PLANNING OF FAST AND SECURE PATHS IN COMPLEX CONFIGURATION SPACES USING SIMPLE PRIMITIVES

K. Verbarg
Department of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany

In this paper, we give a heuristic to compute a collision-free path between two configurations of a general robot. The heuristic only relies on the computation of a very simple primitive, namely the computation of the *clearance* of the robot at a given configuration. This restriction arises in the control of industrial welding robots, where pure computational geometry algorithms are doomed to fail due to technical restrictions. In particular, we are only able to control the robot with the help of a given tool, without knowing the exact position and extensions neither of the robot nor of the obstacles in the workspace. To describe this setting adequately the terms *fast* und *secure* for the computed paths are derived. The efficiency of our method is verified experimentally.

1 Introduction

In this paper we investigate the general robot motion planning problem [1,2,3]. There, the position of a robot living in workspace can be uniquely described by a configuration vector. This vector for example encodes the angles between consecutive links of a robot manipulator. The length of this vector is the number of degrees of freedom (dof) of the robot and the dimension of the configuration space. For a scene of obstacles in the workspace the motion planning problem aims to find a collision-free motion between two given configurations of the robot. Unfortunately, the running time of *all* exact algorithms is exponential in the number f of degrees of freedom in the worst-case[a]. This is due to the fact that the complexity of free space can be $\Omega(n^f)$, where n is the number of (obstacle-)constraints. Obviously, these approaches are not well suited for practical applications, where f is usually larger than three.

So far, only little progress has been made with *heuristic* algorithms. These are algorithms which are not guaranteed to find a solution. Another drawback of this approach is that it may take prohibitive time to find a solution. To overcome this situation, a collection of papers [4,5,6,7,8,9] dedicated to *probabilistic path planning* with many variants of *randomized planners* have been published. Assuming that the free space is well behaved (measured by its "ε-goodness" [4]), they achieve a *randomized completeness*: a path is found with a certain probability, if the algorithm is kept on running long enough. Unfortunately, the assumptions made there rely on some unverifiable properties of the free space. To the best of our knowledge, it is an open problem, whether also assumptions in the workspace can be made to obtain similar results. Alternatively, it would be interesting to investigate the relationship between properties in the workspace and the free space. For example, manipulators with many degrees of freedom may have a corresponding free space with a very small volume. Braun and Corsépius [10] combine several techniques like

[a] We assume that the constraints on the motion of the robot and the obstacles can be algebraically described and the free space can be defined as a semi-algebraic set [3].

a more efficient local planner with the randomized path planning approach and obtain promising experimental results.

In this paper we want to put forward the claim that in practical situations it is not necessary to compute a shortest path. In contrast, it may be preferable to sacrifice the length of the path for the sake of a safer (with a certain clearance from the obstacles), faster (which can be executed faster by the robot) or more efficiently computable path if such a path exists. To this end, it suffices to compute a connected subset of the free space containing the start and the goal configuration, which may have lower complexity than the free space itself by orders of magnitude. Notice that the arrangement of configuration space obstacles may have a very large complexity in general. The underlying idea of our approach is simply to explore the free space starting at one configuration until the goal configuration is reached. The search is guided by the A*-algorithm [11]. We will use only primitive operations such as the computation of the clearance of the robot for a given configuration. Moreover, the strategy takes advantage of certain properties: the volume of the free space compared to the entire feasible configuration space may be small and moreover the path we are seeking may be short and of large minimal clearance.

This paper is organized as follows: In Section 2 we introduce the basic notations and the set of primitive operations, which we will show to be adequate for the special setting of welding robots in Section 3. Section 4 describes our new heuristic. We present a brief overview of informal advantages for the application in mind in Section 5 and some experimental results in Section 6. We conclude this paper with a discussion in Section 7.

2 Clearance of the Robot

The basic primitive we will use is the computation of the *clearance* for a configuration Q of the configuration space $\mathcal{C}$, denoted by $clearance(Q)$. It is defined to be the infimum of the Euclidean distances between any part of the robot, which is positioned at configuration Q, and the obstacles in workspace. Clearly, this infimum is a minimum if we assume the robot and the obstacles to be closed and bounded. We are mainly concerned with the simple and important case that the obstacles and the robot are simple polyhedra. The clearance can then be computed in $\mathrm{O}(N)$ time, where N is the total complexity of the obstacles, if the complexity of the robot is bounded. This can be improved for special settings, for example using spatial data structures [12,13], maintaining a set of nearest objects [14], or using hierarchical representations of the objects [15]. For our purposes, it would also be sufficient to use approximations of the robot or the obstacles, e.g. by a union of spheres [16], as long as we are able to control the maximum absolute error of the distance computations.

We may assume that the mapping $m\colon \mathcal{C} \to$ workspace describing the connection between the configuration space and the workspace is smooth. In the sequel we assume that it also obeys a *Lipschitz condition*

$$||m(X) - m(Y)|| \leq \delta \, ||X - Y|| \quad \text{for all } X, Y \in \mathcal{C}. \tag{1}$$

Notice that such a condition is particularly satisfied if the matrix norm of the derivative m' is bounded on $\mathcal{C}$. Equation (1) corresponds well with our intuition that a robot may not perform "unbounded" accelerations.

Following the approach of Kavraki [4] et al. for any bounded robot $\mathcal{R}$ there exist constants $\varrho = (\varrho_1, \ldots, \varrho_f) > 0$ such that when $\mathcal{R}$ moves along a straight line segment in $\mathcal{C}$ between two configurations $Q = (Q_1, \ldots, Q_f)$ and $Q' = (Q'_1, \ldots, Q'_f)$, no point of $\mathcal{R}$ traces a curve segment longer than

$$\sum_{i=1,\ldots,f} \varrho_i \, |Q_i - Q'_i| = ||\varrho \circ (Q - Q')||_1.$$

in the workspace. Notice that the multiplication of two vectors is understood to be taken component-wise. More precisely, we claim that the Euclidean distance between the two positions of any point of $\mathcal{R}$ with respect to Q and Q' is at most $||\varrho \circ (Q - Q')||_1$.

The vector ϱ is determined by the geometry of $\mathcal{R}$. Due to the use of the L_1-norm above, it suffices to give an estimation for each value of ϱ_i $(i = 1, \ldots, f)$ separately. This means that if $Q - Q' = (0, \ldots, 0, Q_i - Q'_i, 0, \ldots, 0)^\mathsf{T}$, then no part of $\mathcal{R}$ traces a distance longer than $\varrho_i \, |Q_i - Q'_i|$. In our implementation, we chose the coordinates of the position of a reference point to describe the ith translational dof. In this case, $\varrho_i = 1$. Rotational degrees of freedom were described by a rotation angle given in the range $[0, 2\pi[$ with respect to a reference point. Clearly, we can choose ϱ_i as the maximum distance of any part of the robot from the center of rotation in this case.

For manipulators, which are typically long with respect to their sphere of influence, this estimation can be quite pessimistic. In implementations faced with this situation, it is more sensible to use a refined estimation of the geometry of $\mathcal{R}$: Compute a vector ϱ for *every* link separately. We omit the details of the changes for the algorithm here. Notice that for a rigid robot with translational and rotational degrees of freedom it is advantageous to choose the reference point (for the rotation) in the center of the robot to obtain the desired property that the absolute values of each component of the vector ϱ are as small as possible.

Hence, we can conclude that for a configuration $Q \in \mathcal{C}$ with $clearance(Q)$ all configurations Q' with $||\varrho \circ (Q - Q')||_1 < clearance(Q)$ are free. For simplicity, we define the corresponding norm in $\mathcal{C}$ as the *weighted L_1-norm*:

$$||Q - Q'||_\varrho := ||\varrho \circ (Q - Q')||_1. \tag{2}$$

Additionally, we may have to compute $Q - Q'$ for rotational dofs considering that they are 2π-periodic. Then the distance with respect to such a component would have to be taken modulo π.

Based on the notion of clearance, we can now specify the *safety* of a path. It seems to be sensible (cf. also the next section) to allow a larger *velocity* of the robot the larger the clearance is. If we assume the velocity of the robot along a segment of a path to be given by the length of the segment, as we will show to be sensible in the next section, then fastest paths are given by *minimum-link paths* [17], that is the number of segments the path consists of. A path consisting of straight-line segments is called *β-safe*, if the length of every segment is bounded by β times the minimal *clearance* of the points contained by the segment.

3 Welding Robots

In this section we want to motivate that restriction of an algorithm to the computation of *clearance* is sensible in many applications. To this end, we address our experience with the control of *welding robots* at Audi AG Ingolstadt [18]. Such a robot manipulator $\mathcal{R}$ is programmed with a simulation tool, here IGRIP, hiding completely the geometry and kinematics of $\mathcal{R}$ and the description of the obstacles. This tool is a simple front-end to control $\mathcal{R}$ and is shipped together with $\mathcal{R}$. It behaves like the real robot (gravity and kinematic effects) and respects given constraints on exactness and reliability. A point of the $\mathcal{C}$-space of $\mathcal{R}$ is given only by the position and orientation of the end-effector, independent of the number of dofs of $\mathcal{R}$. The primitives provided are the computation of *clearance* and testing whether a given path in $\mathcal{C}$ is free. Notice that the velocity of the execution of a path consisting of consecutive straight-line segments is determined by the number of segments because the simulation tool tries to interpolate (more or less precisely) the given intermediate way-points when executing the motion physically. Moreover, it is sensible to correlate the maximum velocity of a segment with its clearance, to prevent collisions while actually executing the path due to kinematic deviations from the given path.

In this setting, it is not possible to use exact motion planning algorithms, because the technical effort to handle CAD-data and the geometry of $\mathcal{R}$ is prohibitively large. Moreover, companies forced the manufactures of robots to develop simulation tools with a guaranteed accuracy. This accuracy is influenced by industrial and physical constraints (such as torques, dynamics, kinematics and properties of the material), which are not addressed in the classical motion planning model of computational geometry. A lot of effort and money has been invested into this project, such that a "new" approach from scratch appears to be infeasible. Therefore, currently heuristics without any performance guarantee or completeness property are in use. If they fail, on-line programming supervised by an expert is still necessary. Searching on a sufficiently fine grid in $\mathcal{C}$-space seems to be too expensive in terms of running time. A feasible solution may already be the approach of Warren [19] who additionally searches on a "loose" grid.

Besides planning all paths off-line, an efficient solution of the path planning problem gives also a distance between two welding points and enables to optimize the sequence of points which $\mathcal{R}$ travels along. This is basically a traveling salesperson tour problem with some additional (heat) constraints. Moreover, one could then optimize the assignment of robots along the production line to welding points. Hence, any improvement of the heuristic is economically interesting due to savings in programming time, production cycles, robots and improvement of quality. Finally, we may give a feedback to the construction phase of the parts with respect to the simplicity of welding parts together.

4 The Algorithm

We consider the problem of computing a path between a start and target configuration $S, T \in \mathcal{C}$. The basic idea of our algorithm is to explore the free space starting at S guided by an A*-algorithm. Additionally, we control the expansions by the following rule: The larger the clearance is, the larger our steps to the next expansion should be. The overall algorithm is compiled in Algorithm 1.

Algorithm 1 Compute safe and fast path.

Input: Start configuration $S \in \mathcal{C}$ and target configuration $T \in \mathcal{C}$

1 $\mathcal{V} \leftarrow \varnothing$ **{**the explored portion of the free space**}**
2 $\mathcal{F} \leftarrow \{S\}$ **{**the open list**}**
3 Together with every node V we store the circle centered at configuration V with the radius *clearance*(V)
4 $\mathcal{E} \leftarrow \varnothing$
5 **while** $\mathcal{F} \neq \varnothing$ **do**
6 $\quad$ $V \leftarrow$ a vertex in $\mathcal{F}$ with minimum weight $w(V)$
7 $\quad$ Delete V from $\mathcal{F}$
8 $\quad$ If V is covered by a circle in $\mathcal{V}$ then goto Step 5
9 $\quad$ Add V to $\mathcal{V}$ and edge E to $\mathcal{E}$, where E is the edge connecting V with the node which created V in the expansion Step 10
10 $\quad$ Add *expand*(V) to $\mathcal{F}$
11 $\quad$ If T is reachable from V, then stop
$\quad\quad$ **{**A path is found by backtracking the constructed tree**}**
12 **end while**

To apply the A*-algorithm we must implicitly define a graph $\mathbf{G}$ which is to be searched. We will define and construct this graph while searching the free space. Therefore $\mathbf{G} = (\mathcal{V}, \mathcal{E})$ describes the portion of the free space which is already explored. For every vertex $V \in \mathcal{V}$ we store a configuration Q_V and a radius r_V such that all points $X \in \mathcal{C}$ in the circle $\|Q_V - X\|_\varrho \leq r_V$ are contained in the free space. This is achieved by setting $r_V = clearance(Q_V)$. The edges in $\mathcal{E}$ describe the feasible paths between these configurations, that is, Q and Q' are connected by an edge if and only if Q is in the circle of Q' or vice versa. Hence, the straight line connecting Q and Q' in the $\mathcal{C}$-space is free.

The A*-algorithm maintains a list of *open* vertices $\mathcal{F}$ and *closed* vertices $\mathcal{T}$. The open vertices in $\mathcal{F}$ are those to be expanded next. For all open vertices V we compute a weight

$$w(V) = g(V) + h(V),$$

where $g(V)$ is the *cost of the path* from S to V in $\mathbf{G}$ and $h(V)$ is an estimation of the remaining effort to reach T. The open vertex with the least weight will be expanded next. A sensible choice for the functions g and h will be given below.

It remains to specify how an open vertex V is expanded. We must identify a set of new nodes *expand*(V), which are incident to V. When V is expanded, these nodes are added to $\mathcal{F}$. For the parameter $\alpha \in]0, 1[$ of the algorithm we will choose all nodes of *expand*(V) within the distance of $\alpha \cdot r_V$ (with respect to the norm $\|\cdot\|_\varrho$). Again, we will discuss sensible settings for the function *expand* below.

Since $\mathbf{G}$ is computed during the execution of the algorithm we have to ensure that we do not reenter already discovered free space. Therefore, for a node V we want to expand next, we first have to test whether it is already covered by circles in $\mathcal{V}$. If so, we do not need to expand V and have to delete it. A node V for the configuration Q is *covered* by a circle with center Q' and radius r' if $\|Q-Q'\|_\varrho < \alpha r'$.

If s is the total number of circles, the test can be performed in $O(s)$ time by checking each circle separately. The test can be sped up by the application of a spatial data structure. Especially the MBT [20] is well-suited to store the set of circles and support point location queries since it uses circles itself to cluster the stored objects. If additional bounds on the distances of vertices in $\mathbf{G}$ and radii of circles can be employed (e.g. by technical restrictions of the problem) then the time complexity to perform the test can be further reduced. Especially in scenes of low obstacle density it reduces to $O(\log^{f-1} s)$ [21]. If the distances and radii are bounded from below and above, then the time requirement can be improved to be only logarithmic in s [22].

The algorithm is not guaranteed to terminate if no safe path exists. Therefore, we introduce an artificial criterion to stop the algorithm and report failure.

4.1 Expanding Vertices

For a vertex V we construct a set of incident vertices $expand(V)$ such that the free space will be explored thoroughly enough. For simplicity, the cardinality of $expand(V)$ for all V is assumed to be a constant (depending on f).

For some fixed $\alpha \in]0,1[$ a simple choice is to select all vectors from the Cartesian product $\{-1,0,1\}^f$ scaled to the length

$$\alpha \cdot clearance(V)$$

w.r.t. $\|\cdot\|_\varrho$. This is sensible, because it ensures that all nodes stay within the free space and the free space is sufficiently explored. Then $|\,expand(V)| = 3^f - 1$. Moreover, the segment connecting V with a node in $expand(V)$ has a length of at most $\alpha \cdot clearance(V)$ and thus the segment is β-safe with $\beta = 1-\alpha$. Hence, any path computed by Algorithm 1 is also β-safe.

4.2 Weighting Vertices

In our model, we want to determine fast paths, that is *minimum-link paths.* Safe paths can be arbitrarily slow: Figure 1 shows the shortest safe path and a minimum-link safe path from S to T. Remember that for a safe path the length of each segment is bounded by a constant times the clearance of the segment. This is the difference of our approach to the *Voronoi diagram*, where we would use the path through the small bypass instead of the safer (and faster in our terminology) path around the obstacles.

In fact, it is difficult and time-consuming to move through the small bypass in practice, because many consecutive positions must be reached exactly to avoid collision with the triangles. In contrast, the obstacles can be surrounded very easily by a sloppy control of the position. Therefore, it is not sensible to use the length

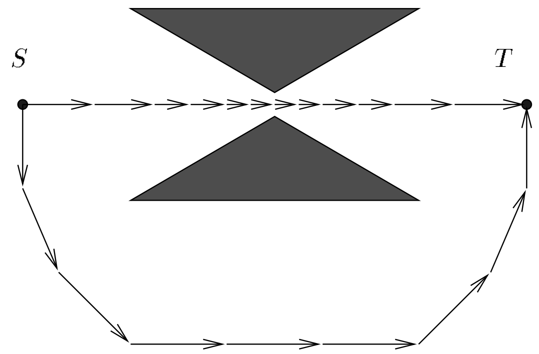

Figure 1: The shortest safe path can be arbitrarily slow compared to the fastest safe path.

of the path or the Euclidean distance to the target T to define the node weights g and h.

To obtain a fastest path, we have to specify $g(V)$ as the number of links the algorithm produced to arrive at V starting in S. Our goal is now to develop an *optimistic* estimation h of the number of remaining links to reach T from V. This will ensure that the A*-algorithm is guaranteed to find a shortest path with respect to the weight w. Let $V^0 = V, V^1, \ldots, V^m = T$ a sequence of consecutive expansions of the algorithm establishing a path from a node V to the goal T. By construction, we have

$$\mathit{clearance}(V^{i+1}) \leq (1+\alpha) \cdot \mathit{clearance}(V^i),$$

and hence due to the choice of $\mathit{expand}(V_j)$

$$\begin{aligned} \|V - T\|_\varrho &\leq \sum_{j=0}^{m-1} \|V_j - V_{j+1}\|_\varrho \\ &\leq \alpha \sum_{j=0}^{m-1} \mathit{clearance}(V^j) \\ &\leq \alpha \sum_{j=0}^{m-1} (1+\alpha)^j \cdot \mathit{clearance}(V) \\ &= \big((1+\alpha)^m - 1\big) \cdot \mathit{clearance}(V). \end{aligned} \tag{3}$$

For an optimistic weight h we set $h(V)$ such that

$$\|V - T\|_\varrho \geq \Big((1+\alpha)^{h(V)} - 1\Big) \cdot \mathit{clearance}(V)$$

and thus

$$
\begin{aligned}
(1+\alpha)^{h(V)} &\geq \frac{\|V-T\|_\varrho}{clearance(V)} + 1 \\
h(V) &\geq \log_{1+\alpha}\left(\frac{\|V-T\|_\varrho}{clearance(V)} + 1\right). \qquad (4)
\end{aligned}
$$

4.3 Improved weight functions

For simplicity, we assume the $\mathcal{C}$-space is bounded ("working cell"). Therefore, the function *clearance* is bounded and we may use as an improved (optimistic) estimation of the remaining effort

$$
\tilde{h}(V) = \frac{\|V-T\|_\varrho}{\text{size of the } \mathcal{C}\text{-space}}. \qquad (5)
$$

The advantage of $\tilde{h}$ is that it is increasing linearly with the distance from T.

In experiments, the following not necessarily optimistic weight function appeared to be extremely efficient:

$$
\hat{h}(V) = \frac{\|V-T\|_\varrho}{clearance(V)}. \qquad (6)
$$

Intuitively, the function prevents the algorithm to run against walls in order to try to find a small passage.

5 Informal Advantages

In this section, we want to present some informal arguments for the practicability of Algorithm 1, which is in principle applicable to general motion planning problems with more than two dofs. Overmars[b] claimed that for multi-link manipulators typically 99% of the $\mathcal{C}$-space is not free. Our method directly takes advantage of a small (in terms of volume) free space, since this greatly reduces the expansions to be made.

Technical constraints, such as safety distances from the obstacles or bounds on feasible and sensible distances between consecutive way-points can be introduced simply. Moreover, feedback for the construction of parts is possible: the minimum clearance to approach welding points may serve as a measure for the difficulty of welding the part.

Compared to the probabilistic planners [4] our method uses less powerful primitives, since no complete planner is necessary to link unconnected portions of the diagram. Also, the explicit construction of configuration space obstacles is not necessary. Even the exact knowledge of the obstacles or the robot is not necessary, as long as we are able to compute the clearance or a conservative estimation of the clearance.

[b]Personal communication.

We can use the computed graph in the free space to accelerate *multiple path queries* by accumulation of the computed approximations of the free space. Clearly, this would increase the space requirements. Let $\mathbf{G}' = (\mathcal{V}', \mathcal{E}')$ be the approximation of the free space already computed during the last search. Rather than starting with an empty graph in Algorithm 1, we add all nodes and edges to $\mathbf{G}$. At the beginning nodes from $\mathcal{V}'$ are marked as closed. If we connect S with a node in $\mathcal{V}'$ then we check whether T is already reachable within $\mathbf{G}$. If this is the case, then we determined a path connecting S and T. Otherwise, we can mark all nodes in $\mathcal{V}'$ which were open in $\mathbf{G}'$ again as open. Of course, the edge weights of these nodes must be reevaluated with respect to S and T. We can thus ensure that the same portion of the free space is not multiply explored.

6 An Experiment

Algorithm 1 has been implemented in its general form to verify and test its performance, that means there are no restrictions of the dimension, the type of the robot and so forth. Only the computation of the clearance and visualizing the obstacles and the robot induces the following restrictions. For the sake of natural representations, the workspace is the Euclidean plane with polygonal obstacles. The robot may consist of a sequence of links which are attached to each other by a rotational degree of freedom. Notice that this description would allow two consecutive links not to be intersecting. Additionally, the first link or the "base" can have a rotational and two translational degrees of freedom. For the case of only one link this models a free-flying rigid robot. Of course, the planner itself is valid also for other types of motion planning problems, but the front end to display the results is restricted as described.

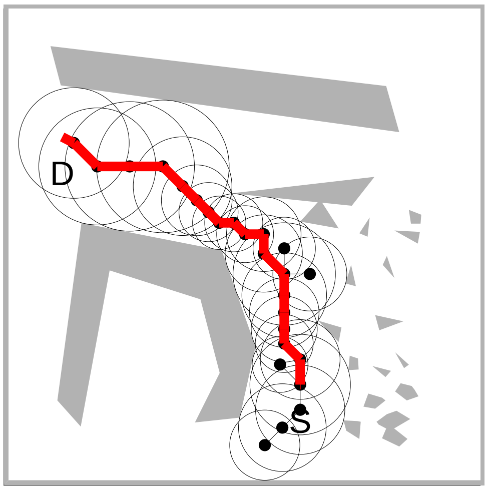

Figure 2: An example of the execution of the algorithm for a point-robot in the plane.

In Figure 2 the execution of the algorithm is visualized for the simple example of a point-robot moving from start S to the goal T with two degrees of freedom. The expanded nodes are illustrated by their respective circles and the finally obtained path is given as a bold line.

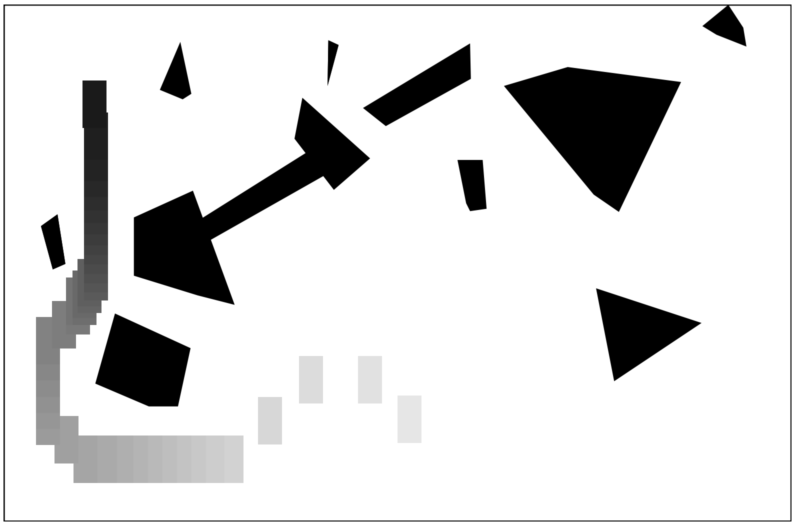

Figure 3: A rigid robot with the shape of a rectangle and only translational degree of freedom. The path length was 41 and the total number of expansions was 229.

Figures 3 and 4 show the computed motion of a free-flying rigid robot with two and three degrees of freedom, respectively. The obstacles are shown as black polygons. In Figure 5, one can see nicely that the algorithm computes the path of the L-shaped robot such that the safety-distance to the obstacles is large.

We also tested the algorithm for *multi-link manipulators*. The results for a 3-dof and 5-dof robot are shown in Figures 6 and 7, respectively. Without any modifications the algorithm got stuck in the last example. To overcome this problem and to achieve the named results, we had to use our additional knowledge that the base of the robot is stationary and the five consecutive links are connected leaving no gap between them. Then, of course, to pass an obstacle the robot must "fold together". This can be included in our weight functions guiding the A*-search e.g. by adding a penalty to the estimation h of the remaining effort depending on the distance of the end-effector from the base.

The example in Figure 8 shows that in fact Algorithm 1 can not do better unless it is provided with the additional information (compiled into the estimation of the remaining effort) that its links are connected without any gaps. The example shows two configurations (solid and dotted) of a multi-link arm, where one link consists of two parts divided by a proper gap. The fastest motion of the arm passing the circular obstacle may just be the one, where the gap of one link slides over the obstacle (without any collision). The algorithm was designed to find the fastest

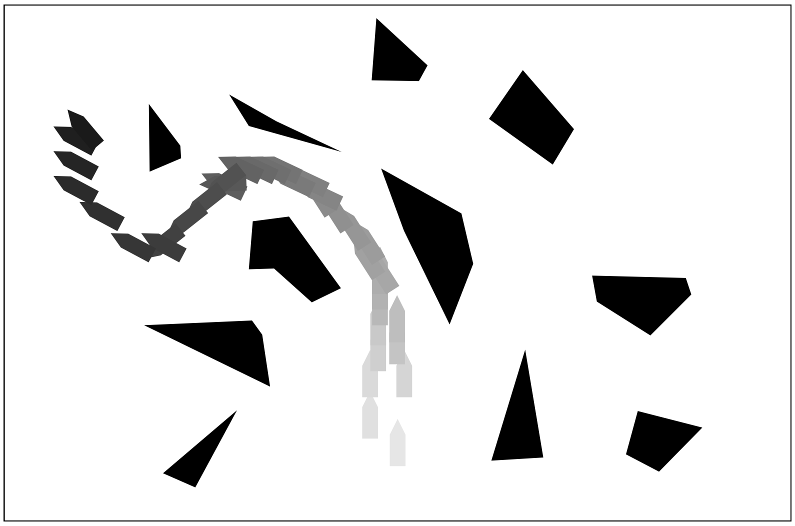

Figure 4: A rigid robot with rotational and translational degrees of freedom. The path length was 36 and the total number of expansions was 419.

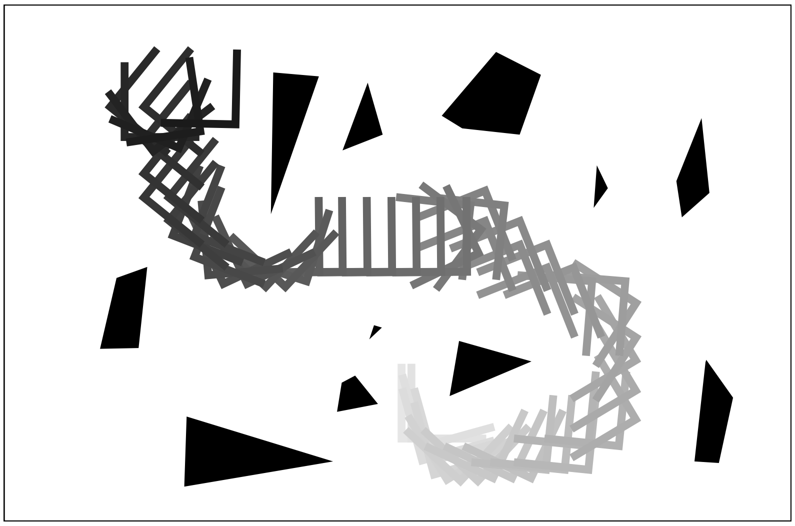

Figure 5: A rigid L-shaped robot with rotational and translational degrees of freedom. The path length was 62 and the total number of expansions was 398.

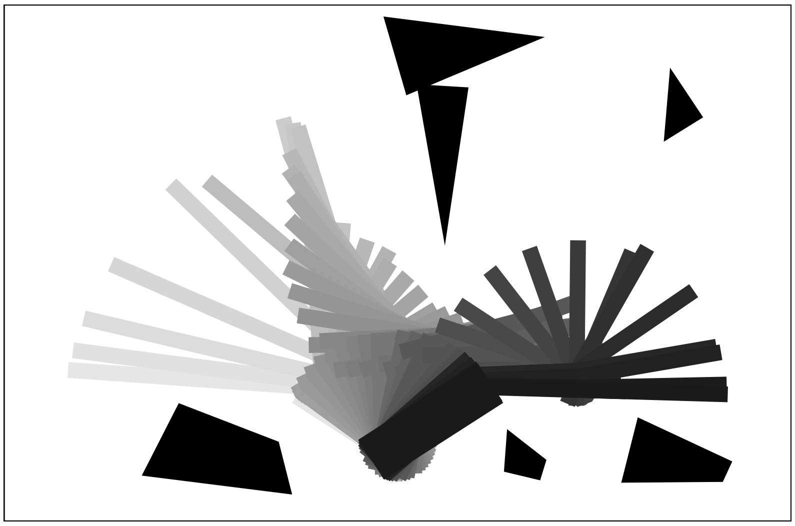

Figure 6: A 3-dof robot consisting of three consecutive links, which may all be rotated. The path length was 81 and the total number of expansions was 6708.

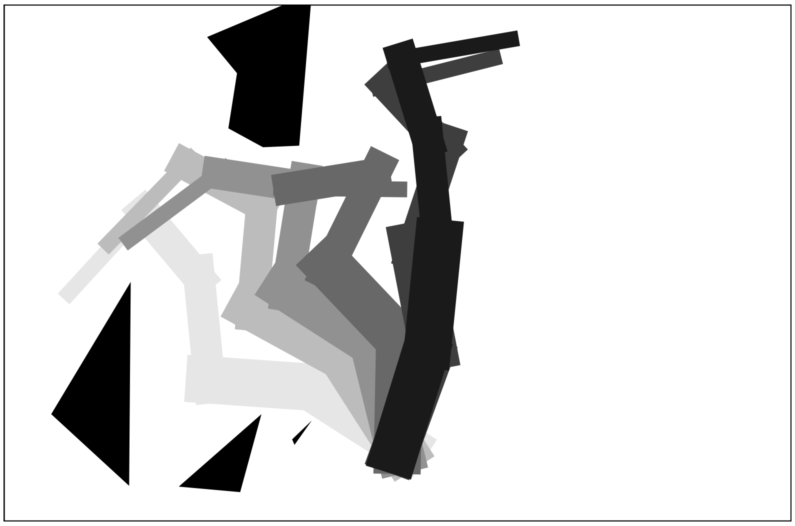

Figure 7: A 5-dof robot consisting of three consecutive links, which may all be rotated. The path length was 141 and the total number of expansions was 211.

motion. Hence, it has to search also many configurations, which do not seem to be profitable for connected manipulators.

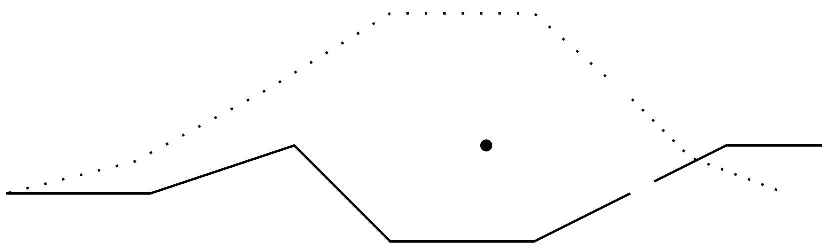

Figure 8: An example of a robot arm with a gap in one of its links.

A more sophisticated approach for simply connected manipulators would be to compute an auxiliary path in three-dimensional workspace only for a reference point on the end-effector subject to the constraint that this path can be continuously deformed into a path including the base of the manipulator. Hence this path may give rise for a feasible motion of the manipulator guided by the end-effector. This problem may be more easy to solve due to the smaller dimension of the space under consideration. Then the function h could be adjusted in order to follow this precomputed path with the entire manipulator.

7 Discussion

We presented an approach to tackle general motion planning problems in high dimensional configuration spaces. The efficiency of our method is confirmed by experiments. It may be possible to estimate the running time of the algorithm and the quality of the path depending on minimal *clearance* and length of a path we assume to exist. This would mean that the algorithm becomes *output-sensitive* with respect to the link-length of the path and sensitive to the difficulty of the problem: The smaller the clearance on a path is, the more effort we may take to compute it.

On the other hand, the relations between workspace and configuration space are still unsettled. Investigations of the volume of the free space or the ε-goodness[4] may give rise what simple configurations are for multi-link manipulators comparable to the assumption of low object density. Possibly, even simple examples may show that the resulting free space is not ε-good for any $\varepsilon > 0$, leaving the question for more adequate assumptions.

Of course, people in computational geometry have to answer the question, whether their methods and models for motion planning are adequate for the applications where the problems originated from. As we have seen in the case of welding robots, we may not be able to use geometric descriptions of the obstacles and the robot to compute a path. Therefore, classical motion planning algorithms are not applicable. On the other hand, welding seems to be relatively simple, since the robots move in the free space without near obstacles unless they enter a point to weld with the end-effector. Therefore, the paths under consideration may be very "simple". Again, it would be interesting to elaborate the properties of these special scenarios.

1. J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.
2. A. F. van der Stappen. *Motion Planning amidst Fat Obstacles.* PhD thesis, University of Utrecht, 1994.
3. J. F. Canny. *The Complexity of Robot Motion Planning.* ACM Doctoral Dissertation Award: 1987. The MIT Press, 1988.
4. Jérôme Barraquand, L. E. Kavraki, J.-C. Latombe, Tsai-Yen Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. In G. Giralt and G. Hirzinger, editors, *Proceedings of the 7th International Symposium on Robotics Research*, pages 249–264. Springer, 1996.
5. L. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
6. L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 353–362, 1995.
7. M. H. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 19–37. A K Peters, 1995.
8. D. Hsu, J.-C. Latombe, and R. Motwani. Path planning and expansive configuration spaces. In *First CGC Workshop on Computational Geometry*, Baltimore, 1996. John Hopkins University. Tech. rep. via http://www.cs.jhu.edu/labs/cgc/cgc_conf.html.
9. P. Švestka. On probabilistic completeness and expected complexity of probabilistic path planning. Technical Report UU-CS-1996-20, University of Utrecht, 1996.
10. B. Braun and R. Corsépius. AMOS: Schnelle Manipulator-Bewegungsplanung durch Integration potentialfeldbasierter lokaler und probabilistischer globaler Algorithmen. In G. Schmidt and F. Freyberger, editors, *Autonome Mobile Systeme 1996*, pages 150–159. Springer Verlag, 1996.
11. N. J. Nilsson. *Principles of Artificial Intelligence.* Springer, Berlin, 1982.
12. D. Henrich and X. Cheng. Fast distance computation for on-line collision detection with multi-arm robots. In U. Rembold and R. Dillmann, editors, *Autonome Mobile Roboter für flexible Fertigungszellen und deren Umgebung*, pages 76–81. Springer, 1993.
13. K. Verbarg and A. Hensel. Hierarchical motion planning using a spatial index. Technical Report 135, University of Würzburg, 1996.
14. B. Mirtich and J. F. Canny. Testing control systems through impulse-based simulation. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry (SCG)*, pages V5–V6, 1996.
15. O. Karch. Hierarchische Approximation von Szenen konvexer und nichtkonvexer Polyeder im $\mathbb{R}^2$ und $\mathbb{R}^3$. Master's thesis, University of Würzburg, 1995.
16. A. P. del Pobil and M. A. Serna. *Spatial Representation and Motion Planning,*

volume 1014 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
17. J. S. B. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles in the plane. *Algorithmica*, 8:431–459, 1992.
18. W. Redinger. Fahrweggenerierung bei Schweißrobotern in der Automobilindustrie (am Beispiel des Seitenteils des Audi AB). Master's thesis, Department of Computer Science, University of Würzburg, 1995.
19. C. W. Warren. Fast path planning using a modified A*-method. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 662–667, 1993.
20. H. Noltemeier, K. Verbarg, and C. Zirkelbach. A data structure for representing and efficient querying large scenes of geometric objects: MB* trees. In G. Farin, H. Hagen, and H. Noltemeier, editors, *Geometric Modelling*, volume 8 of *Computing Supplement*, pages 211–226. Springer, 1993.
21. O. Schwarzkopf and J. Vleugels. Range searching in low-density environments. Submitted to Information Processing Letters, 1996.
22. D. Halperin and M. H. Overmars. Spheres, molecules, and hidden surface removal. In *Proceedings of the 10th Annual ACM Symposium on Computational Geometry (SCG)*, pages 113–122, 1994.