified when we have lost our position. To give an idea of the performance of the self-localization: during a total period of more than two hours our position was known 44% of the time.

Vision object recognition [4] extracts objects like the ball, the goals and robots from the camera images. Due to among others specular reflections the

saturation of the colors can differ for the same object. To separate the objects based on their color, objects are identified by a circular arc in the color UV space. The value of Y forms the intensity. The position of the objects in the image plane is projected on the 2d ground floor on which the robot is driving. Both the color classification and projection of the image plane to the ground floor requires a calibration procedure for the robots. Auto calibration would make the system easier to use and more robust.

Size and position of these objects are estimated and this information is passed on to the World model and to the Player skills.

Driving is controlled by the Motion module, which communicates motor commands to the low-level processor of the Nomad. The Kicker module controls the pneumatic kick mechanism used for shooting at goal.

3.2 Player skills

The *Player skills* module takes care for the basic actions of the robot. When executing these actions it has to take care for other behaviors, which have a higher priority. These are the collision avoidance behavior and the license to kill behavior. The collision avoidance behavior makes sure that a robot does not collide with other robots or walls. If a robot has the ball and sees a large portion of the enemy goal the license to kill behavior makes it shoot at it. These two reactive behaviors get their information directly from the Vision object recognition system. The basic action of the player skill module is

- *Turn (φ)*, rotate the robot around its axis until it reaches the desired heading, relative to the world. Also turns relative to the robot can be specified.
- *TurnToObject(object)*, turn to face the *object*.
- *Shoot()*, kick the ball straight ahead at maximum force.
- *TurnShoot (φ)*, kick the ball at specified angle. This is accomplished by driving while turning followed by shot
- *Goto(x, y, φ , v)*, move to specified position while avoiding obstacles. Either desired heading or desired speed at end point could be requested. Forward and backward motion is supported, but since the robot can only detect obstacles in front of it driving backwards is not recommended during normal operation.
- *GotoObject(object)*, if *object* is *ball* move toward the ball and try to control it. If *object* is one of the goals move toward it until the robot is one meter away from them (you usually don't want it to actually drive across the goal line).
- *Seek(object)*, keep turning until the robot sees the requested object.
- *Dribble(x, y, v)*, carefully drive to the requested position trying to keep control over the ball. If the robot loses the ball don't immediately declare the

action a failure but try to regain it. Dribbling with the ball is difficult due to the shape restrictions and the limit of five motor commands per second.

- *DribbleToObject (object)*, dribble toward one of the goals. The Player skills module keeps a memory of the relative position it last saw the goals, in order to be able to find them again.

3.3 Distributed dynamic world model

In principle, any team strategy requires knowledge of the state (position, orientation, and velocity) of each robot, both for teammates and opponents, as well as the state (position and velocity) of the ball. Each of the robots in our team locally maintains a world model consisting of the robot's own position and the positions of the other objects in the game.

Modeling the system as a Markovian state space model requires two stochastic models, a state transition model in the form that relates previous states with new ones according to the kinematics of the robots and the ball, and an observation model that assigns likelihood values to sensor profiles observed at certain states. With the above definitions, a generic state prediction scheme can be obtained is given by an iterative scheme based on the Bayes' rule, which is described in [5]. This is a general probabilistic framework which has been successfully used in the context of single robot localization [6], where the main assumption is the existence of a single dynamic object (the robot) and a static environment. Known prediction mechanisms like the Kalman filter or the recent particle filter [7] are particular instances of the iterative update.

The central aspect that distinguishes a multi-robot system from a single robot one is the fact that each robot can observe other teammate robots and then communicate this information around [8,9]. This

way, each robot can have an estimate of its own state based on a self-localization mechanism like the one described above, but it can also have additional, fused estimates of its own state based on the communicated information from its teammates. Similarly, the state of the opponent robots and the ball can be collaboratively estimated by all team mate robots.

In our robot team, the kinematics model is given by odometry and the observations are from the vision localization. In the odometry module the translation and/or rotation of the robot is calculated from the rotation of the wheels, where the uncertainty due to slip increases with time. For the observation model, the above discussed vision localization is used for computing the likelihood of an observation based on a matching procedure of observed line segments to a model of the soccer field. Details about this method can be found in [2].

A second characteristic of multi-robot systems is that the communicated information may arrive with delays due to limited network resources (latencies, low bandwidth, etc.), and the above inference cannot always be carried out instantaneously. For this reason, the robots must maintain a history of state posteriors and carry out a sort of *lag* filtering every time observations from other teammate robots arrive that correspond to previous time steps. To ensure the consistency of the communicated messages, a timestamp mechanism is used.

Since decision making must be based on the most recent information, the state posterior is continuously updated, while action commands, observations, and state distributions, are stored in memory. If a message from another teammate robot arrives at time t that corresponds to a lagged observation at time $t - \tau$ then the filtered posterior is propagated repeatedly up to time t [5].



Fig. 3 Finite state machine to determine the next team behavior.

3.4 Team skills

The main benefit of a distributed dynamic world model is that team coordination is greatly simplified. Sharing the same world model enables team robots to reason not only about their own actions but they can also predict the next action a teammate robot will take. Communication on the team coordination level is used to improve robustness.

Our team uses a *global team strategy* from which each robot derives its individual action. The team skills module determines the current team strategy using a simple finite state machine which takes as input the question "Who has the ball?". We have defined three team strategies, which are shown in figure 3 together with the transitions between them.

There are about half a dozen different *individual behaviours* (or roles) available, all of which have an attacking, defending, or intercepting purpose [10]. Utility functions based on potential fields [11] are used to distribute the roles among the team members. This technique is similar to the ones used by other participants in the middle Size league [12, 13]. Each robot calculates its own scores for the individual behaviours useful at the moment, communicates them and then calculates the scores of all its teammates. Incoming scores from one of your teammates overwrite the score you calculated for him. This introduces a level of redundancy, which improves the team performance. When everybody knows the scores for the individual role of every team member the roles can be easily distributed.

A Markov decision process is used to search through the state space of a robot's future states. First we generate a number of possible actions and their parameters with respect to the current situation and individual behavior. For each of these actions the probability of success is calculated. The impact on the world of an action and the expected concurrent actions of teammates and enemy players is evaluated. Modeling teammate actions however is relatively easily compared to modeling your opponent with reasonable accuracy. Now the action with the maximum expected utility can be chosen and the process can be repeated to look further ahead.

Another key problem of the method described above is the choice of evaluation criteria for rating the future world an action produces. Each of them has its own weight associated with it, since some things are more important in soccer than others.

First criterion is a check whether the ball is in one of the goals. This criterion obviously has the highest weight of all, scoring a goal is the goal of the game and avoiding a goal for your opponents is also very important.

Next criterion is the concept of ball possession. This score is at its maximum when our team has the ball and at its minimum when one of the opponents has it. Third criterion is a role evaluation. For this criterion the potential fields from the role distribution process are used. Each individual behavior has its own attractors and repellers. A defender for instance likes to be in the area just in front of his own goal area. The last criterion concerns obstacle avoidance and strategic positioning. This criterion also uses potential fields, but this time only static obstacles, moving objects and your own heading are being considered.

3.5 Improving robustness

In a real world domain like the RoboCup middle size league one has to take care to design a robust system. Common problems are temporary communication failures of vision based self-localization which is not able to provide position updates for some time. This is the reason our team skills module has a relative mode next to the absolute mode described above. In this relative mode no actions are generated which involve absolute coordinates (like "dribble to position x, y " or "shoot at angle a "). Instead more reactive actions of the underlying player skills module are being considered (like "chase ball" or "seek enemy goal"). The system goes into relative mode when it notices the error estimate of its own position has risen above a certain threshold or when the vision system notices its own position cannot be correct. The team skills module switches back to absolute mode when the vision system has reclaimed its position.

When a robot initiates a "chase ball" action it broadcasts to all team members an estimate of time it thinks it needs to reach the ball. When it wants to chase the ball itself it checks whether or not it will be the first to arrive there. If it is not going to be first and the other robot who is chasing the ball is only a few seconds away the first robot chooses another action. Using this mechanism we want to prevent two robots trying to chase the ball at the same time, even in relative mode.

4. CONCLUSIONS AND FUTURE DEVELOPMENTS

In this paper we have shown that besides the fun of the competitions Robot Soccer also involves real scientific challenges, which are representative for the application of Real-World Multi Agent Systems in practical dynamic situations. We have illustrated these challenges by the approach taken by the Dutch RoboCup team: clockwork orange. Future Real-World Multi-Agents will be "intelligent on-line embedded systems" which are able to operate in human habited dynamic environments. In particular

local intelligence en mutual communication make systems very robust for erroneous perception or malfunctioning of one or more robots. Robot Soccer competitions form a platform to compare different approaches to these problems and to evaluate them in practice.

REFERENCES

- [1] WWW Site RoboCup: www.robocup.org
- [2] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 5, Oct.1996.
- [3] F. de Jong, J. Caarls, R. Bartelds, and P. P. Jonker. A two-tiered approach to self-localization. In *Proc. RoboCup 2001 Int. Symposium*, Seattle, USA, Aug. 2001.
- [4] P. P. Jonker, J. Caarls, and W. Boldiove. Fast and accurate robot vision for vision-based motion. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup 2000: Robot Soccer World Cup II*, volume 2019 of *lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001
- [5] F.C.A. Groen, J. Roodhart, M. Spaan, R. Donkervoort, and N. Vlassis. **A distributed world model for robot soccer that supports the development of team skills**. In Ben Kröse, Maarten de Rijke, Guus Schreiber, and Maarten van Someren, editors, *Proc. 13th Belgian-Dutch Conf. on Artificial Intelligence, BNAIC'01*, pages 389-396, Amsterdam, The Netherlands, October 2001.
- [6] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):9-109, 2000.
- [7] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.
- [8] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robot*"; 8(3):325-344, 2000.
- [9] S. I. Roumeliotis. *Robust Mobile Robot Localization: From Single-Robot Uncertainties to Multi-Robot Interdependencies*. PhD thesis, University of Southern California, May 2000.
- [10] M. Spaan, N. Vlassis, F.C.A. Groen, High-level coordination of agents based on multiagent Markov decision processes with roles. *Proceedings Workshop WS7 on Cooperative Robotics*, Editor A.Saffiotti, October 2002
- [11] O. Khatib. Realtime obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90-98, 1986.
- [12] C. Castelpietra, L. Iocchi, M. Piaggio, A. Scalzo, and A. Sgorbissa. Communication and coordination among heterogeneous mid-size players. In *Proceedings of the 4th International Workshop on RoboCup*, pages 14~158, Melbourne, Australia, Aug. 2000.
- [13] T. Weigel, W. Auerbach, M. Dietl, B. Dümmler, J. Gutmann, K. Marko, K. Millier, B. Nebel, B. Szerbakowski, and M. Thiel. CS Freiburg: Doing the right thing in a group. To appear in P. Stone, G. Kraetzschmar, T. Balch, *RoboCup 2000*, Springer-Verlag, 2001.