CHAPTER
FOUR

INCREASING PATH CLEARANCE

Many algorithms have been proposed that create a path for a robot in an environment with obstacles. Since it can be hard to create such a path, 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 preferably be *short* because redundant motions will take longer to execute. In addition, the path also has to keep some amount of minimum *clearance* to the obstacles because it can be difficult to measure and control the precise position of a robot. Traveling along a path with a certain amount of minimum clearance reduces the chances of collisions due to these uncertainties.

In this chapter, we will study two algorithms 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.

4.1 Introduction

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 is extracted by a 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.

Kim *et al.* [88] use an augmented version of Dijkstra's algorithm to extract a path 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* [107], the edges will in general not lie on the medial axis, and hence, the extracted path does not have an optimal amount of clearance.

Another category of algorithms create a path along the Generalized Voronoi Diagram (GVD). 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 4.1 that 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 [35]. 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.

Halperin *et al.* [155] 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 [150] 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.* [111], incrementally constructs 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.* [67] 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.

The above preprocessing methods create a data structure from which paths can be extracted. Brock and Khatib [26] 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 chapter, we will study techniques to improve the clearance along a given path. Later, in Chapter 7, we will discuss a preprocessing approach in which a roadmap with high-clearance paths is computed. First, we will provide some preliminaries in Section 4.2. In Section 4.3, we describe 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 in Section 4.4 where an algorithm is presented 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 in Section 4.5 and conclude in Section 4.6 that clearance along paths can be increased without using complex data structures and algorithms. The results can be used, for example, to obtain high-clearance paths in high-cost environments such as a factory in which a manipulator arm operates.

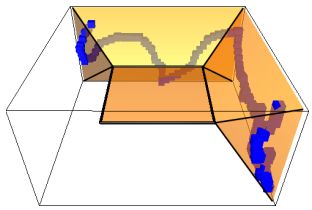


Figure 4.1 A part of the medial axis of an environment that only consists of a bounding box.

4.2 Preliminaries

We say that two configurations π' and π'' are adjacent if the distance $d(\pi', \pi'')$ is at most a predetermined distance *step*. This step size is chosen dependent on the robot and obstacles. The reader is referred to Section 1.2.3 for details on computing distances. We can now define a *discrete path* and a *discrete local path*.

Definition 4.1 (Discrete path Π)**.** *A* discrete path Π *is a series of adjacent configurations* $\pi_0, \cdots, \pi_{n-1}$ *such that* $d(\pi_i, \pi_{i+1}) \leq step$*, where step is the maximum step size.*

Definition 4.2 (Discrete local path LP)**.** *A* discrete local path $\text{LP}[\pi', \pi'']$ *is a series of interpolated configurations* $\pi_0, \cdots, \pi_{n-1}$ *on the local path between configurations* π' *and* π'' *such that* $d(\pi_i, \pi_{i+1}) \leq step$*.*

As this chapter 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 Solid (see Section 1.2.2). We define the clearance of a configuration as follows:

Definition 4.3 (Clearance of a configuration π)**.** *The clearance of a configuration* π *is the Euclidean distance between the pair of closest points on the robot whose placement in the workspace corresponds to configuration* π *on the one hand and the obstacles in the workspace on the other hand.*

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 4.4 (Average clearance of discrete path Π)**.** *Let* Π *be a discrete path. Then the* average clearance *equals to* $\frac{1}{n}\sum_{i=0}^{n-1} Clearance(\pi_i)$*.*

4.3 Rigid, translating bodies

In this section, we describe an algorithm that adds clearance to a path traversed by a translating, rigid body. Our problem 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 scene. 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.* [156] 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.

4.3.1 Retraction algorithm

We will first show how to retract a single sample, corresponding to configuration $\pi \in \Pi$, to the medial axis. Algorithm 4.1 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.[2] 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 4.2 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 $step$.

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

[2] This calculation is delegated to Solid, see Section 1.2.2.

Algorithm 4.1 RETRACTCONFIGURATION(configuration π)

Require: free configuration π, obstacles O, precision *step*

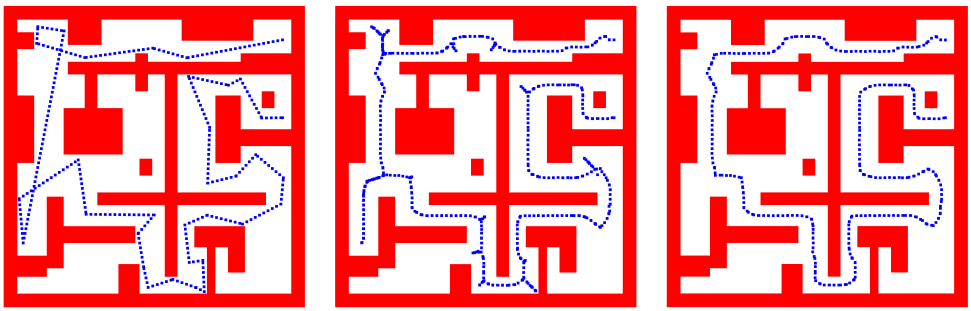
$(cp_\pi, cp_o) \leftarrow$ CLOSESTPAIR(π, O)
$cp_{o'} \leftarrow cp_o$
while $cp_{o'} = cp_o$ **do**
  $\pi' \leftarrow \pi$
  $\pi \leftarrow \pi + cp_\pi - cp_o$
  $(cp_\pi, cp_{o'}) \leftarrow$ CLOSESTPAIR(π, O)
while $d(\pi, \pi') > step$ **do**
  $\pi_{mid} \leftarrow$ INTERPOLATE$(\pi, \pi', 0.5)$
  $(cp_\pi, cp_o) \leftarrow$ CLOSESTPAIR(π_{mid}, O)
  if $cp_{o'} = cp_o$ **then** $\pi \leftarrow \pi_{mid}$ **else** $\pi' \leftarrow \pi_{mid}$
return π_{mid}

Algorithm 4.2 $\mathcal{W}$-RETRACTION(discrete path Π)

retracted path $\Pi' \leftarrow \emptyset$
for all $\pi_i \in \Pi, 0 \leq i < n$ **do**
  $\pi' \leftarrow$ RETRACTCONFIGURATION(π_i)
  $\pi_r \leftarrow$ the last configuration of path Π'
  if $d(\pi_r, \pi') > step$ **then**
    $\Pi' \leftarrow \Pi' \cup \mathcal{W}$-RETRACTION(LP $[\pi_r, \pi']$)
  $\Pi' \leftarrow \Pi' \cup \pi'$
return Π'

Algorithm 4.3 REMOVEBRANCHES(discrete path Π)

$i \leftarrow 1$
while $i < |\Pi| - 1$ **do**
  if $d(\pi_{i-1}, \pi_{i+1}) < step$ **then**
    $\Pi \leftarrow \Pi \backslash \pi_i$
    if $i > 1$ **then** $i \leftarrow i - 1$
  else $i \leftarrow i + 1$
return Π



(a) Original path (b) Retracted path (c) Removed branches

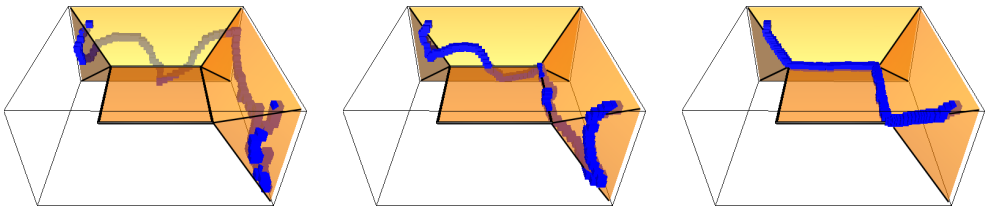
Figure 4.2 Retraction of a path traversed by a square robot in a 2D workspace. Picture (a) shows the query path. In (b), this path has been retracted to the medial axis of the workspace. In (c), its branches have been removed.

Algorithm 4.2 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 $\text{LP}[\pi_0, \pi_0']$ and/or local path $\text{LP}[\pi_{n-1}', \pi_{n-1}]$.

The path will now follow the medial axis. As an example, we applied the algorithm on a square translating in a 2D environment. We borrowed this environment from Lavalle's Motion Strategy Library [100]. See Figure 4.2. The first picture 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 4.3 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 4.2(c) shows the resulting path following the medial axis without traversing a sub-branch twice.

4.4 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 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 maxi-



(a) Initial path (b) Retracted path (c) Optimal path

Figure 4.3 Retraction of a path in a 3D environment that only consists of 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 4.2. Figure (c) shows a path having a larger amount of average clearance. This path was obtained by Algorithm 4.4.

mal 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 4.3 for an example. The crooked path from Figure 4.3(a) was retracted to the medial axis by the algorithm from the previous section. Figure 4.3(b) shows that the retracted samples sway on the medial axis surfaces. In Figure 4.3(c), the path has obtained a larger amount of clearance.

4.4.1 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. Our problem now 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 4.4 outlines our approach. Globally speaking, our solution consists of several iterations. In each iteration, we choose a random direction *dir* which incorporates all DOFs.[3] 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 4.4) does not improve anymore.

By updating the configurations, the path is forced to stretch and shrink dur-

[3]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).

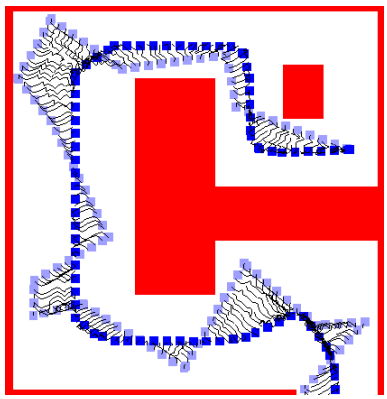


Figure 4.4 An impression of the retraction algorithm. The algorithm retracts the initial path (traversed by a square robot) to the medial axis. For each configuration in the discrete path, the guided random walk (small curve) is drawn.

ing the retraction which causes the following two problems. First, the distance between two adjacent 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 visualized in Figure 4.4. This figure shows an initial and a discrete 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.

Algorithm 4.4 $\mathcal{C}$-RETRACTION(discrete path Π)

```
1: loop
2:    Π' ← Π
3:    dir ← RANDOMDIRECTION(step)
4:    for all π'_i in Π' do
5:       π_new ← π'_i ⊕ dir
6:       if CLEARANCE(π_new) > CLEARANCE(π'_i) then
7:          π'_i ← π_new
8:    Π ← VALIDATEPATH(Π, Π')
9: return Π
```

4.4.2 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*. Distances are computed by the distance metric from Section 1.2.3. That is, the distance between configurations q and r is calculated by summing the weighted partial distances for each DOF $0 \leq i < n$ that describes the configurations; remember that we distinguish three types of DOFs – translation, rotation$_1$ and rotation$_3$, so

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

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 for the algorithm.

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 4.1 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 4.1 shows that θ must be set to $\frac{rnd_i * step}{\sqrt{rnd \cdot w}}$.

Lemma 4.1. *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

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

By substitution, we get

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

and hence,

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

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,

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

■

Lemma 4.2. *Let q and q' be two rotational$_1$ values. If the range of angle q' is set to $-\pi \leq q' \leq \pi$, 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,

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

By substitution, we get

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

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

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

■

Lemma 4.3. *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 $0 \leq \theta \leq \pi$, 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 equals 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

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

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

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

■

Theorem 4.1. *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 4.2 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 4.1 implies that

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

Finally, Lemma 4.3 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 4.1 implies that

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

■

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 4.5 VALIDATEPATH(discrete path Π, discrete path Π')

```
1: i ← 0
2: valid path Π'' ← ∅
3: while i < |Π| − 1 do
4:    Π'' ← Π'' ∪ π'_i
5:    if d(π'_i, π'_{i+1}) > step then
6:       π'_int ← INTERPOLATE(π'_i, π'_{i+1}, 0.5)
7:       if CLEARANCE(π'_int) > CLEARANCE(π_{i+1}) then
8:          Π'' ← Π'' ∪ π'_int
9:       else
10:         Π'' ← Π'' ∪ π_{i+1}
11:   i ← i + 1
12: Π'' ← Π'' ∪ π'_i
13: Π'' ← REMOVEBRANCHES(Π'')
14: return Π''
```

Path validation

As a path is forced to stretch and shrink during the retraction, the path may not be valid anymore after an iteration of Algorithm 4.4. 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 4.5 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}. The reader is referred to Section 1.2.4 for details on interpolation.

After the iterations, we add the last configuration of path Π' to the valid path Π''. Finally, to remove superfluous configurations, we apply Algorithm 4.3 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 4.2. *Algorithm 4.5 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 4.5 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.

4.5 Experiments

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

4.5.1 Experimental setup

We considered the environments and their corresponding paths depicted in Figure 4.5. They have the following properties (see Table 4.1 for their dimensions):

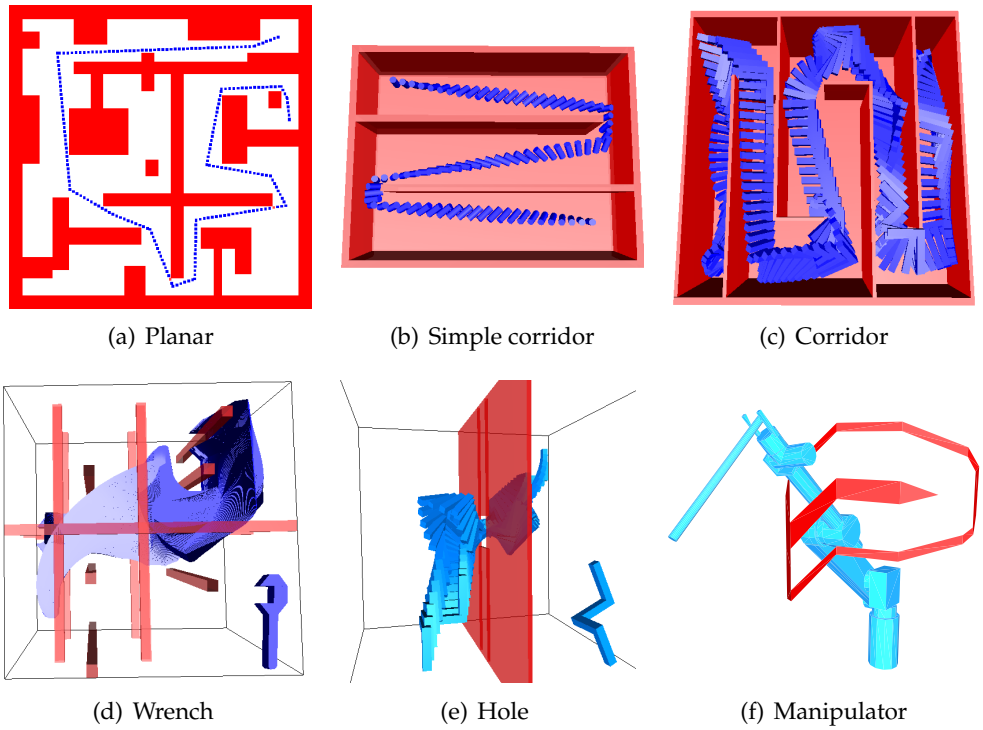
Planar This simple two-dimensional environment contains a path traversed by a square robot that can only translate in the plane. As the robot has two translational DOFs, a retraction in the workspace will result in a path having the optimal amount of clearance. The experiments will show whether a retraction in the $\mathcal{C}$-space is competitive.

Simple corridor This simple three-dimensional environment with ample free space to maneuver features a path traversed by a small free-flying cylinder. Both algorithms will introduce an extra amount of clearance as they both move the robot to the middle of the corridors. However, the $\mathcal{C}$-retraction algorithm should outperform the $\mathcal{W}$-retraction algorithm as it also considers rotational DOFs.

Corridor The environment consists of a winding corridor that forces a 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 fairly large free-flying object (wrench) in a workspace that consists of thirteen crossing beams. The wrench is rather constrained at the start and goal positions. 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, which has six DOFs (three translational DOFs and a rotation$_3$ DOF), consists of four legs and 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.



(a) Planar (b) Simple corridor (c) Corridor

(d) Wrench (e) Hole (f) Manipulator

Figure 4.5 The six test environments and their corresponding paths. For the Wrench and Hole environments, the robot has been depicted separately at the lower right.

Manipulator The articulated robot has six rotational DOFs and 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 4.2. The distance metric from Section 1.2.3 uses weights w_i for the DOFs of a robot. These are listed in Table 4.3.

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, i.e. $step/10$.

	Dimensions of the bounding box	
	environment	**robot**
Planar	100×100	1×1
Simple corridor	$40 \times 11 \times 30$	$0.2 \times 0.2 \times 0.75$
Corridor	$40 \times 17 \times 40$	$5 \times 1 \times 5$
Wrench	$160 \times 160 \times 160$	$68 \times 24 \times 8$
Hole	$40 \times 40 \times 40$	$5 \times 5 \times 10$
Manipulator	$10 \times 10 \times 10$	variable

Table 4.1 The axis-aligned bounding boxes of the environments and robots.

	step size
Planar	1.0
Simple corridor	0.4
Corridor	0.7
Wrench	3.0
Hole	1.0
Manipulator	0.1

Table 4.2 The step sizes for 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	

Table 4.3 The weights for each DOF of the robots.

4.5.2 Experimental results

The results are listed in Table 4.4 and visualized in Figure 4.6.

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 reaches these optimal values. However, for robots having two translational DOFs, we recommend the $\mathcal{W}$-retraction technique as this technique is considerably faster.

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 with 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 much time as the environment was larger compared to 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 showed 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 with 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 4.6.

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

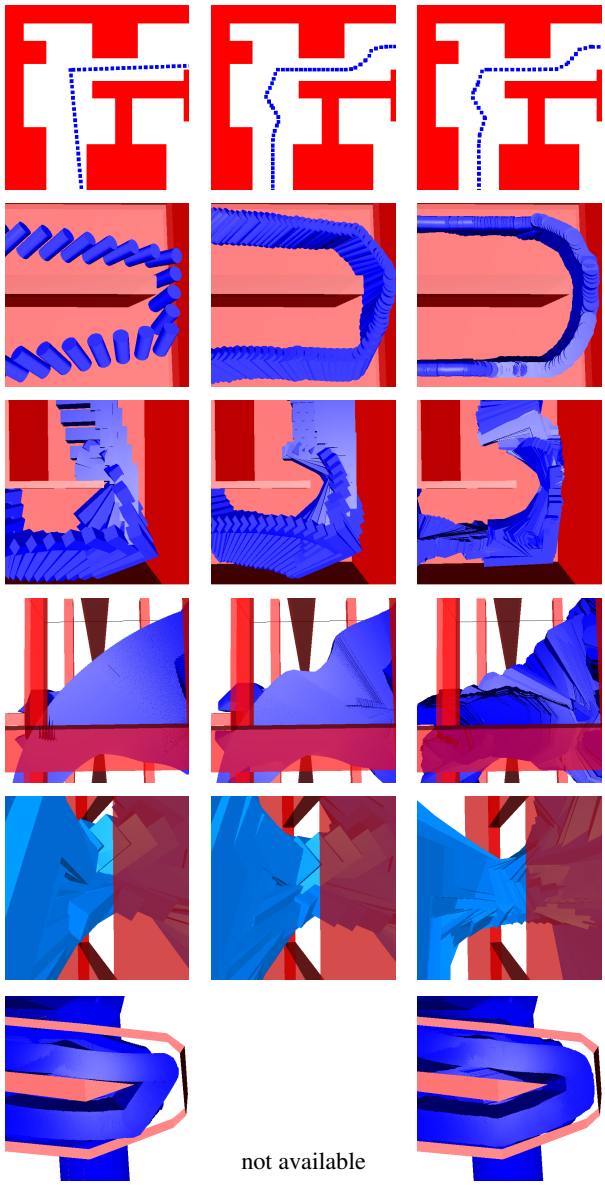
Corridor	Clearance			Time
	min	**avg**	**max**	**s**
Initial path	0.01	0.59	2.44	-
$\mathcal{W}$**-retraction**	0.22	1.15	3.22	1.0
$\mathcal{C}$**-retraction**	0.27	1.87	4.57	27.6

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

Table 4.4 Clearance statistics for the six environments. A larger clearance indicates a better result. The $\mathcal{C}$-retraction statistics are the averages over 100 independent runs.



(a) Initial paths (b) $\mathcal{W}$-retraction (c) $\mathcal{C}$-retraction

Figure 4.6 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 in the right have been created by one particular run of the $\mathcal{C}$-retraction technique.

4.6 Discussion

We presented two new simple algorithms that improve 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 may be too slow to be applied online. 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 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 needed, a complete roadmap can be retracted to the medial axis in the preprocessing phase. We will show in Section 7.3 that a path can indeed be extracted from such a preprocessed roadmap in real-time.