Prioritized Motion Planning for Multiple Robots

Jur P. van den Berg Mark H. Overmars
Institute of Information and Computing Sciences
Utrecht University, The Netherlands
{*berg, markov*}*@cs.uu.nl*

***Abstract*— In this paper we address the problem of motion planning for multiple robots. We introduce a prioritized method, based on a powerful method for motion planning in dynamic environments, recently developed by the authors. Our approach is generically applicable: there is no limitation on the number of degrees of freedom of each of the robots, and robots of various types –for instance free-flying robots and articulated robots– can be used simultaneously. Results show that high-quality paths can be produced in less than a second of computation time, even in confined environments involving many robots. We examine three issues in particular in this paper: the assignment of priorities to the robots, the performance of prioritized planning versus coordinated planning, and the influence of the extent by which the robot motions are constrained on the performance of the method. Results are reported in terms of both running time and the quality of the paths produced.**

***Index Terms*— motion planning, multiple robots, prioritized**

I. Introduction

This paper addresses the problem of motion planning for multiple robots, which is an important topic in the field. The task is to plan trajectories for the robots that bring each robot from some start configuration to some goal configuration without mutual collisions and collisions with static obstacles. This problem has been studied extensively.

Most research has focused on *coordinated* approaches. They are often categorized along the spectrum between *centralized* and *decoupled* planning [12]. A centralized planner computes a path in the composite configuration space, which is formed by the Cartesian product of the configuration spaces of the individual robots [15], [16]. In a decoupled approach a path is computed for each robot independently, and a coordination diagram is used to plan a collision-free trajectory for each robot along its path [12], [14], [17]. Approaches that only weakly constrain the motions of the robots before considering interactions between the robots can be categorized in the middle of the spectrum. They typically use *roadmaps* for each of the robots that cover each of their free configuration spaces well [8], [12], [18].

The various approaches along the spectrum trade off completeness for speed and applicability. Centralized approaches are complete, but are in general computationally demanding, or only applicable to simple robots operating in simple environments. Decoupled approaches are applicable to robots of any kind, but the paths they compute may be far from optimal. Approaches that use a roadmap provide a compromise between the two extremes.

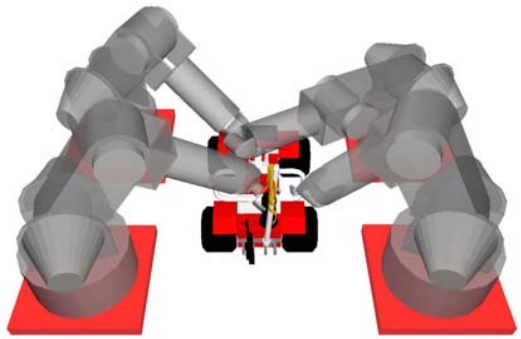

Fig. 1. An environment with four articulated robots manipulating a car.

A less studied approach to motion planning for multiple robots, which is nevertheless often used in practice, is *prioritized* planning [2], [5], [6]. In a prioritized approach each of the robots is assigned a priority. Then in order of decreasing priority, the robots are picked. For each picked robot a trajectory is planned, avoiding collisions with the static obstacles as well as the previously picked robots, which are considered as *dynamic obstacles*.

This reduces the multi-robot motion planning problem to the problem of motion planning for a single robot in a known dynamic environment, which is a difficult problem in itself. Also in this case a spectrum can be defined along which the extent is varied by which the motion of the robot is constrained. In [9], the robot motion is not constrained, and in [7] the motion of the robot is constrained to a path that is collision-free with respect to the static obstacles. Recently, the authors introduced a method that constrains the robot motion to a roadmap [3].

In this paper, we introduce a prioritized method for motion planning for multiple robots, based on the method presented in [3]. Each robot is constrained to move over a preprocessed roadmap that is collision-free with respect to the static obstacles. Our method is applicable to any number of robots with any number of degrees of freedom in both two- and three-dimensional environments (see e.g. Fig. 1). Also, robots of different type can be used simultaneously. Experiments were performed in typical and confined environments involving many robots. Results show that high-quality paths can be produced in less than a second of computation time.

Three issues in particular are addressed in this paper:

- How are the priorities assigned to each of the robots, and how does this influence the performance of prioritized planning?

- How does our prioritized approach compare to coordinated approaches, and how do the approaches perform when the number of robots increases?
- What is the effect on the performance when we vary the extent by which the robot motions are constrained?

We report results in terms of both the optimality of the paths produced and the running time of the methods.

The rest of this paper is organized as follows. In the next section we formally define the problem to be solved. In section III we introduce our prioritized method, and in sections IV to VI we address each of the three issues raised above. We conclude the paper in section VII.

II. Problem Definition

A. Definition

The problem is formally defined as follows. Given are n robots $A_1, \ldots, A_n$, and a two- or three dimensional environment in which the robots can move. For each robot A_i a roadmap R_i is constructed that covers the free configuration space of A_i, and for each robot a start configuration $s_i \in R_i$ and a goal configuration $g_i \in R_i$ is defined. We use the notation $A_i(x)$ to refer to robot A_i configured at x, where $x \in R_i$.

The task is to compute for each robot A_i a trajectory $X_i : [0, T_i] \rightarrow R_i$, such that $X_i(0) = s_i$ and $X_i(T_i) = g_i$, without collisions with other robots (note that the static obstacles are not of concern; the robots move over roadmaps that are already guaranteed to be collision-free with respect to the static obstacles). T_i can be considered as the arrival time of robot A_i at its goal configuration. We refer to $(X_1, \ldots, X_n)$ as the *composite path* of the robot.

For experimental reasons, we assume that none of the robots can collide with any other robot when it is at its start or goal configuration [1]. This ensures that a solution exists to any planning problem, by simply executing the trajectories of the each of the robots one after the other.

B. Discretization

The problem is discretized by choosing a small time step Δt and a maximal velocity v_i for each robot A_i. Each robot may move with a velocity $|v_i|$ or 0 over its roadmap and may only change it at given times $k\Delta t$, where k is an integer. This subdivides each of the edges of the roadmap in small steps of length $v_i \Delta t$. At each time step the robot may move one step in either direction along the edge, or halt at its current position. If the robot is on a vertex of the roadmap, it can choose among all of the outgoing edges of the vertex. Such a discretization is common in motion planning [3], [7], [12].

Hereafter we refer to R_i as the discretized set of states that robot A_i may adopt on its roadmap.

C. Quality Measure

An important aspect of motion planning for multiple robots is to define a quality measure of the composite path, which should be optimized. This measure is usually defined on the vector $(T_1, \ldots, T_n)$ of arrival times of each of the robots. In some previous work, all pareto-optimal solutions are generated [8], [12], but in most cases a single scalar-valued function is optimized, for instance the average arrival time of the robots. The choice of scalarization is quite arbitrary and may depend on the specific application. In this paper we choose to take the maximum of the arrival times as quality measure, i.e. we optimize the arrival time of the latest robot.

III. Prioritized Planning

Prioritized motion planning for multiple robots is a simple approach, already introduced by Erdmann and Lozano-Pérez [6] in 1987. It works as follows: Each of the robots is assigned a priority. Next, the robots are picked in order of decreasing priority. For each picked robot a trajectory is planned, avoiding collisions with the static obstacles as well as the previously picked robots, which are considered as *dynamic obstacles*.

The approach requires two important ingredients: a method to plan motions for a single robot in known dynamic environments, and a scheme to prioritize the robots. In this section we discuss how we implemented them in our method.

A. Motion Planning in Dynamic Environments

According to the definition of section II-A, we need a method that – given a robot, a roadmap, start and goal configurations s and g in the roadmap, and the scripted motions of the dynamic obstacles – computes a path for the robot starting at s at $t = 0$, and arriving as soon as possible at g.

A method for motion planning in dynamic environments that does exactly the above has recently been introduced by the authors in [3]. We briefly review its properties here:

- It is applicable to *any robot type* in configuration spaces of *any dimension*. The method only requires a roadmap that covers the free configuration space of the robot well, and that is collision-free with respect to the static obstacles.
- The shapes and motions of the dynamic obstacles are *unconstrained*: they may move with any speed following any trajectory, as long as the motions are known beforehand. That is, given a position of the robot at a time t we must be able to determine whether the robot collides with a dynamic obstacle [4].
- The roadmap and the time-axis are discretized as described in section II-B. Under these constraints, the method computes a *time-optimal* trajectory on the roadmap that arrives as early as possible at the goal configuration.
- The method achieves *interactive performance* in complicated dynamic environments.

The method finds a trajectory in the discretized roadmap-time space by an efficient multi-layer search. We will not discuss this in detail here, but refer the interested reader to [3].

The method is used in our prioritized approach as follows: Let each robot be given a priority (how these

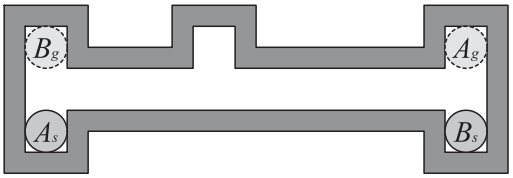

Fig. 2. An example scene in which the prioritization has a significant influence on the result. If robot A has priority over robot B, robot B can only start moving after A has reached its goal. If robot B has priority, A can use the cavity in the passageway to let robot B pass, giving a much better result.

priorities are assigned is discussed below). Then iteratively a trajectory is planned for each robot, in order of decreasing priority. If it is the k'th robot's turn, the trajectories of the $k-1$ robots that have previously been planned are considered as dynamic obstacles. This repeats until a trajectory has been planned for the robot with the least priority. The maximum of the arrival times measures the quality of the planned composite path.

B. Prioritization

An important question is how to assign the different priorities to the robots. There can of course be natural reasons to assign different priorities, for instance based on the importance of the tasks or on different starting times of the robots, but we will not assume that here.

If we have n robots, there are $n!$ different priority schedules, so in general we cannot try them all and select the best one. Yet, the order in which the trajectories are planned can have a significant influence on how optimal the resulting path is (see Fig. 2). To still find a near-optimal prioritization in only a few iterations, a randomized search with hill-climbing is applied in [2]. However, this still means that the multi-robot motion planning problem is solved multiple times, which costs valuable CPU-time.

We propose a simple heuristic to assign priorities to the robots. Let $d_i(s_i, g_i)$ be the number of steps in the discretized roadmap R_i of robot A_i on the shortest path between its start configuration s_i and its goal configuration g_i. Then, if the other robots do not stand in the way, robot A_i can reach its goal in $d_i(s_i, g_i)\Delta t$ time. We call this number the *query distance*.

In our heuristic the priority of a robot equals its query distance. The rationale behind this heuristic is – keeping in mind that we aim to minimize the maximum of the arrival times – that robots that have to traverse long distances (and hence need much time) should be able to do this relatively unhindered, while robots that have to traverse short distances can afford to spend time on avoiding robots with higher priority.

The query distances are straightforwardly computed by performing Dijkstra's shortest path algorithm [11] on each of the roadmaps.

IV. Analyzing the Prioritization

The scene of Fig. 2 was carefully chosen to thwart the prioritized method, and one would typically not encounter such a scene in practice. In this section we examine the influence of the choice of prioritization in more typical environments, and we assess the performance of our prioritization heuristic. We report results in terms of the quality (i.e. the latest of the arrival times) of the produced paths.

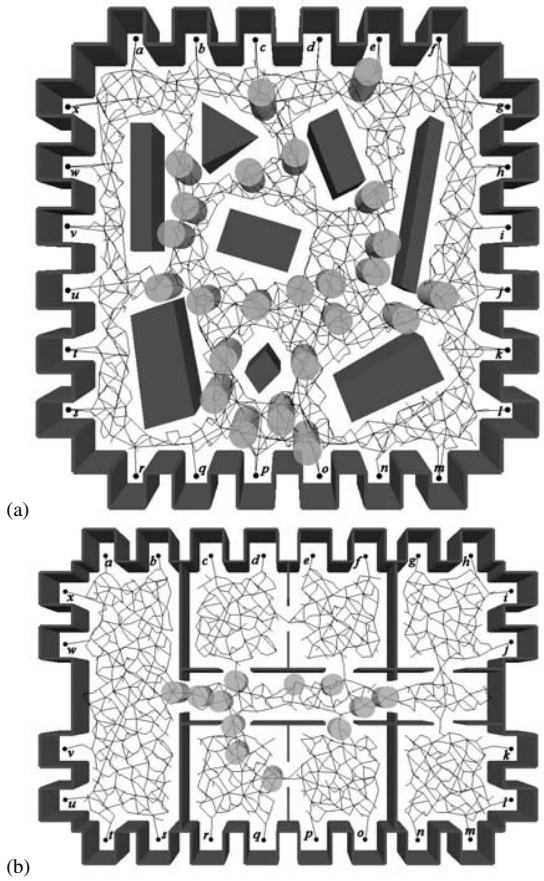

Fig. 3. Two environments and their roadmaps with 24 query configurations (a to x). (a) The 'clutter' scene. 24 robots (cylinders) are shown in a configuration along a composite path. The roadmap contains 1500 vertices. (b) The 'office' scene. Here 12 robots are shown. The roadmap contains 1218 vertices.

We experimented with our method in two environments: the 'clutter' scene and the 'office' scene (see Fig. 3). All the robots are cylinders that can translate in the plane. Since they are all the same, they can use the same roadmap (note that this is not a requirement of our method). For both scenes we constructed a roadmap using a probabilistic roadmap (PRM) method [10]. To have a choice of alternative routes we used a variant that allows the creation of cycles in the roadmap [13].

In both roadmaps we have defined 24 query configurations (a to x – see Fig. 3). Our experiments involve 12 robots with the queries $a \rightarrow n$, $c \rightarrow p$, $e \rightarrow r$, $g \rightarrow t$, $i \rightarrow v$, $k \rightarrow x$, $m \rightarrow b$, $o \rightarrow d$, $q \rightarrow f$, $s \rightarrow h$, $u \rightarrow j$ and $w \rightarrow l$. We performed the prioritized method for 500 randomly picked priority schedules. The results are given in Fig. 4. In the charts we plotted the priority schedules in order of decreasing quality relative to the optimal path quality (i.e. the maximum of the query distances, which

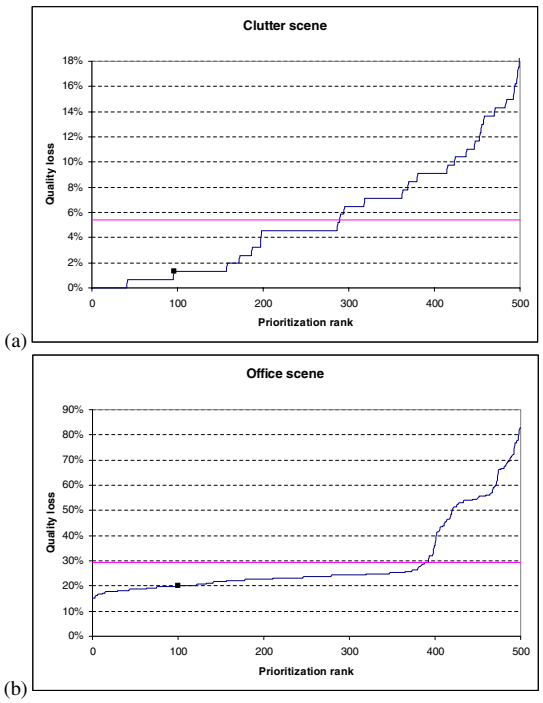

Fig. 4. Results for 500 random prioritizations in the clutter scene (a) and the office scene (b). The prioritizations are shown in order of decreasing quality relative to the optimal path quality. The dotted line indicates the average quality and the quality of our heuristic is indicated with a square.

gives a lower bound on the achievable arrival time). This gives a good indication of how the choice of prioritization influences the quality of the resulting path. The quality of our prioritization heuristic is indicated with a square.

From the results we can see that for the clutter scene the choice of prioritization has a moderate influence on the quality of the resulting path. The average path (dotted line) is slightly more than 5% longer than the optimal path. Some prioritizations actually achieve this optimum. Our prioritization heuristic performs well; it performs less than 2% worse than the optimal prioritization and far better than the average.

The office scene is more confined, and here we see that the theoretical optimum cannot be achieved. The average prioritization produces paths that are about 30% longer. A remarkable result for this scene is that the vast majority of the prioritizations produce paths of more or less equal quality and better than the average. A considerable minority however, produces paths of poor quality. Our heuristic again performs well below average; for both scenes approximately 20% of the prioritizations perform better than our heuristic, and 80% worse.

Overall we can say that the choice of prioritization can have a significant influence on the resulting path quality. Yet, our heuristic provides an efficient way to generate high-quality paths, even in confined environments.

Our prioritized method is very fast: In the above experiment, the running time to produce a composite path for 12 robots was – averaged over all 500 prioritizations – only 0.76 seconds for the clutter scene, and 0.95 seconds for the office scene. In the environment of Fig. 1 we computed a path for four articulated robots between their upright positions and the manipulating configurations as shown in the figure. Again, it took only 0.89 seconds. The experiments were performed on a Pentium IV 3.0GHz with 1 GByte of memory.

V. Comparison with a Coordinated Approach

We assess the performance of our prioritized approach, in terms of both the quality of the paths produced and the running time, by comparing it to a roadmap coordination approach introduced in [12]. The method gives the optimal robot coordination, and therefore paths with optimal quality.

A. Optimal Roadmap Coordination

Given n robots $A_1, \ldots, A_n$, discretized roadmaps $R_1, \ldots R_n$, and start and goal configurations s_i and g_i for each of the robots, the method of [12] finds an optimal composite path in the *roadmap coordination space* $\mathcal{R}$, formed by the Cartesian product $R_1 \times \cdots \times R_n$ of the roadmaps of the individual robots. The task is to find a path in $\mathcal{R}$ from $(s_1, \ldots, s_n)$ to $(g_1, \ldots g_n)$.

In each time step, the robots may either move one step over the roadmap, or halt at the current position, according to the discretization described in section II-B. This defines a neighborhood in the roadmap coordination space.

The composite path with the minimal latest arrival time is searched using the A*-algorithm [11]. Each state that is encountered during the search is checked for mutual collisions of the robots.

The complexity of the search space is exponential in the number of robots, but the algorithm exploits the *cylindrical* nature of the obstacles in the roadmap coordination space, allowing the number of collision-checks to grow only quadratic with the number of robots.

B. Experimental Results

We implemented the above algorithm and performed some experiments in the office scene of Fig. 3b. The coordinated approach turned out to be impractical for more than 3 robots. This is mainly caused by the exponential growth of the search space in the number of robots, and the large configuration space that each of the robots have. Somewhat surprisingly, the chosen optimization criterion also plays a major role: in the A*-method, the multi-robot state in the 'open'-list with the minimal estimated arrival time of the latest robot is expanded. If for multiple states this value is equal, the estimated arrival time of the second latest robot is considered, etc.

The problem with this optimization criterion is that the move costs of robots with a small query distance are hardly counted. This causes the A*-algorithm to first explore the entire reachable space of the robot with a small query distance (as long as it does not become the robot with the latest estimated arrival time) before considering a (waiting) step of another robot that *does* affect the latest estimated

TABLE I

PRIORITIZED VS. COORDINATED PLANNING

query set		Coord.	Coord. (sum)		Prioritized	
start	goal	**run. time**	**run. time**	**qual. loss**	**run. time**	**qual. loss**
(w, j)	(k, v)	1.81s	1.17s	0.0%	0.03s	5.6%
(k, v)	(w, j)	1.53s	0.84s	0.0%	0.03s	7.8%
(e, d)	(c, f)	0.70s	0.30s	1.3%	0.01s	2.5%
(a, k, v)	(n, b, j)	6.44m	31.6m	0.5%	0.06s	2.1%
(s, l, j)	(g, v, b)	7.12m	2.27m	0.0%	0.13s	5.1%
(v, c, f)	(k, o, r)	0.01s	0.02s	0.0%	0.02s	0.0%
(k, v, o)	(w, j, r)	>60m	12.53s	0.0%	0.03s	7.8%
(n, g, k)	(h, m, j)	1.65m	21.45s	0.0%	0.05s	0.0%

arrival time, even though this step may be crucial for finding an optimal coordination.

This results in large parts of the roadmap coordination space being (unnecessarily) explored. Yet, if we want to find the coordination with the optimal latest arrival time, there is no other option. Other optimization criteria, for instance the sum of the arrival times of the robots, do not have this specific problem, but may perform poorly in other situations. In general, a drawback of the A*-method is that it continuously backtracks to states whose rank in the open-list relies heavily on too optimistic arrival time estimates.

Nevertheless, we performed some experiments with the coordinated method for two optimization criteria. We optimized the maximum and the sum of the arrival times. The experiments involve two and three robots in the office scene. We compare the running time and the path quality to the prioritized approach, for which the prioritization was chosen according to the heuristic we introduced in the previous section. To have a fair comparison between the two methods, both of them use the same roadmaps. The results are given in Table I. The path quality is reported relative to the coordinated approach, which computes optimal paths.

From the results it is clear that the prioritized approach is much faster than the coordinated approach, at the expense of only a small increase in path length. Due to the problems described above, the coordinated method could not solve the seventh experiment, even though it is a rather simple problem. When we optimize the sum of the arrival times instead of the latest arrival time, this problem is solved relatively fast. In the fourth experiment however, this optimization criterion performs poorly. In general we can conclude that the running times of the coordinated methods too heavily depend on the optimization criterion and on the specific query set to be solved. In practice, it cannot be applied to problems involving 4 or more robots.

In contrast, the performance of the prioritized method scales well with the number of robots. In the clutter scene we performed an experiment involving up to 24 robots. The results are given in Fig. 5. Even for 24 robots (the scene gets very crowded then – see Fig. 3a) the prioritized method returns a path within reasonable running time.

Theoretically, the running time of the prioritized method grows quadratically with the number of robots: If the number of robots is n, then n times a trajectory is planned avoiding $O(n)$ other robots. In the figure we see that the

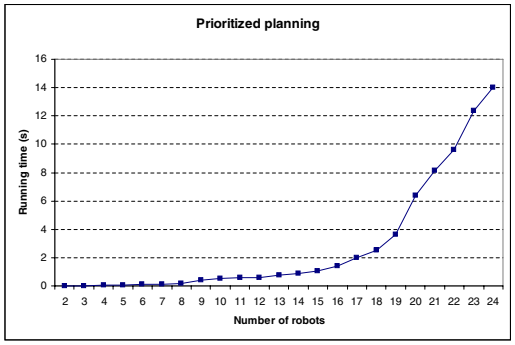

Fig. 5. Running times of the prioritized method in the clutter scene for an increasing number of robots. To each experiment one extra query is added in this order: $a \to n$, $m \to b$, $c \to p$, $o \to d$, $e \to r$, $q \to f$, $g \to t$, $s \to h$, $i \to v$, $u \to j$, $k \to x$, $w \to l$, $n \to a$, $b \to m$, $p \to c$, $d \to o$, $r \to e$, $f \to q$, $t \to g$, $h \to s$, $v \to i$, $j \to u$, $x \to k$ and $l \to w$. In each experiment the queries are prioritized according to our heuristic.

running time grows quickly for more than 20 robots. This is explained by the fact that the space gets more confined when the number of robots is high, giving more difficult problems to solve.

VI. ANALYZING THE SPECTRUM

As stated in the introduction, most approaches to motion planning for multiple robots can be categorized along a spectrum indicating how much the motions of the robots are constrained. In the one extreme the motions are constrained to a path, and in the other extreme the motions are not constrained at all. In this section we examine the effect of the extent by which the motions are constrained on the performance of our prioritized method. Again, we report results in terms of both path quality and running times.

Our method was developed to constrain the robot motions to a roadmap, and it can as such be categorized in the middle of the spectrum. It is also applicable when the robots are constrained to paths, which are special instances of roadmaps. Our method is not applicable to unconstrained configuration spaces, but we approximate this situation by increasing the density of the roadmap. As such, we slide along the spectrum between the two extremes.

We experimented in the clutter scene and the office scene with 12 robots having the same queries as in section IV. We constructed collision-free paths for the robots by taking the shortest paths from the roadmaps of Fig. 3. The density of the roadmaps was varied by tuning the number of vertices. To avoid strong deviations in the results as a consequence of the randomness involved in creating roadmaps, we *extend* roadmaps over the experiments: In each experiment we add a number of vertices to the roadmap of the previous experiment. The number of states in the roadmap grows more or less proportionally. The results are shown in Table II. The path quality is represented here as the absolute arrival time of the latest robot.

From the results we can see that constraining the motions to individual paths indeed gives low running times, but also yields composite paths of poor quality. This is caused

TABLE II

SLIDING ALONG THE SPECTRUM

	Clutter scene			Office scene		
roadm. #vertices	**roadm.** #states	**run. time**	**path length**	**roadm.** #states	**run. time**	**path length**
path	∼140	0.18s	20.44	∼150	0.17s	26.32
200	1980	0.12s	11.06	2187	0.30s	19.11
500	3728	0.35s	10.64	4194	0.59s	19.88
1000	5730	0.35s	10.36	6696	0.56s	14.49
1500	7305	1.09s	10.78	8479	0.70s	14.42
2000	8846	0.97s	10.01	9978	0.86s	14.42
3000	12033	1.33s	10.01	13100	1.76s	14.35
5000	17215	1.82s	10.01	19204	1.80s	14.35
7500	22501	2.38s	10.01	25014	2.15s	14.35

by robots that move in opposite directions and share large portions of the same path. This means that one robot has to wait a long time until the other robot has cleared the path, more or less similar to the situation of Fig. 2.

When we constrain the motions to roadmaps we see that the path quality improves as the roadmaps more densely cover the free configuration space of the individual robots. Moreover, the optimal quality is approached very quickly. After some rather low threshold value (about 2000 vertices) it seems that adding more states to the roadmap hardly helps anymore. This leads us to the conclusion that planning in unconstrained spaces is not useful; constraining motions to roadmaps is much cheaper [12] and approximates the continuous situation very well.

As can be deduced from the results, the running time of our prioritized method only grows linearly with the number of states in the discretized roadmap.

VII. CONCLUSION

In this paper we introduced a generally applicable prioritized approach to motion planning for multiple robots. The method is fast; even problems involving as many as 24 robots in confined environments can be solved in mere seconds of computation time. Choosing a good prioritization is a crucial ingredient for a successful planner. Our heuristic appeared to perform well in practice.

In this paper we assumed that the robots are collision-free on their query configurations, so that we did not have to deal with failures in the experiments. Our method, however, is also applicable without this assumption. In the experiment of Fig. 5, for instance, the assumption was violated in the runs involving more than 12 robots. It is possible though, that without this assumption the prioritized method will not find a solution, even if one exists. This is, however, unlikely to happen in most environments.

Coordinated approaches do not have this disadvantage (they compute optimal paths), but appeared not to be generally applicable to 4 or more robots. The prioritized approach, in contrast, scales well with the number of robots, at the expense of only a small increase in path length in typical scenes. Prioritized planning has more advantages. It is easily extended to motion planning for multiple robots in dynamic environments. Also, it is easily applicable in situations where multiple robots have different start times and continuously share a common environment.

Finally, we have shown that constraining the robot motions to roadmaps hardly affects the quality of the produced paths: A roadmap of only moderate density already well approximates the continuous, unconstrained configuration space. An additional advantage of using roadmaps is that they are created in a preprocessing phase. This substantially relieves the query phase (e.g. of narrow-passage problems).

ACKNOWLEDGMENTS

The authors like to thank Marin Hekman for constructing the scenes used in this paper, and Patrick Min for his careful proofreading. This research was supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2001-39250 (MOVIE - Motion Planning in Virtual Environments).

REFERENCES

[1] S. Akella, J. Peng; Time-Scaled Coordination of Multiple Manipulators. *IEEE Int. Conf. on Robotics and Automation*, pp. 3337-3344, 2004.

[2] M. Bennewitz, W. Burgard, S. Thrun; Finding and Optimizing Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots. *Robotics and Autonomous Systems* 41(2), pp. 89-99, 2002.

[3] J. P. van den Berg, M. H. Overmars; Roadmap-based Motion Planning in Dynamic Environments. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1598-1605, 2004.

[4] G. van den Bergen; *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann Publishers, San Francisco, 2004.

[5] C. M. Clark, T. Bretl, S. Rock; Applying Kinodynamic Randomized Motion Planning with a Dynamic Priority System to Multi-Robot Space Systems, *Proc. IEEE Aerospace Conference*, pp. 3621-3631, 2002.

[6] M. Erdmann, T. Lozano-Pérez; On Multiple Moving Objects. *Algorithmica* 2(4), pp. 477-521, 1987.

[7] Th. Fraichard; Trajectory Planning in a Dynamic Workspace: a 'State-Time' Approach. *Advanced Robotics*, 13(1):75-94, 1999.

[8] R. Ghrist, J. M. O'Kane, S. M. LaValle; Pareto optimal coordination on roadmaps. *Proc. Int. Workshop on the Algorithmic Foundations of Robotics*, 2004.

[9] D. Hsu, R. Kindel, J.-C. Latombe, S. Rock; Randomized Kinodynamic Motion Planning with Moving Obstacles. *Int. J. Robotics Research*, 21(3):233-255, 2002.

[10] L. Kavraki, P. Švestka, J.-C. Latombe, M. H. Overmars; Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Trans. on Robotics and Automation* 12(4), pp. 566-580, 1996.

[11] J.-C. Latombe; *Robot motion planning*. Kluwer Academic Publishers, Boston, 1991.

[12] S. M. LaValle, S. A. Hutchinson; Optimal Motion Planning for Multiple Robots Having Independent Goals. *IEEE Trans. on Robotics and Automation* 14(6), pp. 912-925, 1998.

[13] D. Nieuwenhuisen, M. H. Overmars; Useful Cycles in Probabilistic Roadmap Graphs. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 446-452, 2004.

[14] J. Peng, S. Akella; Coordinating the Motions of Multiple Robots with Kinodynamic Constraints. *IEEE Int. Conf. on Robotics and Automation*, pp. 4066-4073, 2003.

[15] G. Sánchez, J.-C. Latombe; Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2112-2119, 2002.

[16] J. T. Schwartz, M. Sharir; On the piano movers' problem III: Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal obstacles. *Int. Journal of Robotics Research* 2(3), pp. 46-75, 1983.

[17] T. Siméon, S. Leroy, J.-P. Laumond; Path Coordination for Multiple Mobile Robots: a resolution complete algorithm. *IEEE Trans. on Robotics and Automation* 18(1), pp. 42-49, 2002.

[18] P. Švestka, M. H. Overmars; Coordinated path planning for multiple robots. *Robotics and Autonomous Systems* 23(3), pp. 125-152, 1998.