Multi Robot Path Planning for Dynamic Environments: A Case Study

Sebastian Buck, Ulrich Weber, Michael Beetz, and **Thorsten Schmitt**
TU München, Institut für Informatik
Orléansstr. 34, 81667 München
{buck,weberu,beetzm,schmittt}@in.tum.de

*Most multi robot navigation planning methods make assumptions about the kind of navigation problems they are to solve and the capabilities of the robots they are to control. Because these assumptions are implicit and not well understood, selecting the **right** planning method for a given application domain is an art.*

In this paper, we propose to select problem-adequate navigation planning methods based on empirical investigations, that is the robots should learn by experimentation to use the best planning methods. To support this development strategy we provide software tools that enable the robots to automatically learn predictive models for the performance of different navigation planning methods in a given application domain. We show, in the context of robot soccer, that a hybrid planning method that selects planning methods based on a learned predictive model outperforms the individual planning methods. The results are validated in extensive experiments using a realistic and accurate robot simulator that has learned the dynamic model of the real robots.

1 Introduction

Navigation planning is one of the fundamental computational problems in autonomous robot control. A wide variety of navigation planning algorithms have been developed and studied. Several textbooks, such as [9], give systematic accounts of the navigation problem and possible solution methods.

While the computational properties of planning algorithms, that is their correctness, their completeness, the optimality of their results, and their time and space complexity have been thoroughly investigated, the problem of how to choose the ***right*** planning algorithm for a particular application and parameterize it optimally has received surprisingly little attention.

Consider, for example, navigation problems that arise in autonomous robot soccer. In robot soccer (mid-size league) two teams of four autonomous robots play against each other. In order to make a particular play, the robots of one team have to perform a joint navigation task, given by a target position for each robot. Thus to make competent plays the robots must be capable of solving the following category of navigation planning problems: given the current positions of three robots, the field players, and their respective target positions, compute a navigation path for each robot such that when the three navigation plans are executed concurrently the expected time for reaching the goal configuration is minimal. Figure 1(a) depicts a single robot navigation task in a typical game situation and the navigation plans proposed by different navigation planning algorithms. The figure illustrates that the paths computed by the different methods are qualitatively very different. While one path is longer and keeps larger distances to the next obstacles another one is shorter but requires more abrupt directional changes. The performance that the paths accomplish depends on many factors that the planning algorithms have not taken into account (see fig. 1(b)). These factors include whether the robot is holonomic or not, the dynamical properties of the robot, the characteristics of change in the environment, and so on.

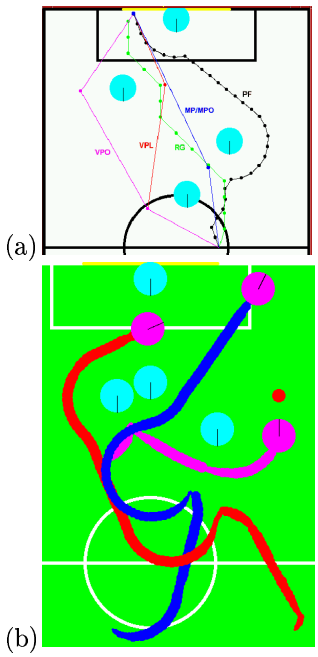

Figure 1: Single robot navigation task in a typical robot soccer scenario. Subfigure (a) shows navigation plans for one robot proposed by different path planning methods. Subfigure (b) shows the trajectories that the robots followed in a multi robot scenario. The width of the trajectory indicates the robot's translational velocity.

The conclusion that we draw from this example is that the choice of problem-adequate navigation planning methods should be based on empirical investigations. In this paper we develop a method and software tools for choosing the appropriate navigation planning algorithms and parameterizations based on empirical investigations. Because in many multi robot applications the navigation tasks are heterogeneous and there are typical distributions of navigation tasks we develop a feature language which allows us to classify navigation tasks along dimensions that challenge planning methods. We will then explain how a robot can make up classes of navigation tasks and choose the right planning mechanisms for a given task.

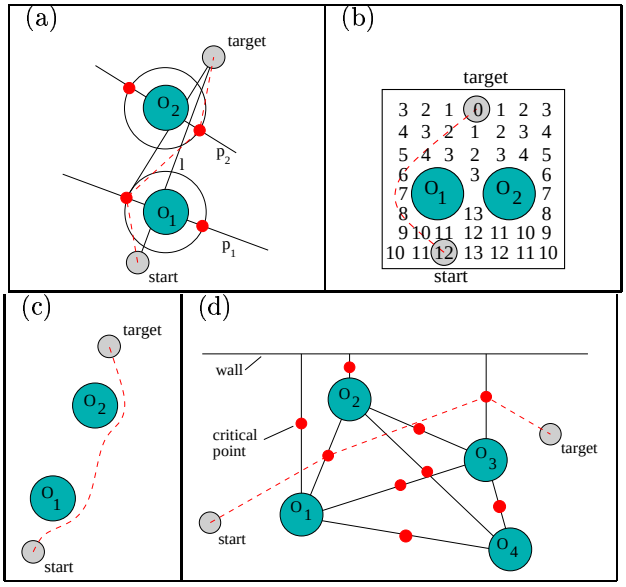

Figure 2: The figure depicts the operation of different categories of single robot navigation planning methods: the circumnavigation (a), the shortest path (b), the potential field (c), and the maximum clearance path planning method (d).

The main contributions of this paper are the following ones. First, a simulator that can learn the dynamical properties of a given robot automatically. Second, a feature language for characterizing navigation tasks. Third, a method for learning the relative performance of a set of given algorithms for different kinds of navigation tasks that occur in the application. The remainder of this paper includes a classification of multi robot path planning methods and gives a short description of the planning algorithms used in our empirical investigation (Sect. 2). Section 3 describes a set of features that can be used to measure the characteristics of multi robot navigation problems along certain interesting dimensions. Section 4 summarizes the results of our investigation. We conclude with a review of related work and a discussion of the results.

2 Multi Robot Navigation Planning

In a nutshell one can classify multi robot navigation planning methods into ones that plan the joint navigation planning task such as [2, 11] who operate in a combined configuration time-space and ones that essentially decompose a multi robot navigation task into single navigation tasks and merge and repair the resulting plans in order to avoid conflicts with the navigation plans of other robots [7].

In this paper we restrict ourselves to those multi robot navigation planning methods that apply single robot methods and integrate the results through plan merging and plan repair afterwards. Thus in the remainder of this section we will first introduce different methods for single robot navigation planning (section 2.1) and then sketch different methods for merging individual plans and repairing their negative interferences (section 2.2).

2.1 Single Robot Path Planning

Let us now introduce the path planning methods that we will use in our subsequent experiments. We categorize these methods according to the objectives they try to maximize into four categories. We consider methods that are based on attraction forces to the destination, ones that try to minimize path length, ones that consider a path as a sequence of circumnavigation steps, and ones that aim at maximizing the clearance of the paths.

Potential Field Method. The basic idea of potential field navigation planning [5] is to assign to each location in the environment a potential that results from the superposition of an attractive force towards the destination and repulsive forces caused by obstacles. The navigation plan then is the steepest descent path from the robot's current position to its destination (fig. 2(c)). A drawback of the basic algorithm is the possibility of local minima, which causes the algorithm to return paths that get stuck in a local optimum. These problems have been eliminated in a number of extensions of the basic algorithm (see, for example,[1, 15, 16]). The plans generated by potential field methods trade off safety, the distance to the closest obstacles, and path length.

Shortest Path Method. Another class of planning algorithms aim at minimizing the path length [10, 8]. These algorithms tessellate the environment into small grid cells and assign to each grid cell the length of the shortest path to the destination (fig. 2(b)). This class of planning methods have drawbacks in that they do not have a notion of path curvature, which may result in paths that require frequent changes of the robot's direction and speed. In addition, in their basic realization the methods prefer paths that are close to obstacles. This problem can be alleviated by either artificially growing the obstacles or by viewing navigation as a Markov decision process [6] where the actions have nondeterministic effects. The latter approach, however, tends to yield computationally expensive solution methods.

Circumnavigating Obstacles. The third class of methods consider path planning as finding a sequence of navigation actions that circumnavigate the obstacles that intersect the straight line paths to the destinations. This category of path planning methods include the viapoint method [14], fig 2(a) and the elastic band algorithm (for dynamic obstacles) and visibility graph planning [9] (for static obstacles). These methods sometimes cause problems when the starting and target positions are too close to obstacles.

Maximizing the Clearance. This category of navigation planning algorithms aim at the computation of paths that keep maximal distance to the obstacles. A well known member of this family is the *Voronoi path planning* algorithm [9] that guides the robots on paths that have equal distances to the closest obstacles. Another planning algorithm that aims at maximizing the clearance in the face of moving obstacles is the *maximum clearance* planning algorithm used in [3] that we

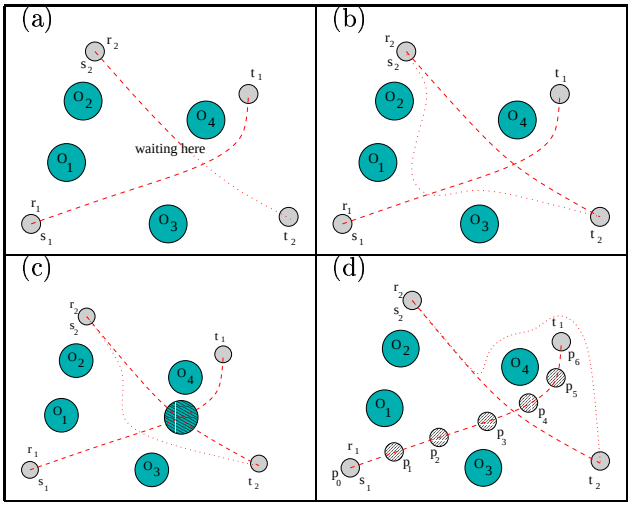

Figure 3: Several strategies for merging and repairing multi robot navigation plans by combining single robot navigation plans. Figure (a) depicts the waiting strategy. Figures (b) - (d) depict different path repair strategies including path revision through intermediate target positions (b), through intersection point obstacle (c), and robot obstacle (d).

use in our experiments as well. Due to the large average distance to obstacles the algorithms tend to propose paths that have a lower curvature than those proposed by other algorithms at the cost that the paths are becoming longer.

2.2 Plan Merging and Repair

After having detailed the algorithms for single robot path planning we will now address the question of how to combine the individual plans in order to obtain a good performance on the joint navigation tasks. In addition to the algorithms discussed in this section, there exist a number of dynamic motion planning methods which consider the obstacles' velocities for plan merging. For example, Kant and Zucker [7] propose an algorithm for navigation problems in dynamic environments that decomposes navigation planning into planning a collision free path with respect to the static obstacles and then determines the velocity along this path in order to avoid collisions with the moving obstacles. Unfortunately the application of such methods for planning in dynamic environments requires predictive models of the obstacles' motion, which in the RoboCup application cannot be provided for the robots of the opponent team because of frequent, abrupt, and unpredictable changes in the speed and orientation of their movements.

Waiting. Simply combining the paths computed by each robot without taking further precautions entails the danger that two robots might collide if their paths intersect. The simplest fix is to let one robot wait when two robots are to reach an intersection at about the same time until the other robot has crossed the intersection (see figure 3(a)). One can make this strategy smarter by assigning priorities to the different robots, such that the robot with the more urgent task can go first. [4] assigns higher priority to a robot that can drive on a straight line.

Path Replanning. The remaining methods try to revise the individual plans such that no negative interferences will occur. Again we assign priorities to the robots according to the importance of their navigation tasks and ask the robots with lower priority to revise their plans to avoid conflicts with the paths of the higher priority robots. We have considered three different methods for path revision. The first one modifies the path by introducing additional intermediate target points. The second one hallucinates additional obstacles at the positions where collisions might occur. The third one simply considers the other robot at its respective position as a static obstacle.

Defining Temporary Targets. One way to avoid possible collisions is to constrain a path by specifying additional intermediate target points. A natural constraint to impose on the path of the lower priority robot is that it is to intersect the path of the other robot behind the robot. This can be easily accomplished by setting an additional intermediate target point behind the robot with the higher priority (fig. 3(b)). The disadvantage with respect to the waiting strategy is that the path of side stepping robot becomes longer. The advantage with respect to the waiting strategy is that the side stepping robot can keep a better dynamical state, that is it does not have to stop.

Inserting New Obstacles. Another alternative for path revision is the insertion of additional artificial obstacles at the problematic path intersections. In this solution the robot with the lower priority hallucinates an obstacle at the intersection position and therefore plans a path around the critical area. The success of this path replanning technique depends on a competent placement of the intermediate target points. Figure 3(c) shows the insertion of an additional obstacle at the intersection point. Figure 3(d) repeatedly inserts the additional obstacles at the current position of the higher priority robot. However, both insertion tactics have drawbacks in that they might cause the robot to repeatedly replan or cause paths with unnecessarily high curvature.

3 Characterizing Multi Robot Navigation Problems

We have seen that a large variety of different single robot navigation planning and plan merging methods exist, that these methods have different strengths and weaknesses, and that they make different assumptions about the navigation problems at hand. This observation suggests that we need a language in which we can describe the characteristics of navigation problems in a given robot control application and use these characteristics to select the appropriate new planning method.

In our investigations we will use 7 different features (see fig. 4). These features are: (1) the number of intersections between the line segments that represent the navigation tasks, (2) the size of the bounding box

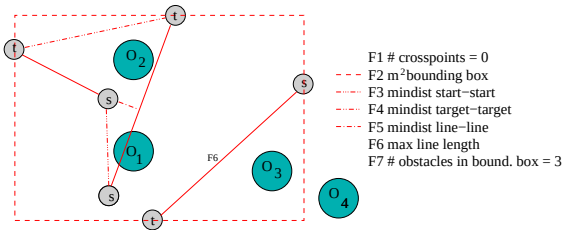

Figure 4: Visualization of navigation task features that are used for classifying navigation tasks.

of the navigation tasks, (3) the minimal linear distance between different starting positions, (4) the minimal linear distance between different target positions, (5) the minimal distance between the line segments that represent the navigation tasks, (6) the maximum length of the linear distances of the individual navigation tasks, and (7) the number of obstacles in the bounding box of the joint navigation task.

The number of intersections between navigation tasks gives us a measure of the expected complications caused by negative interactions of the individual navigation tasks. The size of the bounding box, the minimal linear distances between different starting positions as well as target positions is intended to provide us with a measure of crowdedness caused by the individual navigation tasks. Navigation tasks with line segments that are close to each other can be expected to require a higher degree of synchronization. The maximum length of the linear distances of the individual navigation tasks gives us a crude measure of the expected duration of the joint navigation task. Finally, the number of obstacles in the bounding box of the joint navigation task should be correlated with the necessary jinks.

4 An Empirical Investigation in Robot Soccer

After having introduced different navigation strategies and a language for characterizing multi robot navigation problems we will now try to assess the strengths and weaknesses of the different methods in a particular application: autonomous robot soccer. We will do so by first describing a simulation tool that enables us to collect the necessary amount of realistic data, second making a simple quantitative comparison with respect to the average performance of the individual methods, third, trying to find clusters of navigation tasks within the navigation tasks in the application that the individual methods solve well or poorly. Finally, we use that learned characterization of the kind of navigation problems that a method solves well in order to choose the navigation strategies in problem specific ways. We will show that such a deliberating navigation planner outperforms the individual methods that it uses.

4.1 A Learning Simulator

Comparing the performance of different multi robot path planning algorithms requires an accurate model of the robots' dynamics, as well as controllability and repeatability of experiments. For this reason we have developed a tool that simulates how the dynamical state of the robot changes as the robot's control system issues new driving commands such as setting the target translational and rotational velocities.

The dynamical state of the robot is summarized as a quintuple $\zeta = \langle x, y, \varphi, V_{tr}, V_{rot} \rangle$, where x and y are coordinates in a global system, φ is the orientation of the robot and V_{tr} and V_{rot} are the translational and rotational velocities. The robot control system issues driving commands that can be represented as a tuple $\xi = \langle V_{tr}, V_{rot} \rangle$, where $V_{tr}(V_{rot})$ denotes the translational (rotational) target velocity.

The dynamical model used by the simulator is acquired by learning the mapping $\Delta : \zeta_i \times \xi_i \mapsto \zeta_{i+1}$ from experience, that is recorded data from real robot runs. Our simulator learns this mapping using a simple multi layer neural network and supervised learning with the RPROP algorithm [13]. In a net driving time of more than five hours we have executed a wide variety of navigation scenarios that correspond to soccer plays. In these scenarios we have recorded the command state pairs at a frequency of $10Hz$. We have collected a total of more than 10000 training patterns. The accuracy for navigation tasks (including acceleration) is around 99%, although we haven't yet performed extensive and detailed statistical tests. The accuracy decreases to about 92% in situations where both velocities, the translational and rotational one, are changed abruptly at the same time. These inaccuracies are caused by the lack of representative training patterns as well as the high variance in navigation behavior with maximal acceleration.

4.2 Comparative Experiments

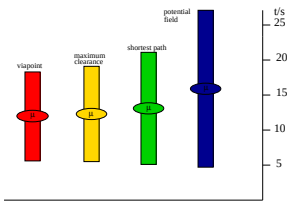

Figure 5: Mean values μ and standard deviation (in seconds) of the compared algorithms.

Because we are interested in RoboCup we set the number of obstacles to $m = 4$ (which is the team size of the mid size league). We carry out experiments with $n = 3$ robots (resulting from 3 field players in RoboCup). All our experiments underlie the same randomly generated situations: A robot starts at a randomly defined position in its configuration space and needs to get to an also randomly defined target position. The obstacles move linearly from one randomly generated start point to a randomly defined target. If an obstacle reaches its target a new target is defined immediately. Obstacles move with a randomly chosen but constant velocity from one point to another. Figure 5 pictures the mean value and the standard deviation of the time resources required to complete a joint navigation task using different planning methods. The data for the statistics was acquired by performing 1000 randomly chosen navigation problems. The results show that based on the empirical data we cannot determine a single method that out-

performs the other ones in a statistically significant way. This suggests that we should try to identify specializations of the navigation problems for which one planning method outperforms the other ones. We will look at this problem in section 4.3.

4.3 Predicting the Performance of Planning Methods

A natural way for encoding a predictive model of the expected performance of different navigation planning methods in a given application domain is the specification of rules that have the following form:

if $c_1 \wedge ... \wedge c_n$
then fastest-method($\langle$*srpm,prm* $\rangle$)

In this rule pattern the c_is represent conditions over the features that we use to classify navigation problems (see section 3). The **then**-part of the rule asserts that for navigation problems that satisfy the conditions c_i the combination of the single robot planning method *srpm* together with the plan repair method *prm* can be expected to accomplish the navigation task faster then any other combination of single robot planning and plan repair method.

The advantage of a predictive model that consists of such rules is that it can be learned automatically by decision tree learning [12][1]. We obtain the data set that is necessary for the learning task in the following way.

1. generate a random navigation task from a given distribution of navigation tasks and compute the feature vector $\langle f_1, ...f_7 \rangle$ of the navigation task.
2. solve the navigation task with each combination $\langle$*srpm,prm*$\rangle$ of the single robot planning methods and plan repairs to be investigated.
3. store the data record $\langle f_1, ...f_7, f \rangle$ where f is the combination of single robot planning methods and plan repairs that achieved the best performance in the data set that is used for learning the decision tree.

Using this data collection method we have collected a training set of 1000 data records and used it for decision tree learning. From this training set the C4.5 algorithm with standard parameterization and subsequent rule extraction has learned a set of 10 rules including the following two:

1. **if** there is one intersection of the navigation problems
   $\wedge$ the navigation problems cover a small area ($\leq 10.7m^2$)
   $\wedge$ the target points are close to each others ($\leq 1.1m$)
   $\wedge$ the starting/target point distances are small ($\leq 5m$)
   then fastest-method($\langle$*potential field,temp. targets*$\rangle$)
2. **if** there is no intersection of the navigation problems
   $\wedge$ the navigation problems cover a medium area
   $\wedge$ the distance between target points is moderate
   then fastest-method($\langle$*max. clearance,temp. targets*$\rangle$)

The first rule essentially says that the potential field method is appropriate if there is only one intersection and the joint navigation problem covers at most one fourth of the field, and the target points are close to each others. This is because the potential field algorithm tends to generate smooth paths even for cluttered neighborhoods. The second rule says that for typical size navigation problems where the navigation problems do not intersect, the maximum clearance method performs well. The rationale of the second rule is the maximal clearance causes the individual robots to take similar paths and therefore it becomes more likely that the robots pass the intersection area at the same time.

The accuracy of the ruleset for predicting the fastest navigation method is about 50% both for the training and the test set. A substantially slower algorithm was chosen only in very few cases. The inaccuracies of the rules have several reasons. First, the feature language as it has been introduced is not yet expressive enough. We expect that the accuracy of the rules can be substantially increased by adding additional features such as the angle between the robot's current orientation and the direction to the target position. Second, in many navigation problems different methods achieved almost the same performance. In those cases we only selected the best one even when the margins where very narrow. Obviously, these data records are very noise sensitive. Third, in many runs collisions were caused by dynamic obstacles and have caused robots to get stuck. These runs resulted in outlier results that are not caused by the planning methods. Thus, for higher accuracy those runs have to be handled differently.

The conclusions that we draw from this experiment are that even with crude feature languages, without sophisticated data transformations and outlier handling a robot can learn useful predictive models for the performance of different navigation methods in a given application domain.

4.4 A Hybrid Navigation Planner

An obvious idea for exploiting the predictive model that we have learned in section 4.3 is the implementation of a hybrid method navigation planner that uses a set of navigation methods and picks for every navigation problem the navigation method that is proposed by the decision tree. Thus for the next experiment we have implemented such a hybrid navigation planner and compared its performance with the performance obtained by the individual navigation methods.

We have performed a bootstrapping t-test based on 1000 different joint navigation tasks in order to empirically validate that the hybrid navigation planner performs better than the individual planning methods that we are using. Based on these experiments we obtained a 99.9% confidence in the test set (99.9% in the training set) that the hybrid method outperforms the potential field method (with its respective parameterization). The respective probabilities for the shortest path method are 99.9% (99.9%), for the maximum clearance method 99.84% (99.71%), and for the viapoint method 96.25% (94.62%). This means that

[1]For our experiments, we have used the public domain version of Quinlan's C4.5 algorithm

Algorithm	Mean time values of 1000 training and 1000 test problems			
	TRAIN		TEST	
	μ/sec	significance level $P(\mu_{tree} < \mu)$	μ/sec	significance level $P(\mu_{tree} < \mu)$
Simple Potential Field	15.49	99.99 %	15.92	99.99 %
Shortest Path	13.36	99.99 %	13.14	99.99 %
Maximum Clearance	12.35	99.71 %	12.31	99.84 %
Viapoint	12.14	94.62 %	11.95	96.25 %
Decision Tree	11.64	50.00 %	11.44	50.00 %

Table 1: Results of four evaluated algorithms and the trained decision tree. The significance level is based on a t-test.

our hypothesis that the hybrid planner dominates the other planning methods could be validated with statistical significance ($\geq 95\%$).

5 Conclusions

In this paper we have shown that in complex multi robot navigation planning domains it is extremely difficult to predict the performance of different kinds of planning methods. This is because the different planning methods make different kinds of assumptions. The maximum clearance method, for example, assumes that it is better to keep larger distances to the objects and follow a longer path with higher speed, on the other hand, the circumnavigation methods assume that it is better to pass obstacles at a closer distance. This, however, requires the robot to have more accurate control over its dynamics, for example to be equipped with an omnidirectional drive. Therefore, it is unclear for most application domains to which degree the navigation tasks match the assumptions of the different methods. Even worse, yet within a single application domain we can identify classes of navigation problems for which the algorithms' suitability varies.

In this paper, we have therefore proposed to select problem-adequate navigation planning methods based on empirical investigations, that is the robots should learn by experimentation to use the best planning methods. To support this development strategy we have provided a feature language for classifying navigation problems and software tools that enable the robots to automatically learn predictive models for the performance of different navigation planning methods in a given application domain. We have shown, in the context of robot soccer, that a hybrid planning method that selects planning methods based on a learned predictive model outperforms the individual planning methods. The results were validated in extensive experiments using a realistic and accurate robot simulator that has learned the dynamic model of the real robots.

Even though these results are conclusive we expect that the performance can be substantially improved by using a more sophisticated feature language for the classification of navigation problems and by devising more sophisticated preprocessing methods for the data elements used for learning. In particular, we would expect that a competent module for rejecting outliers would reduce the deviations in the data substantially and improve the predictability drastically. These extensions of our basic framework are subject of our future investigations.

References

[1] J. Barraquand, B. Langlois, and J. Latombe: *Numerical potential field techniques for robot path planning.* IEEE Transactions on Systems, Man, Cybernetics, 22(2):224–241, March/April 1992.

[2] M. Bennewitz and W. Burgard: *A Probabilistic Method for Planning Collision-free Trajectories of Multiple Mobile Robots.* Proc. of the workshop Service Robotics - Applications and Safety Issues in an Emerging Market at the 14th ECAI, 2000.

[3] S. Buck, R. Hanek, M. Klupsch, and T. Schmitt: *Agilo RoboCuppers: RoboCup Team Description.* RoboCup 2000: Robot Soccer World Cup IV, Springer, 2000.

[4] S.J. Buckley: *Fast motion planning for multiple moving objects.* Proc. of the IEEE ICRA 1989.

[5] Y. K. Hwang and H. Ahuja: *A Potential Field Approach to Path Planning.* IEEE Transactions on Robotics and Automation, vol. 8, 1, 23-32, 1992.

[6] L. Kaelbling and A. Cassandra and J. Kurien: *Acting under uncertainty: Discrete bayesian models for mobile-robot navigation.* Proceedings of the IEEE/RSJ IROS 1996.

[7] K. Kant and S. Zucker: *Toward efficient trajectory planning: the path-velocity decomposition.* Int. Journal of Robotics Research, 5(3):72–89, Fall 1986.

[8] K. Konolige: *A Gradient Method for Realtime Robot Control.* Proc. of the IEEE/RSJ IROS 2000.

[9] J.-C. Latombe: *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[10] J. Lengyel, M. Reichert, B. Donald, and D. Greenberg: *Real–time robot motion planning using rasterizing computer graphics hardware.* Proceedings of SIGGRAPH, pages 327–335, August 1990.

[11] S. Leroy, J.P. Laumond, and T. Simeon: *Multiple path coordination for mobile robots: A geometric algorithm.* Proceedings of the IJCAI 1999.

[12] R. Quinlan: *Induction of decision trees*, Machine Learning 1 (1), 1986

[13] M. Riedmiller and H. Braun: *A direct adaptive method for faster backpropagation learning: the Rprop algorithm.* Proceedings of the ICNN 1993.

[14] A. Schweikard: *A simple path search strategy based on calculation of free sections of motions.* Engineering Applications of Artificial Intelligence, 5, 1, 1 - 10, 1992.

[15] P. Tournassoud: *A strategy for obstacle avoidance and its application to multi-robot systems.* Proceedings of the IEEE ICRA 1986, pp. 1224-1229.

[16] R. Volpe and P. Khosla: *Manipulator Control with Superquadratic Artificial Potential Functions: Theory and Experiments.* IEEE Transactions on Systems, Man, and Cybernetics, vol. 20, 6, 1423-1436, 1990.