A Behavior Architecture for Autonomous Mobile Robots Based on Potential Fields*

Tim Laue and Thomas Röfer

Bremer Institut für Sichere Systeme, Technologie-Zentrum Informatik, FB 3,
Universität Bremen, Postfach 330 440, 28334 Bremen, Germany.
E-Mail: {timlaue,roefer}@tzi.de

Abstract. This paper describes a behavior-based architecture which integrates existing potential field approaches concerning motion planning as well as the evaluation and selection of actions into a single architecture. This combination allows, together with the concept of competing behaviors, the specification of more complex behaviors than the usual approach which is focusing on behavior superposition and is mostly dependent on additional external mechanisms. The architecture and all methods presented in this paper have been implemented and applied to different robots.

1 Introduction

Artificial potential fields, developed by [1] and also in detail described by [2, 3], are a quite popular approach in robot motion planning because of their capability to act in continuous domains in real-time. By assigning repulsive force fields to obstacles and an attractive force field to the desired destination, a robot can follow a collision-free path via the computation of a motion vector from the superposed force fields. Especially in the RoboCup domain, there also exist several applications of potential functions for the purposes of situation evaluation and action selection [4–6].

Most approaches consider potential fields only as a tool being embedded in another architecture, e. g. a planner [4, 7]. In these cases, the potential field implementations are limited to special tasks such as obstacle avoidance or the computation of an appropriate passing position. Due to some limitations, which are discussed in Sect. 2.1, the standard approach is not able to produce a behavior of an adequate complexity for several tasks, e. g. robot soccer.

The approach presented here combines existing approaches in a behavior-based architecture [3] by realizing single competing behaviors as potential fields. The architecture has generic interfaces allowing its application on different platforms for a variety of tasks. The process of behavior specification is realized via a description language based on XML.

* The Deutsche Forschungsgemeinschaft supports this work through the priority program “Cooperating teams of mobile robots in dynamic environments”.

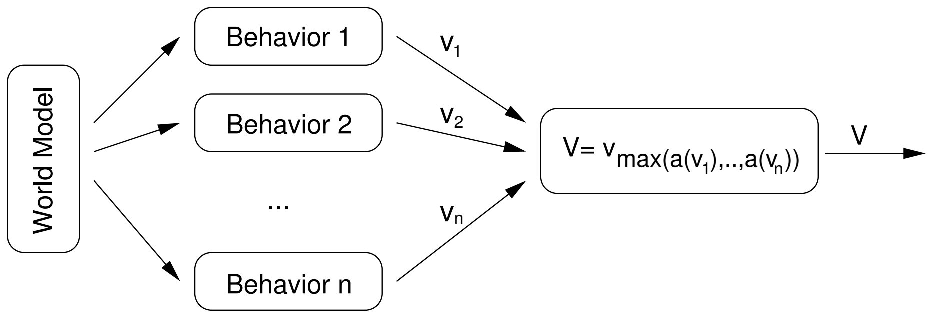


Fig. 1. The basic scheme of behavior selection. Only the behavior the output v of which has the highest activation value $a(v)$ will be executed.

This paper, which is based on the works of [8], is organized as follows: Section 2 gives an overview of the structure of the architecture, the mechanisms for behavior selection and the way of behavior specification, Section 3 shows the possibilities of modeling the environment, the Sections 4 and 5 describe the used approaches and some extensions for motion planning and action evaluation. Previous applications of the architecture are presented in Section 6 and the paper ends with the conclusion in Section 7.

2 Architecture

The architecture described in this section represents a framework for modeling the environment and a set of behaviors. It is also responsible for the task of behavior selection.

2.1 Competition Instead of Exclusive Superposition

As above-mentioned, potential fields are based on the superposition of force fields. Being a quite smart technique for obstacle avoidance, this approach fails when accomplishing more complex tasks including more than one possible goal position. An obvious example is the positioning of a goalkeeper: The usage of attractive force fields for its standard defense position as well as for a near ball to be cleared would lead to a partial erasement of the fields causing an unwanted behavior. This problem could be solved by using an external entity which selects the most appropriate goal, but this proceeding would affect the claim of a stand-alone architecture. Therefore, different tasks have to be splitted into different competing behaviors. This applies also to tasks based on action evaluation, especially since they use a different computation scheme, as explained in Sect. 5.

The approach of action selection by [9, 3], as shown in Fig. 1, has been considered as being most suitable for this architecture. A number of independent behaviors without any fixed hierarchy as in [10] compete for execution by the

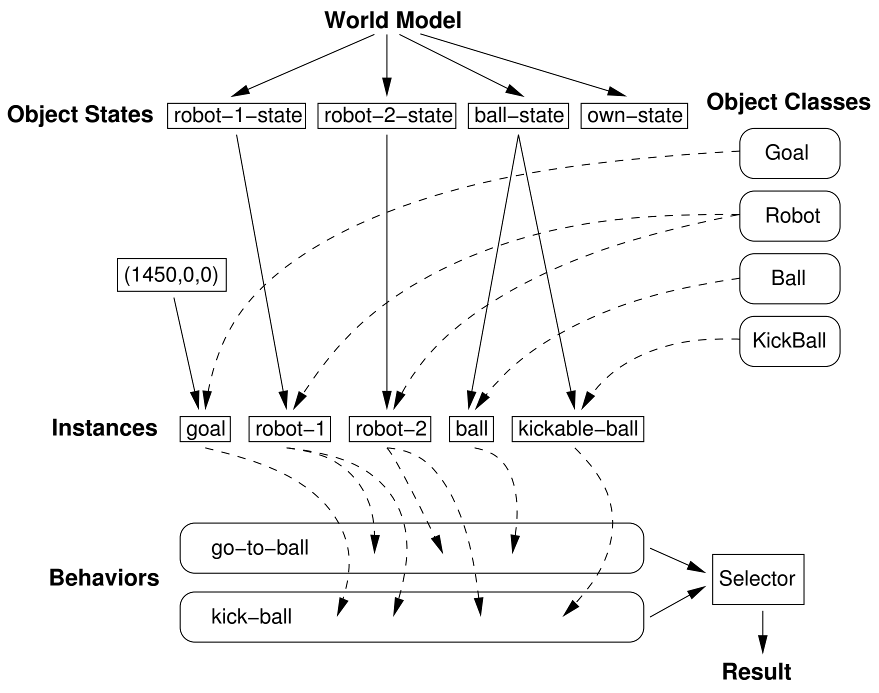


Fig. 2. A simplified example of a behavior for playing soccer, showing the elements of the architecture and their interdependencies.

respective computation of activation values which represent the current appropriateness.

As potential fields are rather based on functions and vectors than on symbolic representations [11], there are several facile ways to gain activation values from single behaviors, e. g. the value of a potential function at a certain position or the length of a computed motion vector. Constant activation values for standard behaviors are also possible.

To refine the behavior selection scheme, various mechanisms could be used. Among the blocking and keeping of behaviors under certain circumstances, behaviors can be combined with others to realize small hierarchies. For instance, this allows the usage of a number of evaluation behaviors differentiating situations (e. g. *defense* or *midfield play* in robot soccer) respectively combined with appropriate motion behaviors.

2.2 The Design of the Architecture

As to be seen in Fig. 2, the architecture consists of several interdependent elements:

Object States An object state contains information about an object in the environment. The appropriate data has to be converted from the robot's sensor layer, as described in Sect. 2.3.

Object Classes An object class is used as a template for object instances. Due to its complex configuration, it is described in a separate section (3).

Object Instances Obtaining all necessary parameters and data from classes and object states, instances may be assigned to one or more behaviors. In the case of a world model containing absolute positions, their state can also be static.

Behaviors In this architecture a behavior equals a potential field for motion planning respectively an action evaluation via potential functions. These methods are described in more detail in the Sections 4 and 5

2.3 Embedding the Architecture in Robot Systems

To use the architecture within a special robot system, two components have to be programmed to connect the generic interfaces with both the robot's sensor layer and the execution layer. Concerning the sensor component, it does not matter whether the robot system uses localization techniques to compute a world model using absolute positions or just reacts on sensor measurements, as long as an appropriate mapping of the data structures may be found. The second component has to be an interpreter which translates the abstract motion parameters or activates chosen actions, e. g. a kick or a ball catching motion, both by setting the corresponding motor parameters.

2.4 Using XML for Behavior Specification

As afore mentioned, the architecture consists of several elements. Specifying a behavior is therefore equivalent to specifying all entities and their interdependencies. Such modeling tasks may be managed in a better way by using external configuration files than by programming all needed elements in a programming language such as e. g. C++. Thus, a format for behavior specification has been described in XML. Figure 3 shows extracts from an example behavior. The reasons to use XML instead of defining a new grammar from scratch are its structured format as well as the big variety and quality of existing editing, validation, and processing tools. Many XML editors, for example, are able to check the validity of a behavior specification at run time or even assist the user during the selection of elements and their attributes. Actually, a behavior specification has to be transformed into an intermediate code which may easily be interpreted, since on many embedded computing platforms, XML parsers are not available due to resource and portability constraints.

3 Modeling of the Environment

Since the methods for motion planning and action selection, which will be explained in the following sections, are not directly configurable, the whole behavior depends on the model of the environment, in particular on the parameters a)

```
<object name="Opponent-Robot" type="repulsive">
    <asymptotic-function range="200"
                         at-zero="1000"
                         const-interval="1"/>
    <point-field/>
    <circle radius="90"/>
</object>
```

b)

```
<motionfield name="go-to-ball">
   <return-const value="-1"/>
   <include name="ball"/>
   <include name="own-penalty-area"/>
   <include-group name="all-robots"/>
</motionfield>
```

Fig. 3. Extracts from an XML behavior specification: a) An object class describing the attributes of an opponent robot. b) A motion behavior for moving towards a ball.

of the object classes. The architecture offers various options allowing a detailed description. An object class O may be considered as the following tuple:

$$O = (f_O, G_O, F_O) \tag{1}$$

With f_O being a potential function, which may be chosen among several standard functions [2] and a *social function* introduced by [12]. The range of the field as well as its gradient depend directly on the parameters which have to be assigned to f_O. A geometric primitive G_O is used to approximate an object's shape. The kind of field F_O determines the shape of the region influenced by O, which may, for example, be a circumfluent area around G_O or a tangential field around the position of the instance.

As shown in Fig. 4, an object instance excites in its environment both a potential field and a *charge* [4] based on its potential function. They may be computed separately in the following way: Let P be an arbitrary position in the environment of an object instance and $\boldsymbol{v}_O\left(P, S, G_O, F_O\right)$ a function computing a vector from P to an instance I of the class O given its geometry and kind of field as well as the object state S assigned to the instance, the value $\varphi_I\left(P\right)$ of the potential function is to be computed as follows:

$$\varphi_I\left(P\right) = f(|\boldsymbol{v}_I\left(P, S, G_O, F_O\right)|) \tag{2}$$

Analogous, the field vector $\boldsymbol{v}_I\left(P\right)$ of the potential field is:

$$\boldsymbol{v}_I\left(P\right) = f'(|\boldsymbol{v}_I\left(P, S, G_O, F_O\right)|)\frac{\boldsymbol{v}_I\left(P, S, G_O, F_O\right)}{|\boldsymbol{v}_I\left(P, S, G_O, F_O\right)|} \tag{3}$$

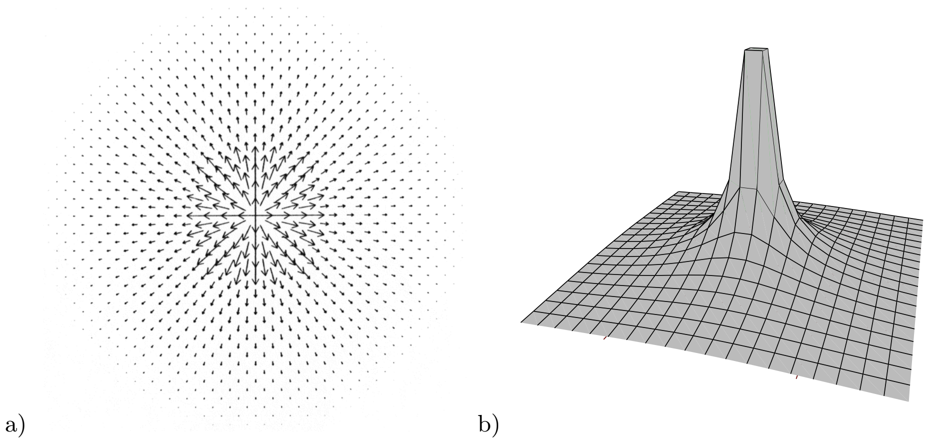


Fig. 4. In the environment of an object instance both a) a potential field and b) the value of the potential function may be computed.

Due to this duality, each object instance may be assigned to and used by motion behaviors as well as for the evaluation of actions.

4 Motion behaviors

All motion behaviors are mainly based on the standard motion planning approach by [1]. Some of the main extensions have been the integration of relative motions which allow the robot to behave in spatial relations to other objects, e. g. to organize in multi-robot formations, and the implementation of a path planner to avoid local minima.

4.1 The General Procedure of Motion Planning

Following the standard approach as described in [1–3], a vector $\boldsymbol{v}$ can be computed by the superposition of the force vectors $\boldsymbol{v}_I$ of all n object instances assigned to a behavior:

$$\boldsymbol{v} = \sum_{I=1}^{n} \boldsymbol{v_i}\,(R) \tag{4}$$

With R being the current position of the robot, $\boldsymbol{v}$ can be used to determine the robot's direction of motion, rotation and speed. Due to the equal interface of all objects, this method does not have to distinguish between attractive and repulsive objects.

4.2 Relative Motions

Assigning force fields to single objects of the environment allows the avoidance of obstacles and the approach to desired goal positions. Nevertheless, moving to

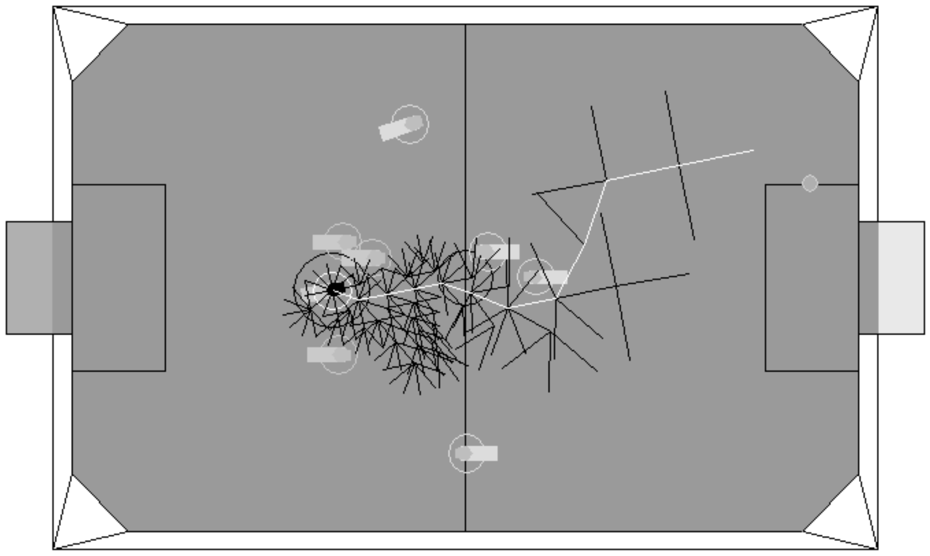


Fig. 5. Planning a path to the ball on a Sony Four-legged Robot League field. The dark lines show the generated search tree. The chosen path is drawn bright.

more complex spatial configurations, e. g. positioning between the ball and the penalty area or lining up with several robots is not possible directly. An extension of the *Motor-Schema* approach [13] to dynamically form multi-robot formations has been developed by [14]. A quite similar technique has been used in this architecture, but it is not limited to a set of special formations. Relative motions are realized via special objects which may be assigned to behaviors. Such an object consists of a set of references to object instances and a spatial relation, e. g. *between* or *relative-angle*. These are used to dynamically compute a geometric representation of the desired destination region, which will subsequently be used as the object's G_O. Via an additional assignment of a potential function, such objects may be integrated into the process of motion planning in a transparent way among all other object instances.

4.3 Dealing with Local Minima

Local minima are an inherent problem of potential fields [15] which has, of course, to be discussed when using this approach for motion planning since an optimal standard solution does not exist. In the past, several heuristics have been proposed [3], but they cannot guarantee to be effective. Due to the increasing computing resources even in embedded systems, the usage of path planners becomes practicable in real-time applications [16].

This approach uses the A* algorithm [17] for path planning. To discretize the continuous environment, a search tree with a dynamic number and size of

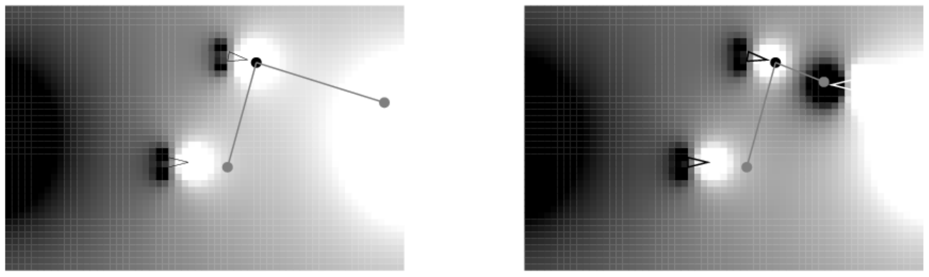


Fig. 6. Two examples of action evaluation from [4]. In both cases a kick to the side as well as a forward kick of a ball are evaluated. Bright regions represent a better evaluation.

branches, similar to [16] is built up (see Fig. 5). The potential functions of the objects in the environment are used to determine the path costs.

The path planner is integrated in such a way that it may be used by every motion behavior. There also exist mechanisms to detect whether the robot enters or leaves a local minimum and thus to use the path planner only on demand.

5 Behaviors for Action Evaluation

In this architecture, actions are considered to be indivisible entities which have to be executed by the robot after their selection, e. g. the activation of a kick motion. It is also possible that an action evaluation behavior is combined with a motion behavior to determine the appropriateness of its execution.

There exist several approaches using potential functions to evaluate certain situations or the results of planned actions. Most of them rasterize the environment into cells of a fixed size [5, 7] and compute the value of each cell to determine the most appropriate position. A quite different approach is the *Electric Field Approach* [4] which is computationally much less expensive. By computing the anticipated world state after an action, only relevant positions have to be evaluated, as shown in Fig. 6. Due to its efficient computation and the direct mapping of possible actions, a similar approach has been integrated into the architecture.

5.1 The Procedure of Action Evaluation

Analogous to the computation of a field vector, the value $\varphi(P)$ (for which [4] use the term *Charge*) may be determined at an arbitrary position P, being the sum of the potential functions of all object instances assigned to the behavior:

$$\varphi(P) = \sum_{I=1}^{n} \varphi_I(P) \qquad (5)$$

To use this method to evaluate a certain action which changes the environment, e. g. kicking a ball, this action has to be mapped to a geometric transformation

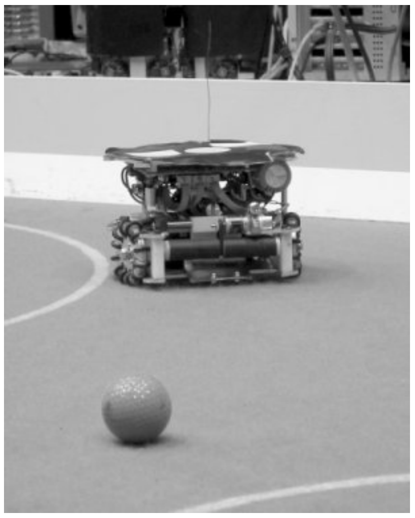
a) b) 

Fig. 7. The behavior architecture has been used in RoboCup competitions as well on a) Sony ERS-210A robots as to control b) Small-size robots.

in order to describe the motion of the manipulated object. A set off different transformations, inter alia including rotation, translation, and tracing the potential field gradient, has been implemented, together with mechanisms to check for collisions and practicability of the action. External mechanisms as planners are not needed. To describe more complex actions, e. g. turning with a ball and subsequently kicking to the goal, sequences of actions may also be specified.

Having computed an anticipated future state, the value φ may also be determined along with a value φ_d representing the change of the environment which will be caused by an execution of the action. Both values may be used as activation values of the behavior.

6 Applications

Up to now, the architecture has been applied to two different platforms, both being RoboCup teams of the Universität Bremen, to be seen in Fig.7.

The main test and development platform have been the robots of the *Bremen Byters*, which are a part of the *GermanTeam* [18], a team which competes in the Sony Four-legged Robot League. To specify the environment of that domain, about 40 object instances based on 15 different classes have been used. Each player has a different role and therefore a different set of behaviors. For playing soccer, about 10–15 behaviors have been needed, e. g. *Go to Ball*, *Go to Defense Position* or *Kick Ball Forward*. Due to the large number of degrees of freedom, these robots are able to perform many different motions, resulting in a variety of kicks. After measuring the effect of each kick on the position of the ball, they could directly be specified as behaviors and therefore be evaluated and selected by the architecture. Also sequences of actions have been used, allowing a quite forward-looking play. Except for the handling of special game states, for which all members of the German Team use XABSL [19], the complete behavior could be realized via the architecture presented in this paper.

The second platform has been the control program of B-Smart [20], running on an external PC and computing the behaviors for a team of omni-directional driven small-size robots, which are able to move with a speed of up to two meters per second. Due to the generic nature of the architecture and the similarity of the domains, large parts of the Bremen Byters' behaviors and their specification of the environment were copied leaving only several changes of parameters to be made, e. g. the dimensions of the field and the ranges of the potential functions. Through that, the portability of this approach of behavior modeling has been shown. Nevertheless, the direct interchange of unchanged behaviors is not a primary goal. Especially considering robot soccer competitions, optimizations for a single system are often more important.

7 Conclusion and Future Works

This paper presents a behavior-based architecture for autonomous mobile robots, integrating several different approaches for motion planning and action evaluation into a single general framework by dividing different tasks into competing behaviors. This approach turned out to be able to specify an overall behavior sufficiently for handling complex tasks in the robot soccer domain. The architecture's abstract design using external behavior specifications in XML files has also appeared to be well manageable.

In the future, the authors intend to port the architecture to other platforms to test and extend the capabilities of this approach. There exist several features already implemented but not adequately tested, e. g. the integration of object instances based on a probabilistic world model. In addition, the behavior selection process is currently extended to deal with a hierarchy of sets of competing behaviors similar to [21, 22], allowing the specification of even more complex overall behaviors.

At the RoboCup 2004, the architecture will be used again by B-Smart and will also be integrated in the behavior control of the German Team.

Acknowledgements

The authors would like to thank the members of the GermanTeam as well as the members of B-Smart for providing the basis of their research.

References

1. Khatib, O.: Real-time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research **5** (1986) 90–98

2. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers, Boston, USA (1991)
3. Arkin, R.C.: Behavior-Based Robotics. MIT Press, Cambridge, Massachusetts, USA (1998)
4. Johannson, S.J., Saffiotti, A.: Using the Electric Field Approach in the RoboCup Domain. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: RoboCup 2001: Robot Soccer World Cup V. Volume 2377 of Lecture Notes in Artificial Intelligence., Springer (2002)
5. Meyer, J., Adolph, R.: Decision-making and Tactical Behavior with Potential Fields. In Kaminka, G.A., Lima, P., Rojas, R., eds.: RoboCup 2002: Robot Soccer World Cup VI. Volume 2752 of Lecture Notes in Artificial Intelligence., Springer (2003)
6. Weigel, T., Gutmann, J.S., Dietl, M., Kleiner, A., Nebel, B.: CS-Freiburg: Coordinating Robots for Successful Soccer Playing. IEEE Transactions on Robotics and Automation **18** (2002) 685–699
7. Ball, D., Wyeth, G.: Multi-Robot Control in Highly Dynamic, Competitive Environments. In Browning, B., Polani, D., Bonarini, A., Yoshida, K., eds.: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
8. Laue, T.: Eine Verhaltenssteuerung für autonome mobile Roboter auf der Basis von Potentialfeldern. Diploma thesis, Universität Bremen (2004)
9. Maes, P.: How To Do the Right Thing. Connection Science Journal **1** (1989) 291–323
10. Brooks, R.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation **2** (1986) 14–23
11. Brooks, R.: Intelligence without representation. Artificial Intelligence Journal **47** (1991) 139–159
12. Reif, J.H., Wang, H.: Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots. In Goldberg, K., Halperin, D., Latombe, J.C., Wilson, R., eds.: The Algorithmic Foundations of Robotics. A. K. Peters, Boston, MA (1995) 331 – 345
13. Arkin, R.C.: Motor Schema-Based Mobile Robot Navigation. The International Journal of Robotics Research **8** (1989) 92–112
14. Balch, T., Arkin, R.C.: Behavior-based Formation Control for Multi-robot Teams. IEEE Transactions on Robotics and Automation **14** (1999) 926–939
15. Koren, Y., Borenstein, J.: Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, California, USA (1991) 1398–1404
16. Behnke, S.: Local Multiresolution Path Planning. In Browning, B., Polani, D., Bonarini, A., Yoshida, K., eds.: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
17. Hart, P., Nilsson, N.J., Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths in Graphs. IEEE Transactions on Systems Science and Cybernetics **SSC-4** (1968) 100–107
18. Röfer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Jüngel, M., Laue, T., Lötzsch, M., Nistico, W., Spranger, M.: GermanTeam 2004. The German RoboCup National Team. In: 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.

19. Lötzsch, M., Bach, J., Burkhard, H.D., Jüngel, M.: Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL. In Browning, B., Polani, D., Bonarini, A., Yoshida, K., eds.: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
20. Kurlbaum, J., Laue, T., Lück, B., Mohrmann, B., Poloczek, M., Reinecke, D., Riemenschneider, T., Röfer, T., Simon, H., Visser, U.: Bremen Small Multi-Agent Robot Team (B-Smart) Team Description for RoboCup 2004. In: 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
21. Behnke, S., Rojas, R.: A hierarchy of reactive behaviors handles complexity. In: Proceedings of Balancing Reactivity and Social Deliberation in Multi-Agent Systems, a Workshop at ECAI 2000, the 14th European Conference on Artificial Intelligence, Berlin (2000)
22. Jäger, H., Christaller, T.: Dual Dynamics: Designing Behavior Systems for Autonomous Robots. In Fujimura, S., Sugisaka, M., eds.: Proceedings of the International Symposium on Artificial Life and Robotics (AROB '97), Beppu, Japan (1997) 76–79