Abstract. We propose a novel approach to planning the safe motion of a robot - the motion is considered safe if the robot stays as distant as possible from the nearest obstacle. The approach is based on approximating the medial axis of the free configuration space through randomized probes of the full configuration space.

Experiments with 2- and 3-DOF manipulators show that the approach generates longer but significantly safer motion when compared with naive planning to simply find a shortest motion. In addition, once preprocessing is complete our approach spends much less time planning motion than the naive one. As it is based on probing randomly, rather than at points on a grid, our approach promises to extend effectively to robots with many DOF.

1 Introduction

Motion planning is a central problem in robotics research. We refer the reader to [4][6][12] for a comprehensive survey. The problem involves finding a feasible (continuous and collision-free) path for a given robot from some start configuration to a goal configuration. Precisely: Let $R$ be a $k$-DOF (degrees of freedom) robot with a $k$-dimensional configuration space $C$, where each point in $C$ represents a placement of $R$ in the workspace. Let $C_f \subseteq C$ be the set of free configurations of $R$ at which $R$ does not intersect any obstacles in the workspace. Let $s, g \in C_f$ be the designated start and goal configurations, respectively. The goal of motion planning is to find a continuous path from $s$ to $g$ in $C_f$ that represents a physically executable motion of $R$.

A motion of the robot $R$ can be considered "safe" if the path in $C_f$ corresponding to the motion travels mostly along or near the medial axis (or the Voronoi diagram) [1] of $C_f$, which is defined to be the set of points in $C_f$ that are equally distant from two closest points on the boundary of $C - C_f$ [1][10]. This assertion is based on the observation that (1) a point on the medial axis of $C_f$ is "as far away as possible" from neighboring regions of $C - C_f$, and (2) a configuration of $R$ that is distant from $C - C_f$ corresponds to a physical placement of $R$ that is also distant from the obstacles in some reasonable metric. Safety, of course, is an important consideration as the actual path of a robot is subject to "noise" and may stray from the planned path.

The notion of using the medial axis to plan safe motion has been applied previously [7][13] to a 2-DOF rigid robot (such as a disc) in a 2-dimensional workspace occupied by obstacles that are polygons, line segments or simple curve segments. Note, however, that, for such a robot, the free configuration space $C_f$ is essentially similar to the subspace of the workspace not occupied by the obstacles, and hence the computation of the medial axis (more often called the Voronoi diagram in this context) is relatively easy from a description of the workspace [2][13]. In [3], though, we successfully apply the classical grid-based approach [4] - which consists essentially of probing the configuration space at points of a grid to reduce motion planning to a discrete problem of connecting grid points - to plan safe motion of reasonably complex 2 and 3-DOF manipulators. Our method in [3] is to approximate the medial axis of the free configuration space by probing the full configuration space at points of a grid. However, as is well-known a straightforward extension of grid-based methods to high-dimensional configuration spaces can be prohibitively expensive. A cell-division-based method has recently been proposed in [11] for computing an approximate Voronoi diagram for a set of disjoint convex objects in even in high-dimensional spaces, but it requires exact description $C - C_f$, which is often expensive to compute.

Recently, though, a motion planning method based on random probing has proved successful for high DOF robots [5][8]. Our motivation in this paper is to adapt random probing to approximate medial axes and plan safe motion. Though the computation of medial axes is well-investigated, e.g., in image processing [9], to our knowledge none of the existing techniques apply random probing.

Specifically, in a preprocessing phase we approximate the space of free configurations $C_f$ through probing: configurations are generated randomly in the full configuration space $C$, and each is determined to be valid (i.e., in $C_f$) or invalid.

Heuristics are then used to group invalid configurations into distinct, connected regions, that we term forbidden regions, in the configuration space. Invalid configurations in distinct forbidden regions are tagged with distinct labels $o_i$. Each valid configuration $c$ is then labelled with both the label $o_i$ of the nearest forbidden region, as well as the distance $d_i$ of $c$ from the nearest invalid configuration in the forbidden region $o_i$. The distance $d_i$ is a measure of the safety of $c$.

Once all free configurations have been labeled the medial axis of $C_f$ is approximated. This approximation is accomplished in two stages.

In the first stage, a subset of free configurations that are "believed" to lie on the medial axis of $C_f$, based on certain heuristics, is selected. These selected configurations may, however, not produce a connected skeleton for each connected region of $C_f$. Note that the medial axis of $C_f$
consists of one connected component for each connected region of $C_j$, and, ideally, any skeleton approximating each of these components should also be connected. To ensure that at the end of the preprocessing phase the skeleton has few disconnected components (or, hopefully, only one connected component) for each connected region of $C_j$, in the second stage so-called bridges are built between disconnected components. This is done by a repeated application of a Dijkstra-type algorithm that finds connecting paths lying in safe regions of $C_j$. The totality of configurations, either selected originally by a heuristic or found as part of a bridge, then constitute our approximation of the medial axis (i.e., a skeleton) of $C_j$.

In the final stage of preprocessing, we use an inexpensive local planner to attempt to connect each configuration in the skeleton to its neighbors that are also in the skeleton. An edge is then inserted between configuration pairs that are successfully connected by the local planner to form a graph, called the axis graph.

In a subsequent planning phase, an attempt is first made to connect the start and goal configurations $s$ and $g$ to nodes of the axis graph. If this is successful a path is computed from $s$ to $g$ through the graph.

We apply the approach described above to 2- and 3-DOF manipulators. Our experimental results, albeit with these low DOF configuration spaces, show random probing to be remarkably effective and efficient in planning safe motion, and we are now in the process of extending the implementation to high DOF manipulators.

In Section 2 we describe our method for approximating the forbidden regions, and then approximating the medial axis. Section 3 gives the implementation details of the proposed approach for 2- and 3-DOF manipulators, and the analysis of the experimental results. Concluding remarks are in Section 4.

## 2 Approximating Forbidden Regions, Medial Axes, and Planning Motion

### Approximating Forbidden Regions

To approximate forbidden regions by probing, we label all invalid configurations that lie within a certain threshold distance $T_\varepsilon$ of each other with the same label. In other words, if $L(c)$ represents the label of the configuration $c$ and $D(c_1, c_2)$ represents the distance between the configuration $c_1$ and $c_2$, calculated with respect to some metric, then $L(c_1) = L(c_2)$ if $c_1$ and $c_2$ are invalid configurations such that $D(c_1, c_2) \leq T_\varepsilon$.

The random generation of configurations in the configuration space makes approximating forbidden regions somewhat difficult. If the value of $T_\varepsilon$ is too small, then groups of invalid configurations that do in fact belong to the same forbidden region may be erroneously determined to belong to distinct forbidden regions. On the other hand if the value of $T_\varepsilon$ is too large, then two distinct forbidden regions may be grouped as one.

### Approximating Medial Axes

Each valid configuration $c_i$ is labeled with a tuple $(o_i, d_i)$, where $o_i$ indicates the forbidden region that $c_i$ is closest to and $d_i$ indicates the shortest distance of $c_i$ to an invalid configuration belonging to $o_i$. In other words, if $c_i$ is an invalid configuration closest to $c_i$, then $o_i = L(c_i)$ and $d_i = D(c_i, c_j)$.

The selection of the configurations that lie on the medial axis is done using a heuristic as follows: First, let us define the neighbors of a valid configuration $c_i$ as $\{c : c \text{ is a valid configuration and } D(c, c_i) \leq T_\varepsilon\}$, where $T_\varepsilon$ is a predetermined threshold value. Now, each valid configuration $c_i$ reads the value $o_i$ of each of its neighbors $c_j$. If two neighboring configurations $c_i$ and $c_j$ have labels $(o_i, d_i)$ and $(o_j, d_j)$, respectively, where $o_i \neq o_j$ (i.e., they are closer to distinct forbidden regions), then both $c_i$ and $c_j$ are selected to lie on the medial axis $M$.

The motivation for this heuristic is that configurations on the medial axis are exactly those that are equidistant from two distinct forbidden regions.

It turns out, however, that pieces of the medial axis of $C_j$ may be “missing” from the approximation $M$ obtained above, owing to the granularity of the probes, especially for dimensions higher than two. This can cause a disconnection in the approximation. In most cases we do not have a priori knowledge of the number of connected regions of $C_j$. or, therefore, of the number of connected components that its medial axis approximation $M$ should have. With this in mind, if $M$ is found to have two or more components we attempt to connect these components by building paths, or, as we term them, bridges. We do this by applying a Dijkstra-type shortest path algorithm that is modified to find safe paths by assigning weights to nodes (i.e., valid configurations) that are inverse to their distance from the closest forbidden region.

Fig. 1 shows the configuration space $C$ of a 2-DOF manipulator, where the hollow circles represent the configurations in $C - C_j$, and the points in $M$ chosen by our heuristic are shown as filled circles. Details of this manipulator will be given in the next section.

We note that $M$, in fact, gives a connected skeleton that lies in apparently safe regions of $C_j$.

Finally, we convert $M$ into a graph, to be used to plan safe motion, by applying an inexpensive local planner to attempt to connect each configuration in the skeleton to its neighbors that are also in the skeleton. If the local planner is, in fact, able to connect two configurations, then an edge between them is added to a graph, called the axis graph.

![Figure 1: Approximation $M$ of the medial axis of $C_j$ for the 2-DOF manipulator shown in Fig. 2. The darker hollow circles represent the configurations in $C - C_j$, and the black dots the points in $M$. The path shown corresponds to the motion of the manipulator shown in Fig. 4(a).](image-url)
Planning Motion

In the planning phase, we first attempt to connect both the given start and goal configurations to \( M \) by a simple "hill-climbing" heuristic. Specifically, we again apply our local planner to successively attempt connections to a neighboring valid configuration \( c_1 \) having the largest \( d_i \) of all neighboring labels \((a_i, d_i)\), until a configuration already in the axis graph is reached. Sometimes, though, the hill climbing method will not connect to the axis graph, owing most likely to one of two reasons: (a) the local planner fails to make a connection, or (b) at some point in the movement towards \( M \) we are trapped at a local maxima of \( d_i \) values. In this case we assume we are unable to plan a path.

Once configurations \( s \) and \( g \) are connected to \( s' \) and \( g' \) of \( M \), respectively, we attempt to find a safe path between \( s' \) and \( g' \) in \( M \) using the same modified Dijkstra-type algorithm that we use to build bridges amongst components of \( M \).

Selection of the threshold \( T_r \) is critical to a good approximation of the medial axis. If \( T_r \) is large then the number of configurations selected to lie on the medial axis will also be large, as neighbors that are far apart but close to distinct forbidden region tend to get selected. The adverse effects of a large axis graph include (a) increased processing time for path planning, and (b) deterioration of the quality of paths as the axis graph contains configurations relatively close to forbidden regions.

On the other hand, if \( T_r \) is small the likelihood increases of getting stuck at a local maxima in the path planning stage, when trying to connect the source or the goal configuration to the axis graph using hill-climbing.

For the current version of the paper we experiment with different values of \( T_r \), but as future work plan to learn estimating good values for \( T_r \) from a preliminary survey of the workspace itself.

3 Experiments with Manipulators and Analysis

3.1 A 2-DOF Manipulator

The 2-DOF manipulator that we use is a robot arm of length 200 pixels that can rotate about a pivot \( p \) as well as slide along \( p \). See Fig. 2. The configuration space is therefore two dimensional, each configuration being a 2-tuple \((l, \theta)\), where \( l \) represents the distance of one distinguished end of the arm from the pivot, and \( \theta \) its orientation.

We next describe our local planner \( P \) that we use to attempt to move the arm from one configuration to another: given two configurations \( c_1 = (l_1, \theta_1) \) and \( c_2 = (l_2, \theta_2) \), \( P \) first plans the motion from \( c_1 \) to \( c_2 \) as a rotation about \( p \) along the smaller of the two angles (one clockwise and the other counterclockwise) between orientations \( \theta_1 \) and \( \theta_2 \), to reach the intermediate configuration \((l_1, \theta_2)\), followed by sliding along \( p \) to reach \((l_2, \theta_2)\). Then \( P \) checks if this plan indeed defines a collision-free motion from \( c_1 \) to \( c_2 \). If so it is successful in planning motion from \( c_1 \) to \( c_2 \), if not it fails.

Scenes

We experiment with moving the robot arm in five different scenes, shown in Fig. 3, each being a square of side 500 pixels containing obstacles. The pivot \( p \) is at the center of each square. To keep the implementation simple our obstacles are all circles of the same radius.

We compare our approach to a simpler one, referred to as the naive approach, that does not take into account safety. In the naive approach, configurations are first generated randomly in the configuration space as in our approach (the configurations are identical in both the methods). A graph is then created by simply inserting an edge between every two neighboring configurations \( c_1 \) and \( c_2 \), if the local planner \( P \) succeeds in planning a motion from \( c_1 \) to \( c_2 \). Finally, a motion between the start and goal configuration is planned by finding a shortest path on this graph using Dijkstra's algorithm.

Fig. 4(a) shows a movement of the manipulator found by our approach using the approximate medial axis shown in Fig. 1. The naive approach, on the other hand, yields the motion shown in Fig. 4(b) for the same start-goal configuration pair. Note that the arm stays farther away from the obstacles during the motion shown in Fig. 4(a).

For each scene, we fixed \( T_r \) and generated different numbers of configurations and used different values of \( T_r \), to determine the improvement in performance of either approach in terms of the quality of the path and also the time required to answer the queries. Further, for each scene we randomly generated 1000 queries, each consisting of a start-
goal configuration pair (in $C_f$, so that both configurations are free). If in the planning phase a path is found for a given start-goal pair, we say that the corresponding query has been successfully answered. Note though that some start-goal configuration pairs may not admit any collision-free motion for the arm – we do not precheck to exclude such pairs.

We collect data including the following parameters for both our approach and the naive approach:

- **NG**: Number of configurations generated randomly.
- **NQ**: Number of queries answered successfully.
- **ATQ**: Average CPU time (sec) required to answer a query (on a DEC 3000/600).
- **APL**: Average path length for successfully answered queries. Path lengths are measured in the configuration space, rather than the workspace, by the number of edges.
- **ASI**: Average safety index of paths found for successfully answered queries. The safety index of a path is the average of the weights of all the configurations that lie on the path.
- **PT**: Preprocessing time (sec).

### 3.2 A 3-DOF Manipulator

Our 3-DOF manipulator is obtained from the 2-DOF manipulator described in the previous section by adding another degree of freedom by allowing the pivot $p$ to slide along a straight-line track $T$ of length 100 pixels. See Fig. 2. The configuration space is therefore three dimensional, with each configuration being a 3-tuple $(t, l, \theta)$ where $t$ represents the position of the pivot on the track, and $l$ and $\theta$ are as before.

To plan a motion from $(t_1, l_1, \theta_1)$ to $(t_2, l_2, \theta_2)$, the local planner $P$ first tries to translate the arm along the track to reach $(t_2, l_1, \theta_1)$, and then from $(t_2, l_1, \theta_1)$ to $(t_2, l_2, \theta_2)$ by rotation followed by sliding as for the 2-DOF manipulator.

### Scenes

We experimented with moving the robot arm in five different scenes, called Scenes 1–5, that are similar to those shown in Fig. 3 for the 2-DOF case. Each scene is a square of size 700 pixels containing obstacles that are circles.

As in the 2-DOF case, for each scene motion is planned by both our approach and the naive approach, for 1000 start-goal configuration pairs generated randomly in $C_f$, with different numbers of total configurations and different values of $T_T$. See Figs. 5(a) and 5(b) for examples of motion planned by either approach. The same parameters that were introduced for the 2-DOF case are used here.

### 3.3 Results and Analysis

The charts in Figs. 6–9 show a few of our experimental results, comparing parameters NQ, ATQ, APL and ASI for both our and the naive methods, for both the 2- and 3-DOF manipulators in different scenes. The four different lines represent the ratios of the values of the four parameters, precisely, $(\text{value with our method})/(\text{value with the naive method})$, plotted against NG.

As the ASI curve shows, our method is almost always safer than the naive method by a factor of about 1.4–1.6 for the 2-DOF manipulator, and at least 1.6 for the 3-DOF manipulator. The APL curve indicates that the paths we find are almost always at most 1.4 times longer. Further, as may be seen from the ATQ curve, we answer individual queries more than twice as quickly as the naive method. The reason for this is that, as we take safety into account, the axis graph is considerably smaller than the graph searched in the naive method. The NQ curve indicates that we almost always solve as many queries as the naive method. Looking at raw data for PT and ATQ, that are not reproduced here, we conclude that the initial investment in preprocessing time in our method amortizes after approximately 100 queries.

### 4 Concluding Remarks

We have presented a new approach for planning safe motion based on the construction of an approximate medial axis of the free configuration space of a robot. The experimental results indicate that the approach does indeed succeed in
finding significantly safer motion.

We are considering variations of the methods presented in this paper, where the required threshold distances are learned from the workspace itself and does not require user intervention for that.

References


Figure 6: 2-DOF, scene 1, $T_o = 15$, $T_r = 15$.

Figure 7: 2-DOF, scene 2, $T_o = 15$, $T_r = 15$. 
Figure 8: 3-DOF, scene 1, $T_x = 30, T_y = 20$.

Figure 9: 3-DOF, scene 2, $T_x = 30, T_y = 20$. 