A Genetic Approach to the Cleaning Path Planning Problem

Martin Dlouhý[*], František Brabec[†] and Petr Švestka[‡]

February 9, 2000

Abstract

The cleaning path problem is an important and challenging problem in the field of robotics and robot motion planning. It can be described as a task where given a vehicle equipped with a cleaning device and an area filled with obstacles, a path needs to be computed for the vehicle such that by following the path the vehicle adequately cleans or traverses all the area. Usually, the path should also satisfy additional cleaning quality requirements. This is a rather fundamental problem with numerous practical applications [1, 2]. Cleaning interiors, using a broom or a vacuum cleaner, sowing seed or plowing a field, lawn mowing, and searching for objects using metal detectors are just a few examples.

Despite the wide applicability of algorithms for the computation of cleaning paths, this problem has received far less attention than the general path planning problem [3]. The difficulty of the task depends on the characteristics of the involved vehicle or robot and the complexity of the area to be cleaned. For example, if the robot is a free-flying disc then the problem is quite simple and exact solutions can be computed efficiently [4]. For machines with limited maneuverability as for example car-like robots [5, 6, 7], the problem can in non-trivial environments become very complicated and approximate or heuristic algorithms must be employed.

We propose a new approach to the general cleaning path problem and allow different shapes of the robot and various restrictions on its maneuverability. Historically, this problem was solved only for simple areas where no obstacles were present. Our solution is based on the Probabilistic Path Planner [8, 9, 10, 11] and incorporates genetic algorithms as a method for optimization of the desired function. The strength of the method is shown in experiments using car-like robots operating in cluttered environments.

*Martin Dlouhý is with the Department of Computer Science, Charles University Prague, Malostranské náměstí 25, Praha 1, 118 00, Czech Republic. E-mail: dlouhy@ksvi.mff.cuni.cz

†František Brabec is with the Department of Computer Science, University of Maryland, College Park, MD 20742, USA. E-mail: brabec@cs.umd.edu

‡Petr Švestka is with the Department of Computer Science, Utrecht University, The Netherlands. Deceased

1 Introduction

In this paper we propose solutions for what we refer to as the *cleaning path problem.* Consider a robot (as in the path planning problem) but with an additional feature — *a broom.* The broom represents an additional tool mounted on the robot. This tool performs a certain task (e.g. sweeps the floor) as the robot moves. Therefore, the area covered (e.g. physical touched) by the broom at some point during that motion is considered to be "cleaned". The task is to develop an algorithm whose input is the workspace description along with the robot's starting position, shape and maneuverability. The algorithm then outputs a path such that if the robot follows this path, its broom will cover all the reachable part of the workspace and at the same time manage to avoid all obstacles.

There are some other noticeable differences when compared to a classical path planning. For example, the cleaning path planner can be invoked only once for a given static environment. Because of this, the speed of the algorithm computing the cleaning path is not as crucial as for standard path planning. Moreover the path can be incrementally improved, so a cleaning robot performing routine cleaning every day can continuously increase its productivity.

Our solution of the cleaning path problem is based on a probabilistic path planner, which is a very general method of path planning for robots of arbitrary shape, degrees of freedom and restrictions to their maneuverability. To our knowledge, it is also the only practical method which can be used for car-like robots moving only forward [10]. This was the model used for our initial focus and was later extended to make a general cleaning algorithm.

The final algorithm can be applied to a cleaning space of arbitrary dimension but for most applications usually two-dimensional space suffices (for example, a floor to be cleaned). The same holds for the robot's workspace, which we also restrict to a two-dimensional plane.

The maneuverability restriction is the crucial part which makes the cleaning problem very hard. Suppose a circular car-like robot moving only forward and a mathematical example (see Figure 1), when the robot is directed to perfectly clean a circle with a circular hole of radius smaller then turning radius minus robot's perimeter. The robot

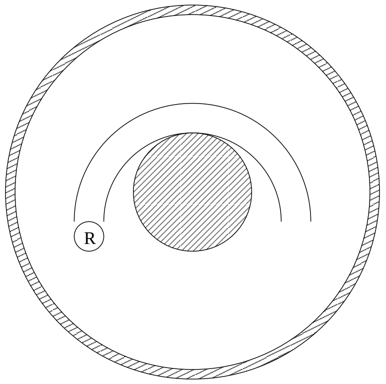

Figure 1: Infinite cleaning path for circle with circular hole

can clean such an area but the resulting path is necessarily infinite. The existance of such a model is the reason why we found it necessary to focus only on approximate or heuristic algorithms.

The paper is organized as follows. First, in section 2 we introduce the terminology and specify the cleaning path problem in detail. Section 3 gives an overview of the probabilistic path planner (PPP) with extension of the cleaning segments. Basic terms are also specified and the definition of cleaning path problem is refined for direct application of PPP. Then, in section 4, we describe several options for how cleaning segments can be generated. Next, in section 5, we employ a generic algorithm to find a suboptimal solution to cleaning path problem understood as an optimization problem. Finally, in section 6 we present the results of experiments in which the various methods were used. Section 7 concludes the paper.

2 Cleaning Path Problem

In this section we will define basic notions necessary for a more exact definition of the cleaning path problem.

A *cleaning robot* is a planar robot with a *broom* associated with it. This broom is a fixed object attached to the robot. As the robot moves, the area overlapped by the broom becomes “clean”. In this paper we assume a circular broom. This assumption simplifies our discussion yet is realistic in most scenarios. The main algorithms presented in this paper can be easily extended to robots with non-circular brooms.

Definition 1 (Cleaned areas) *The area covered by the robot's broom when placed at configuration c is referred to as the area cleaned by c and is denoted by Clean(c). Given a path P, we denote $\cup_{c\in P}$ Clean(c) by Clean(P). We refer to Clean(P) as the area cleaned by P. This is the portion of the workspace that is cleaned by the robot when it follows the path P. In general, given a set of configurations V, we denote the total area cleaned by (the configurations in) V by Clean(V).*

To define the cleaning path problem, we need a notion of what we call the *traversable* configurations.

Definition 2 (Traversable configurations) *Let $\mathcal{A}$ be a cleaning robot in a workspace $\mathcal{W}$ containing some set of obstacles; let $\mathcal{C}_f$ be the associated free configuration space and let h be the home configuration of our cleaning robot $\mathcal{A}$. A configuration $c \in \mathcal{C}_f$ is* traversable *iff there exists a free path connecting h to c and there exists a free path connecting c to h.*

Restriction to traversable configurations will ensure that the path will end in some position from which it will always be possible to return to the starting (home) position.

Definition 3 (Cleaning path) *Let $\mathcal{A}$ be a cleaning robot in a workspace $\mathcal{W}$ containing some set of obstacles; let $\mathcal{C}_f$ be the associated free configuration space and let h be the home configuration of our cleaning robot $\mathcal{A}$. Let $\mathcal{T}(h)$ be the set of all traversable configurations. A path P is a cleaning path iff P starts at home configuration h and $P \in \mathcal{T}(h)$.*

Definition 4 (Cleanable area) *Let $\mathcal{A}$ be a cleaning robot in a workspace $\mathcal{W}$ containing some set of obstacles, let $\mathcal{C}_f$ be the associated free configuration space, and let h be the home configuration of our cleaning robot $\mathcal{A}$. Let $\mathcal{T}(h)$ be the set of all traversable configurations. We define the* cleanable area *$\mathcal{CA}(h)$ as the subset of $\mathcal{W}$ consisting of all points covered by $\mathcal{A}$'s broom when located at some $c \in \mathcal{T}(h)$. That is: $\mathcal{CA}(h) = \cup_{c\in\mathcal{T}(h)}$ Clean(c)*

The goal now is to compute path P that cleans the whole cleanable area, that is Clean$(P) = \mathcal{CA}(h)$. We will call such a path the *exact cleaning path*. Unfortunately, as we mentioned in section 1, such a path can be unacceptably long. For most practical cases, the computation of the cleaning path covering the whole cleanable area is not a feasible option and thus it is necessary to compute only approximations.

While it is simple for an exact cleaning path to decide which path is better[1], this is not the case for approximate cleaning. Among several possible definitions we picked the following — for a given path length bound the path that produces the larger cleaned area is preferred.

Definition 5 (Optimal cleaning path for threshold length l_{max}) *Let P be a cleaning path. Then we say that*

[1]The cleaned area is the same, so a path which can be realized faster or which is simply shorter is preferred.

P is an optimal cleaning path for threshold length l_{max}, if length(P) $\leq l_{max}$, and for every path P' length(P') $\leq l_{max}$ is $|Clean(P')| \leq |Clean(P)|$.

3 Probabilistic Path Planner

In this section, we will introduce the Probabilistic Path Planner *PPP* which we will later apply to solve the general cleaning path problem. PPP is a special case of road-map methods. It constructs a graph whose nodes are free configurations and whose edges correspond to free paths interconnecting these nodes. The nodes are probabilistically chosen from the free configuration space and a so-called local planner searches for paths linking pairs of nearby nodes. Whenever such a local connection is found, a corresponding edge is added to the graph. Motion planning problems can then be solved by performing searches in this graph, which we also refer to as a probabilistic road-map. We will use its adapted form to solve the cleaning path problem.

Let $R = (V, E)$ be a probabilistic road map for our robot (e.g., a forward car-like robot), where V is a set of free configurations and E is a set of edges representing collision-free paths computed by the local planner.

While in a normal probabilistic road-map all edges are computed by the local planner, here we introduce one additional type of edges. The original paths generated by the local planner are denoted E_{loc}. The paths added artificially are called *cleaning segments* and are denoted by E_{seg}. The idea is that E_{seg} contains potentially more complicated descriptions of a path entered either by the user or generated by some heuristic algorithm.

We also make another adaption of R — a reduction to traversable nodes (with respect to the road-map R). This means that all nodes n such that there is no connecting path between n and the home configuration and between the home configuration and n are discarded from the road map along with all edges that contain n.

Given a road-map R with home position h and restricted to traversable nodes, every path starting at h following the edges of R is a cleaning path (see section 2). Because the path length can be simply reduced to l_{max}, we have a subset of possible suboptimal cleaning paths with threshold length l_{max}. The cleaning path problem then becomes an optimization problem which we will apply the genetic algorithm on. Before we focus on the genetic algorithm however, we will first mention some approaches generating cleaning segments which can later improve the possible suboptimal solution.

4 Generation of Cleaning Segments

In this section we mention several possibilities how to heuristically generate cleaning segments.

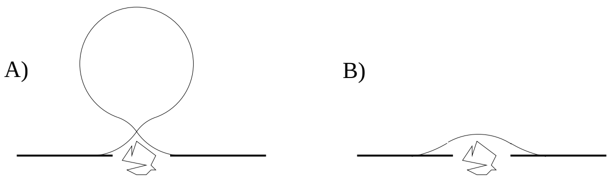

Figure 2: Two possible connections of cleaning segments

The simplest case is a regular pattern, for example a pattern of parallel lines with spacing given by broom diameter. This pattern that covers the whole plane is then restricted to traversable positions. This way, both endpoints are accessible from start position h and there is a path back to the start position h.

Another option is to start from known obstacles and to perform so called contour cleaning. Generated segments can efficiently cover the whole cleaning area but they do not have to satisfy maneuverability restrictions. For this reason, they are split into smaller segments first, then they are further processed as in the previous case.

The last option is to start with a given cleaning path, which does not take obstacles into account. The input for generating such a path can be for example a rectangular area, for which a cleaning path can be easily computed. After adding obstacles, the original path breaks into several smaller segments. Then, small connecting links are added. This is the main difference between this method and the former method. The links are supposed to connect two neighboring parts of the original path. If there is a short link, the segments are merged. There may be many links available and the shortest one does not necessarily have to be the desired one. A method used for making the selection was to evaluate paths using the sum of the length of the link and the length of the unused part of the original path. This ensured that a connecting link was sufficiently short while still using the original path. This method worked very well.

In all cases there is a problem with restricting to only traversable positions. If the extreme (boundary) position is used, then the connecting path can be complicated and unnecessarily long (see Figure 2). This can be avoided by cutting the ends of the segment into several smaller pieces and letting the search algorithm to decide which part will be used and which will not.

5 Genetic Algorithm

In this section we will propose a genetic algorithm as a solution to the cleaning path problem based on a probabilistic road-map R.

5.1 Notation

In the following text we will express this problem in terms of genetic algorithms. The origin of genetic algorithms comes from molecular biology, where each individual is completely coded by DNA sequences (in the simplified version). New individuals is created by recombination of two such sequences. Mutations can also be produced by a random local modification of a sequence. Finally, the whole process is controlled by a selection criteria, in which only the best individuals survive.

Individual — is a path in graph $R = (V, E)$. It is a list $[c_1, c_2, c_3, \ldots, c_m]$ with each $c_i \in V$, each $(c_i, c_{i+1}) \in E$, and $c_1 = h$, where h is the home position. It defines a cleaning path for which the cleaned area can be computed. The length of the path is bounded by a given limit l_{max}. If the length of the path exceeds this limit, it is shortened by removing nodes at its end until the maximum length of the path is reached within the last segment. Then, only the part of the last segment is considered that is within this given limit.

Mutation — there are three types of mutations described in genetics: deletion, insertion, and replacement. We can implement all of them, but only some combinations are allowed. Deletion of element c_d is a replacement of path (individual) $[c_1, c_2, c_3, \ldots, c_m]$ by $[c_1, \ldots, c_{d-1}, c_{d+1}, \ldots, c_m]$ if $c_{d-1}, c_{d+1} \in E$. Insertion of element d can result in path $[c_1, \ldots, c_i, d, c_{i+1}, \ldots, c_m]$ if (c_i, d) and (d, c_{i+1}) are both in E. Finally, replacement of element c_i by a new element d can be done if both (c_{i-1}, d) and (d, c_{i+1}) are in E.

Recombination — let the parents be $[c_1, c_2, \ldots, c_n]$ and $[d_1, d_2, \ldots, d_m]$. Identify all pairs (c_i, d_j) such that $c_i = d_j$ (if no such pair exists then recombination fails here and no children can be generated). Arbitrarily pick one such pair and denote it by (c_i, d_j). Its children will now be $[c_1, \ldots, c_i, d_{j+1}, \ldots, d_m]$ and $[d_1, \ldots, d_j, c_{i+1}, \ldots, c_n]$.

Generation — a set of n individuals. We will require a fixed number of individuals per generation.

Unfortunately, the definition of recombination is not sufficient for pairing if the number of nodes in R is large compared to the average number of nodes used to represent paths. Thus, a more complicated recombination rule is required.

The idea for how to increase the probability of successful recombination is as follows. Let $[c_1, c_2, \ldots, c_n]$ and $[d_1, d_2, \ldots, d_m]$ be the parents. Identify all pairs $(c_i, d_j) \in E$. If no such pair exists, the recombination fails and no children can be generated. Otherwise, arbitrarily pick one such pair (c_i, d_j). Then, the only child is $[c_1, \ldots, c_i, d_j, \ldots, d_m]$.

One disadvantage of this approach is that only one child is the result of the recombination. We could enforce the existence of both *bridges* (c_i, d_j) and (d_{j-1}, c_{i+1}) so that they are both in E, but it would reduce the chance of successful recombination.

The chance for a successful recombination depends strongly on the graph and the selected paths. However, since the selected paths are going to cover all parts of the cleaning area, it is very likely that the chance for successful recombination will be near 100%.

5.2 Evolution

Let P_0 (population) be the set of n initial cleaning paths. This set can be generated for example by a random walk. Now, we let the population evolve by letting P_0 create new generation P_1. This creates new generation P_2 and so on, until we decide to stop. Generation P_{j+1} is created from P_j according to the following algorithm:

1. Let P_{i+1} be empty.
2. While size of P_{i+1} is less than n (population size)
   - 2.1 Randomly pick two distinct individuals from P_i.
   - 2.2 Process recombination of these individuals to obtain a new child.
   - 2.3 If this succeeds:
     - 2.3.1 With some probability mutate the child.
     - 2.3.2 Add the child to P_{i+1}

The selection of individuals for recombination (or crossover) and the rate in which the mutation operators are applied are the most important factors in this process. If all individuals are picked with equal probabilities, we get results worse than in the case of randomly generated paths. It is crucial that individuals with better evaluation (larger cleaned area for given fixed length) are preferred over others. This ensures that they will have more children for the next generation.

A classical approach to the problem of the selection of individuals is based on the probability distribution proportional to the value of an optimized function. Let $a_i = area(p_i)$ denote the size of the area cleaned by path p_i. The probability that path p_i is selected equals $\frac{a_i}{\sum_{i+1}^{n} a_i}$. This method did not work well to solve our cleaning problem because the population degenerated quickly once there was a path with a much higher value than all others.

An alternative solution to this problem is an order-based distribution. Let all paths (individuals) be sorted ascendantly by the cleaned space area. Then the probability that

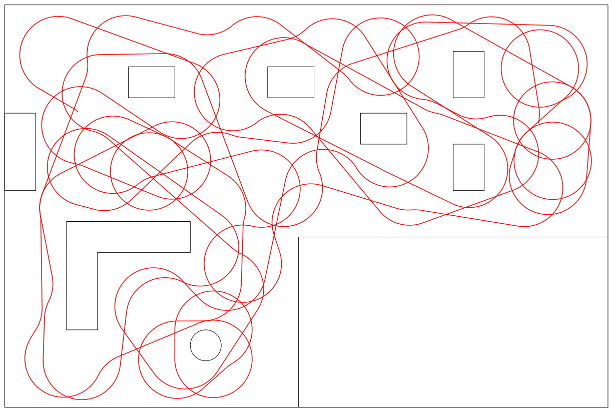

Figure 3: The best path obtained from genetic algorithm

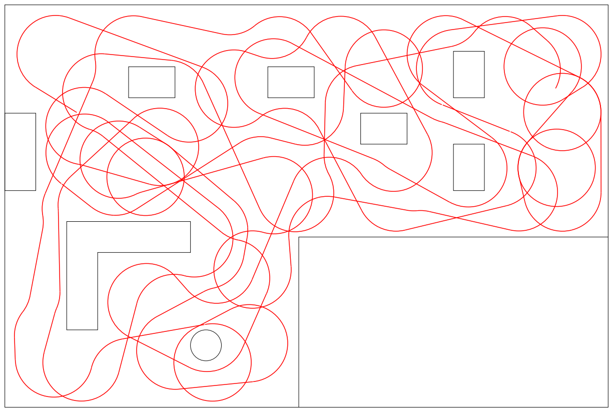

Figure 4: Example of smoothed cleaning path

the i-th element becomes selected is $\frac{i}{\frac{n}{2}(n-1)}$. This way the best individuals are preferred while there is still a chance for others to participate in recombination.

We also need to set the second parameter - the probability of mutation. When the algorithm starts, mutations usually improve selected individuals. Later, however, they tend to degrade them. Another contribution of mutations is that they ensure that the population doesn't easily degenerate. The value has to be set experimentally — we used mutation with a probability of 50% for replacing a single element.

The last parameter which has to be set is the population size. It has to be big enough to cover the variability of possible cleaning paths. On the other hand, it cannot be too large or it would slow down the iteration process. For our experiments the number varied between 20 and 100.

6 Experiments

In this section we present experiments and compare results generated by the random search and our genetic algorithm. The first example is an environment with no cleaning segments given. The whole cleaning was done only on connections generated by the local planner.

The example shown in Figure 3 is the best cleaning path for the evolution of the population of size 100.

The path can be further smoothed by making little changes to the positions of the control nodes (nodes used by the path). We use the new coordinates if the value of the cleaned area increases. This can lead to significant improvements (see Figure 4).

The distributions of path lengths for random and genetic algorithms are shown in Figure 5. One can see that the average area covered during the evolution of paths is comparable with the maximum cleaned area for the random walk. It means that the genetic algorithm finds a better cleaning path much sooner than the random walk.

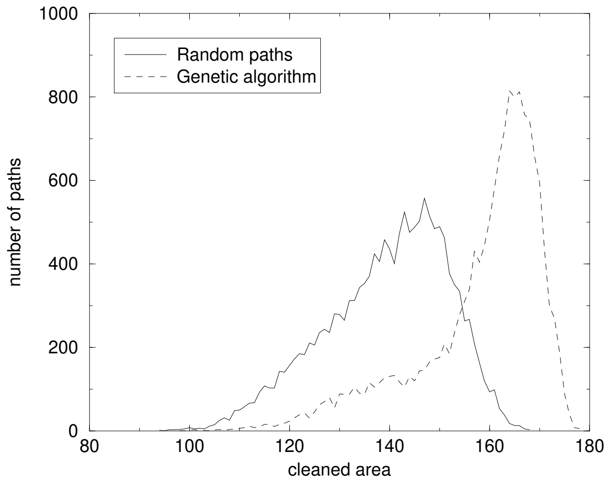

Figure 5: Distribution of cleaned area for fixed path length

Figure 6 shows an application of cleaning segments. In this case, the cleaning segments were generated as a regular raster with a step width of the size of the broom.

7 Conclusions

In this paper we presented a new approach to the general cleaning path problem, in which we allowed different shapes for the robot and restrictions on its maneuverability. In the past, this problem had been solved only for simple areas where no obstacles were present. Our solution is based on the Probabilistic Path Planner (PPP) extended by cleaning segments and incorporates genetic algorithms as a method for optimization of the desired function. The strength of

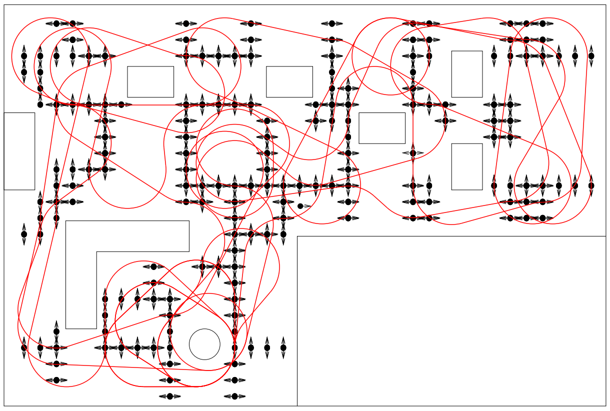

Figure 6: Traversable nodes (taken from a regular grid) together with the cleaning path

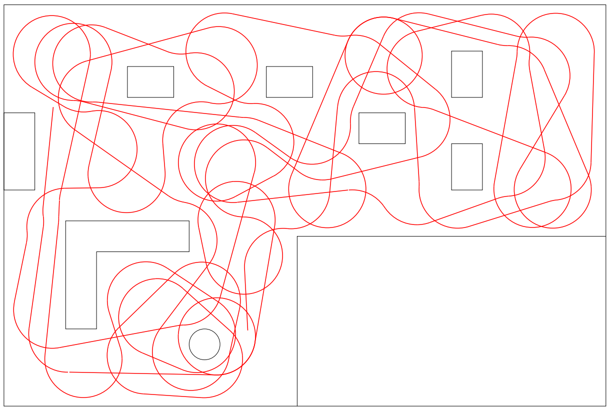

Figure 7: Smoothed cleaning path from Figure 6

the method was shown in experiments with car-like robots in cluttered environments.

References

[1] M. Schofield, "Cleaning robots," *Service Robot: An International Journal*, vol. 1, no. 3, pp. 11–16, 1995.

[2] M. Schofield, "Cleaning robots: the need for an industrial centered approach to autonomy and extended systems thinking," *Autonomic Robotic Systems*, 1997.

[3] *Handbook of Theoretical Computer Science*, chapter 8, Algorithmic Motion Planning in Robotics, pp. 393–430, Elseview Science Publishers B.V., 1990.

[4] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon decomposition," in *International Conference on Field and Service Robotics*, Canberra, Australia, 1997.

[5] S. Fortune and G. Wilfong, "Planning constrained motion," in *Proc. 20th Annu. ACM Sympos. Theory Comput.*, 1988, pp. 445–459.

[6] G. Desaulniers and F. Soumis, "An efficient algorithm to find a shortest path for a car-like robot," in *IEEE Transactions on Robotics and Automation*, Dec 1995, vol. 11, pp. 819–827.

[7] M. T. Laumond, P. E. Jacobs and R. M. Murray, "A motion planner for nonholonomic mobile robots," in *IEEE Transactions on Robotics and Automation*, Oct 1994, vol. 10, pp. 577–593.

[8] J.-C. Latombe, "Controllability, recognizability, and complexity issues in robot motion planning," *IEEE*, pp. 484–500, 1995.

[9] L. E. Kavaraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *IEEE Transaction on Robotics and Automation*, Aug 1996, vol. 12, pp. 566–579.

[10] P. Švestka and M. H. Overmars, "Motion planning for carlike robots using a probabilistic learning approach," *The International Journal of Robotics Research*, vol. 16, no. 2, pp. 119–143, Apr 1997.

[11] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," Tech. Rep., Department of Computer Science, Utrecht University, 1996.