Creating High-quality Paths for Motion Planning

Roland Geraerts and Mark H. Overmars

The International Journal of Robotics Research 2007; 26; 845
DOI: 10.1177/0278364907079280

Roland Geraerts
Mark H. Overmars
Institute of Information and Computing Sciences,
Utrecht University 3508 TB Utrecht, the Netherlands,
{roland,markov}@cs.uu.nl

Creating High-quality Paths for Motion Planning

Abstract

Many algorithms have been proposed that create a path for a robot in an environment with obstacles. Most methods are aimed at finding a *solution. However, for many applications, the path must be of a good quality as well. That is, a path should be short and should keep some amount of minimum clearance to the obstacles. Traveling along such a path reduces the chances of collisions due to the difficulty of measuring and controlling the precise position of the robot. This paper reports a new technique, called Partial shortcut, which decreases the path length. While current methods have difficulties in removing all redundant motions, the technique efficiently removes these motions by interpolating one degree of freedom at a time. Two algorithms are also studied that increase the clearance along paths. The first one is fast but can only deal with rigid, translating bodies. The second algorithm is slower but can handle a broader range of robots, including three-dimensional free-flying and articulated robots, which may reside in arbitrary high-dimensional configuration spaces. A big advantage of these algorithms is that clearance along paths can now be increased efficiently without using complex data structures and algorithms. Finally, we combine the two criteria and show that high-quality paths can be obtained for a broad range of robots.*

KEY WORDS—motion planning, path quality, path length, path clearance

1. Introduction

Motion planning is one of the fundamental problems in robotics. The motion planning problem can be defined as finding a path between a start and goal placement of a robot in an environment with obstacles. During the last 15 years, efficient techniques, such as the Probabilistic Roadmap Method (Kavraki et al. 1996) and the Rapidly-exploring Random Tree (Kuffner and LaValle 2000), have been devised to tackle this problem. However, many techniques generate low quality paths. These paths often contain many unnecessary and jerky motions. Many applications require a short path since redundant motions will take longer to execute. In addition, the path has to keep some amount of minimum clearance from obstacles to reduce the chance of collisions with these obstacles. In this paper, we study several techniques for reducing the path length and for increasing the path clearance.

The International Journal of Robotics Research
Vol. 26, No. 8, August 2007, pp. 845–863
DOI: 10.1177/0278364907079280
©2007 SAGE Publications
Figures 1–11 appear in color online: http://ijr.sagepub.com

1.1. Path Length

The first problem we study is removing redundant motions from a path. A simple technique that decreases the path length is called *Path pruning*. This technique assumes that a path is represented by a list of nodes $v_0, \cdots, v_{n-1} \in V$ which may originate from a graph. For collision checking purposes, the path must be converted to a discrete path Π using a local planner (see Definition 3). The path pruning technique removes a node v_{i+1} from a path Π if the local path $\text{LP}[v_i, v_{i+2}]$ between nodes v_i and v_{i+2} is collision-free.

The most widely applied method is the *Shortcut* heuristic because of its effectiveness and simple implementation. The Shortcut heuristic takes two random configurations on the path. If the part between these two configurations can be replaced by a new shorter path, produced by a local planner, then the original part is replaced by the new path. This technique outperforms the Path pruning heuristic as not only nodes are considered on the path, but also all intermediary configurations. The configurations can be chosen randomly (Chen and Hwang 1998; Geem et al. 1999; Kavraki and Latombe 1998; Sánchez and Latombe 2002; Sekhavat et al. 1998; Švestka 1997) or deterministically (Baginski 1997; Hsu et al. 1999; Isto 2002). Also several variants of this heuristic have been used (Baginski 1997; Berchtold and Glavina 1994; Hsu et al. 1999; Isto 2002; Kavraki and Latombe 1998).

We will show in Section 3 that the Path pruning and Shortcut heuristics will not remove all redundant motions. This is because interpolations are performed between all degrees of freedom (DOFs) simultaneously. We propose the *Partial shortcut* heuristic, which takes only one DOF into account at each optimization step. Experiments will show that this efficient method creates shorter paths than the other two methods.

1.2. Path Clearance

The second problem we study is increasing the clearance along a path. Algorithms that produce paths with high clearance can be divided into two categories. The first category creates a roadmap (or graph) which represents the high-clearance collision-free motions that can be made by the moving object in an environment with obstacles. From this graph a path can be extracted using Dijkstra's shortest path algorithm. Since the calculations to create the high-clearance paths are performed off-line, we refer to this technique as a preprocessing approach. The second category optimizes a given path. The optimization is usually performed on-line in a post-processing stage.

The Generalized Voronoi Diagram (GVD) is a roadmap which can be used to extract high-clearance paths. The GVD (or medial axis) for a robot with n degrees of freedom is defined as the collection of k-dimensional geometric features $(0 \leq k < n)$ which are $(n + 1 - k)$-equidistant to the obstacles. As an example, consider Figure 1 which shows a bounding box and a part of the medial axis for a translating robot. The medial axis of this robot consists of a collection of surfaces, curves and points. The surfaces are defined by the locus of 2-equidistant closest points to the bounding box. The curves have 3-equidistant closest points and the points have 4-equidistant closest points to the bounding box. These features are connected if the free space in which the robot operates is also connected (Choset and Burdick 2000). Hence, the GVD is a complete representation for motion planning purposes. Most importantly, paths on the GVD have appealing properties such as large clearance from obstacles.

Wein et al. (2005) introduce a hybrid between the visibility graph and the Voronoi diagram of polygons in the plane. A shortest path with a preferred minimum amount of clearance can be extracted in real time.

Unfortunately, an exact computation of the GVD is not practical for problems involving many degrees of freedom (DOFs) and many obstacles as this requires an expensive and intricate computation of the configuration space obstacles. Therefore, the GVD is approximated in practice.

Vleugels and Overmars (1998) approximate the GVD by applying spatial subdivision and isosurface extraction techniques. Although the calculations are easy and robust, and they can be generalized to higher dimensions, the technique only works for disjoint convex sites and consumes an exponential amount of memory, making this technique impractical for problems involving many DOFs. Another approach, proposed by Masehianand et al. (2003), incrementally constructs

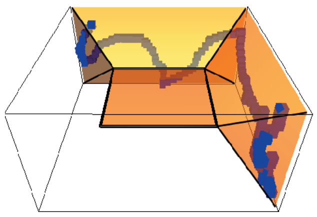

Fig. 1. Part of the medial axis of an environment that consists only of a bounding box.

the GVD by finding the maximal inscribed disks in a two dimensional discretized workspace. Although this algorithm is also extensible to handle higher-dimensional problems, it suffers from the same drawback as the preceding algorithm. Hoff et al. (2000) describe a technique that exploits the fast computation of a GVD using graphics hardware for motion planning in complex static and dynamic environments. However, the technique is limited to a three-dimensional workspace for rigid translating robots.

Kim et al. (2003) use an augmented version of Dijkstra's algorithm to extract a path from a graph (which does not have to be a GVD) based on other criteria than length. The minimum clearance along the path is maximized by incorporating a higher cost for edges that represent a small amount of clearance. Such a path rarely provides an optimal solution because it is restricted to the randomly generated nodes in the roadmap. Even if the nodes are placed on the medial axis (Lien et al. 2003), the edges will in general not lie on the medial axis, and hence, the extracted path does not have an optimal amount of clearance.

The above preprocessing methods create a data structure from which paths can be extracted. Brock and Khatib (2002) present a (post-processing) framework that provides an efficient method for performing local adjustments to a path in dynamic environments. This path is represented as an elastic band. Subjected to artificial forces, the elastic band deforms in real time to a short and smooth path that maintains clearance from the obstacles. The method can be applied to a broad range of robots, but many parameters have to be set to get the framework running. It is also not clear whether the resulting path will and can have an optimal amount of clearance.

In this paper, we will study techniques to improve the clearance along a given path. First, we will provide some preliminaries in Section 2. After dealing with creating short paths in Section 3, we describe in Section 4 our first algorithm that adds clearance to a path by retracting it to the medial axis of the workspace. The algorithm is limited to translating, rigid bodies. Although it provides optimal clearance paths for rigid, translating bodies in the plane, the paths may not be optimal for robots operating in a three-dimensional environment. We remove these limitations by presenting an algorithm that provides high-clearance paths for a broad range of robots residing in arbitrary high-dimensional configuration spaces. We apply these algorithms to six different environments and conclude that clearance along paths can be increased without using complex data structures and algorithms.

Finally, we show that short paths having some minimal amount of clearance can be easily created by combining the techniques. The results can be used, for example, to obtain high-quality paths in high-cost environments such as a factory in which a manipulator arm operates.

2. Preliminaries

As this paper deals with improving the robot's clearance along a path, we need a way to compute the clearance of the robot to the obstacles. This calculation is delegated to our collision checker Solid (van den Bergen 2003). We define the clearance of a configuration as follows:

Definition 1 [Clearance of a configuration π] Let R be the set of all points on the robot whose placement in the environment corresponds to configuration π. Furthermore, let O be the set of all points on all obstacles in the environment. Then the clearance of configuration π is the shortest Euclidean distance between r and o: $r \in R \wedge o \in O$.

To define a path, we need the following definition of adjacent configurations.

Definition 2 [Adjacent configurations] The configurations $\pi_0, \cdots, \pi_{n-1}$ are adjacent if the distance $d(\pi_i, \pi_{i+1})$ is at most a predetermined distance *step*.

The step size is chosen dependent on the robot and obstacles. We will describe below how to compute distances. We can now define a *discrete path*, *discrete local path*, and a *node path*.

Definition 3 [Discrete path Π] A *discrete path* Π is a series of n adjacent configurations $\pi_0, \cdots, \pi_{n-1}$.

Definition 4 [Discrete local path LP] A *discrete local path* $\text{LP}\left[\pi', \pi''\right]$ is a series of n interpolated adjacent configurations $\pi_0, \cdots, \pi_{n-1}$ on the local path between π' and π''.

Definition 5 [Node path N] A *node path* N is a series of n nodes $\nu_0, \cdots, \nu_{n-1}$ such that the local paths $\text{LP}[\nu_i, \nu_{i+1}]$ are collision-free.

The average clearance of a path gives an indication of the amount of free space in which the path can be moved without colliding with the obstacles:

Definition 6 [Average clearance of discrete path Π] Let Π be a discrete path. Then the *average clearance* is given by $\frac{1}{n}\sum_{i=0}^{n-1} \text{Clearance}(\pi_i)$.

3. Path Length

We study three techniques to decrease path length. In Section 3.1, we study the *Path pruning* technique which considers all nodes of the path. Then, we study the *Shortcut* technique in Section 3.2 which considers all configurations on the path. While this technique performs interpolations between all DOFs simultaneously, the *Partial shortcut* technique, described in Section 3.3, takes only one DOF into account in each optimization step. Finally, we conduct experiments in Section 3.4.

3.1. Path Pruning

In this section, we assume that a path is represented by a list of nodes, see Definition 5. As these nodes are often generated randomly, the path will be jerky. A very simple technique that decreases the path length considerably is to remove all redundant nodes. A node ν_{i+1} on node path N is redundant if the configurations on the local path $\text{LP}[\nu_i, \nu_{i+2}]$ are collision-free, i.e. $\text{LP}[\nu_i, \nu_{i+2}]$ is in the free space $\mathcal{C}_{\text{free}}$. Besides being simple, the technique is efficient and deterministic. See Algorithm 1 for more details.

Algorithm 1 REMOVEREDUNDANTNODES(node path N)

```
1: $i \leftarrow 0$
2: while $i < |N| - 2$ do
3:     if $\text{LP}[\nu_i, \nu_{i+2}] \in \mathcal{C}_{\text{free}}$ then
4:         $N \leftarrow N \backslash \nu_{i+1}$
5:         if $i > 0$ then $i \leftarrow i - 1$
6:     else
7:         $i \leftarrow i + 1$
8: return $N$
```

3.2. Shortcuts

While the previous method only considers the nodes of a path, the shortcut method considers all configurations on a discrete path Π (see Definition 3). Therefore, this method is expected to create shorter paths at the cost of increased computation time. The method tries to iteratively improve the path (see Algorithm 2). In each iteration, path Π is randomly split in three parts. Let π_a and π_b denote the begin and end configurations of the middle part. If the local path $\text{LP}(\pi_a, \pi_b)$ is collision-free, then this local path replaces the middle part. As we use a straight-line local planner, all DOFs are interpolated simultaneously. When another local planner is used, e.g. a local planner for non-holonomic robots, care has to be taken that path Π' and Π'' keep satisfying the constraints of the local planner.

Algorithm 2 SHORTCUT(discrete path Π)

```
1: loop
2:   number of configurations $n \leftarrow |\Pi|$
3:   $a, b \leftarrow$ two random indices $0 \leq a + 1 < b < n$
4:   $\Pi' \leftarrow \pi_0, \cdots, \pi_{a-1}$
5:   $\Pi'' \leftarrow \pi_a, \cdots, \pi_b$
6:   $\Pi''' \leftarrow \pi_{b+1}, \cdots, \pi_{n-1}$
7:   if LP$(\pi_a, \pi_b) \in \mathcal{C}_{\text{free}}$ then
8:     $\Pi \leftarrow \Pi' \cup$ LP$(\pi_a, \pi_b) \cup \Pi'''$
```

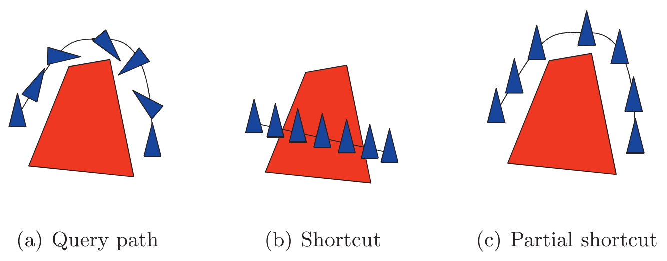

Fig. 2. Translation is required to navigate around the obstacle and rotation can only be optimized by considering large portions of the path.

3.3. Partial Shortcuts

Redundant motions (like unnecessary rotations) will not be removed by the previous two heuristics as they can only be removed by considering large portions of the path. But if we consider such a large portion, some other degrees of freedom (DOFs) are necessary to navigate around obstacles. Hence, applying the local planner to such a long portion is not going to succeed (see Figure 2).

The standard optimization technique (Shortcut) replaces pieces of the path by a straight-line segment in the configuration space. In this way, all DOFs are interpolated simultaneously. Some of them might be necessary to move around the obstacles while others are not. The translational DOFs in particular are often necessary to guide the object around an obstacle while the rotational DOFs might be less relevant. Consequently, applying the local planner on such a part of the path will fail. Applying the local planner to optimize shorter pieces of the path will not remove the redundant rotations either because the two positions on the path will often have rather different orientations. Therefore, the rotation is required locally, while more globally, it might be redundant (see Figure 2).

We created a new technique, called *Partial Shortcut*, which takes only one DOF f into account in each optimization step. Algorithm 3 outlines the technique. In line 2, the chance that a particular DOF is chosen is dependent on its weight, i.e. $P(\text{DOF } i) = w_i / \sum_{i=0}^{n-1} w_i$. In this expression, we consider rotation in 3D as one DOF. Then, we split path Π in the same way as we did in the Shortcut algorithm. (We assume that there are no constraints which the configurations on the path have to satisfy.) Now let $\pi''_i[f]$ indicate the value for the f^{th} DOF of configuration π''_i of path Π''. We replace in each configuration π''_i the value of the f^{th} DOF by the value interpolated between $\pi''_0[f]$ and $\pi''_{m-1}[f]$, where m is the number of configurations on path Π''. After creating partial shortcuts, it can occur that the distance between two adjacent configurations on path Π'' is larger than the step size. In such a case, we validate Π'' by inserting extra configurations such that $\forall i : d(\pi_i, \pi_{i+1}) \leq step$. If path Π'' is collision-free, then Π'' replaces the original middle part. In this path all DOFs behave in the same way as in the original path except for DOF f.

We expect that this method will be slower than the Shortcut heuristic as only one DOF is taken into account in each iteration. However, we expect that more redundant motions can be removed.

Algorithm 3 PARTIALSHORTCUT(discrete path Π)

```
1: loop
2:   $f \leftarrow$ a random degree of freedom
3:   number of configurations $n \leftarrow |\Pi|$
4:   $a, b \leftarrow$ two random indices: $0 \leq a + 1 < b < n$
5:   $\Pi' \leftarrow \pi_0, \cdots, \pi_{a-1}$
6:   $\Pi'' \leftarrow \pi_a, \cdots, \pi_b$
7:   $\Pi''' \leftarrow \pi_{b+1}, \cdots, \pi_{n-1}$
8:   $m \leftarrow |\Pi''|$
9:   for all $\pi''_i \in \Pi''$ do
10:    $\pi''_i[f] \leftarrow$ INTERPOLATE$(\pi''_0[f], \pi''_{m-1}[f], i/(m-1))$
11:  VALIDATEPATH$(\Pi'')$
12:  if $\Pi'' \in \mathcal{C}_{\text{free}}$ then
13:    $\Pi \leftarrow \Pi' \cup \Pi'' \cup \Pi'''$
```

3.4. Experiments

In this section, we apply the three techniques to six different paths which have been created by the Probabilistic Roadmap Method (Kravaki and Latombe 1998; Kravaki et al. 1996) (which took up to 1.2 seconds). These paths have been selected such that an optimization step cannot easily change the homotopic class of a path. Our goal is to investigate the extent to which these techniques can improve the paths.

3.4.1. Experimental Setup

All techniques were integrated in our motion planning system called SAMPLE (System for Advanced Motion PLanning Experiments), implemented in Visual C++ under Windows XP.

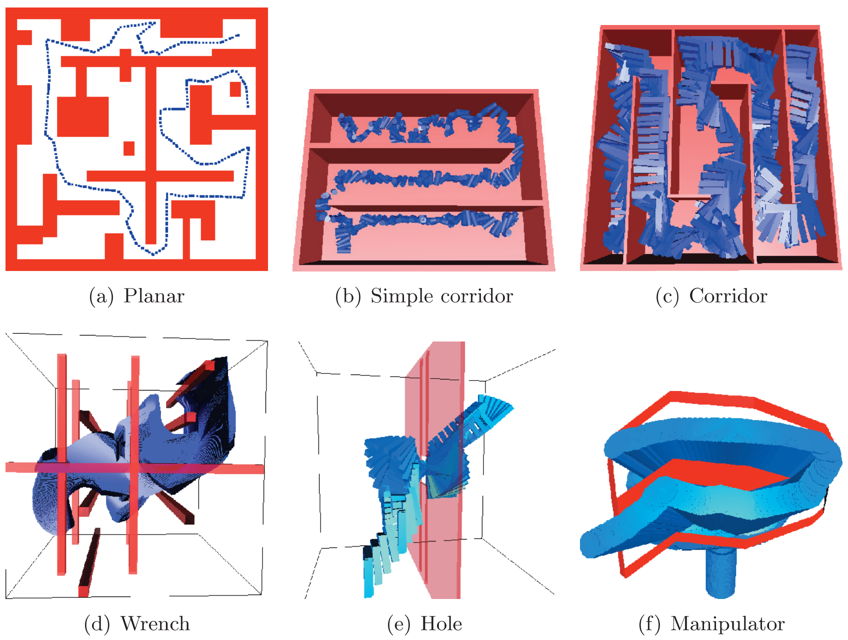

Fig. 3. The six test environments and their corresponding *initial* paths.

SAMPLE automates conducting experiments, i.e. statistics are automatically generated and processed, decreasing the chance of errors. All experiments were run on a 3 GHz Pentium 4 processor with 1 GB internal memory. We used Solid as basic collision checking package (van den Bergen 2003). The statistics were all averaged over 100 independent runs.

To test the quality of the three techniques, we considered the environments depicted in Figure 3. Their properties are stated in Table 1. The step sizes are based on the environments and robots. When these values are very high, it can occur that collisions are not being detected. However, if they are very small, the running times of the algorithms will increase. We conducted preliminary experiments to find reasonable values.

Table 1. The axis-aligned bounding boxes of the environments and robots, and the step sizes for the robots.

	Dimensions of the bounding box		step size
	environment	**robot**	
Planar	100 × 100	1 × 1	1.0
Simple corridor	40 × 11 × 30	0.2 × 0.2 × 0.75	0.4
Corridor	40 × 17 × 40	5 × 1 × 5	0.7
Wrench	160 × 160 × 160	68 × 24 × 8	3.0
Hole	40 × 40 × 40	5 × 5 × 10	1.0
Manipulator	10 × 10 × 10	variable	0.1

Planar This simple two-dimensional environment contains a path traversed by a square robot that can only translate in the plane. We use this environment to check whether the techniques can reach the optimal solution.

Simple corridor This simple three-dimensional environment with ample free space to maneuver features a jerky path traversed by a small free-flying cylinder. Many motions are redundant. We expect that they can be removed easily by all techniques.

Corridor The environment consists of a winding corridor that forces a free-flying elbow-shaped robot to rotate. Redundant rotations can only be removed by considering large portions of the path. Hence, we expect that the Partial shortcut technique will out perform the other techniques.

Wrench This environment features a fairly large free-flying object (wrench) in a workspace that consists of 13 crossing beams. The wrench is rather constrained at the start and goal positions. In contrast to the previous three environments, rotational DOFs are now more important than translational ones. Again, we expect that the Partial shortcut technique outperforms the other techniques as this technique handles each DOF independently.

Hole The free-flying robot, which has six DOFs, consists of four legs and must rotate in a complicated way to get through the hole. The path contains many redundant (rotational) motions. Only where the robot passes through the hole, the clearance is small, which may cause difficulties in removing the redundant rotational part of the motions for the Path pruning and Shortcut methods.

Manipulator The articulated robot has six rotational DOFs and operates in a constrained environment. In this environment, there is a major difference in importance of the six rotational DOFs. That is, the link that is closest to the base is more important than the gripper of the manipulator. As only the Partial shortcut technique recognizes this difference, we again expect this technique to outperform the others.

We need a distance measure to discuss path length. The importance of choosing a good distance metric is discussed in Amato et al. (1998). Such a metric often incorporates weights (w_i) which are chosen such that a small displacement of a configuration in $\mathcal{C}$-space leads to a small displacement of the corresponding placement of the robot in the workspace. We distinguish three types of DOFs: translation, rotation(1) (rotation about the x-, y-, or z-axis) and rotation(3) (rotation in S^3). For example, a free-flying robot can be described by three translational DOFs and one rotational(3) DOF, and an articulated robot with six joints can be described by six rotational(1) DOFs. Clearly, the rotational DOFs need to have larger weights than the translational ones. The weights used for the different environments are listed in Table 2. These weights are also used in the Partial shortcut technique.

As we want to distinguish between rotational and translational DOFs, we compute the length of a discrete path Π as follows:

$$\begin{aligned} d(\Pi) &= d_r(\Pi) + d_t(\Pi), \quad \text{where} \\ d_r(\Pi) &= \sum_{i=0}^{n-2} d_r(\pi_i, \pi_{i+1}) \quad \text{and} \\ d_t(\Pi) &= \sum_{i=0}^{n-2} d_t(\pi_i, \pi_{i+1}). \end{aligned}$$

Let $q = \pi_i$ and $r = \pi_{i+1}$. Then, for all k rotational DOFs $0 \leq j < k$ and for all $(l-k)$ translational DOFs $k \leq j < l$:

$$\begin{aligned} d_r(q,r) &= \sqrt{\sum_{j=0}^{k-1} [w_j d(q_j, r_j)]^2} \quad \text{and} \\ d_t(q,r) &= \sqrt{\sum_{j=k}^{l-1} [w_j d(q_j, r_j)]^2}. \end{aligned}$$

Table 2. The weights for each DOF of the robots.

	Type of DOF of the robot		
	translational	**rotational(1)**	**rotational(3)**
Planar	1, 1		
Simple corridor	1, 1, 1		3
Corridor	1, 1, 1		7
Wrench	6, 6, 6		30
Hole	1, 1, 1		11
Manipulator		6, 6, 6, 2, 2, 2	

The calculation of distance $d(q_j, r_j)$ is dependent on the type of the DOF:

- For translation, we set $d(q_j, r_j)$ to $|q_j - r_j|$.
- We split the calculation for a rotational(1) DOF in two parts: if the range is smaller than 2π, which often occurs for revolute joints in manipulator arms, we take the same distance measure as for a translational DOF. If the rotational DOF is periodic, i.e. the orientation at 0 radians equals the orientation at 2π radians, we take the smallest angle. More formally, we set $d(q_j, r_j)$ to $|q_j - r_j|$ if r is not periodic, otherwise $d(q_j, r_j) = \min\{|q_j - r_j|, q_j - r_j + 2\pi, r_j - q_j + 2\pi\}$.
- We use unit quaternions to represent rotations in 3D. The distance between two quaternions $q_j(x, y, z, w)$ and $r_j(x, y, z, w)$ can be calculated by taking the shortest angle over a 4D-sphere, i.e. $d(q_j, r_j) = \min\{2\arccos(q_j \cdot r_j), 2\pi - 2\arccos(q_j \cdot r_j)\}$. The dot product $q_j \cdot r_j$ is defined as $q_j \cdot r_j = q_j.x * r_j.x + q_j.y * r_j.y + q_j.z * r_j.z + q_j.w * r_j.w$.

We express path length as a percentage relatively to the 'optimal' path length to facilitate the comparison between different optimization techniques. Let d be the path length and d_{opt} be the optimal path length. Then we calculate the percentage Δd as $100\% * \frac{d - d_{opt}}{d_{opt}}$. The closer this number approaches zero, the closer to optimal the path is. The optimal path lengths were defined as the paths of minimum length over all experiments conducted and are stated in Table 3 and depicted in Figure 4. Even though we cannot guarantee that these are indeed optimal, visual inspection strongly suggests that they are very close to optimal. Table 4 shows the initial relative lengths. The paths are far from being optimal. For example, the path in the Simple corridor environment is 408% longer than the shortest path encountered in all experiments with this environment.

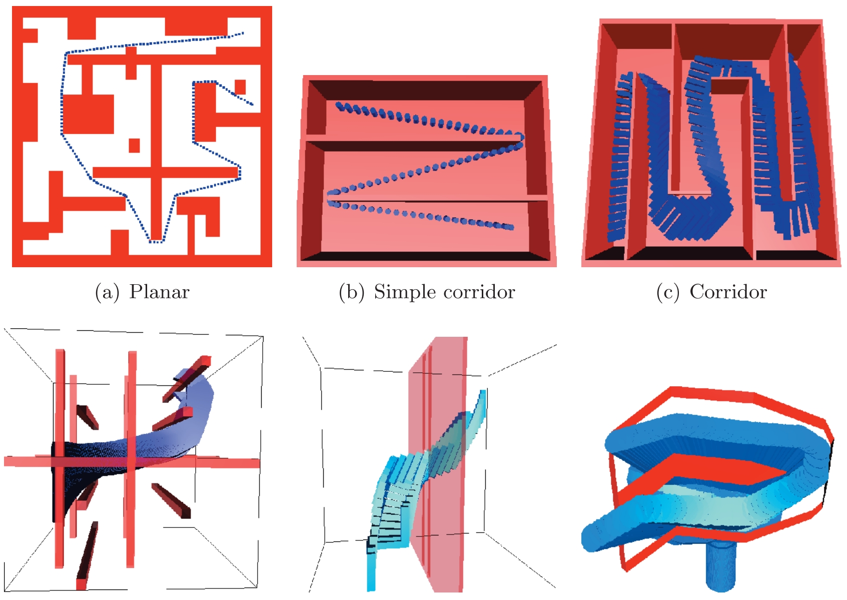

Fig. 4. The six test environments and their corresponding *optimal* paths.

Table 3. The shortest absolute lengths of the paths.

	Shortest path length		
	$d_{r_{opt}}$	$d_{t_{opt}}$	d_{opt}
Planar	–	300.12	300.12
Simple corridor	0.12	100.84	100.96
Corridor	19.31	162.59	181.90
Wrench	827.63	138.99	966.62
Hole	6.87	36.76	43.63
Manipulator	10.73	–	10.73

Table 4. Relative length statistics of the initial paths. The numbers are expressed as percentages relatively to the optimal path lengths.

	Relative path length		
	Δd_r	Δd_t	Δd
Planar	–	40	40
Simple corridor	213,917	154	408
Corridor	1,296	132	256
Wrench	113	112	113
Hole	628	61	150
Manipulator	55	–	55

We investigate the extent to which the three heuristics can improve the six paths from Figure 3. We run the Path pruning heuristic once for each experiment as this technique is deterministic. We use these paths as input for the Shortcut and Partial shortcuts heuristics. As these techniques are non-deterministic, we run them 100 times for each experiment and report the average results. To ensure the exploitation of the full potential of the heuristics, we run each non-deterministic experiment for 120 seconds as preliminary experiments showed that all paths converge within this time.

In the second batch of experiments we determine how long the Shortcut and Partial shortcut heuristics should be applied to obtain reasonably short paths. This is useful for practical purposes.

3.4.2. Experimental Results

The results of the experiments are stated in Table 5. This table shows the relative path length (rotational, translational and total length) for the initial paths and the three heuristics.

The table shows that the Path pruning heuristic improved the paths considerably in all cases. Note that the rotational distance (Δd_r) is still far from optimal, although the translational distance (Δd_t) has been decreased considerably. The running times of this deterministic heuristic were between 5 and 54 ms. The Shortcut heuristic was able to decrease the path length even more, i.e. the paths obtained a length that was 3–28% larger than the optimal paths. However, the paths still contained many redundant rotational motions. The Partial shortcut heuristic was much better able to remove the redundant (rotational) motions than the previous heuristics since only one

Table 5. Relative length statistics of the resulting paths. The numbers are expressed as percentages relatively to the optimal path lengths. The closer a number approaches zero, the closer to optimal it is.

Planar	Relative path length		
	Δd_r	Δd_t	Δd
Initial	–	40	40
Path pruning	–	15	15
Shortcut	–	3	3
Partial shortcut	–	1	1

Simple corridor	Relative path length		
	Δd_r	Δd_t	Δd
Initial	213,917	154	408
Path pruning	15,817	10	28
Shortcut	11,633	3	17
Partial shortcut	383	1	1

Planar	Relative path length		
	Δd_r	Δd_t	Δd
Initial	1,296	132	256
Path pruning	326	34	65
Shortcut	133	8	21
Partial shortcut	35	4	7

Wrench	Relative path length		
	Δd_r	Δd_t	Δd
Initial	113	112	113
Path pruning	71	71	71
Shortcut	28	28	28
Partial shortcut	3	3	3

Hole	Relative path length		
	Δd_r	Δd_t	Δd
Initial	628	61	150
Path pruning	462	17	87
Shortcut	155	0	25
Partial shortcut	27	0	5

Manipulator	Relative path length		
	Δd_r	Δd_t	Δd
Initial	55	–	55
Path pruning	45	–	45
Shortcut	8	–	8
Partial shortcut	3	–	3

DOF was interpolated at a time. In addition (preliminary experiments showed that) using different weights sped up the convergence. Finally, the translational lengths of the paths became close to optimal.

We will now examine the results more closely for each environment.

Planar Both the Shortcut and Partial shortcut techniques reached the optimal solution. The latter one produced paths that were on average only 1% larger than the optimal path.

Simple corridor The initial path contained many redundant (rotational) motions which could not be removed completely by the Path pruning and Shortcut heuristics. However, the Partial shortcut technique was able to produce paths that are very close to the optimal path as only one DOF is optimized during an iteration of the algorithm. Note that the relative rotational path length seems to be very large while the total relative path length was only 1. This is because the optimal (absolute) rotational distance was very close to zero (0.12).

Corridor Also in this environment, the Partial shortcut heuristic outperformed the other techniques. A large number of the redundant rotational motions were removed as large portions of the path could be replaced by less redundant motions.

Wrench The optimal path corresponds to a smooth motion traversed by the wrench. Again, the Partial shortcut heuristic was able to produce such a path as the resulting paths were only 3% worse than the optimal path.

Hole All techniques removed the redundant translational motions, i.e. the translational relative path lengths for the Shortcut and Partial shortcut heuristics were only 0.49% and 0.33%. However, it was difficult to remove the rotational motions as the moving object was rather constrained near the hole. However, the Partial shortcut heuristic obtained a path that was on average 5% longer than the optimal path.

Manipulator The Shortcut and Partial shortcut methods created short paths which are comparable to the optimal path depicted in Figure 4. However, the latter one outperformed the Shortcut method and produced paths that were on average only 3% larger than the optimal path.

In our experiments, we ran the heuristics for 120 seconds as we wanted to see their full potential. In all cases, the Partial

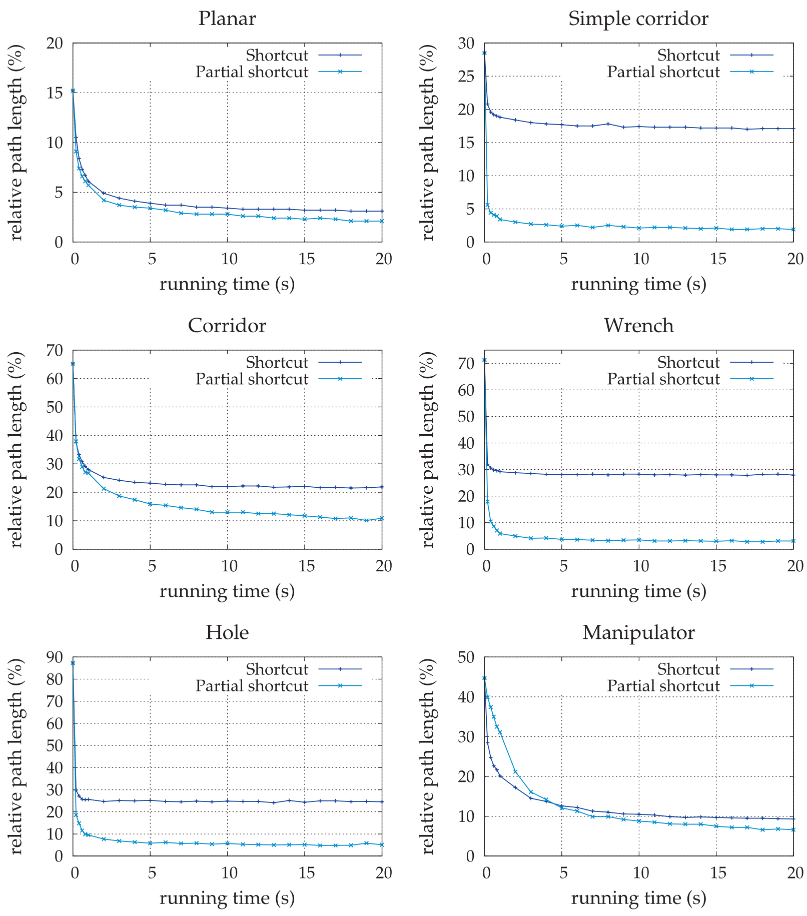

Fig. 5. Convergence of the Shortcut and Partial shortcut heuristics in the six environments.

shortcut heuristic outperformed the Shortcut heuristic. Hence, when optimal path quality is desired (in terms of path length), the Partial shortcut algorithm should be used.

However, the running times may be too high for on-line use. An important question is how well the heuristics perform when there is less computation time available. Figure 5 shows for each environment the relationship between the running time and the relative path length of both the Shortcut and Partial shortcut heuristics. Each marker in the graphs represents the averaged relative path length over 100 independent runs. In all but one environment (Manipulator), the Partial shortcut heuristic always outperforms the Shortcut heuristic. Therefore, the Partial shortcut heuristic should be preferred. Furthermore, it can be observed that the relative path length decreases rapidly as the available computation time increases. When the path is relatively simple (such as in the Planar, Simple corridor, Wrench and Hole environment), only one second of computation time is required to obtain a path that is about 5% longer than the optimal path. For more complex paths (such as in the Corridor and Manipulator environment), the paths obtained after one second are about 25% longer than the optimal path.

We conclude that reasonably short paths can be obtained for all tested environments when the (Partial shortcut) algorithm is run for one second.

4. Path Clearance

We study two techniques to increase the clearance along a path. In Section 4.1, we propose the $\mathcal{W}$-RETRACTION technique. This technique is fast, but can only deal with rigid, translating bodies. In Section 4.2, we propose the $\mathcal{C}$-RETRACTION technique. This technique is slower but can handle a broader range of robots. Finally, we conduct experiments in Section 4.3 and show in Section 4.4 that high-quality paths can be obtained which have some minimum amount of clearance while being short.

4.1. Rigid, Translating Bodies

In this section, we describe an algorithm that adds clearance to a path traversed by a translating, rigid body. The problem we want to solve is as follows. Convert a given discrete path Π into a path Π' such that each robot placement that corresponds to $\pi'_i \in \Pi'$ has (at least) two-equidistant nearest points to the obstacles in the environment. We initially assume that the start and goal configurations lie on the medial axis.

We will increase the clearance along a path by retracting its individual placements of the robot (which we refer to as *samples*) to the medial axis of the free workspace. Our approach is based on a technique from Wilmarth et al. (1999) which retracts samples to the medial axis.[1] Such a retracted sample will have (at least) two-equidistant nearest points to the obstacles in the workspace, resulting in a large clearance. As the retraction is performed in the workspace, only the clearance along paths traversed by translating, rigid bodies can be improved.

4.1.1. Retraction Algorithm

We first show how to retract a single sample, corresponding to configuration $\pi \in \Pi$, to the medial axis. Algorithm 4 outlines our approach. Let cp_π be the point on the robot in the workspace that corresponds to configuration π that is closest to the point cp_o on an obstacle in the workspace. We first calculate the pair (cp_π, cp_o) of closest points between the robot and obstacles. (This calculation is delegated to our collision checker.) Then, we iteratively move in direction $\overrightarrow{cp_o cp_\pi}$ until the closest point on the obstacles changes. In each iteration, the largest distance it can move such that the robot will not collide with the obstacles equals its clearance which is defined as the Euclidean distance between cp_π and cp_o. Finally, we use binary search between the original closest point cp_o and changed closest point $cp_{o'}$ (with precision $step$) to find the configuration π_{mid} that has two-equidistant nearest points to the obstacles in the workspace.

Algorithm 4 RETRACTCONFIGURATION(configuration π)

```
Require: free configuration $\pi$, obstacles $O$, precision $step$
 1: $(cp_\pi, cp_o) \leftarrow$ CLOSESTPAIR($\pi$, $O$)
 2: $cp_{o'} \leftarrow cp_o$
 3: while $cp_{o'} = cp_o$ do
 4:     $\pi' \leftarrow \pi$
 5:     $\pi \leftarrow \pi + cp_\pi - cp_o$
 6:     $(cp_\pi, cp_{o'}) \leftarrow$ CLOSESTPAIR($\pi$, $O$)
 7: while $d(\pi, \pi') > step$ do
 8:     $\pi_{mid} \leftarrow$ INTERPOLATE($\pi$, $\pi'$, 0.5)
 9:     $(cp_\pi, cp_o) \leftarrow$ CLOSESTPAIR($\pi_{mid}$, $O$)
10:     if $cp_{o'} = cp_o$ then $\pi \leftarrow \pi_{mid}$ else $\pi' \leftarrow \pi_{mid}$
11: return $\pi_{mid}$
```

1. While their technique retracts single samples to the medial axis, our technique retracts a complete path.

Algorithm 5 $\mathcal{W}$-RETRACTION(discrete path Π)

```
1: retracted path $\Pi' \leftarrow \emptyset$
2: for all $\pi_i \in \Pi, 0 \leq i < n$ do
3:     $\pi' \leftarrow$ RETRACTCONFIGURATION($\pi_i$)
4:     $\pi_r \leftarrow$ the last configuration of path $\Pi'$
5:     if $d(\pi_r, \pi') > step$ then
6:         $\Pi' \leftarrow \Pi' \cup \mathcal{W}$-RETRACTION(LP $[\pi_r, \pi']$)
7:     $\Pi' \leftarrow \Pi' \cup \pi'$
8: return $\Pi'$
```

Algorithm 5 shows how to retract a discrete path Π to the medial axis. We retract each configuration $\pi \in \Pi$ to the medial axis. If the distance between two consecutive configurations of the retracted path Π' exceeds $step$, we generate extra configurations by applying the algorithm onto the local path that is defined by these two configurations until the distance between any two consecutive configurations is less than $step$.

Algorithm 5 will only work correctly when the start configuration π_0 and/or goal configuration π_{n-1} lie on the medial axis. If not, the retracted path is concatenated with the local path LP$[\pi_0, \pi'_0]$ and/or local path LP$[\pi'_{n-1}, \pi_{n-1}]$.

The path will now follow the medial axis. As an example, we applied the algorithm on a square translating in a 2D environment. We took this environment from Lavalle's Motion Strategy Library (LaValle 2006). The first picture in Figure 6 shows the original path. The retracted path is visualized in the second picture. As we can see, the moving object sometimes traverses the same position twice. This detour is caused by the injective mapping of configurations and can be detected by looking for reversals in a sub-branch of the path. Algorithm 6 removes those redundant branches in linear time in $|\Pi|$. For each triple $\{\pi_{i-1}, \pi_i, \pi_{i+1}\}$, we remove π_i if the distance between π_{i-1} and π_{i+1} is smaller than $step$. Figure 6(c) shows the resulting path following the medial axis without traversing a sub-branch twice. This path was computed within one second.

Algorithm 6 REMOVEBRANCHES(discrete path Π)

```
1: $i \leftarrow 1$
2: while $i < |\Pi| - 1$ do
3:     if $d(\pi_{i-1}, \pi_{i+1}) < step$ then
4:         $\Pi \leftarrow \Pi \backslash \pi_i$
5:         if $i > 1$ then $i \leftarrow i - 1$
6:     else $i \leftarrow i + 1$
7: return $\Pi$
```

4.2. Robots with Many Degrees of Freedom

The retraction method from the previous section provides an accurate retraction of paths for rigid, translating bodies to the

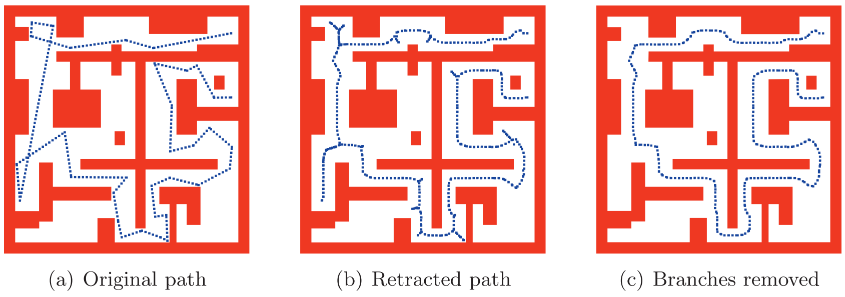

Fig. 6. Retraction of a path traversed by a square robot in a 2D workspace. Picture (a) shows the query path. In (b), this path has been retracted to the medial axis of the workspace. In (c), its branches have been removed.

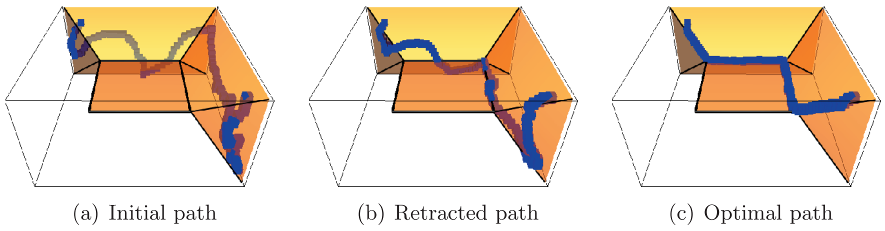

Fig. 7. Retraction of a path in a 3D environment that consists of only a bounding box. A part of the medial axis inside this box is shown. Figure (a) shows the initial path. This path is retracted to the medial axis by Algorithm 5. Figure (c) shows a path having a larger amount of average clearance. This path was obtained by Algorithm 7.

medial axis. As the retraction is performed by a series of translations of the robot, the method is not suitable for increasing the clearance along a path traversed by an articulated robot or a free-flying robot for which the rotational DOFs are important for a solution of the problem. In addition, the method will in general not produce a maximal clearance path because the retraction is completed when the samples are placed somewhere on the medial axis. Many samples could have had a larger clearance if they were further retracted toward configurations representing a higher clearance. See Figure 7 for an example. The crooked path from Figure 7(a) was retracted to the medial axis by the algorithm from the previous section. Figure 7(b) shows that the retracted samples sway on the medial axis surfaces. In Figure 7(c), the path has obtained a larger amount of clearance.

4.2.1. Retraction Algorithm

Our new retraction algorithm attempts to iteratively increase the clearance of the configurations on the path by moving them in a direction for which the clearance is higher. The problem we want to solve is as follows. Convert a given path Π into a path Π' such that for each $\pi'_i \in \Pi'$ the clearance is locally maximal wherever possible. A configuration represents a locally maximum clearance when there is no direction in which the clearance is larger. Algorithm 7 outlines our approach.

Algorithm 7 $\mathcal{C}$-RETRACTION(discrete path Π)

```
1: loop
2:     $\Pi' \leftarrow \Pi$
3:     $dir \leftarrow$ RANDOMDIRECTION($step$)
4:     for all $\pi'_i$ in $\Pi'$ do
5:         $\pi_{new} \leftarrow \pi'_i \oplus dir$
6:         if CLEARANCE($\pi_{new}$) > CLEARANCE($\pi'_i$) then
7:             $\pi'_i \leftarrow \pi_{new}$
8:     $\Pi \leftarrow$ VALIDATEPATH($\Pi, \Pi'$)
9: return $\Pi$
```

Globally speaking, our solution consists of several iterations. In each iteration, we choose a random direction dir which incorporates all DOFs. We use a random direction because alternative choices will require too many time-consuming distance calculations (such as moving in the direction of the steepest descent). Then, we try to move each configuration π_i in the chosen direction, i.e. $\pi'_i \leftarrow \pi_i \oplus dir$. (The operator $\oplus$ will be defined below.) If the clearance of π'_i is larger than the clearance of π_i, then π_i is replaced by π'_i. We stop retracting the path when the *average* clearance of the path (see Definition 6) does not improve anymore.

By updating the configurations, the path is forced to stretch and shrink during the retraction which causes the following two problems. First, the distance between two adjacent

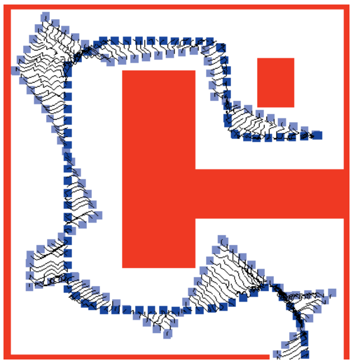

Fig. 8. An impression of the retraction algorithm. The algorithm retracts the initial path traversed by a square robot to the medial axis. For each configuration in the discrete path, the guided random walk (small curve) is drawn.

configurations in the path can become *larger* than the maximum step size. This happens for example when the path is pushed away from the obstacles. If this occurs, we insert an appropriate configuration between them. Second, several configurations can be mapped to a small region in which the distance between two non-adjacent configurations is *smaller* than the step size. This occurs for example when pieces of the path are traversed twice. As we have seen in the previous section, they can be removed easily.

An impression of a retraction is given in Figure 8. This figure shows an initial and a retracted path traversed by a square robot in a simple two-dimensional workspace. The line segments between the paths are the guided random walks of the configurations. We call these walks *guided* because a configuration is updated only if its clearance increases. Note (by close inspection) that extra configurations have been inserted at some places while configurations have been removed at other places. After 40 iterations, the initial path has been successfully retracted to the medial axis, resulting in a path with large clearance. Although this example shows a retraction for a robot with only two DOFs, the retraction can also be applied to robots with more DOFs such as an articulated robot with six joints.

4.2.2. Algorithmic Details

A discrete path consists of a series of configurations. We require that the distance between each pair of adjacent configurations is at most $step$. We calculate the distance between two configurations q and r by summing the weighted partial distances for each DOF $0 \leq i < n$ that describes the configurations, i.e.

$$d(q,r) = \sqrt{\sum_{i=0}^{n-1} [w_i d(q_i, r_i)]^2}.$$

The calculation of distance $d(q_i, r_i)$ can be found in Section 3.4.1. The clearance of the configurations is improved by iteratively moving them in a random direction. We will show how to compute such a direction and how to add this direction to a configuration. After an iteration of the algorithm, the distance between two adjacent configurations may have changed. We will show how to insert and delete appropriate configurations to maintain a valid path. Finally, we discuss how to choose an appropriate termination criterion.

Random direction vector Our goal is to create a random direction q' such that the distance from configuration q to $q \oplus q'$ equals $step$. The direction q' is composed of values for each DOF q_i such that $\sqrt{\sum_{i=0}^{n-1} p_i^2} = step$, where $p_i = w_i d(q_i, q_i \oplus q_i')$. This expression shows how much each DOF i contributes to the total distance. Let rnd be a vector of random values between 0 and 1 such that $\sum_i^{n-1} rnd_i = 1$, and let rnd_i be the random value for DOF i. Furthermore, let $w = (w_0, \cdots w_{n-1})$ be the weight vector. Theorem 1 shows that the translational and rotational(1) components of q' must be set to $q_i' = \pm\frac{rnd_i * step}{\sqrt{rnd \cdot w}}$. The calculation of the rotational(3) component is more complicated. We represent this component as a random 3D unit axis $a = (a_x, a_y, a_z)$ and an angle of revolution θ about that axis. (Since a revolution about a random axis of more than π radians is redundant, we constrain θ to $0 \leq \theta \leq \pi$.) This representation can easily be converted to a quaternion, i.e. $q_i' = (a_x \sin(\theta/2), a_y \sin(\theta/2), a_z \sin(\theta/2), \cos(\theta/2))$. Theorem 1 shows that θ must be set to $\frac{rnd_i * step}{\sqrt{rnd \cdot w}}$.

Lemma 1. *If the distances $d(q_i, q_i \oplus q_i')$ are set to $\pm\frac{rnd_i * step}{\sqrt{rnd \cdot w}}$, then $d(q, q \oplus q') = step$.*

Proof: The distance between q and $q \oplus q'$ is defined as $\sqrt{\sum_{i=0}^{n-1} p_i^2}$, where $p_i = w_i d(q_i, q_i \oplus q_i')$.

When setting $d(q_i, q_i \oplus q_i')$ to $\pm\frac{rnd_i * step}{\sqrt{rnd \cdot w}}$, we must prove that $\sqrt{\sum_{i=0}^{n-1} p_i^2} = step$. By definition, we have

$$d(q, q \oplus q') = \sqrt{\sum_{i=0}^{n-1} \left(w_i d(q_i, q_i \oplus q_i')\right)^2}.$$

By substitution, we get

$$d(q, q \oplus q') = \sqrt{\sum_{i=0}^{n-1} \left(\pm\frac{rnd_i * w_i * step}{\sqrt{rnd \cdot w}}\right)^2},$$

and hence,

$$\begin{aligned} d(q, q \oplus q') &= step * \sqrt{\sum_{i=0}^{n-1} \left(\pm \frac{rnd_i * w_i}{\sqrt{rnd \cdot w}}\right)^2} \\ &= step * \sqrt{\sum_{i=0}^{n-1} \frac{(rnd_i * w_i)^2}{rnd \cdot w}}. \end{aligned}$$

Since $rnd \cdot w = \sum_{j=0}^{n-1} (rnd_j * w_j)^2$, we get

$$d(q, q \oplus q') = step * \sqrt{\sum_{i=0}^{n-1} \frac{(rnd_i * w_i)^2}{\sum_{j=0}^{n-1} (rnd_j * w_j)^2}},$$

and hence,

$$d(q, q \oplus q') = step. \quad \blacksquare$$

Lemma 2. *Let q and q' be two rotational(1) values. If the range of angle q' is set to $-\pi \le q' \le \pi$, then $d(q, q \oplus q') = |q'|$.*

Proof: The distance between two rotational(1) values q and r is defined as

$$d(q, r) = \min\{|q - r|, q - r + 2\pi, r - q + 2\pi\}.$$

Let $r = q \oplus q'$. Then,

$$\begin{aligned} d(q, q \oplus q') &= \min\{|q - (q + q')|, q - (q + q') \\ &+ 2\pi, (q + q') - q + 2\pi\}. \end{aligned}$$

By substitution, we get

$$d(q, q \oplus q') = \min\{|q'|, 2\pi - q', q' + 2\pi\}.$$

Because $-\pi \le q' \le \pi$, it holds that $2\pi - q' \ge \pi$ and $q' + 2\pi \ge \pi$. Since $q' \le \pi$, the minimum is determined by $|q'|$. Hence,

$$d(q, q \oplus q') = |q'|. \quad \blacksquare$$

Lemma 3. *Let q and q' be two quaternions. The quaternion q' represents a random (unit) axis and an angle of revolution θ about that axis. If the range of θ is set to $0 \le \theta \le \pi$, then the distance $d(q, q' * q) = \theta$.*

Proof: The distance between two quaternions q and r is defined as $d(q, r) = \min\{2 \arccos(q \cdot r), 2\pi - 2 \arccos(q \cdot r)\}$. Since θ is positive, the dot product $q \cdot r$ is also positive and lies between 0 and 1. As a consequence, the arccos of the dot product will lie between 0 and π. Hence, the distance between q and r is equal to $d(q, r) = 2 \arccos(q \cdot r)$.

Let $q = q' * r$. The quaternion q' represents a rotation by θ around a unit axis $a = (a_x, a_y, a_z)$, i.e. $q' = (a_x \sin(\theta/2), a_y \sin(\theta/2), a_z \sin(\theta/2), \cos(\theta/2))$. Hence, we get

$$d(q, q' * q) = 2 \arccos((q' * r) \cdot r).$$

After substitution, this can be shown to be equivalent to

$$d(q, q' * q) = 2 \arccos\left((r \cdot r) \cos\left(\frac{\theta}{2}\right)\right).$$

The length of a quaternion that represents a rotation is always equal to 1. Hence, $r \cdot r = 1$. By using $0 \le \theta \le \pi$ and substitution, we get

$$d(q, q' * q) = \theta. \quad \blacksquare$$

Theorem 1. *By setting the translational and rotational(1) components of q' to $q'_i = \pm\frac{rnd_i * step}{\sqrt{rnd \cdot w}}$ and the rotational(3) components of q' to $q'_i = (a_x \sin(\theta/2), a_y \sin(\theta/2), a_z \sin(\theta/2), \cos(\theta/2))$, where $a = (a_x, a_y, a_z)$ is a random unit axis and angle $\theta = \frac{rnd_i * step}{\sqrt{rnd \cdot w}}$, it holds that $d(q, q \oplus q') = step$.*

Proof: The distance between two translational values q_i and $q_i \oplus q'_i$ equals

$$d(q_i, q_i \oplus q'_i) = d(q_i, q_i + q'_i) = |q_i - (q_i + q'_i)| = |q'_i|.$$

Furthermore, Lemma 2 showed that the distance between two rotational(1) values q_i and $q_i \oplus q'_i$ equals

$$d(q_i, q_i \oplus q'_i) = |q'_i|.$$

As we set q'_i to

$$q'_i = \pm\frac{rnd_i * step}{\sqrt{rnd \cdot w}},$$

Lemma 1 implies that

$$d(q, q \oplus q') = step.$$

Finally, Lemma 3 showed that the distance between two quaternions q_i and $q_i \oplus q'_i$ equals

$$d(q_i, q_i \oplus q'_i) = d(q_i, q'_i * q_i) = \theta,$$

where $q'_i = (a_x \sin(\theta/2), a_y \sin(\theta/2), a_z \sin(\theta/2), \cos(\theta/2))$ and $a = (a_x, a_y, a_z)$ is a random unit axis. By setting θ to

$$\theta = \frac{rnd_i * step}{\sqrt{rnd \cdot w}},$$

Lemma 1 implies that

$$d(q, q \oplus q') = step. \quad \blacksquare$$

Besides choosing a random vector, we need to add a direction q' to configuration q. For translational and rotational(1) DOFs, we add up the values. If the rotational(1) DOF is periodic, we have to make sure that the value remains in the range between 0 and 2π. For the rotational(3) DOF, q'_i is multiplied by q_i.

Algorithm 8 VALIDATEPATH(discrete path Π, discrete path Π')

```
1: $i \leftarrow 0$
2: valid path $\Pi'' \leftarrow \emptyset$
3: while $i < |\Pi| - 1$ do
4:    $\Pi'' \leftarrow \Pi'' \cup \pi'_i$
5:    if $d(\pi'_i, \pi'_{i+1}) > step$ then
6:       $\pi'_{int} \leftarrow$ INTERPOLATE$(\pi'_i, \pi'_{i+1}, 0.5)$
7:       if CLEARANCE$(\pi'_{int}) >$ CLEARANCE$(\pi_{i+1})$ then
8:          $\Pi'' \leftarrow \Pi'' \cup \pi'_{int}$
9:       else
10:         $\Pi'' \leftarrow \Pi'' \cup \pi_{i+1}$
11:   $i \leftarrow i + 1$
12: $\Pi'' \leftarrow \Pi'' \cup \pi'_i$
13: $\Pi'' \leftarrow$ REMOVEBRANCHES$(\Pi'')$
14: return $\Pi''$
```

Path validation As a path is forced to stretch and shrink during the retraction, the path may not be valid after an iteration of Algorithm 7. A discrete path Π is valid if $\forall i : d(\pi_i, \pi_{i+1}) \leq step$. In this section we will show how to construct a new valid path. Algorithm 8 outlines our approach.

Let Π be the original path and Π' be the updated path. Furthermore, let π_i be the i^{th} configuration on path Π and π'_i be the corresponding (possibly updated) configuration on path Π'. We construct a new path Π'' which will initially contain all configurations from Π' and possibly new configurations to assure that Π'' is valid.

In each iteration of the loop, we concatenate configuration π'_i to the valid path Π''. Then we check whether the distance between two adjacent configurations on the updated path Π' is larger than the step size, i.e. we check if $d(\pi'_i, \pi'_{i+1}) > step$. If this condition is true, we have to add an extra configuration to path Π'' to assure that Π'' keeps valid. We consider two candidate configurations and choose the one that has the largest clearance. The first one is the original configuration π_{i+1} and the second one is created by interpolating halfway between configurations π'_i and π'_{i+1}.

After the iterations, we add the last configuration of path Π' to the valid path Π''. Finally, to remove superfluous configurations, we apply Algorithm 6 on path Π''. Recall that this algorithm removes a configuration π''_i from path Π'' for which it holds that $d(\pi''_{i-1}, \pi''_{i+1}) \leq step$.

Theorem 2. *Algorithm 8 assures that discrete path Π'' is valid.*

Proof: The input of the algorithm is a valid path Π and a possibly invalid path Π'. We have to prove that the algorithm creates a path Π'' such that $\forall i : d(\pi''_i, \pi''_{i+1}) \leq step$.

Lines 4 and 12 imply that path Π'' will contain each configuration of the updated path Π'. The maximum distance between two adjacent configurations π'_i and π'_{i+1} of path Π' is $2 * step$ which occurs when one of them is updated while the other one is left unchanged. (Note that when both configurations are updated, their relative distance remains the same, and hence, they do not cause the path to be invalid.) We insert one of the following two configurations to path Π''. The first candidate, π'_{int}, is the configuration in the middle of the straight-line in $\mathcal{C}_{\text{free}}$ between π'_i and π'_{i+1}. As the distance between π'_i and π'_{i+1} is halved, $d(\pi'_i, \pi'_{int}) \leq step$ and $d(\pi'_{int}, \pi'_{i+1}) \leq step$. The second candidate is configuration π_{i+1}. It holds that $d(\pi'_i, \pi_{i+1}) \leq step$ and $d(\pi_{i+1}, \pi'_{i+1}) \leq step$. As path Π'' contains the sequence π'_i, the candidate configuration, and π'_{i+1}, path Π'' will remain valid. Finally, as Algorithm REMOVEBRANCHES only removes a configuration π''_i when $d(\pi''_{i-1}, \pi''_{i+1}) < step$, it will not invalidate the path. Hence, Algorithm 8 constructs a valid path Π''. ■

Termination criterion An important issue is when to terminate the algorithm. In each iteration of the algorithm, we only update a configuration if its clearance increases. Such an update can lead to insertions and deletions of configurations. If a configuration π is inserted, then the clearance of π will be equal to or higher than the clearance of the configuration before it was updated. If a configuration π' is deleted, it will not play a role anymore. However, π' could have a high clearance while a possibly inserted configuration could have a low clearance. Hence, while each configuration can obtain a higher clearance, the average clearance can actually decrease. As this worst-case scenario may occur incidentally, we have to take this into account in our termination criterion.

We terminate the algorithm when the improvement of the average clearance in k consecutive iterations is smaller than some small threshold δ. We conducted experiments to find appropriate values for these parameters. We observed that setting k to 25 and δ to $step/10$ led to mature convergence.

4.3. Experiments

In this section, we investigate the extent to which the $\mathcal{W}$-retraction and $\mathcal{C}$-retraction algorithms can improve the clearance along six paths.

4.3.1. Experimental Setup

We considered the environments and their corresponding paths depicted in Figure 9. These are the same test environments as before. The paths were obtained by applying the Path pruning heuristic on the paths of Figure 3. They have the following properties:

Planar As the robot has two translational DOFs, a retraction in the workspace will result in a path having the optimal amount of clearance. The experiments will show whether a retraction in the $\mathcal{C}$-space is competitive.

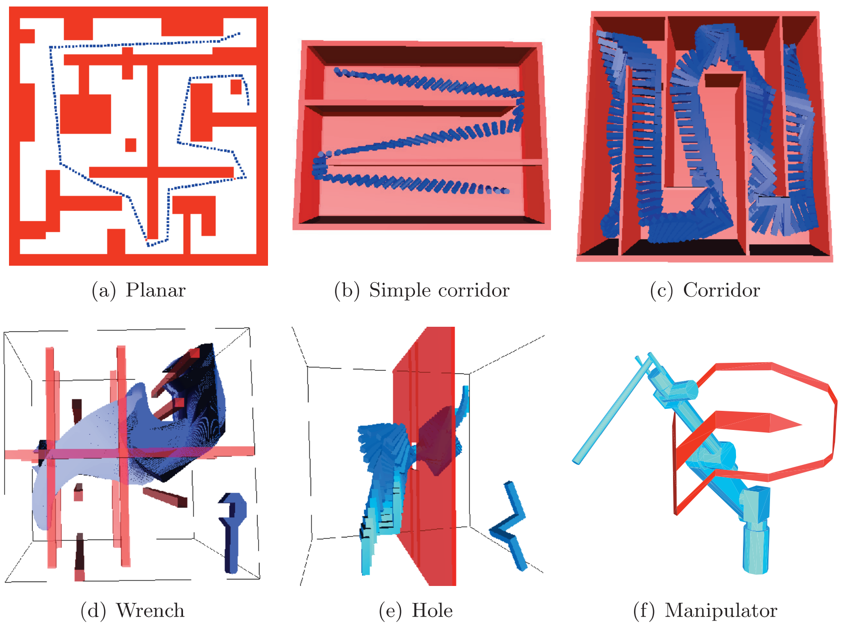

Fig. 9. The six test environments and their corresponding initial paths. For the Wrench and Hole environments, the robot has been depicted separately at the lower right.

Simple corridor This environment has ample free space in which the small free-flying cylinder operates. Both algorithms will introduce an extra amount of clearance as they both move the robot to the middle of the corridor. However, the $\mathcal{C}$-retraction algorithm should outperform the $\mathcal{W}$-retraction algorithm as it also considers rotational DOFs.

Corridor The winding corridor forces the free-flying elbow-shaped robot to rotate. As there is little room between the walls of the corridor and the robot, it may be hard to increase the clearance along the path.

Wrench This environment features a large moving object in a small workspace. We expect that the $\mathcal{W}$-retraction algorithm will be outperformed by the $\mathcal{C}$-retraction algorithm as the rotational DOFs are of major concern in this environment.

Hole The free-flying robot must rotate in a complicated way to get through the hole. Only where the robot passes through the hole, the clearance is small. Hence, the improvement of the minimum amount of clearance along the path shows the potential of the $\mathcal{C}$-retraction algorithm.

Manipulator The articulated robot operates in a constrained environment. The clearance along the path is very small. The $\mathcal{W}$-retraction algorithm cannot be applied as it cannot handle rotational DOFs. Again, an increase in the minimum and average amounts of clearance along the path will show the potential of the $\mathcal{C}$-retraction algorithm.

We subdivided each path in consecutive configurations such that the distance between each two adjacent configurations is at most some predetermined distance $step$. The step sizes for the paths can be found in Table 1. Our distance metric uses weights w_i for the DOFs of a robot. These are listed in Table 2.

In each run, we recorded the minimum, maximum and average clearance of the path. As the $\mathcal{C}$-retraction algorithm is non-deterministic, we ran this algorithm 100 times for each experiment and calculated the averages. Each run was terminated when the improvement of the average clearance in 25 consecutive iterations was smaller than some small threshold, $step/10$.

4.3.2. Experimental Results

The results are listed in Table 6 and visualized in Figure 10.

Planar A retraction in the workspace results in a path having the optimal amount of clearance. The statistics show that the $\mathcal{C}$-retraction technique reached these optimal values. However, for robots having two translational DOFs, we recommend the $\mathcal{W}$-retraction technique as this technique is considerably faster.

Table 6. Clearance statistics for the six environments. A larger clearance indicates a better result. The $\mathcal{C}$-retraction statistics are averages over 100 independent runs.

Planar	Clearance			Time
	min	**avg**	**max**	**s**
Initial path	0.00	2.47	7.15	–
$\mathcal{W}$-retraction	1.79	4.49	8.32	0.8
$\mathcal{C}$-retraction	1.79	4.49	8.32	9.4

Simple corridor	Clearance			Time
	min	**avg**	**max**	**s**
Initial path	0.16	1.91	3.83	–
$\mathcal{W}$-retraction	0.62	2.62	3.96	0.7
$\mathcal{C}$-retraction	1.21	3.64	4.25	6.0

Corridor	Clearance			Time
	min	**avg**	**max**	**s**
Initial path	0.01	0.59	2.44	–
$\mathcal{W}$-retraction	0.22	1.15	3.22	1.0
$\mathcal{C}$-retraction	0.27	1.87	4.57	27.6

Wrench	Clearance			Time
	min	**avg**	**max**	**s**
Initial path	0.00	4.17	11.32	–
$\mathcal{W}$-retraction	2.11	7.12	12.38	12.4
$\mathcal{C}$-retraction	1.99	7.83	15.03	373.8

Hole	Clearance			Time
	min	**avg**	**max**	**s**
Initial path	0.28	1.81	5.97	–
$\mathcal{W}$-retraction	0.79	3.08	6.85	0.6
$\mathcal{C}$-retraction	1.05	3.44	7.24	12.7

Manipulator	Clearance			Time
	min	**avg**	**max**	**s**
Initial path	0.00	0.14	0.35	–
$\mathcal{W}$-retraction	n.a.	n.a.	n.a.	n.a.
$\mathcal{C}$-retraction	0.05	0.29	0.43	26.8

Simple corridor As expected, a large increase in clearance was introduced by the retraction algorithms. At the expense of five extra seconds of computing time, the $\mathcal{C}$-retraction technique doubled the minimum amount of clearance and increased the average clearance by 39% with respect to the $\mathcal{W}$-retraction technique.

Corridor Although there is little room between the walls of the corridor and the robot, the techniques were still able to increase the clearance along the path. Again, the $\mathcal{C}$-retraction technique outperformed the $\mathcal{W}$-retraction technique but this took much more computation time.

Wrench Both algorithms needed relatively long times as the environment was larger than the other ones. Both algorithms were successful in increasing the clearance. The $\mathcal{C}$-retraction technique performed slightly better with respect to increasing the average and maximum clearance. However, the $\mathcal{W}$-retraction technique was 6% better with respect to the minimum clearance. This was due to the early termination of the $\mathcal{C}$-retraction, as shown by decreasing the termination threshold.

Hole The $\mathcal{W}$-retraction technique doubled the amount of minimum and average clearance along the path. The $\mathcal{C}$-retraction technique outperformed the $\mathcal{W}$-retraction technique because all DOFs were taken into account. The minimum amount of clearance along the path was further improved by 33%.

Manipulator The minimum clearance along the initial path was nearly zero. The $\mathcal{C}$-retraction technique successfully introduced some clearance along the path. Although there is little room for the manipulator to move, the algorithm doubled the average clearance along the path. This extra clearance may be crucial in high-cost environments to guarantee safety. For clarity, we only visualized a part of the sweep volume of the manipulator in Figure 10.

4.4. Combining Path Length and Path Clearance

We have now studied techniques which separately compute short or high-clearance paths. In practice, these criteria are combined, i.e. most applications require a short path keeping some minimum amount of clearance ($c_{\min}$) from the obstacles. Notice that these criteria seem to contradict each other: creating shortcuts will pull the robot to the obstacles while adding clearance pushes it away.

The stated two-objective optimization problem can be addressed by first adding clearance to the path. (To reduce the running time of the $\mathcal{C}$-retraction technique, the retraction of a configuration could be stopped when its clearance exceeds the value of $c_{\min}$.) Then, the path length is reduced by applying the Partial shortcut technique while respecting the minimum amount of clearance. This can be easily achieved by adding the following code in the *for* loop to Algorithm 3: **if** CLEARANCE(π''_i) $< c_{\min}$ **then goto** 2.

As an example, we used the first three retracted paths of Figure 10. We set $c_{\min}$ to 1.0, 0.4 and 0.7, respectively, which

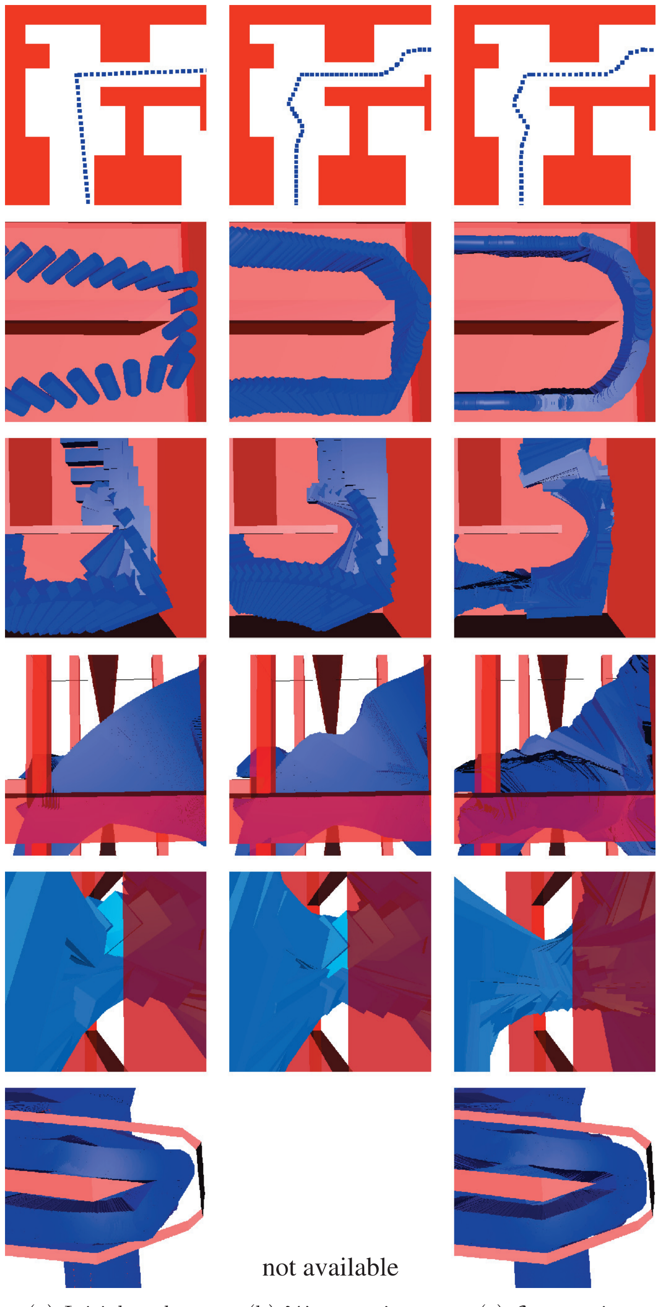

Fig. 10. A close-up of the paths in the six environments. The pictures in the left column show parts of the initial paths. The paths in the middle column are the result of the $\mathcal{W}$-retraction technique while the paths on the right have been created by one particular run of the $\mathcal{C}$-retraction technique.

equal to the step sizes listed in Table 1. Then, we applied the Partial shortcut technique (for 20 seconds) while respecting the minimum amount of clearance. Figure 11 shows the results, together with the high-clearance and short paths to facilitate the comparison. Visual inspection makes clear that the two criteria can easily be combined, resulting in high-quality paths.

5. Conclusions

For many applications, a path should be *short* and should keep some amount of minimum *clearance* from obstacles. These two criteria seem to contradict each other: a short path will pull the robot to obstacles while clearance pushes it away. A high-quality path can be obtained, removing the redundant motions after a particular amount of clearance is added to the path. In this paper, we proposed novel techniques for decreasing the path length, increasing the clearance along a path, and a technique that combines these optimization criteria.

We compared three simple heuristics to decrease the path length, and showed that the Path pruning heuristic is a fast and effective technique that can be used to decrease the path length. The length can be further decreased by the Shortcut heuristic, which is often used as this technique is easy to implement. However, this technique can have difficulties removing all redundant (rotational) motions as all DOFs are interpolated simultaneously. We presented a new technique, Partial shortcut, which takes only one DOF into account in each optimization step. Experiments showed that these redundant motions are now successfully removed. Another advantage of this technique is that the Partial shortcut technique creates shorter paths than the Shortcut technique. Reasonably short paths are obtained within one second of computation time on a modern personal computer.

In this paper, we focused on paths traversed by holonomic robots. An interesting topic for future research is to extend the Shortcut and Partial Shortcut method such that non-holonomic constraints are satisfied.

We also presented two new simple algorithms that increase the clearance along paths. They improve on existing algorithms since higher amounts of clearance for a larger diversity of robots are obtained. Moreover, they do not need complex data structures and (manual) preprocessing. The first algorithm, $\mathcal{W}$-retraction, is fast but it can only deal with rigid, translating bodies. The second algorithm, $\mathcal{C}$-retraction, is slower but it outperforms the workspace-based algorithm as higher amounts of clearance along the paths are obtained. Furthermore, it can handle a broader range of robots which may reside in arbitrary high-dimensional configuration spaces.

The running times indicate that improving the clearance along paths is too slow to be applied online. (They are much larger than the running times for creating the initial paths.) However, in applications where safety is important, the running times are not that crucial. For example, due to the difficulty of measuring and controlling the precise position of a manipulator arm, the arm can be damaged if it moves near obstacles. Improving the clearance at the cost of a few minutes

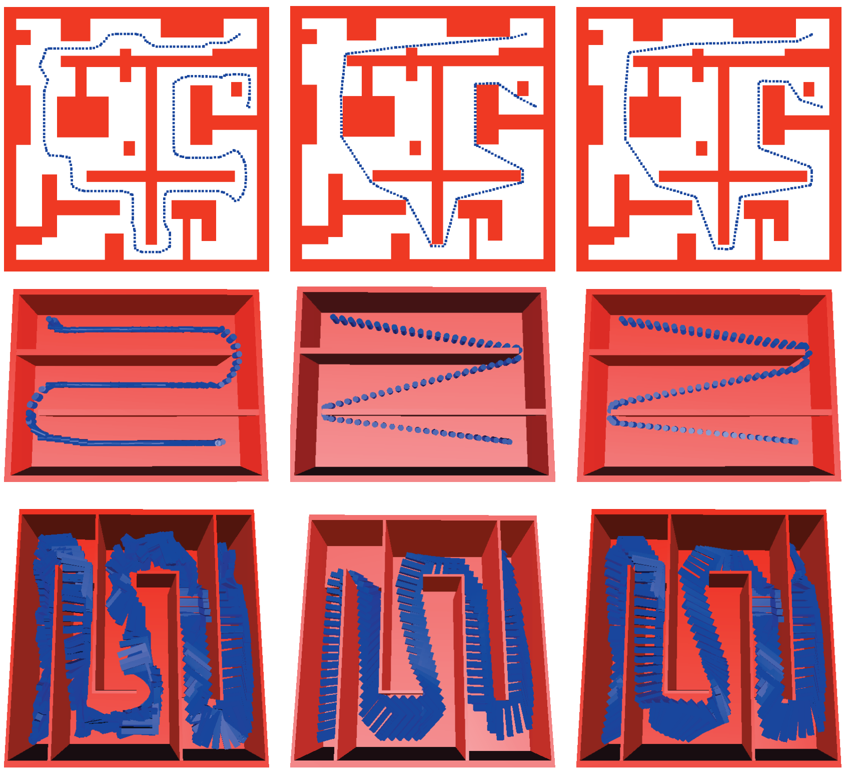

Fig. 11. Paths with high-clearance, short length and their combination.

of calculation time can prevent damage to the robot and its environment.

We expect that the running times of the $\mathcal{C}$-retraction algorithm can be dramatically decreased by incorporating learning techniques. This is a topic of future research. However, when on-line performance is required, a complete roadmap should preferably be retracted to the medial axis in the preprocessing phase. We show in Geraerts and Overmars (2006) that a path can indeed be extracted from such a pre-processed roadmap in real time.

Acknowledgements

Part of this research has been funded by the Dutch BSIK/BRICKS Project.

References

Amato, N.M., Bayazit, O., Dagle, L., Jones, C., and Vallejo, D. (1998). Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE International Conference on Robotics and Automation*, pp. 630–637.

Baginski, B. (1997). Efficient motion planning in high dimensional spaces: The parallelized Z3-method. *International Workshop on Robotics in the Alpe-Adria-Danube Region*, pp. 247–252.

Berchtold, S. and Glavina, B. (1994). A scalable optimizer for automatically generated manipulator motions. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1796–1802.

van den Bergen, G. (2003). *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, San Francisco, CA.

Brock, O. and Khatib, O. (2002). Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research*, **21**: 1031–1052.

Chen, P. C. and Hwang, Y. K. (1998). SANDROS: A dynamic graph search algorithm for motion planning. *IEEE Transactions on Robotics and Automation*, **14**: 390–403.

Choset, H. and Burdick, J. (2000). Sensor-based exploration: The hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, **19**: 96–125.

Geem, C., Siméon, T., Laumond, J.-P., Bouchet, J.-L., and Rit, J.-F. (1999). Mobility analysis for feasibility studies in CAD models of industrial environments. *IEEE Interna-*

tional Conference on Robotics and Automation, pp. 1770–1775.

Geraerts, R. and Overmars, M. H. (2006). Creating high-quality roadmaps for motion planning in virtual environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4355–4361.

Hoff, K., Culver, T., Keyser, J., Lin, M., and Manocha, D. (2000). Interactive motion planning using hardware-accelerated computation of generalized Voronoi diagrams. *IEEE International Conference on Robotics and Automation*, pp. 2931–2937.

Hsu, D., Latombe, J.-C., and Sorkin, S. (1999). Placing a robot manipulator amid obstacles for optimized execution. *IEEE International Symposium on Assembly and Task*, pp. 280–285.

Isto. P. (2002). Constructing probabilistic roadmaps with powerful local planning and path optimization. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2323–2328.

Kavraki, L. E. and Latombe, J.-C. (1998). Probabilistic roadmaps for robot path planning. In *Practical Motion Planning in Robotics: Current Approaches and Future Directions* (eds K. Gupta and A. del Pobil), pp. 33–53, John Wiley.

Kavraki, L. E., Švestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, **12**: 566–580.

Kim, J., Pearce, R., and Amato, N. M. (2003). Extracting optimal paths from roadmaps for motion planning. *IEEE International Conference on Robotics and Automation*, pp. 2424–2429.

Kuffner, J. J. and LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. *IEEE International Conference on Robotics and Automation*, pp. 995–1001.

LaValle, S. M. (2006). Planning Algorithms. http://planning.cs.uiuc.edu. Cambridge University Press, New York, NY.

Lien, J.-M., Thomas, S., and Amato, N. M. (2003). A general framework for sampling on the medial axis of the free space. *IEEE International Conference on Robotics and Automation*, pp. 4439–4444.

Masehianand, E., Admin-Naseri, M. R., and Khadem, S. E. (2003). Online motion planning using incremental construction of medial axis. *IEEE International Conference on Robotics and Automation*, pp. 2928–2933.

Sánchez, G. and Latombe, J.-C. (2002). On delaying collision checking in PRM planning. Application to multi-robot coordination. *International Journal of Robotics Research*, **21**: 5–26.

Sekhavat, S., Švestka, P., Laumond, J.-P., and Overmars, M. H. (1998). Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *International Journal of Robotics Research*, **17**: 840–857.

Vleugels, J. and Overmars, M. H. (1998). Approximating Voronoi diagrams of convex sites in any dimension. *International Journal of Computational Geometry and Applications*, **8**: 201–221.

Švestka, P. (1997) Robot Motion Planning Using Probabilistic Road Maps. PhD thesis, Utrecht University.

Wein, R., van den Berg, J. P., and Halperin, D. (2005). The Visibility-Voronoi complex and its applications. *Annual Symposium on Computational Geometry*, pp. 63–72.

Wilmarth, S. A., Amato, N. M., and Stiller, P. F. (1999). MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. *IEEE International Conference on Robotics and Automation*, pp. 1024–1031.