CHAPTER
FIVE

DECREASING PATH LENGTH

Paths which are created by e.g. sampling-based motion planning methods usually contain many redundant motions. However, most applications require short paths because they take less time to execute. A popular algorithm for decreasing the path length is the *Shortcut* method. This method iteratively takes two random configurations on the path and creates a shortcut between them if this shortcut does not cause the robot to collide with the obstacles. Unfortunately, this heuristic will not remove all redundant motions. This is mainly caused by simultaneously interpolating all degrees of freedom of the configurations when creating a shortcut. We propose a new algorithm, *Partial shortcut*, which successfully deals with these difficulties by interpolating one degree of freedom at a time.

5.1 Introduction

Due to their probabilistic nature, sampling-based planners create low-quality paths. These paths often contain many unnecessary and jerky motions. However, many applications require a short path since redundant motions will take longer to execute. In this chapter, we will study several techniques for reducing the path length.

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 $\nu_0, \cdots, \nu_{n-1} \in V$ which may originate from a graph. When a robot must move along this path, it should first be converted to a discrete path Π using a local planner (see Definition 4.1). The path pruning technique removes a node ν_{i+1} from a path Π if the local path $\text{LP}[\nu_i, \nu_{i+2}]$ between nodes ν_i and ν_{i+2} is collision-free. Details will be given in Section 5.2.

In Section 5.3, we investigate the *Shortcut* heuristic, which is the most often applied method 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 the 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 [31, 50, 84, 131, 140, 151], or deterministically [5, 72, 75]. Also several variants of this heuristic have been used [5, 72, 75, 84].

We will show in Section 5.4 that the previous two 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 in each optimization step.

The experiments in Section 5.5 will show that the Partial shortcut method creates shorter paths than the other methods.

5.2 Path pruning

In this section, we assume that a path is represented by a list of nodes, see Definition 5.1. 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. Besides being simple, the technique is efficient and deterministic. See Algorithm 5.1 for more details.

Definition 5.1 (Node path N)**.** *A* node path N *is a series of nodes* $\nu_0, \cdots, \nu_{n-1}$ *such that the local paths* $\text{LP}[\nu_i, \nu_{i+1}]$ *are collision-free.*

Algorithm 5.1 REMOVEREDUNDANTNODES(node path N)

```
1: i ← 0
2: while i < |N| − 2 do
3:    if LP[ν_i, ν_{i+2}] ∈ C_free then
4:       N ← N\ν_{i+1}
5:       if i > 0 then i ← i − 1
6:    else
7:       i ← i + 1
8: return N
```

5.3 Shortcuts

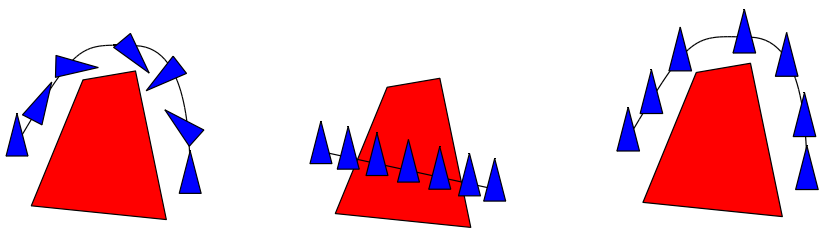
While the previous method only considers the nodes of a path, the shortcut method considers all configurations on a discrete path Π (see Definition 4.1). Therefore, this method is expected to create shorter paths at the cost of increased computation time. The method tries to iteratively improve the path (see Algorithm 5.2). 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[1], all DOFs are interpolated simultaneously. See Section 1.2.4 for details on interpolation.

5.4 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 5.1).

The standard optimization technique (Shortcut) replaces pieces of the path by a straight-line segment in the configuration space. In this way, all DOFs

[1]When the Shortcut method is used for another local planner (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.



(a) Query path (b) Shortcut (c) Partial shortcut

Figure 5.1 Translation is required to navigate around the obstacle and rotation can only be optimized by considering large portions of the path.

Algorithm 5.2 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** $\text{LP}(\pi_a, \pi_b) \in \mathcal{C}_{\text{free}}$ **then**
8: $\Pi \leftarrow \Pi' \cup \text{LP}(\pi_a, \pi_b) \cup \Pi'''$

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 5.1).

We created a new technique, called *Partial Shortcut*, which takes only one DOF f into account in each optimization step. Algorithm 5.3 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.[2] 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 then 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 5.3 PARTIALSHORTCUT(discrete path Π)

```
1: loop
2:    f ← a random degree of freedom
3:    number of configurations n ← |Π|
4:    a, b ← two random indices: 0 ≤ a + 1 < b < n
5:    Π' ← π_0, ··· , π_{a−1}
6:    Π'' ← π_a, ··· , π_b
7:    Π''' ← π_{b+1}, ··· , π_{n−1}
8:    m ← |Π''|
9:    for all π''_i ∈ Π'' do
10:      π''_i[f] ← INTERPOLATE(π''_0[f], π''_{m−1}[f], i/(m − 1))
11:   VALIDATEPATH(Π'')
12:   if Π'' ∈ C_free then
13:      Π ← Π' ∪ Π'' ∪ Π'''
```

5.5 Experiments

In this section, we will apply the three techniques from the previous sections to six different paths. These paths have been selected with a view 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.

[2] We assume that there are no constraints which the configurations on the path have to satisfy.

5.5.1 Experimental setup

To test the quality of the three techniques, we considered the environments depicted in Figure 5.2. These are the same test environments as used in Chapter 4. Their properties are stated in Table 4.1 and Table 4.2. Since they have already been introduced in Section 4.5, we only discuss the properties of the paths:

Planar We use this simple problem involving two DOFs to check whether the techniques can reach the optimal solution.

Simple corridor The environment contains 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 Traversing corridors requires rotation of the elbow shaped object. The path has little clearance. Redundant rotations can only be removed by considering large portions of the path. Hence, we expect that the Partial shortcut technique outperforms the other techniques.

Wrench This environment features a large moving object in a small workspace. The moving object is rather constrained at the start and goal. 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 path contains many redundant (rotational) motions. Only where the path 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 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 other ones.

We need a distance measure to discuss path length. As we have to distinguish between rotational and translational DOFs, we compute the length of a discrete path Π as follows:

$$d(\Pi) = d_r(\Pi) + d_t(\Pi),$$

where

$$d_r(\Pi) = \sum_{i=0}^{n-2} d_r(\pi_i, \pi_{i+1}) \qquad \text{and} \qquad d_t(\Pi) = \sum_{i=0}^{n-2} d_t(\pi_i, \pi_{i+1}).$$

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 $k \leq j < l$:

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

The partial distances $d(q_j, r_j)$ are calculated in the same way as in Section 1.2.3. The weights w_j that are used for the different environments are listed in Table 4.3.

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

$$\Delta d = 100\% * \frac{d - d_{opt}}{d_{opt}}.$$

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 5.1 and depicted in Figure 5.3. Even though we cannot guarantee that these are indeed optimal, visual inspection strongly suggests that they are very close to optimal. Table 5.2 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.

We will investigate the extent to which the three heuristics can improve the six paths from Figure 5.2. We will run the Path pruning heuristic once for each experiment as this technique is deterministic. We will 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 we have learned that all paths converges within this time.

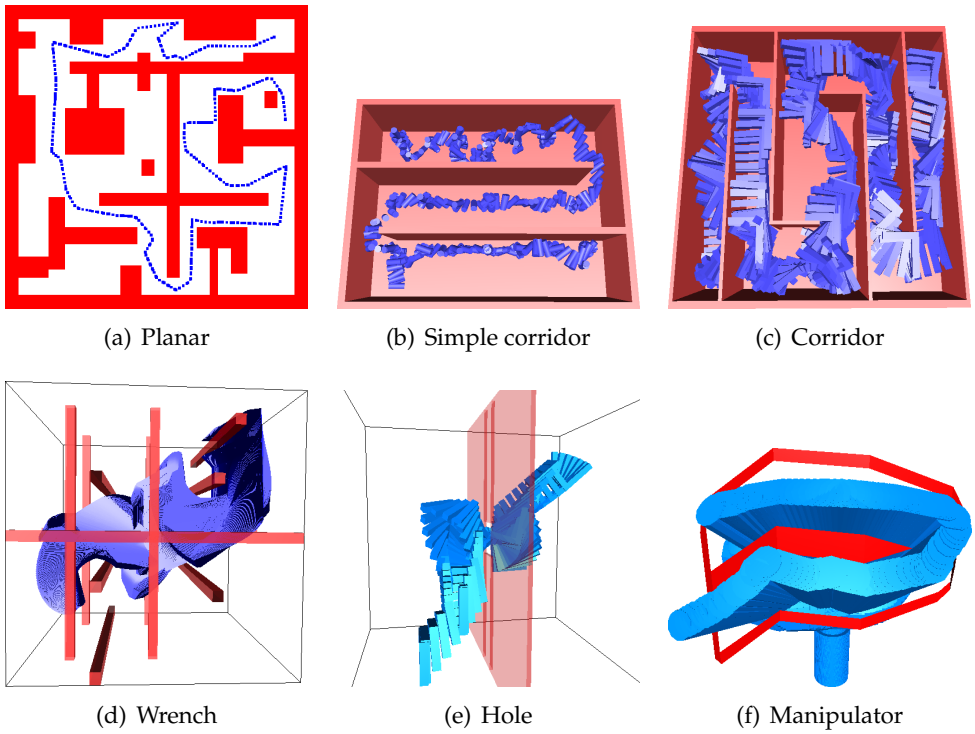
In the second batch of experiments we determine how long the Shortcut and Partial shortcut heuristics should be applied to obtain reasonably short paths. This is useful for practical purposes.

	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 5.1 The shortest absolute lengths of the paths.

	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

Table 5.2 The relative length statistics of the initial paths. The numbers are expressed as percentages relatively to the optimal path lengths.



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

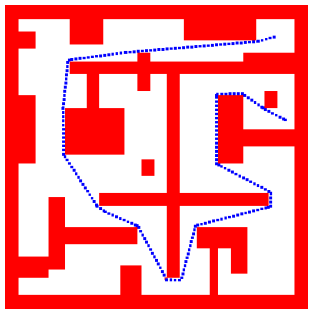
(d) Wrench (e) Hole (f) Manipulator

Figure 5.2 The six test environments and their corresponding *initial* paths.

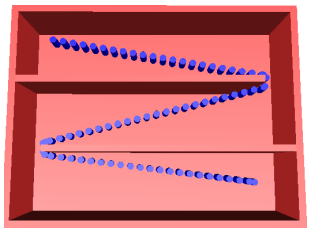
5.5.2 Experimental results

The results of the experiments are stated in Table 5.3. This table shows the relative path length (rotational, translational and total length) for the initial paths and the three heuristics.

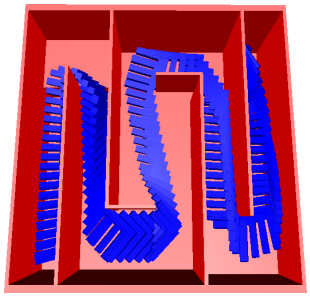
The table shows that the Path pruning heuristic improved the paths considerably in all cases. Note that the rotational distance (Δd_r) is still far from optimal, although the translational distance (Δd_t) has been decreased considerably. The running times of this deterministic heuristic were between 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 to 28 percent 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. In addition, the translational lengths of the paths became close to optimal.



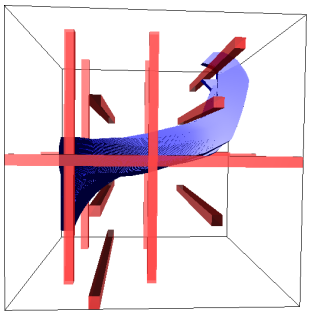
(a) Planar



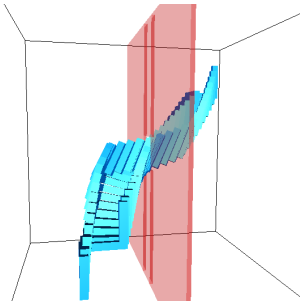
(b) Simple corridor



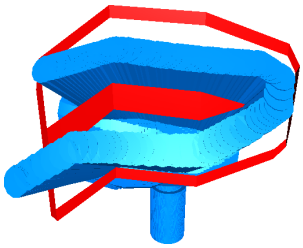
(c) Corridor



(d) Wrench



(e) Hole



(f) Manipulator

Figure 5.3 The six test environments and their corresponding *optimal* paths.

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

Planar Both the Shortcut and Partial shortcut techniques reach the optimal solution. The latter one produces paths that are 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 part of the redundant rotational motions

was 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 5.3(f). 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 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 5.4 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.

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

Corridor	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

Table 5.3 The 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.

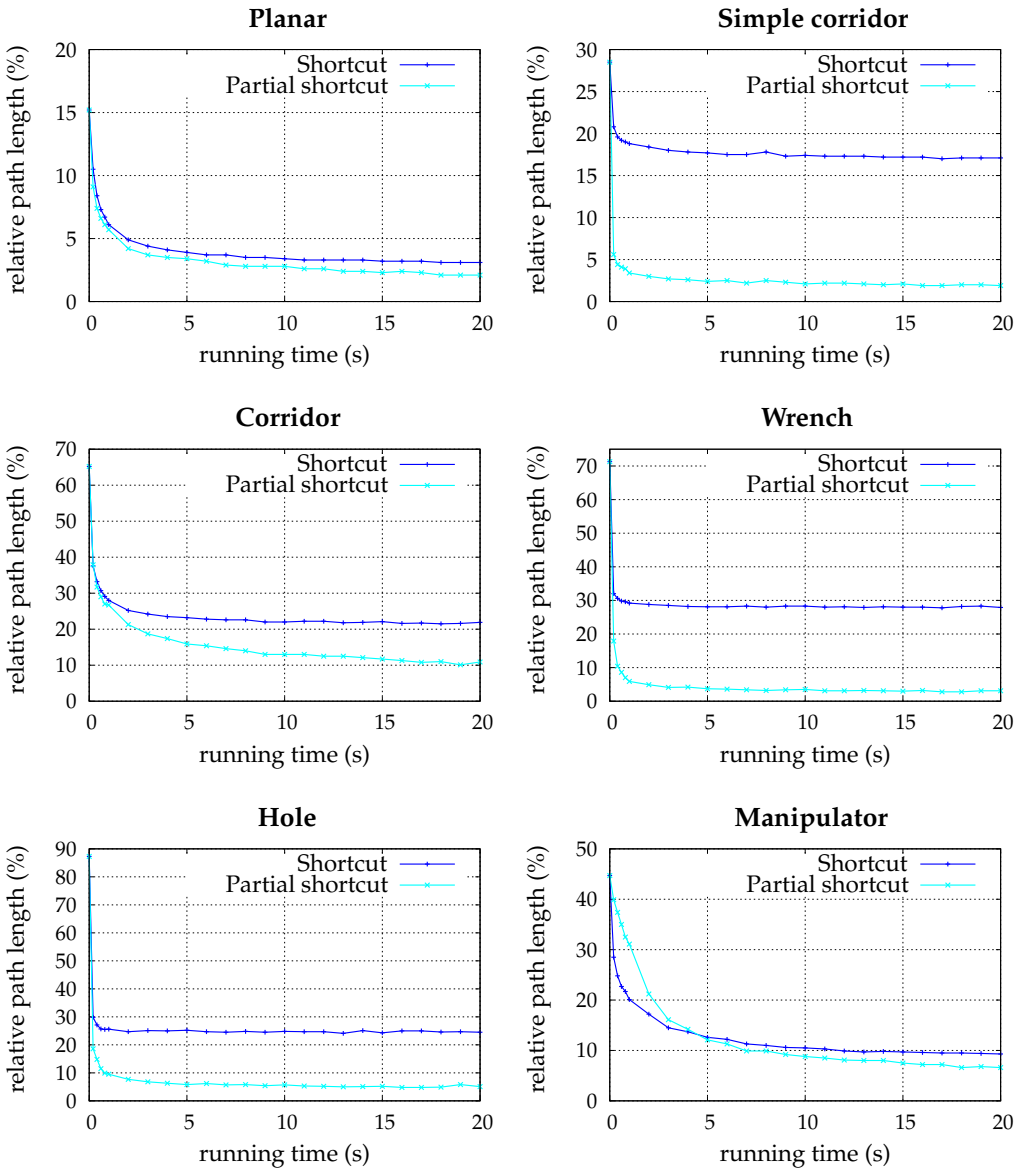


Figure 5.4 Convergence of the Shortcut and Partial shortcut heuristics in the six environments.

5.6 Discussion

We compared three simple heuristics to decrease the path length. We 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 chapter, 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.