

# The Rise of Potential Fields in Real Time Strategy Bots

**Johan Hagelbäck and Stefan J. Johansson**

Department of Software and Systems Engineering  
Blekinge Institute of Technology  
Box 520, SE-372 25, Ronneby, Sweden  
email: [sja@bth.se](mailto:sja@bth.se), [jhg@bth.se](mailto:jhg@bth.se)

## Abstract

Bots for Real Time Strategy (RTS) games are challenging to implement. A bot controls a number of units that may have to navigate in a partially unknown environment, while at the same time search for enemies and coordinate attacks to fight them down. Potential fields is a technique originating from the area of robotics where it is used in controlling the navigation of robots in dynamic environments. We show that the use of potential fields for implementing a bot for a real time strategy game gives us a very competitive, configurable, and non-conventional solution.

## Keywords

Agents:Swarm Intelligence and Emergent Behavior, Multi-disciplinary Topics and Applications:Computer Games

## Introduction

A *Real-time Strategy* (RTS) game is a game in which the players use resource gathering, base building, technological development and unit control in order to defeat their opponents, typically in some kind of war setting. The RTS game is not turn-based in contrast to board games such as Risk and Diplomacy. Instead, all decisions by all players have to be made in real-time. Generally the player has a top-down perspective on the battlefield although some 3D RTS games allow different camera angles. The real-time aspect makes the RTS genre suitable for multiplayer games since it allows players to interact with the game independently of each other and does not let them wait for someone else to finish a turn.

In 1985 Oussama Khatib introduced a new concept while he was looking for a real-time obstacle avoidance approach for manipulators and mobile robots. The technique which he called *Artificial Potential Fields* moves a manipulator in a field of forces. The position to be reached is an attractive pole for the end effector (e.g. a robot) and obstacles are repulsive surfaces for the manipulator parts (Khatib 1986). Later on Arkin (Arkin 1987) updated the knowledge by creating another technique using superposition of spatial vector fields in order to generate behaviours in his so called motor schema concept.

Many studies concerning potential fields are related to spatial navigation and obstacle avoidance, see e.g. (Borenstein & Koren 1991; Massari, Giardini, & Bernelli-Zazzera 2004). The technique is really helpful for the avoidance of simple obstacles even though they are numerous. Combined with an autonomous navigation approach, the result is even better, being able to surpass highly complicated obstacles (Borenstein & Koren 1989).

Lately some other interesting applications for potential fields have been presented. The use of potential fields in architectures of multi agent systems is giving quite good results defining the way of how the agents interact. Howard et al. developed a mobile sensor network deployment using potential fields (Howard, Matarić, & Sukhatme 2002), and potential fields have been used in robot soccer (Johansson & Saffiotti 2002; Röfer *et al.* 2004). Thureau et al. (Thureau, Bauckhage, & Sagerer 2004b) has developed a game bot which learns reactive behaviours (or potential fields) for actions in the First-Person Shooter (FPS) game Quake II through imitation.

First we describe the domain followed by a description of our basic MAPF player. That solution is refined stepwise in a number of ways and for each and one of them we present the improvement shown in the results of the experiments. We then discuss the solution and conclude and show some directions of future work. We have previously reported on the details of our methodology, and made a comparison of the computational costs of the bots, thus we refer to that study for these results (Hagelbäck & Johansson 2008).

## ORTS

Open Real Time Strategy (ORTS) (Buro 2007) is a real-time strategy game engine developed as a tool for researchers within artificial intelligence (AI) in general and game AI in particular. ORTS uses a client-server architecture with a game server and players connected as clients. Each time-frame clients receive a data structure from the server containing the current game state. Clients can then issue commands for their units. Commands such as move unit  $A$  to  $(x, y)$  or attack opponent unit  $X$  with unit  $A$ . All client commands are executed in random order by the server.

Users can define different type of games in scripts where units, structures and their interactions are described. All type of games from resource gathering to full real time strat-

egy (RTS) games are supported. We focus here on one type of two-player game, *Tankbattle*, which was one of the 2007 ORTS competitions (Buro 2007). In Tankbattle each player has 50 tanks and five bases. The goal is to destroy the bases of the opponent. Tanks are heavy units with long fire range and devastating firepower but a long cool-down period, i.e. the time after an attack before the unit is ready to attack again. Bases can take a lot of damage before they are destroyed, but they have no defence mechanism of their own so it may be important to defend own bases with tanks. The map in a tankbattle game has randomly generated terrain with passable lowland and impassable cliffs.

The game contains a number of neutral units (sheep). These are small indestructible units moving randomly around the map making pathfinding and collision detection more complex.

### The Tankbattle competition of 2007

For comparison, the results from our original bot against the four top teams were reconstructed through running the matches again (see Table 1). To get a more detailed comparison than the win/lose ratio used in the tournament we introduce a game score. This score does not take wins or losses into consideration, instead it counts units and bases left after a game. The score for a game is calculated as:

$$score = 5(ownBasesLeft - oppBasesLeft) + ownUnitsLeft - oppUnitsLeft \quad (1)$$

### Opponent descriptions

The team *NUS* uses finite state machines and influence maps in high-order planning on group level. The units in a group spread out on a line and surround the opponent units at *Maximum Shooting Distance* (MSD). Units use the cool-down period to keep out of MSD. Pathfinding and a flocking algorithm are used to avoid collisions.

*UBC* gathers units in squads of 10 tanks. Squads can be merged with other squads or split into two during the game. Pathfinding is combined with force fields to avoid obstacles and a bit-mask for collision avoidance. Units spread out at MSD when attacking. Weaker squads are assigned to weak spots or corners of the opponent unit cluster. If an own base is attacked, it may decide to try to defend the base.

*WarsawB* uses pathfinding with an additional dynamic graph for moving objects. The units use repelling force field collision avoidance. Units are gathered in one large squad. When the squad attacks, its units spread out on a line at MSD and attack the weakest opponent unit in range.

*Uofa06* Unfortunately, we have no description of how this bot works, more than that it was the winner of the 2006 year ORTS competition. Since we failed in getting the 2007 version of the UofA bot to run without stability problems under the latest update of the ORTS environment, we omitted it from our experiments.

## MAPF in ORTS, V.1

We have implemented an ORTS client for playing Tankbattle based on Multi-agent Potential Fields (MAPF) following the

Team	Win %	Wins/games	Avg units	Avg bases	Avg score
NUS	0%	(0/100)	0.01	0.00	-46.99
WarsawB	0%	(0/100)	1.05	0.01	-42.56
UBC	24%	(24/100)	4.66	0.92	-17.41
Uofa.06	32%	(32/100)	4.20	1.45	-16.34
Average	14%	(14/100)	2.48	0.60	-30.83

Table 1: Replication of the results of our bot in the ORTS tournament 2007 using the latest version of the ORTS server.

Figure 1: Part of the map during a tankbattle game. The upper picture shows our agents (light-grey circles), an opponent unit (white circle) and three sheep (small dark-grey circles). The lower picture shows the total potential field for the same area. Light areas has high potential and dark areas low potential.

proposed methodology of Hagelbäck and Johansson (Hagelbäck & Johansson 2008). It includes the following six steps:

1. Identifying the objects
2. Identifying the fields
3. Assigning the charges
4. Deciding on the granularities
5. Agentifying the core objects
6. Construct the MAS architecture

Below we will describe the creation of our MAPF solution.

### Identifying objects

We identify the following objects in our applications: Cliffs, Sheep, and own (and opponent) tanks, and base stations.

### Identifying fields

We identified four tasks in ORTS Tankbattle: Avoid colliding with moving objects, Hunt down the enemy's forces, Avoid colliding with cliffs, and Defend the bases. This leads us to three types of potential fields: *Field of Navigation*, *Strategic Field*, and *Tactical field*.

The field of navigation is generated by repelling static terrain and may be pre-calculated in the initialisation phase. We would like agents to avoid getting too close to objects where they may get stuck, but instead smoothly pass around them.

The strategic field is an attracting field. It makes agents go towards the opponents and place themselves at appropriate distances from where they can fight the enemies.

Our own units, own bases and sheep generate small repelling fields. The purpose is that we would like our agents to avoid colliding with each other or bases as well as avoiding the sheep.

### Assigning charges

Each unit (own or enemy), base, sheep and cliff have a set of charges which generate a potential field around the object. All fields generated by objects are weighted and summed to

form a total field which is used by agents when selecting actions. The initial set of charges were found using trial and error. However, the order of importance between the objects simplifies the process of finding good values and the method seems robust enough to allow the bot to work good anyhow. We have tried to use traditional AI methods such as genetic algorithms to tune the parameters of the bot, but without success. The results of these studies are still unpublished. We used the following charges in the V.1 bot:<sup>1</sup>

### The opponent units

$$p(d) = \begin{cases} k_1 d, & \text{if } d \in [0, MSD - a[ \\ c_1 - d, & \text{if } d \in [MSD - a, MSD] \\ c_2 - k_2 d, & \text{if } d \in ]MSD, MDR] \end{cases} \quad (2)$$

Unit	$k_1$	$k_2$	$c_1$	$c_2$	MSD	$a$	MDR
Tank	2	0.22	24.1	15	7	2	68
Base	3	0.255	49.1	15	12	2	130

Table 2: The parameters used for the generic  $p(d)$ -function of Equation 2.

**Own bases** Own bases generate a repelling field for obstacle avoidance. Below in Equation 3 is the function for calculating the potential  $p_{ownB}(d)$  at distance  $d$  (in tiles) from the center of the base.

$$p_{ownB}(d) = \begin{cases} 5.25 \cdot d - 37.5 & \text{if } d \leq 4 \\ 3.5 \cdot d - 25 & \text{if } d \in ]4, 7.14] \\ 0 & \text{if } d > 7.14 \end{cases} \quad (3)$$

**The own tanks** The potential  $p_{ownU}(d)$  at distance  $d$  (in tiles) from the center of an own tank is calculated as:

$$p_{ownU}(d) = \begin{cases} -20 & \text{if } d \leq 0.875 \\ 3.2d - 10.8 & \text{if } d \in ]0.875, l] \\ 0 & \text{if } d > l \end{cases} \quad (4)$$

**Sheep** Sheep generate a small repelling field for obstacle avoidance. The potential  $p_{sheep}(d)$  at distance  $d$  (in tiles) from the center of a sheep is calculated as:

$$p_{sheep}(d) = \begin{cases} -10 & \text{if } d \leq 1 \\ -1 & \text{if } d \in ]1, 2] \\ 0 & \text{if } d > 2 \end{cases} \quad (5)$$

Figure 1 shows an example of a part of the map during a Tankbattle game. The screen shot are from the 2D GUI available in the ORTS server, and from our own interface for showing the potential fields. The light ring around the opponent unit, located at maximum shooting distance of our tanks, is the distance our agents prefer to attack opponent units from. The picture also shows the small repelling fields generated by our own units and the sheep.

<sup>1</sup> $I = [a, b[$  denote the half-open interval where  $a \in I$ , but  $b \notin I$

## Granularity

We believed that tiles of 8\*8 positions was a good balance between performance on the one hand, and the time it would take to make the calculations, on the other.

### Agentifying and the construction of the MAS

We put one agent in each unit, and added a coordinator that took care of the coordination of fire. For details on the implementation description we have followed, we refer to Hagelbäck and Johansson (Hagelbäck & Johansson 2008).

### Weaknesses and counter-strategies

To improve the performance of our bot we observed how it behaved against the top teams from the 2007 years' ORTS tournament. From the observations we have defined a number of weaknesses of our bot and proposed solutions to these. For each improvement we have run 100 games against each of the teams NUS, WarsawB, UBC and Uofa.06. A short description of the opponent bots can be found below. The experiments are started with a randomly generated seed and then two games, one where our bot is team 0 and one where our bot is team 1, are played. For the next two games the seed is incremented by 1, and the experiments continues in this fashion until 100 games are played.

By studying the matches, we identified four problems with our solution:

1. Some of our units got stuck in the terrain due to problems finding their way through narrow passages.
2. Our units exposed themselves to hostile fire during the cool down phase.
3. Some of the units were not able to get out of local minima created by the potential field.
4. Our units came too close to the nearest opponents if the opponent units were gathered in large groups.

We will now describe four different ways to address the identified problems by adjusting the original bot V.1 described earlier (Hagelbäck & Johansson 2008). The modifications are listed in Table 3.

### Increasing the granularity, V.2

In the original ORTS bot we used 128x128 tiles for the potential field, where each tile was 8x8 positions in the game world. The potential field generated from a game object, for example own tanks, was pre-calculated in 2-dimensional arrays and simple copied at runtime into the total potential field. This resolution proved not to be detailed enough. In

Properties	V.1	V.2	V.3	V.4	V.5
Full resolution		✓	✓	✓	✓
Defensive field			✓	✓	✓
Charged pheromones				✓	✓
Max. potential strategy					✓

Table 3: The implemented properties in the different experiments using version 1–5 of the bot.

the tournament our units often got stuck in terrain or other obstacles such as our own bases. This became a problem, since isolated units are easy targets for groups of attacking units.

The proposed solution is to increase the resolution to 1x1 positions per tile. To reduce the memory requirements we do not pre-calculate the game object potential fields, instead the potentials are calculated at runtime by passing the distance between an own unit and each object to a mathematical formula. To reduce computation time we only calculate the potentials in the positions around each own unit, not the whole total potential field as in the original bot. Note that the static terrain is still pre-calculated and constructed using 8x8 positions tiles. Below is a description and formulas for each of the fields. In the experiments we use weight  $1/7 \approx 0.1429$  for each of the weights  $w_1$  to  $w_7$ . The weight  $w_7$  is used to weight the terrain field which, except for the weight, is identical to the terrain field used in the original bot. The results from the experiments are presented in Table 4. Below is a detailed description of the fields.

*The opponent units and bases.* All opponent units and bases generate symmetric surrounding fields where the highest potentials surround the objects at radius  $D$ , the  $MSD$ ,  $R$  refers to the *Maximum Detection Range*, the distance from which an agent starts to detect the opponent unit. The potentials  $p_{oppU}(d)$  and  $p_{oppB}(d)$  at distance  $d$  from the center of an agent are calculated as:

$$p_{oppU}(d) = w_1 \cdot \begin{cases} 240/d(D-2), & \text{if } d \in [0, D-2[ \\ 240, & \text{if } d \in [D-2, D] \\ 240 - 0.24(d-D) & \text{if } d \in ]D, R] \end{cases} \quad (6)$$

$$p_{oppB}(d) = w_6 \cdot \begin{cases} 360/(D-2) \cdot d, & \text{if } d \in [0, D-2[ \\ 360, & \text{if } d \in [D-2, D] \\ 360 - (d-D) \cdot 0.32 & \text{if } d \in ]D, R] \end{cases} \quad (7)$$

*Own units — tanks.* Own units generate repelling fields for obstacle avoidance. The potential  $p_{ownU}(d)$  at distance  $d$  from the center of a unit is calculated as:

$$p_{ownU}(d) = w_3 \cdot \begin{cases} -20 & \text{if } d \leq 14 \\ 32 - 2 \cdot d & \text{if } d \in ]14, 16] \end{cases} \quad (8)$$

*Own bases.* Own bases also generate repelling fields for obstacle avoidance. Below is the function for calculating the potential  $p_{ownB}(d)$  at distance  $d$  from the center of the base.

$$p_{ownB}(d) = w_4 \cdot \begin{cases} 6 \cdot d - 258 & \text{if } d \leq 43 \\ 0 & \text{if } d > 43 \end{cases} \quad (9)$$

*Sheep.* Sheep generate a small repelling field for obstacle avoidance. The potential  $p_{sheep}(d)$  at distance  $d$  from the center of a sheep is calculated as:

$$p_{sheep}(d) = w_5 \cdot \begin{cases} -20 & \text{if } d \leq 8 \\ 2 \cdot d - 25 & \text{if } d \in ]8, 12.5] \end{cases} \quad (10)$$

Team	Win %	Wins/games	Avg units	Avg bases	Avg score
NUS	9%	(9/100)	1.18	0.57	-32.89
WarsawB	0%	(0/100)	3.03	0.12	-36.71
UBC	24%	(24/100)	16.11	0.94	0.46
Uofa.06	42%	(42/100)	10.86	2.74	0.30
Average	18.75%	(18.75/100)	7.80	1.09	-17.21

Table 4: Experiment results from increasing the granularity.

Team	Win %	Wins/games	Avg units	Avg bases	Avg score
NUS	64%	(64/100)	22.95	3.13	28.28
WarsawB	48%	(48/100)	18.32	1.98	15.31
UBC	57%	(57/100)	30.48	1.71	29.90
Uofa.06	88%	(88/100)	29.69	4.00	40.49
Average	64.25%	(64.25/100)	25.36	2.71	28.50

Table 5: Experiment results from adding a defensive field.

### Adding a defensive potential field, V.3

After a unit has fired its weapon the unit has a cooldown period when it cannot attack. In the original bot our agents was, as long as there were enemies within  $MSD$  ( $D$ ), stationary until they were ready to fire again. The cooldown period can instead be used for something more useful and we propose the use of a defensive field. This field makes the units retreat when they cannot attack, and advance when they are ready to attack once again. With this enhancement our agents always aim to be at  $D$  of the closest opponent unit or base and surround the opponent unit cluster at  $D$ . The potential  $p_{def}(d)$  at distance  $d$  from the center of an agent is calculated using the formula in Equation 11. The results from the experiments are presented in Table 5.

$$p_{def}(d) = w_2 \cdot \begin{cases} w_2 \cdot (-800 + 6.4 \cdot d) & \text{if } d \leq 125 \\ 0 & \text{if } d > 125 \end{cases} \quad (11)$$

### Adding charged pheromones, V.4

The local optima problem is well known in general when using PF. Local optima are positions in the potential field that has higher potential than all its neighbouring positions. A unit positioned at a local optimum will therefore get stuck even if the position is not the final destination for the unit. In the original bot agents that had been idle for some time moved in a random direction for some frames. This is not a very reliable solution to the local optima problem since there is not guarantee that the agent has moved out of, or will not directly return to, the local optima.

Thurau et al. (Thurau, Bauckhage, & Sagerer 2004a) described a solution to the local optima problem called avoid-past potential field forces. In this solution each agent generates a trail of negative potentials on previous visited positions, similar to a pheromone trail used by ants. The trail pushes the agent forward if it reaches a local optima.

We have introduced a trail that adds a negative potential to the last 20 positions of each agent. Note that an agent is not effected by the trails of other own agents. The negative

Team	Win %	Wins/games	Avg units	Avg bases	Avg score
NUS	73%	(73/100)	23.12	3.26	32.06
WarsawB	71%	(71/100)	23.81	2.11	27.91
UBC	69%	(69/100)	30.71	1.72	31.59
Uofa.06	93%	(93/100)	30.81	4.13	46.97
Average	76.5%	(76.5/100)	27.11	2.81	34.63

Table 6: Experiment results from adding charged pheromones.

Team	Win %	Wins/games	Avg units	Avg bases	Avg score
NUS	100%	(100/100)	28.05	3.62	46.14
WarsawB	99%	(99/100)	31.82	3.21	47.59
UBC	98%	(98/100)	33.19	2.84	46.46
Uofa.06	100%	(100/100)	33.19	4.22	54.26
Average	99.25%	(99.25/100)	31.56	3.47	48.61

Table 7: Experiment results from using maximum potential, instead of summing the potentials.

potential for the trail was set to -0.5 and the results from the experiments are presented in Table 6.

### Using maximum potentials, V.5

In the original bot all potential fields generated from opponent units were weighted and summed to form the total potential field which is used for navigation by our agents. The effect of summing the potential fields generated by opponent units is that the highest potentials are generated from the centre of the opponent unit cluster. This makes our agents attack the centre of the enemy force instead of keeping the MSD to the closest enemy. The proposed solution to this issue is that, instead of summing the potentials generated by opponent units and bases, we add the highest potential any opponent unit or base generates. The effect of this is that our agents engage the closest enemy unit at maximum shooting distance instead of moving towards the centre of the opponent unit cluster. The results from the experiments are presented in Table 7.

## Discussion

The results clearly show that the improvements we suggest increases the performance of our solution dramatically. We will now discuss these improvements from a wider perspective, asking ourselves if it would be easy to achieve the same results without using potential fields.

### Using full resolution

We believed that the PF based solution would suffer from being slow. Because of that, we did not initially use the full resolution of the map. However, we do so now, and by only calculating the potentials in a number of move candidates for each unit (rather than all positions of the map), we have no problems at all to let the units move in full resolution. This also solved our problems with units getting stuck at various objects and having problems to go through narrow passages.

### Avoiding the obstacles

The problems with local optima are well documented for potential fields. It is a result of the lack of planning. Instead, a one step look-ahead is used in a reactive manner. This is of course problematic in the sense that the unit is not equipped to plan its way out of a sub-optimal position. It will have to rely on other mechanisms. The pheromone trail is one such solution that we successfully applied to avoid the problem.

On the other hand, there are also advantages of avoiding to plan, especially in a dynamically changing environment where long term planning is hard.

### Avoiding opponent fire

The trick to avoid opponent fire by adding a defensive potential field during the cool-down phase is not hard to implement in a traditional solution. By adding a state of cool-down, which implements a flee behaviour, that makes the unit run away from the enemies, that could be achieved. The potential problem here is that it may be hard to coordinate such a movement with other units trying to get to the front, so some sort of coordinating mechanism may be needed. While this mechanism is implicit in the PF case (through the use of small repulsive forces between the own units), it will have to be taken care of explicitly in the planning case.

### Staying at maximum shooting distance

The problem we had, to keep the units at the MSD from the *nearest* opponent, was easily solved by letting that opponent be the one setting the potential in the opponent field, rather than the gravity of the whole opponent group (as in the case of summing all potentials). As for the case of bots using planning, we can not see that this really is a problem for them.

### On the methodology

We have used the newer version of the ORTS server for the experiments. On the one hand, it allows us to use the latest version of our bot, which of course is implemented to work with the new server. On the other hand, we could not get one of the last years' participants to work with the new server. Since games like these are not transitive in the sense that if player A wins over player B, and player B wins over player C, then player A will not be guaranteed to win over player C, there is a risk that the bot that was left out of these experiments would have been better than our solution. However, the point is that we have shown that a potential field-based player is able to play significantly better than a number of planning-based counterparts. Although we have no reason to believe that the UofA07 bot would be an exception, we do not have the results to back it up.

The order of the different versions used was determined after running a small series of matches with different combinations of improvements added. We then picked them in the order that best illustrated the effects of the improvements.

However, our results were further validated in the 2008 ORTS tournament, where our PF based bots won the three competitions that we participated in (Collaborative Pathfinding, Tankbattle, and Complete RTS). In the Tankbattle competition, we won all 100 games against NUS, the winner of

Team	Total win %	Blekinge	Lidia	NUS
Blekinge	98	—	96	100
Lidia	43	4	—	82
NUS	9	0	18	—

Table 8: Results from the ORTS Tankbattle 2008 competition.

last year, and only lost four of 100 games to Lidia (see Table 8).

## Conclusions and Future Work

We have presented a five step improvement of a potential field based bot that plays the Strategic Combat game in ORTS. By the full improvement we managed to raise the performance from winning less than 7 per cent to winning more than 99 per cent of the games against four of the top five teams at the ORTS tournament 2007. Our bot did also quite easily win the 2008 tournament.

We believe that potential fields is a successful option to the more conventional planning-based solutions that uses e.g. A\* in Real Time Strategy games.

In the future, we will report on the application of the methodology described in (Hagelbäck & Johansson 2008) to a number of other ORTS games. We will also set up a new series of experiments where we adjust the ability/efficiency trade-off of the bot in real time to increase the player experience.

## Acknowledgements

We would like to thank Blekinge Institute of Technology for supporting our research, the reviewers for their constructive comments, and the organisers of ORTS for providing us with an interesting application.

## References

- Arkin, R. C. 1987. Motor schema based navigation for a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 264–271.
- Borenstein, J., and Koren, Y. 1989. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics* 19:1179–1187.
- Borenstein, J., and Koren, Y. 1991. The vector field histogram: fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation* 7(3):278–288.
- Buro, M. 2007. ORTS — A Free Software RTS Game Engine. <http://www.cs.ualberta.ca/~mburo/orts/> URL last visited on 2008-06-16.
- Hagelbäck, J., and Johansson, S. J. 2008. Using multi-agent potential fields in real-time strategy games. In Padgham, L., and Parkes, D., eds., *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*.
- Howard, A.; Matarić, M.; and Sukhatme, G. 2002. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem.

In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*.

Johansson, S., and Saffiotti, A. 2002. An electric field approach to autonomous robot control. In *RoboCup 2001*, number 2752 in Lecture notes in artificial intelligence. Springer Verlag.

Khatib, O. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research* 5(1):90–98.

Massari, M.; Giardini, G.; and Bernelli-Zazzera, F. 2004. Autonomous navigation system for planetary exploration rover based on artificial potential fields. In *Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) 6th Conference*.

Röfer, T.; Brunn, R.; Dahm, I.; Hebbel, M.; Homann, J.; Jüngel, M.; Laue, T.; Löttsch, M.; Nistico, W.; and Spranger, M. 2004. GermanTeam 2004 - the german national Robocup team.

Thurau, C.; Bauckhage, C.; and Sagerer, G. 2004a. Imitation learning at all levels of game-ai. In *Proceedings of the International Conference on Computer Games, Artificial Intelligence, Design and Education*. University of Wolverhampton. 402–408.

Thurau, C.; Bauckhage, C.; and Sagerer, G. 2004b. Learning human-like movement behavior for computer games. In *Proc. 8th Int. Conf. on the Simulation of Adaptive Behavior (SAB'04)*.