Coordinated Motion Planning for Multiple Car-Like Robots using Probabilistic Roadmaps*

Petr Švestka, Mark H. Overmars
Department of Computer Science, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, the Netherlands
e-mail: petr@cs.ruu.nl, markov@cs.ruu.nl

Abstract

We present a new approach to the multi-robot path planning problem, where a number of robots are to change their positions through feasible motions in the same static environment. The approach is probabilistically complete. That is, any solvable problem is guaranteed to be solved within a finite amount of time. The method, which is an extension of previous work on probabilistic single-robot planners, is very flexible in the sense that it can easily be applied to different robot types. In this paper we apply it to non-holonomic car-like robots, and we present experimental results which show that the method is powerful and fast.

1 Introduction

We present a new method for solving multi-robot path planning problems in known static environments. In these problems, a number of robots move independently in the same workspace (containing obstacles), and the task is to compute feasible paths for the robots which bring each robot from some start configuration to some goal configuration, while avoiding (mutual) collisions.

The multi-robot path planning problem has received a considerable amount of attention in the recent years ([3],[2],[1],[7]). Current approaches basically fall into two classes: *centralized planning methods* and *decoupled planning methods* (See also ([5]). The former are very straight-forward. The idea is that one treats the separate robots as one composite robot, hence transforming the multi-robot problem into a single-robot one (with many degrees of freedom). Standard motion planning methods, for example cell-decomposition methods, can then be used for finding a path in the configuration space of the composite robot. A major drawback however is that the dimension of this configuration space is usually rather large, and, as a result, the time complexity of centralized planning methods is high. Decoupled planning methods plan the paths for the individual robots more or less independently, and, in a second stage, coordinate these paths in a way that mutual robot collisions are avoided. This scheme significantly reduces the amount of computation, but completeness is lost. For example, when two robots are to change place, then they typically follow the same route. Obviously, any coordination of the robot motions along the route will result in collisions.

The method we describe in this paper, which we refer to as the *probabilistic multi-robot method*, builds on previous work that we have done on a general single-robot planning method. This work resulted in efficient and probabilistically complete planners for a broad variety of robots, including free-flying robots ([6]), high dof articulated robots ([4]), and various non-holonomic robots ([8]). The single-robot method incrementally constructs a roadmap from which paths, feasible for the particular robot, can efficiently be retrieved. We refer to roadmaps constructed by the single-robot method as *simple roadmaps*. The multi-robot planner will combine such simple roadmaps into roadmaps for composite robots.

The probabilistic multi-robot method does not fall into either of the two above mentioned classes of multi-robot planning methods. It uses the notion of composite robots, but, unlike current centralized approaches, it requires no computations to be performed in the configuration space of the composite robot. A roadmap for the composite robot is extracted from information stored in a simple roadmap, computed by the single-robot method for the underlying simple robot.

The method is a flexible one, in the sense that it is easily applicable to many different robot types, and it is probabilistically complete. In this paper we apply it

* This research was partially supported by the ESPRIT III BRA Project 6546 (PROMotion) and by the Dutch Organization for Scientific Research (N.W.O.)

IEEE International Conference on Robotics and Automation
0-7803-1965-6/95 $4.00 ©1995 IEEE

to *car-like robots*. We give experimental results which show that the method is very efficient in terms of computation costs. Throughout this paper we will assume that the robots are identical, although the technique we present is conceptually applicable to problems involving non-identical robots as well.

2 The single-robot method

The single-robot planning method, which forms the basis for the multi-robot method that we present in this paper, is conceptually very simple. An undirected graph $G = (V, E)$ is constructed, with nodes corresponding to free configurations of the robot and edges to simple feasible paths.

The construction is done incrementally, by repeatedly adding a *random* free configuration c to V, and trying to connect c to a number of (well chosen) nodes in V, which we refer to as c's *neighbors*, by a *local planner*. Whenever such a connection succeeds, a corresponding edge is added to E. The local planner is a simple but fast planner. Given two argument configurations a and b, it constructs a path connecting a and b which is feasible in absence of obstacles. Then, it tests whether this path intersects any obstacles. If so, failure is returned, and otherwise the local planner succeeds. If the local planner is chosen properly ([8]), then probabilistic completeness of the (global) planner is guaranteed. In the rest of this paper we assume this to be the case.

Once a roadmap has been constructed in the above manner, it can be used for solving *queries*. Given a start configuration s and goal configuration g, the corresponding query consists of trying to connect both s and g to nodes (in the same connected component) of G by feasible paths, for example with the local planner. If this succeeds, then a feasible path connecting s and g can be retrieved, by performing a graph search and concatenating appropriate path segments (For details see [6]).

Of course, queries are likely to succeed only if a sufficient amount of time has been spent on the construction of G, or *learning time*. Experimental results over a wide range of examples show that these required learning times tend to be very low. For free-flying and car-like robots at most a few seconds[1] of learning where required for obtaining roadmaps that solve virtually all queries in far from trivial scenes ([6],[8]). In joint work with Kavraki and Latombe ([4]), we have applied the method to articulated robots with up to 7 degrees of freedom, in highly constrained workspaces. Again, the required learning times where surprisingly low, not exceeding 80 seconds.

The local planner and an associated distance measure, used for selecting the neighbors of a new node, are the only robot dependent components in the single-robot planning method. This makes the method flexible and easily applicable to different robot-types. This flexibility propagates to the multi-robot extension that we present in this paper.

[1] The experiments where performed on a Silicon Graphics Indigo^2 workstation rated with 96.5 SPECfp92 and 90.4 SPECint92.

3 Formalization and discretization of the multi-robot planning problem

We now formalize the multi-robot path planning problem. Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be n instances of some robot $\mathcal{A}$, present in a workspace $\mathcal{W}$, together with a set of obstacles whose union we denote by $\mathcal{B}$. Furthermore, let $\mathcal{C}$ be the space of all possible configurations of $\mathcal{A}$, and let $\mathcal{C}_f$ be a the subset of $\mathcal{C}$ consisting of all configurations c such that $\mathcal{A}$ placed at c intersects no obstacles. That is, $\mathcal{C}_f$ is $\mathcal{A}$'s free configuration space. Given a configuration c, we denote the workspace-area occupied by $\mathcal{A}$, when placed at c, by $\mathcal{A}(c)$.

Definition 1 *A path planning problem for $\mathcal{A}_1, \ldots, \mathcal{A}_n$ is defined as follows: Given start configurations $s_1, \ldots, s_n$ and goal configurations $g_1, \ldots, g_n$ (with $s_i, g_i \in \mathcal{C}_f$), find continuous maps $P_1, \ldots, P_n \in [0,1] \to \mathcal{C}_f$ describing feasible motions for $\mathcal{A}$, such that ($\forall i, j \in \{1, \ldots, n\}$):*

- $P_i(0) = s_i \wedge P_i(1) = g_i$
- $\forall_{t \in [0,1]} : \mathcal{A}(P_i(t)) \cap \mathcal{B} = \emptyset$
- $\forall_{t \in [0,1]} : i \neq j \Rightarrow \mathcal{A}(P_i(t)) \cap \mathcal{A}(P_j(t)) = \emptyset$

We refer to such a problem simply as problem $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$.

Given a graph $G = (V, E)$ storing a simple roadmap for robot $\mathcal{A}$ (computed by the probabilistic single-robot method), we are interested in solving multi-robot problems using G. For ease of presentation, we assume that all start configurations s_i and goal configurations g_i are nodes of G (they can always be added as extra, non-random, nodes), and that, for each node $c \in V$ the edge (c, c) is contained in E. We denote the workspace-area swept by $\mathcal{A}$ when moving along a path corresponding to an edge $e \in E$ by $\mathcal{A}(e)$, and we refer to it as *e's sweep-volume*. The idea is that we seek paths in G along which the robots can go from their start configurations to their goal configurations, but we disallow simultaneous motions along paths corresponding to edges e_1 and e_2 with intersecting sweep-volumes. In this way, we avoid mutual robot collisions,

while robot-obstacle collisions are ruled out by the fact that we move along the simple roadmap. We say that we *discretize* the multi-robot planning problem to G.

Definition 2 *The* G-discretized path planning problem *is defined as follows: Given start configurations* $s_1, \ldots, s_n$ *and goal configurations* $g_1, \ldots, g_n$ *(with* $s_i, g_i \in V$*), find continuous*[2] *maps* $P_1, \ldots, P_n \in \{1, \ldots, m\} \rightarrow E$ *such that* $(\forall i, j \in \{1, \ldots, n\})$*:*

- $s_i \in P_i(1) \wedge g_i \in P_i(m)$,
- $\forall_{k \in \{1,\ldots,m\}} : i \neq j \Rightarrow \mathcal{A}(P_i(k)) \cap \mathcal{A}(P_j(k)) = \emptyset$.

We say that $(P_1, \ldots, P_n)$ *is a* G-discretized path for the composite robot, *solving the problem* $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$.

We now first show that solving G-discretized path planning problems (instead of continuous ones) is sufficient, in the sense that this guarantees probabilistic completeness. Given a set of free configurations W and a graph G computed by the single-robot method, we denote by $G \oplus W$ the graph that is obtained by adding the elements of W to G, as is done with the random configurations.

Theorem 1 *Let* $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$ *be an arbitrary problem for the composite robot, for which there exists a solution in the open free configuration space of the composite robot (That is, one without robot-robot and robot-obstacle contacts). Then, within a finite amount of time, the probabilistic single-robot method will produce a graph* G *such that a* $\tilde{G}$*-discretized solution for the problem exists, where* $\tilde{G} = G \oplus \{s_1, \ldots, s_n, g_1, \ldots, g_n\}$.

Theorem 1 states that, given an arbitrary solvable problem for the composite robot, the probabilistic single-robot method will, within a finite amount of time, construct a graph G with which the problem can be solved, provided that we have means for finding G-discretized paths. This theorem is proved in the full version of this paper.

4 The multi-robot method

The question now is, given a simple roadmap $G = (V, E)$ for a robot $\mathcal{A}$, how to compute G-discretized paths for the composite robot $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ (with all $\mathcal{A}_i$ identical to $\mathcal{A}$).

We have seen that each step in a G-discretized path describes a number of (simultaneous) motions along edges with non-intersecting sweep-volumes. Clearly, during such a step, mutual robot collisions cannot occur, no matter how the individual robot velocities are chosen. Hence, it is feasible to move just one robot at a time. Of course this will increase the lengths of the composite paths, but, as we shall see later, this can be remedied with some simple smoothing techniques. It follows that we can constrain our search to *simple* G*-discretized paths*, that is, G-discretized paths where each step corresponds to the motion of just one robot, while the others stand still (at nodes of G).

For finding simple G-discretized paths, we introduce the notion of *super-graphs*. We say an edge $e \in E$ is *blocked* by a node $x \in V$ if $\mathcal{A}(e) \cap \mathcal{A}(x) \neq \emptyset$.

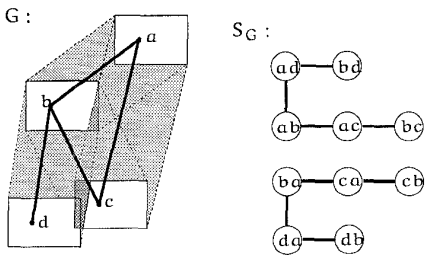

Figure 1: At the left we see a simple roadmap G for the shown rectangular robot $\mathcal{A}$ (shown in white, placed at the graph nodes). We assume here that $\mathcal{A}$ is a translational robot, and the areas swept by the local paths corresponding to the edges of G are indicated in light grey. At the right, we see the G-induced supergraph S_G for $n = 2$. It consists of two separate connected components.

Definition 3 *Given a simple roadmap* $G = (V, E)$*, the* G-induced supergraph $S_G = (V_S, E_S)$ *is defined as follows:*

- $(x_1, \ldots, x_n) \in V^n$ *is a node of* S_G *iff* $i \neq j \Rightarrow \mathcal{A}(x_i) \cap \mathcal{A}(x_j) = \emptyset$. *We refer to the nodes of* S_G *as* super-nodes. *Given a super-node* $X = (x_1, \ldots, x_n)$*, we refer to the* x_i*'s as* X*'s* underlying simple nodes.
- *Two super-nodes* $X = (x_1, \ldots, x_n)$ *and* $Y = (y_1, \ldots, y_n)$ *are connected by an edge of* S_G *if and only if for exactly one* $i \in \{1, \ldots, n\}$ $x_i \neq y_i$*, and* x_i *is connected to* y_i *by an edge in* G *which is not blocked by an* x_j*, with* $j \neq i$. *We refer to the edges of* S_G *as* super-edges.

So each node of S_G corresponds to a feasible placement of the n simple robots at nodes of G, and each edge of S_G corresponds to a feasible motion of one simple robots along an edge of G. See Figure 1

[2] We say a map P of type $\{1, \ldots, m\} \rightarrow E$ is *continuous* iff, for all $k \in \{1, \ldots, m-1\}$, $P_i(k)$ and $P_i(k+1)$ have a node in common.

for an example of a (simple) G-induced supergraph. Any path in the G-induced supergraph describes a simple G-discretized path (for the composite robot), and vice-versa. Hence, the problem of finding G-discretized paths for our composite robot reduces to graph searches in S_G.

The size of a G-induced super-graph, as defined above, is exponential in n (the number of robots). However, the entire data-structure does not have to be stored explicitly. Given a particular super-node X, its neighbors in S_G can be retrieved in constant time provided that, for each $(x, e) \in V \times E$, we know whether $\mathcal{A}(x)$ intersects $\mathcal{A}(e)$. This asks for a data-structure of quadratic size (in the size of G) which for each node-edge pair (x, e) stores whether $\mathcal{A}(x) \cap \mathcal{A}(e) = \emptyset$. Using optimized intersection routines, such a data-structure, which we refer to as *the G-intersection map*, can be computed and updated quite efficiently. Hence, for performing graph searches in S_G, we need only to compute and store the set V_S of super-nodes. If however n is large, then the required amount of memory can still be very (too) large. Such cases ask for reducing the number of super-nodes. In Section 6 we discuss a technique for achieving this.

Our multi-robot approach for solving a composite problem $((s_1, \ldots, s_n), (g_1, \ldots, g_n))$ now consists of the following steps:

1. Compute a simple roadmap G of sufficient density using the probabilistic single robot approach.
2. Add $s_1, \ldots, s_n$ and $g_1, \ldots, g_n$ to G (together with edges connecting them to other nodes).
3. For each node v and each edge e, compute and store whether $\mathcal{A}(v) \cap \mathcal{A}(e) = \emptyset$ (That is, compute the G-intersection map).
4. Construct the supergraph S_G as described above, and store it in an (partially) implicit form.
5. Find the shortest path in S_G between node $(s_1, \ldots, s_n)$ and node $(g_1, \ldots, g_n)$.
6. Transform this path into a feasible path in the configuration space of the composite robot, using the local method.
7. Smooth the (maximal) segments of the composite path where only one robot moves.
8. Combine consecutive motions of different simple robots into simultaneous ones, where possible.

The last two steps of the algorithm require some explanation. In step (7) we identify maximal segments of the (composite) path where just one (simple) robot moves. Each such segment can be then smoothed with the use of standard single-robot smoothing techniques, after the stationary robots have (temporarily) been added to the set of obstacles. Typically, this technique significantly reduces the length of the composite path. However, it does not allow for simultaneous motions of the simple robots. In Step (8) we heuristically identify segments in the composite path where the consecutive robot-motions can be replaced by simultaneous ones, without introducing mutual robot collisions. This step again reduces the length of the composite path (Details will be given in the full paper).

5 Application to car-like robots

We have implemented a multi-robot motion planner for car-like robots, based on the super-graph concept as described in the previous section. In this implementation, we have interleaved the construction of the simple roadmap and that of the super-graph (Steps 1,3, and 4). That is, whenever a simple node is added to G, we immediately extend the super-graph S_G correspondingly. In this way we obtain a method which can be proven to be probabilistically complete; if it runs long enough then any problem which is solvable (in the open free configuration space) will be solved.

Car-like robots are solid planar robots with 3 degrees of freedom, but their motions are non-holonomically constrained in a way that they can only move forwards and backwards, and follow certain curves of a lower-bounded turning radius (for a formal definition, see [5],[8]). Intuitively, they can perform the motions that an ordinary car can.

Applying the method to car-like robots asks for a local planner that computes paths which are feasible for these robots. We use the *RTR local planner*, which uses simple curves (that is, circle arcs of constant turning radii) and straight line motions for building the local paths. Given two argument configurations a and b, the planner constructs the shortest path consisting of a simple curve, followed by a straight line motion, followed by another simple curve, which connects a and b. So the RTR local planner generates paths which describe motions that are compositions of a *Rotation*, followed by a *Translation*, followed by another *Rotation* (This local planner has the properties which guarantee probabilistic completeness of the single-robot method). For more details we refer to previous work ([8]).

We have done experiments on a Silicon Graphics Indigo^2 workstation rated with 96.5 SPECfp92 and 90.4 SPECint92. We present some results for 2 scenes.

See Figure 2 for Scene 1. It consists of a narrow "H-shaped" corridor, and there are three rectangular car-like robots present. The problem in the scene is that, if one robot is to change its position, the others often

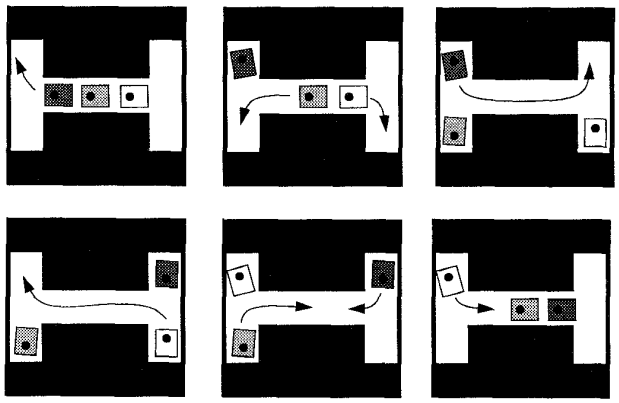

Figure 2: Scene 1. Three car-like robots in a narrow corridor. The white robot has to swap places with the dark robot.

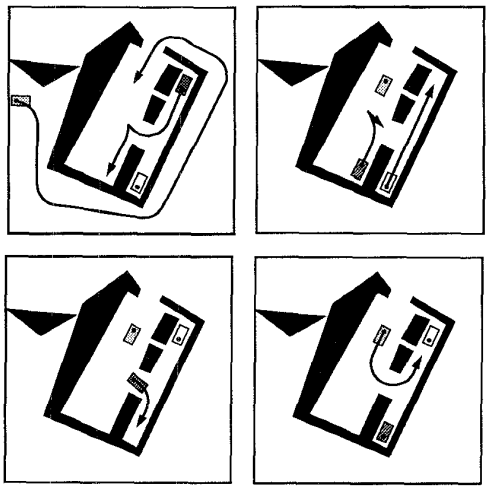

Figure 3: Scene 2. Three car-like robots in a workspace with wide and narrow areas.

have to make large detours. A decoupled planning method will not solve such problems.

Within 2 seconds a super-graph was obtained which solved most problems in this scene, among which the shown "swapping-problem" (Figure 2). This super-graph contained about 20.000 super-nodes. The computation of a shortest path in S_G took about 1 second, as did the smoothing of the path.

Scene 2 (Figure 3), which also involves 3 car-like robots, required larger super-graphs to be constructed. This is due to the fact that, in comparison with the previous scene, the free space is quite large, and, hence, a larger number of simple nodes (i.e., nodes of the simple roadmap) are required to obtain a sufficient "covering". The problem shown required about 4 seconds for the construction of a suitable super-graph (which contained about 150.000 nodes), and another 5 seconds for the shortest-path search in the super-graph. Again, smoothing took about 1 second.

6 Reducing the super-graph size

As pointed out before, in some cases it is desirable/necessary to reduce the number of nodes stored in a super-graph in order to obtain a practical planner. For this we introduce the notion of *clustered super-graphs*. We say two nodes a and b of G are G-*proximate*, if and only if a has an outgoing edge e_a and b an outgoing edge e_b such that $\mathcal{A}(e_a) \cap \mathcal{A}(e_b) \neq \emptyset$. In words, two nodes are G-proximate if they have outgoing edges with intersecting sweep-volumes. Clustered super-graphs are formalized in Definition 4.

Definition 4 *A* G-induced clustered super-graph $\hat{S}_G$ *is defined defined as follows:*

- *A super-node* X *is a* clustered super-node *(that is, a node of* $\hat{S}_G$*) iff for each underlying simple node* a *of* X *there exists another underlying simple node* b *of* X *with* a *and* b *being* G*-proximate.*

- *Two clustered super-nodes* $X = (x_1, \ldots, x_n)$ *and* $Y = (y_1, \ldots, y_n)$ *are connected by a* clustered super-edge *if and only if for exactly one* $i \in \{1, \ldots, n\}$ $x_i \neq y_i$*, and* x_i *is connected to* y_i *by a path in* G *not containing any edges* e *which are blocked by an* x_j*, with* $j \neq i$*.*

Intuitively, within each clustered super-node, the simple nodes are grouped in "clusters", that is, groups of mutually G-proximate nodes. While a regular super-edge is defined by an (non-blocked) *edge* in G, a clustered super-edge is defined by a (non-blocked) *path* in G.

It can be proven that two nodes X and Y are graph-connected in a G-induced clustered super-graph $\hat{S}_G$ if and only if they are graph-connected in the (regular) super-graph S_G, provided that G is connected. Since it is always possible to go from a (regular) super-node to a clustered super-node (if G is connected), all problems solvable by a regular super-graph can be solved by the corresponding clustered super-graph as well. Hence, in terms of solvable problems, G-induced super-graphs and G-induced clustered super-graphs are equivalent. For more details we refer to the full version of this paper.

The major advantage of clustered super-graphs is their relatively small size. In fact, if G is of bounded degree and for every edge $e \in E$ the number of edges $\tilde{e} \in E$ with $\mathcal{A}(e) \cap \mathcal{A}(\tilde{e})$ is bounded by a constant, then it is easy to prove that the number of nodes in the G-induced clustered super-graph is $\Theta(|V|^{\lfloor \frac{n}{2} \rfloor})$. So, for example, for $n = 3$ the clustered super-graph still has just a linear number of nodes (that is, linear in $|V|$). Even if the above conditions do not hold, the sizes of clustered super-graphs tend to be substantially smaller than those of regular super-graphs.

For example, we have seen that the problem in Scene 2 (previous section) is solved by a G-induced super-graph containing about 150.000 nodes. The (equivalent) G-induced clustered super-graph contains only approximately 20.000 nodes. In Scene 1 the gain is somewhat smaller, due to the "compact" structure of the scene.

The usage of clustered super-graphs however also has some drawbacks. The computation of the clustered super-edges is more expensive than that of regular super-edges, due to the fact that for each clustered super-edge that is added a graph search in G has to be performed. Hence, the edges must be explicitly stored (they cannot be retrieved in constant time). This however does not have to increase the complexity of the graph, since one can constrain the clustered super-graph to be a forest (edges creating cycles are discarded).

7 Conclusions

We have presented a new multi-robot motion planning approach. Roadmaps for the composite robot are derived from roadmaps for the underlying simple robots. This approach avoids (expensive) computations in the configuration space of the composite robot, without loss of probabilistic completeness. The roadmaps for the simple robots are computed via a probabilistic single-robot learning method, which we have recently developed and applied to a broad variety of robots. This single-robot method is very time-efficient and flexible, in the sense that it can easily be applied to different robots. We have shown that this flexibility propagates to the presented multi-robot extension by applying the multi-robot method to car-like robots. Experimental results indicate that the method is very time-efficient. Another nice property of the method is that it is, to some extent, a learning approach. That is, a super-graph constructed for a particular problem can be reused and, where necessary, extended for solving other multi-robot problems (in the same scene) as well. However, a considerable portion of the running-time of our multi-robot method is consumed by path searches in the super-graph, which must be performed for every individual query.

In this paper we required the underlying simple robots to be identical. The ideas developed in this paper are however also applicable to composite robots consisting of distinct underlying robots. In that case, a separate simple roadmap for each underlying robot can be constructed, and a super-graph can be derived from the information stored in the simple roadmaps in a straightforward manner. More computation time will however be required for computing/updating the G-intersection map. We are currently investigating this. Another interesting direction of research, which we plan to persue, is motion planning among movable obstacles. Again, the super-graph notion seems to be useful for tackling such problems, since movable obstacles can be regarded as (multiple) robots, though constrained in their motions by the fact that, in order to move, they have to be manipulated by some (real) robot.

References

[1] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. Syst. Man Cybern.*, 22:224–241, 1992.

[2] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Internat. J. Robot. Res.*, 10:628–649, 1991.

[3] M. Erdmann and T. Lozano-Peréz. On multiple moving objects. Technical Report 883, MIT, Massachusets, USA, 1986.

[4] L. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. Technical Report UU-CS-94-32, Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, August 1994.

[5] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, USA, 1991.

[6] M. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In *The Algorithmic Foundations of Robotics.* A. K. Peters, Boston, MA, 1995.

[7] J. H. Reif and H. Wang. Social potential fields: A distributed behavioral control for automonous robots. In *Proc. The First Workshop on the Algorithmic Foundations of Robotics.* A. K. Peters, Boston, MA, 1994.

[8] P. Švestka and M. Overmars. Motion planning for car-like robots using a probabilistic learning approach. Technical Report UU-CS-1994-33, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, August 1994.