DECISION MAKING FOR MIROSOT SOCCER PLAYING ROBOTS

Uwe Egly[1], **Gregor Novak**[2], **Daniel Weber**[1]

[1]Institute of Information Systems,
Vienna University of Technology, Vienna, Austria
uwe@kr.tuwien.ac.at
daniel.weber@clara.tuwien.ac.at

[2]Institute of Computer Technology,
Vienna University of Technology, Vienna, Austria
novak@ict.tuwien.ac.at

Abstract A key property of autonomous systems like mobile robots is their ability to make decisions by their own without a master or supervisor. In this paper, we describe a rule-based fuzzy decision making mechanism for a system, very similar to the MiroSOT robots. Unlike these MiroSOT robots, which are usually controlled by a master computer, the proposed decision making mechanism is integrated into an extended version of a MiroSOT robot. This allows the robot to make decisions and to act autonomously.
In many cases, the decisions are based on local sensor data of the robot instead of global data obtained, e.g., from a global (camera) picture of the scene via the master computer.

Keywords: Robot Soccer, Reasoning, Fuzzy, Decision Making

1. Introduction

A key property of autonomous systems like mobile robots is their ability to make decisions by their own without a master or supervisor. In this paper[1], we propose a decision making mechanism for autonomous robots based on fuzzy rules. Since our work is based on an extension of a MiroSOT[2] robot, we first describe in Fig. 1 the vision-based FIRA[3] environment [1] based on a master computer system.
In this environment, two robot teams play soccer against each other. The size of the robots is limited to a cube with an edge length of 75mm. A master computer control all the robots of a team. Above the playground, each team is allowed to mount a camera, which transmits pictures to the master computer. In order to simplify the position and movement detection of entities on the playground, there are some constraints:

- The playground is black.
- The (golf) ball is orange.
- Each robot is marked on its top with at least one color.

In the master computer, a vision system detects the position, the orientation and the movement of object on the playground. This global information is the basis for decision making which is performed by the master. A possible result of the decision making process is a desired movement each robot of the team should perform. With a radio communication, these desired movement is sent (as a trajectory) to each of the robots. They execute the orders and follow the trajectory.
Although we describe our concepts in the context of robot soccer, it is not our intention to build up a typical MiroSOT system with an external and centralized decision making unit. The idea is to develop a universal applicable reasoning engine for several autonomous robot systems, like Tinyphoon[4] or Nano [2].

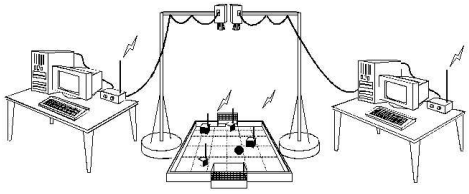

Fig. 1: Overview of the vision-based FIRA system.

2. Problem Analysis and Related Work

Playing robot soccer is a demanding application. The robots have to act extremely fast and the whole reasoning and decision making process must be finished after a few milliseconds (in other words, the system must meet real-time requirements). The demanding time constraints are one reason for the centralized robot team organization with a master computer.
Besides these timing problems, decision making in such systems is often based on more or less inaccurate and noisy sensor data. Such unreliable data are a problem in the approach with a master, but the problems are even bigger for autonomous systems. The reason for the increasing difficulties in the latter systems is the replacement of the "global view" by a "local view" with probably less information.
The robots' global goals and behaviors have to be described in a way which allows for flexible changes and updates.

[1]This paper was written within the Center of Excellence for Autonomous Systems of the Vienna University of Technology (CEAS).

[2]Micro Robot Soccer Tournament

[3]http://www.fira.net/games/mirosot.html

[4]Official homepage of Tinyphoon *http://www.tinyphoon.com*

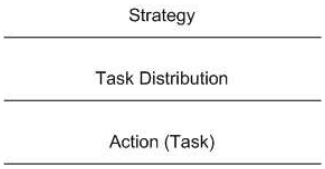

Fig. 2: The three layered architecture for decision making.

Moreover, the rules of the game also have to be described. The physical constraints together with the rules restrict the allowed or possible behavior of the robot.
Another main problem is the definition of the behavior for soccer playing including opponent modeling and tactics. MiroSOT systems usually use inflexible hard coded if-then rule sets which prevent easy updates or changes. The usage of vision based systems for position detection arises problems of noisy data which makes usual systems ineffective but hybrid control solutions exist [3] for the robot soccer domain. In a hybrid or fully autonomous control structure, the planning activities of a single robot has to be coordinated within the team [4].
In the literature, there exist several approaches for decision making, often combined with path planning. A famous representative of this approach is the *Evolutionary Artificial Potential Field* (EAPF) method. A few suggestions for the MiroSOT domain exist [5]. Basically, this is just a solution for path planing amongst moving obstacles. Decision making can be done by specifying some special potential field functions. Some implementations use genetic algorithms to optimize parameters [6] for these functions in advance.

3. Decision Making

The responsibility of the decision making is to select actions in order to achieve the global team goal which is divided into subgoals for each robot. The global goal is, e.g., to win a soccer game. Decision making generates single tasks which are computed by considering a set of rules (specifying, e.g., the behavior) as well as world information. This world information is obtained by fusion of the information delivered by several sensors. The rules are set at the start up phase of the robot. Tasks consist of simple subtasks, small enough to be solved easily by the robot. In case of soccer robots, an example for a simple subtask is "*go behind ball*", i.e., the robot is nearby the ball and the ball is between the robot and the opponent's goal. Such a subtask can be solved by moving the robot to a target position which satisfies the requirements.
On the basis of this scenario, the robot decision making unit determines the appropriate behavior to reach the target position. The algorithm uses a fuzzy logic based method combined with a state machine system to determine the best action out of a given set of actions. In a team related problem-solving strategy, some measures have to be taken into account in order to avoid two robots taking the same course of action. In non-autonomous systems with a central control, such problems do not occur.
Main design principles of the decision making system were to keep it simple and to allow its use in fully autonomous co-

```
- <Action ref="ShootAtGoal">
    <Weight ref="Forward" value="1.0" />
    <Weight ref="Defender" value="0.3" />
    <Weight ref="Goalkeeper" value="0.05" />
  - <RuleSet>
    - <And>
        <Value term="FreeToBall" />
        <Value term="NearestToBall" />
        <Value term="BehindBall" />
        <Value term="FacingBall" />
        <Value set="DistanceToBall" term="Near" />
      - <Not>
          <Value term="OpponentNearBall" />
        </Not>
      </And>
    </RuleSet>
  </Action>
```

Fig. 3: The *ShootAtGoal action* in an XML representation.

operating mobile robots. The system consists of three layers depicted in Fig. 2. The top layer *Strategy* decides the main behavior or strategy (offensive, defensive) for the team, in order to reach the global goal. Role assignment is done in the *Task Distribution* layer. Based on the chosen strategy, fuzzy values and estimation on teammates' behavior, a role is assigned to each member (avoiding concurrent role assignments, e.g., there is only one goalkeeper). The final layer, the basic *Action* layer, depends on the robot's role. Classic actions established in robot soccer are used in this implementation. To ensure the generality of these actions, they can be further divided into base-actions (representing the simple moves above a single goto-Position command). A mechanism to avoid the toggling of decisions because of noisy senor data is provided by using fuzzy-based evaluations on every layer. The system works like a state machine, where the last selected state influences the next state selection as long as it is valid. If, for some cycles, this state has not been re-approved, the next computed state is entered. The state transition is also based on fuzzy logic. Every item on every layer of the hierarchical decision making approach can be seen as states of the machine.
In common multi robot systems, the adaptation of the behavior of the robots is very complex, if the working environment or the goal changes significantly. If the behavior is hard-coded, e.g., as if-then rules, then changes are cumbersome. We propose a more flexible approach based on a "description language" based on XML, where environmental and behavior information can be described in colloquial terms. The environment of a robot can contain dimensions of rooms and obstacles and many more parameters. Behavioral information specifies the actions of the robot in a specific situation. The *ShootAtGoal* action, specified in XML form, is used as an example in Fig. 3
In this *ShootAtGoal* action, there are three different weight fields, namely one for a forward player, one for a defender, and one for a goal keeper. The rule consists of six conjunctively connected preconditions; five are positive and one is negative. The action can be performed if the conjunction of the preconditions is satisfied.
Such rules are combined within the hierarchical approach of decision making described above. Whenever rules are evaluated, they influence the rules of the next lower level by changing their weight.

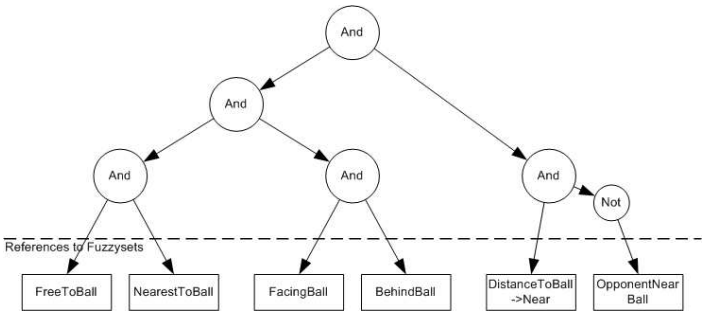

Fig. 4: Rules Logictree

The lowest levels comply with the basic actions, a robot can execute. On every level, it is possible to specify detailed information for determining values of parameters required for this level (e.g., if, in some level, it is required to select a target position, rules for choosing the appropriate positions are provided within this level). As we have already seen in the example rule above, it is possible to build up rules with logical operators (*and*, *or*, *not*) at any desired level of complexity. The values referenced in the precondition of the rule are given or calculated from sensor data. However, not the absolute values are used, but the values are "categorized" or "fuzzified". For instance, the attribute *DistanceToBall* is a distance which can be very near, near, middle, far and very far. The mapping of, e.g., the value of a range sensor to these categories is application dependent and can be specified in a separate XML file. Moreover, there is an additional XML schema file which contains all the attributes with their names and allowed categories. Such a schema file allows for consistency checks.

Rule evaluation on every layer is done by traversing logic trees (see Fig. 4 for a tree representing the rule in Fig. 3). Such a tree is a representation of the logical expression which forms the precondition of the rule. The logic trees are build when the corresponding XML files are parsed at system start up. The items in the precondition are connected by the logical connectives *and*, *or* and *not*.

At the beginning of every cycle, the current sensor and environment variables are fuzzified. In order to avoid the time-consuming complete evaluation of all rules every time in every layer, special "activation rules" are used to focus on relevant information. For instance, if the current strategy is *defense*, then all rules handling other strategies are ignored in the current cycle.

4. Simulation Software

In order to test the functionality and the quality of the decision making component, a simulation application has been implemented. A game situation with three robots per team is depicted in Fig. 5. The main goal of the simulation software is to allow for testing the decisin making module independently from the real hardware platform. The simulator is written in C# and imports the robots reasoning library (which itself is written in C++). A simple physical model of rigid bodies with collision detection and collision response is included. Properties of every simulation entity

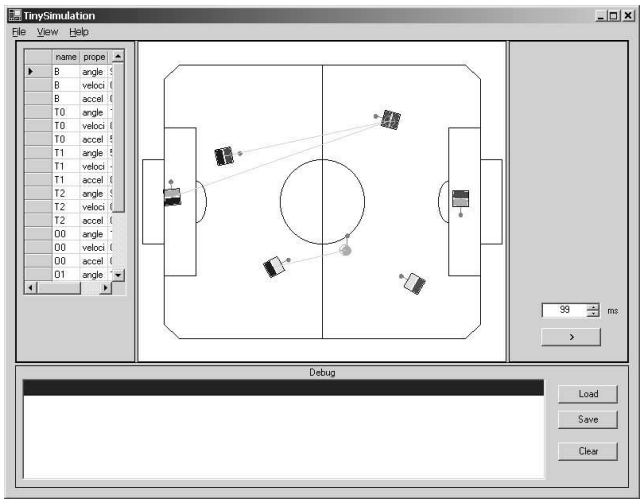

Fig. 5: A snapshot from the simulation program.

can be defined to adapt the physical behavior to represent the reality (mass, velocities, dimensions etc.). In order to make changes of the behavior and the environment easy, all the parameters are represented in XML.

The application focuses on simulating the temporal behavior of the robots control cycle. Therefore, the simulator's control loop manages two time scales; one time scale for the physical model and one time scale for the graphical display. The result of the simulation is independent from the computer system time as well as from the system performance. On fast computer systems, the simulation runs in real time, wehereas on less powerful systems, a slowddown is possible. For instance, if we want to simulate ten seconds of the game, a fast computer performs the whole simulation just in time, whereas the slower system might use 20 seconds. Although the elapsed time may differ for different computers, the number of iterations of the control loop is the same on every system.

In the current version of the simulation environment, opponents cannot be simulated yet. It is possible, however, to define actions of the opponent robost (like a path which a specific opponent robot has to follow) in order to check the reaction of the own robots. These reactions are calculated by the decision making module.

In the robot soccer domain, the sensor data mostly consists of positions computed with an image recognition module. Therefore, the position data is often noisy. In order to enable more realistic results, the simulation environment contains a mechanism to attach noise to object positions. The complete simulation session can be saved including all positions, time stamps and robot properties. A main reason for implementing this application was the need to test the decision making and to analyze the reason why a specific decision has been taken in a given situation.

The interface of the simulation allows the control of the robot positions and messages from other programs (e.g., a MiroSOT-based vision system). Further improvements will allow communication between the robots belonging to the team, the management of multiple XML decision files and the attachment of a reasoning library to the opponent robots.

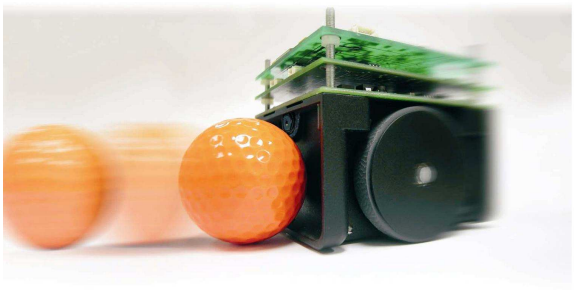

Fig. 6: Tinyphoon in action.

5. Further Research

The reasoning engine is designed to be implemented on the Tinyphoon robot. Tinyphoon2003 (depicted in Fig. 6) is the further development of Roby-Go [8], which was first released in spring 2000. Contrary to Tinyphoon, Roby-Go was developed basically as a pure soccer playing robot for the category MiroSOT. Tinyphoon represents a completely autonomous robot with integrated digital camera. Since it is derived from Roby-Go, it is still a two-wheeled differentially driven robot with the dimensions of a cube with an edge length of 75mm. The first version of Tinyphoon was finished and introduced in September 2003. It consists of a motion unit containing the motor driver, a micro controller, digital encoders for the measurement of the wheels' speed, acceleration sensors and a yaw rate sensor. Furthermore, it consists of a mono vision system unit [9] equipped with a CMOS camera and a DSP (Digital Signal Processor). Both units are connected via CAN (Controller Area Network) bus.
Integrating the proposed reasoning engine requires updating of the Tinyphoon, because of lack of computing resources in the current version. The next version of Tinyphoon will get, beside a pivoted stereo visioning unit [10], a reasoning unit equipped with a 32bit processor.
Besides the improvement of the hardware, it is also planned to improve the handling of sensor data. Since these data are noisy, it is required that the sensor submits, besides the actual acquired measurement value, a quality factor for this value. It is most common that sensors measure more accurate in some ranges, and less accurate in some others. The quality factor should be available for the sensor and has to be transmitted with the measurement value. The handling of quality factors in rule systems and their propagation during rule applications has to be improved. A possible method in the literature is based on *certainty factors*, which are used to handle inaccurate information in knowledge-based systems.

6. References

[1] G. Novak. Multi Agent Systems - Robot Soccer. Doctoral dissertation, Vienna University of Technology, Vienna, Austria, 2002.

[2] H. Tappeiner. Nano - Entwicklung eines kleinen sechsbeinigen Roboters nach biologischem Vorbild. Master's thesis, Vienna University of Technology, Austria, March 2004.

[3] H.-S. Shim, M.-J. Jung, H.-S. Kim, J.-H. Kim and P. Vadakkepat. A hybrid control structure for vision based soccer robot system. In *Int. J. Intelligent Automation and Soft Computing, 6*, pages 89–101, 2000.

[4] R. Alami, F. Ingrand and S. Qutub. A scheme for coordinating multi-robot planning activities and plans execution. In *13th European Conference on Artificial Intelligence*, pages 617–621, 1998.

[5] P. Vadakkepat, T. H. Lee and L. Xin. Application of evolutionary artificial potential field in robot soccer system. In *Joint 9th IFSA World Congress and 20th NAFIS International Conference*, pages 2781–2785, 2001.

[6] P. Vadakkepat, K. C. Tan and W. Ming-Liang. Evolutionary artificial potential fields and their application in real time robot path planning.

[7] L. A. Zadeh. Fuzzy sets. In *Inform. and Contr. vol. 8*, 1965.

[8] G. Novak. Roby-go, a prototype for several mirosot soccer playing robots. In *IEEE International Conference on Computational Cybernetics (ICCC04)*, pages 207–212, Vienna, Austria, August 29 - September 1 2004.

[9] S. Mahlknecht, R. Oberhammer and G. Novak. A real-time image recognition system for tiny autonomous mobile robots. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS04)*, pages 324–330, Toronto, Canada, May 25-28 2004. IEEE Computer Society.

[10] G. Novak, A. Bais and S. Mahlknecht. Simple stereo vision system for real-time object recognition for an autonomous mobile robot. In *IEEE International Conference on Computational Cybernetics (ICCC04)*, pages 213–216, Vienna, Austria, August 29 - September 1 2004.