Using Skeletons for Nonholonomic Path Planning among Obstacles

Brian Mirtich *
John Canny †

Computer Science Division
University of California
Berkeley, CA 94720

Abstract

This paper describes a practical path planner for nonholonomic robots in environments with obstacles. The planner is based on building a one-dimensional, maximal clearance skeleton through the configuration space of the robot. However rather than using the Euclidean metric to determine clearance, a special metric which captures information about the nonholonomy of the robot is used. The robot navigates from start to goal states by loosely following the skeleton; the resulting paths taken by the robot are of low "complexity." We describe how much of the computation can be done off-line once and for all for a given robot, making for an efficient planner. The focus is on path planning for mobile robots, particularly the planar two-axle car, but the underlying ideas are quite general and may be applied to planners for other nonholonomic robots.

1 Introduction

With an abundance of theoretical results and algorithms already existing for solving the classical piano mover's problem, research has recently focused on solving more general versions of this problem. Prominent among these generalizations is path planing for nonholonomic robots. A useful technique in attacking the piano mover's problem is to transform it from physical space to configuration space, replacing the robot with a single point in configuration space and each obstacle with a corresponding configuration obstacle. Planning for the point robot in configuration space is equivalent to planning for the actual robot in physical space [17]. An underlying assumption in the classic version of this problem is that (ignoring obstacles) the point can locally move in any direction of configuration space, that is it can move at any velocity in the tangent space of the manifold of configurations. When planning for nonholonomic robots, this assumption must be relaxed, greatly increasing the difficulty of the problem.

*Supported by NSF Graduate Fellowship, David and Lucile Packard Fellowship and National Science Foundation Presidential Young Investigator Award (# IRI-8958577).

†Supported in part by David and Lucile Packard Fellowship and National Science Foundation Presidential Young Investigator Award (# IRI-8958577).

1.1 Path complexity

Much recent research has focused on important theoretical considerations such as exactly when a system is completely controllable [1, 3]. A completely controllable robot can reach every point in free configuration space within the same connected component of its current configuration. Assuming a completely controllable robot, the object is to plan nice trajectories between start and goal configurations which avoid obstacles. "Nice" is of course a rather nebulous concept; a somewhat better description would be trajectories of low complexity. This begs the question of how to measure path complexity. A complexity measure for general systems is discussed later; for now consider a specific example, the two-axle car in the plane.

It seems intuitive that the "simplest" path between two configurations might be the shortest such path, so one may look to the shortest paths for inspiration in defining complexity. Reeds and Shepp have proven that in the absence of obstacles the shortest path for the car robot never contains more than two reversals, which are places where the car changes direction from forward to backward or vice-versa [20]. As any practiced parallel parker can attest, having to change directions many times does not make for a desirable path. Since it is also desirable to drive as little as possible, path complexity for the car like robot increases with path length and number of reversals. Our planner tries to find paths which minimize the path complexity, although no guarantees on optimality are made since complexity remains a somewhat qualitative concept.

To illustrate this interpretation of complexity, consider the following method of planning feasible paths for the car (a feasible path is one which obeys the nonholonomic constraints of the system). First a nonfeasible path is planned using any algorithm for the classic piano mover's problem. This nonfeasible path may then be approximated to arbitrary closeness by a feasible path. The standard planner has no knowledge of the nonholonomic constraints of the robot, and the path it generates may not be a good one to approximate with a feasible path. The left of figure 1 illustrates this; the resulting feasible path is quite complex involving many reversals. On the other hand, the right of figure 1 shows a less complex path. Qualitatively this appears to be a much better path.

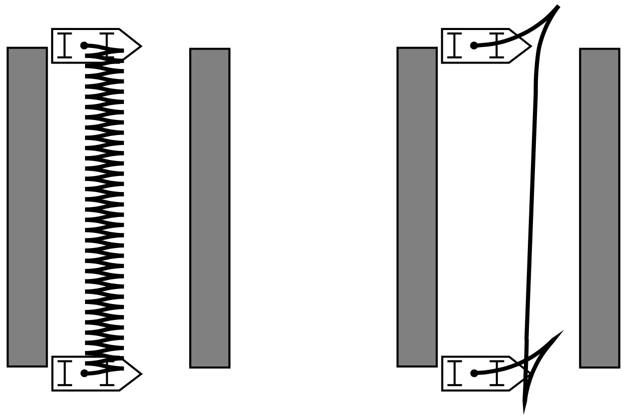

Figure 1: *Left: Path of high complexity involving many reversals. Right: Simpler (and shorter) path.*

1.2 Overview of the algorithm

The basic idea of the algorithm is as follows. For a given environment of the robot, a one dimensional subset (called a skeleton or roadmap) of configuration space is constructed. This subset is path-connected within connected components of configuration space, and the same skeleton may be used repeatedly for solving path planing problems in the given environment. To find a path between start and goal configurations s and g, paths are found which link s and g to the skeleton. Then a path is found which connects the two link points of the skeleton. This latter path is computed by moving roughly "along" the skeleton. Once the skeleton has been constructed, path planing is much simpler since only a one dimensional set of configurations is involved. Furthermore, the method of constructing the skeleton insures that the paths which follow it will be of low complexity if at all possible. For a given robot, a large part of the work can be done off-line so that skeletons for new environments can be computed more rapidly. The algorithm is currently being implemented for the planar two-axle car.

2 Related work

Latombe presents an excellent survey of the problems and algorithms of robot motion planning, including nonholonomic motion planning [12].

Laumond performed some the earliest work in the area of nonholonomic motion planning, planning for mobile robots subject to one nonholonomic constraint in the presence of obstacles [14]. Fortune and Wilfong developed a decision algorithm for determining the existence of a feasible path under given conditions, although the algorithm does not find the actual path [6]. Barraquand and Latombe have attacked nonholonomic motion planing from a different angle; their planner finds a path by performing a systematic search through the the configuration space, using potential field methods to guide the search [1]. They find paths for car and trailer-like robots (in the presence of obstacles) which minimize the number of maneuvers required, and their planner is able to generate paths for robots with relatively many degrees of freedom. Jacobs and Canny have given an algorithm for path planning for a car-like robot amidst obstacles based on reducing the set of a smooth trajectories to a sufficient set of canonical trajectories [7, 8]. The building blocks for their trajectories are "Dubins paths" while our trajectories for the car-like robot are constructed from "Reeds-Shepp paths," the major difference being that the latter may contain reversals along the path between two points. Murray and Sastry developed a method of steering systems with nonholonomic constraints between arbitrary configurations using sinusoidal control inputs [19]. Although their method assumes no obstacles, it provides a simple and efficient means for generating paths and can certainly be incorporated into a planner which take obstacles into account.

Our work most closely parallels the recent work of Jacobs, Laumond, and Taix. They have developed a two-stage planning strategy for the car-like robot [9]. First a path is found for the associated holonomic system (obtained by removing the nonholonomic constraints), and then this initial path is approximated by feasible path segments. Our planner is also based on following a nonfeasible path (a portion of the skeleton). They also introduced the notion of a metric based on shortest paths and the corresponding "Reeds-Shepp ball," which served as inspiration for our planner.

3 Shortest feasible path (SFP) metric

3.1 Definition of the SFP metric

The idea of reducing configuration space to a smaller dimensional subset still complete for path planning is not new. Roadmaps [4] and Voronoi diagrams have been used previously to simplify path planning. What is novel about our approach is the particular skeleton which is constructed and how it is built. It is a maximal clearance skeleton based on a special metric that makes it especially suited to nonholonomic motion planning.

The skeleton is a set of points which are maximally clear from two or more obstacles. Using the car as an example, it is easy to see why the standard Euclidean metric gives a somewhat distorted view of equidistance for nonholonomic systems (see figure 2). Under the Euclidean metric, the car is the same distance from each of the two configuration obstacles. However, it is actually harder for the car to move in the "sideways" direction. When considering only feasible paths, the car must drive a farther distance in order to reach obstacle A than to reach obstacle B. So in some sense obstacle A is further away than obstacle B. This is the essential idea behind our metric, the shortest feasible path (SFP) metric.

Definition. Let C be the configuration space for a nonholonomic robot R. The **SFP metric** is a function $\rho_R : C \times C \to \mathbb{R}$ which assigns to each pair of points $p_1, p_2 \in C$ the (arc) length of the shortest *feasible* path between the two configurations. The real number $\rho_R(p_1, p_2)$ is called the **SFP distance** between points p_1 and p_2.

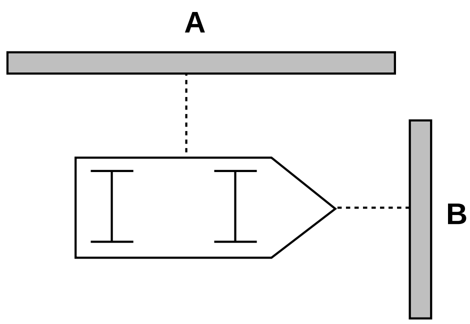

Figure 2: *When considering only feasible paths obstacle B is closer.*

3.2 Justification for using SFP metric

Given start and goal configurations represented by points s_0 and g_0 in configuration space, a path between s_0 and g_0 is found by concatenating three subpaths, all of which must of course avoid obstacles. First subpaths are found from s_0 to s and g_0 to g, where s and g are points on the skeleton. Then a subpath must be found which moves "along" the skeleton, linking s and g. In general, this last portion of the path forms the bulk of the complete path.

In general it is impossible to move between points s and g without leaving the skeleton because the skeleton may sometimes lead in infeasible directions. Although the skeleton lies completely in free space, when the robot steps off the skeleton obstacle avoidance becomes an issue.

Definition. Let C be the configuration space for a nonholonomic robot R, $c \in C$, and $r \in \mathbb{R}$. The **shortest feasible path ball of radius** r **centered at** c is defined as $\mathcal{B}_{SFP}(c, r) = \{q \in C : \rho_R(c, q) < r\}$.

If $\mathcal{B}_{Euc}(c, r)$ is the standard Euclidean ball of radius r about c, then $\mathcal{B}_{SFP}(c, r) \subset \mathcal{B}_{Euc}(c, r)$ since the SFP distance between two points is at least as large as the Euclidean distance.

The basic idea is to cover the section of the skeleton from s to g with SFP balls which lie completely in free space in order to make a series of "jumps" between the points $s = p_0, p_1, \ldots, p_{n-1}, p_n = g$ on the skeleton. If the robot is currently at configuration p_i then the next jump point p_{i+1} is determined by finding an SFP ball $\mathcal{B}_{SFP}(p_i, r_i)$ lying completely in free space, and then choosing the point in the ball furthest along the skeleton path as p_{i+1}. The shortest path from p_i to p_{i+1} has a length less than r_i since $p_{i+1} \in \mathcal{B}_{SFP}(p_i, r_i)$. Any path with length smaller than r_i avoids all configuration obstacles because the set of all reachable configurations $\mathcal{B}_{SFP}(p_i, r_i)$ lies completely in free space. Furthermore, the path from p_i to p_{i+1} is a shortest path and therefore of low complexity. For example the shortest paths of the planar two-axle car involve no more than five straight line or curved segments, and no more than two cusps (changes of directions).

Since the complexity of the path is fixed for the path between successive jump points, the complexity of the final path from s to g is proportional to the number of jump required. The number of jumps can be minimized by making the SFP balls $\mathcal{B}_{SFP}(p_i, r_i)$ as large as possible. As an example, consider the planar

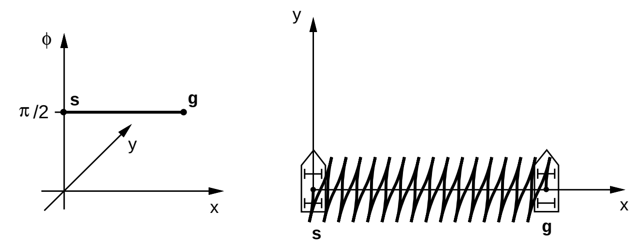

Figure 3: *Left: Portion of a skeleton through configuration space with start and goal points. Right: Resulting path through physical space when the skeleton is covered with small SFP balls.*

two-axle car with a portion of a skeleton shown in the left of figure 3, and suppose the skeleton segment from s to g is covered with small SFP balls. Then the jump points are closely spaced, and the resulting path in physical space would look like the one in the right of the figure. Note that the paths between the successive jump points are simple, but it is necessary to concatenate so many such paths together that the final total path is quite complex. In contrast if the skeleton path from s to g is covered with larger balls less jumps are required resulting in nicer path such as those shown in figure 4. The path on the left of figure 4 involves two jumps to bring the robot from s to g, while the one on the right brings the robot from s to g in a single jump. This latter case would occur if the SFP ball at s was large enough to contain g. For the physical paths in figure 4 the robot's configuration deviates farther from the skeleton, but this is of no consequence since the actual path is still guaranteed to avoid obstacles.

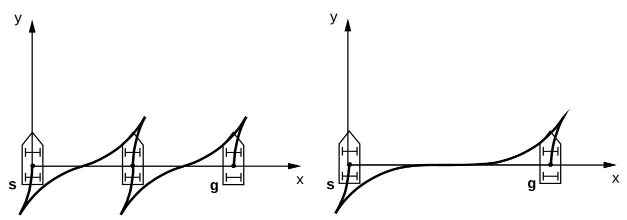

Figure 4: *Resulting paths when the skeleton is covered with larger SFP balls allowing for fewer jumps instead of many smaller ones.*

It should now be clear why the skeleton built as a set of maximal clearance points under the SFP metric is useful in planning paths. The SFP balls about points on the skeleton are as large as possible, and fewer jumps are required to move along the skeleton resulting in less complex final paths. Bellaiche has used SFP balls to determine lower bounds on path complexities in the presence of obstacles [2]. Essentially, the skeleton is designed so that the complexities of the resulting paths are as close as possible to the lower bounds.

4 Computation of the SFP distance

Computing the SFP distance between the robot's current configuration and a configuration obstacle is an important operation used often in constructing the skeleton. The prospect of implementing this operation efficiently initially appears dubious, since finding shortest feasible paths for even a simple nonholonomic robot like the two-axle car in the plane is non-trivial. Worse yet the shortest paths for some robots are unknown, and some approximation or search by brute force enumeration of paths is required. These facts suggest computing a lookup table off-line for determining the SFP distance between two points in configuration space. Several properties of such a scheme are desirable:

1. For efficiency, computing the actual configuration obstacles should be avoided. The calculation of the SFP distance between the robot's configuration and a configuration obstacle must only involve the physical obstacle in physical space.

2. The table should be computed only once for a particular robot. Placing the robot in a new environment should not require a new table.

3. The dimension of the table should be as small as possible. Since ρ_R is a map from $C \times C$, a naive approach would involve a $2n$-dimensional table where n is the dimension of the configuration space. This is too large for any practical robot.

4. There should be a method of planning paths in environments larger than the size of the table. Generally it is not practical to build a table large enough to cover the largest expected environment or to impose size constraints on the environment.

Our method satisfies all of these desired properties. In the following discussion P and C represent the physical and configuration space of the robot.

4.1 Table reduction via rigid body motions

Suppose the SFP distance between a start configuration s and a goal configuration g is to be found. If a rigid body motion is applied to the two robot configurations s and g to obtain new configurations s' and g', then $\rho_R(s,g) = \rho_R(s',g')$ (see figure 5) In fact applying the rigid body motion to the the trace in P of the shortest feasible path between configurations s and g yields the trace in P of the shortest feasible path between s' and g'.

The rigid body motions for $\mathbb{R}^2$ and $\mathbb{R}^3$ are given by $SE(2) \simeq \mathbb{R}^2 \times S_1$ and $SE(3) \simeq \mathbb{R}^3 \times SO(3)$ respectively. The lookup table is simplified if a rigid body motion that brings the start configuration s to some home position is always applied prior to lookup. Then the domain of the table need not be $C \times C$ but only $(C/SE(i)) \times C$ where i is the dimension of physical space. (Technically, the domain is $\tilde{C} \times C$, where $\tilde{C}$ is the natural embedding of $C/SE(i)$ in C.)

For example, with the two-axle car in the plane both C and $SE(2)$ are $\mathbb{R}^2 \times S_1$ and so $C/SE(2)$ is a single point. Hence every initial start configuration s

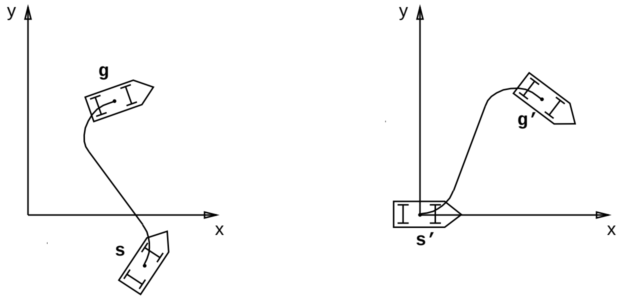

Figure 5: *Applying a rigid body motion to the start and goal configurations does not change the SFP distance between the configurations.*

can be transformed via rigid body motion to the same (arbitrary) home configuration, such as the car positioned at the origin and pointing along the positive x-axis. The domain of the lookup table is isomorphic to C, since we only need to compute the SFP distance between this home configuration and any other configuration. An analogous situation arises if the robot is an aircraft with $P = \mathbb{R}^3$ since the space of rigid body motions $SE(3)$ is again equal to the configuration space C. If the robot is the planar car with k trailers, a slightly different situation arises. The configuration of the robot may be specified by the position and orientation of the lead car plus the orientation of each trailer, so $C = \mathbb{R}^2 \times S_1 \times S_1 \times \ldots \times S_1$ ($k+1$ S_1 factors). Thus $C/SE(2) = S_1 \times S_1 \times \ldots \times S_1$ (k S_1 factors), which is no longer a single point. It is not possible to bring all starting configurations to a single home configuration using rigid body motions. Instead the home position is actually a non-trivial subspace of configuration space. The car can not always be brought to the origin with the trailers lined up along the negative x-axis, however it is always possible to bring the lead car to the origin pointing along the positive x-axis, while leaving the relative orientations of the trailers unchanged. The point is that while applying a rigid body motion to the start and goal configurations does not affect the SFP distance, other motions such as straightening out the trailers do.

4.2 Computing SFP distances in physical space

Attacking the problem of computing SFP distances in physical space leads to a further reduction in the size of the lookup table. Let O be an obstacle in physical space and CO the corresponding configuration obstacle. The SFP distance between the robot at configuration p and the configuration obstacle CO is defined as

$$d_{SFP}(p, CO) = \min_{q \in CO} \rho_R(p, q)$$

Consider a single point $w \in P$ on the physical obstacle O, and let $\mathcal{C}(w) \subset C$ be the configuration obstacle of this single point. Clearly

$$CO = \bigcup_{w \in O} \mathcal{C}(w).$$

Now define a map $\rho'_R : C \times P \to \mathbb{R}$ by

$$\rho'_R(p, w) = \min_{q \in \mathcal{C}(w)} \rho_R(p, q).$$

The SFP distance between the robot's configuration p and the configuration obstacle CO may then be defined as

$$d_{SFP}(p, CO) = \min_{w \in O} \rho'_R(p, w).$$

This is equivalent to the previous formulation, however in the second formulation the minimization is over the points in the physical obstacle O, not the configuration obstacle CO. The lookup table need not store the value of ρ_R over points in $(C/SE(i)) \times C$, but only the value of ρ'_R over points in $(C/SE(i)) \times P$. The size of the lookup table is reduced and the SFP distance computations are performed using physical obstacles rather than configuration obstacles.

4.3 Building the table off-line

Each table entry corresponds to a particular home configuration s and a particular point w in physical space, and contains the length of the shortest feasible path which moves the robot from configuration s to a configuration intersecting w. Hence it is necessary to minimize over all goal configurations g for which the robot intersects the point w in physical space.

As an example, consider computing $\rho'_R(p, w)$ for the two-axle car robot. For this robot $C/SE(2)$ is a single point, so s is the same for all table entries (suppose for example that s is the configuration with the car at the origin pointing along the positive x-axis). For a fixed final orientation ϕ, the shortest path must be found from s to a configuration at which the robot intersects the point w. This problem can be transformed by growing the obstacle w and shrinking the robot to a point. However since the obstacle itself is simply a point, the grown obstacle is just the "negative" of the shape of the robot (figure 6). The problem becomes one of finding the shortest feasible path that brings the point robot to some point on the edge of the grown obstacle. This procedure is performed for each final orientation ϕ. Minimizing over all final orientations yields the value to be stored in the table, that is $\rho'_R(s, w)$. Clearly the calculations do not involve the obstacles in the robot's environment, and thus may be performed completely off-line.

For more complex robots the minimization problem becomes higher dimensional, but the basic approach is the same. For a car with one trailer the minimization would be performed over all pairs (ϕ_1, ϕ_2) describing the final orientations of the car and trailer, rather than just over a single orientation parameter in the previous example. Nonetheless these are still "one time" calculations for a given robot.

4.4 Approximating SFP distances outside table

The requirement that the table be of finite size remains. By bringing the robot to the origin of physical space (but not configuration space) via rigid body motions, the table can be constructed over $(C/SE(i)) \times \tilde{P}$ where $\tilde{P} \subset P$ is some compact region containing the origin. The finite table is then used for computing

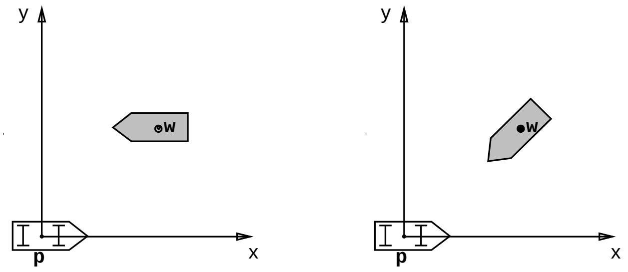

Figure 6: *The shaded regions are the result of growing the point obstacle w and shrinking the robot to a point. Left: $\phi = 0$. Right: $\phi = \pi/4$.*

SFP distances between the robot and *nearby* obstacles. Distant obstacles fall outside the range of the table, however the SFP distances to these may be approximated by a simpler metric. For example the standard Euclidean metric is a very good estimate of the actual SFP metric for the planar two-axle car at large distances.

5 Computing d_{SFP} derivatives

The skeleton is built by incrementally extending a path through configuration space. During this process it is often necessary to move directly away from an obstacle or to move such that the distances to the two or more closest obstacles change by the same amount. For such operations it is useful to know how the distance between a given configuration s and a configuration obstacle CO changes with respect to the coordinates parameterizing s. Recall the distance function $d_{SFP} : C \times \mathcal{CO} \to \mathbb{R}$ where $\mathcal{CO}$ is the set of configuration obstacles. For a given configuration $s \in C$ and configuration obstacle $CO \in \mathcal{CO}$, $d_{SFP}(s, CO)$ is the length of the shortest feasible path from s to a configuration $g \in CO$. If s is specified by the coordinate variables $x_1, x_2, \ldots, x_n$, what is desired are the values $\frac{\partial}{\partial x_i} d_{SFP}(s, CO), i = 1 \ldots n$. As with the distance calculations it is desirable to compute these derivatives using only the physical obstacles rather than computing the configuration obstacles. For efficiency the derivatives should also be computed from a lookup table computed off-line once for a given robot. We now analyze how this may be done.

Let s be the robot configuration, CO a configuration obstacle, and $g \in CO$ the closest point on the configuration obstacle to s under the SFP metric. If s is displaced a distance ϵ the closest point g may move along the surface of CO. However since g is the closest point on CO to s, the first order derivatives of ρ_R as g moves along any curve on the surface of CO are nondecreasing (the derivatives are zero along any curve on the stratum containing g). Thus the first order derivatives of d_{SFP} may be computed assuming g is fixed on CO, that is

$$\frac{\partial}{\partial x_i} d_{SFP}(s, CO) = \frac{\partial}{\partial x_i} \rho_R(s, g).$$

The right hand side of the above equation is easy to compute off-line when the distance table is generated. Let O be the physical obstacle corresponding to CO and w the physical point on O corresponding to g ($g \in \mathcal{C}(w)$ for some $w \in O$). Clearly g is the closest point on $\mathcal{C}(w)$ to s since $\mathcal{C}(w) \subset CO$. The closest point to s on $\mathcal{C}(w)$ is known at the time $\rho'_R(s, w)$ is computed, so the derivatives may be evaluated at this time. In most cases closed form derivatives are not available and the derivatives must be evaluated numerically. The derivatives are stored in the same table as the ρ'_R distances, so each table entry actually consists of a distance and n derivatives where n is the dimension of configuration space.

Note that this scheme allows computing the derivatives of the SFP distance to a configuration obstacle in physical space; the configuration obstacle is never actually computed.

The derivatives taken from the table are derivatives with respect to the configuration coordinates when the robot is at some home configuration. When a rigid body motion is applied to the actual configuration s bring it to a home configuration s', the derivatives computed at s' must be mapped back to the corresponding derivatives at s. This is not a problem since the rigid body transformation bringing s to s' establishes an isomorphism between their tangent spaces.

Outside of the range of the table the Euclidean metric is a good approximation of the SFP metric. Therefore outside of the range of the lookup table the derivatives of $d_{SFP}(s, CO)$ are estimated by their Euclidean analogues.

6 The Planning Algorithm

Once the metric to be used for building the maximum clearance skeleton and its implementation have been defined, the planning algorithm becomes an application of Canny's general roadmap algorithm [4]. We now give a brief description of the algorithm.

The roadmap algorithm is similar to a plane sweep algorithm. One of the coordinates of configuration space is chosen as a sweeping direction, and successive slices of configuration space are taken, each slice being a (hyper) plane through configuration space with the sweeping coordinate held fixed. For example, the car is described by three parameters, a position (x, y) in the plane and an orientation ϕ. Physical space is 2-dimensional. If x is chosen as the sweeping direction, slices taken through configuration space would be planes of constant x value. Another key concept of the roadmap algorithm is that of a silhouette curve.

Definition. Within a given slice of configuration space, let p be a point of maximal clearance from the obstacles, i.e. moving away from p in any direction within the slice decreases the SFP distance to the nearest obstacle. Points corresponding to p may be located in successive slices as the sweeping coordinate is varied; this locus of points is the **silhouette curve** passing through p.

Intuitively, a silhouette curve is generated by first finding a local maxima of the *distance to nearest obstacle* function within a slice, and then tracing this maxima through the slices as the sweeping coordinate is varied. There is also the notion of a critical slice where two or more previously disconnected regions of free space are joined, or a single region splits into disconnected regions. Such slices intersect more than one silhouette curve. The overall planning algorithm is now described.

1. Compute the values of ρ_R and the partial derivatives of d_{SFP} over a compact region about the origin of physical space; store these values in a lookup table.
2. Given a start configuration s_0 and a goal configuration g_0 find curves to points s and g on a skeleton silhouette curve.
3. Trace the unexplored silhouette curves in both directions. If a complete path along silhouette and linking curves is found from s to g, go to step 5.
4. In a critical slice locate points on unexplored silhouette curves and go to step 3.
5. Plan a path from s_0 to g_0 by jumping along the curve linking s_0 to s, jumping along the constructed skeleton from s to g, and finally jumping along the curve linking g to g_0.

The first step is performed only once (off-line) for a particular robot. The final step is performed after the necessary portions of the skeleton have been computed. Steps 2 through 4 are a version of the roadmap algorithm; skeleton construction occurs during these steps. These steps are now described using the planar two-axle car as an example.

The car's configuration is parameterized by a position (x, y) and an orientation ϕ; assume that x is chosen as the sweeping direction. Given an x-slice through configuration space it is simple to find a point in the slice located on a silhouette curve. Beginning at some initial point in the slice, we incrementally move away from the nearest obstacle or obstacles while remaining in the slice. Eventually a point will be reached from which it is impossible to move without moving closer to one of the nearest obstacles. Such a point is maximally clear from the obstacles and is thus by definition on a silhouette curve. This is the procedure that occurs in step 2 above. Note that the derivatives of the d_{SFP} function are used in moving away from the nearest obstacle(s).

In step 3 silhouette curves are "traced." Once a point on a silhouette curve is located, the silhouette curve can be traced by taking successive x-slices separated by some small Δx. Suppose $p = (x, y, \phi)$ is a point on a silhouette curve in the current x-slice. To find the point $p' = (x + \Delta x, y + \Delta y, \phi + \Delta\phi)$ on the silhouette curve in the next slice, Δy and $\Delta\phi$ must be chosen so that the distances to all of the nearest obstacles change by the same amount. An initial approximation to p' can be obtained by choosing Δy and $\Delta\phi$ such that the inner products of the vector $(\Delta x, \Delta y, \Delta\phi)$ with the gradients of the distance functions to the nearest obstacles are all equal. Again the derivatives of the d_{SFP} function are required. Silhouette curves are traced in both directions by sweeping in both directions from the current slice, i.e. taking Δx to be positive and negative.

If after tracing the silhouette curves from s and g there is no path which links the two points, new silhouette curves must be found and traced. The search for unexplored silhouette curves occurs in a critical slice, and the procedure is similar to that of step 2; we look for new local maxima of the *distance to nearest obstacle* function within the critical slice. After points on unexplored silhouette curves are found, the process is repeated starting at step 3. Steps 3 and 4 are repeated as long as necessary, until a complete path is found from s to g.

The above ideas are only an overview of the roadmap algorithm. For a more complete description of the algorithm and how it may be specialized for computing a roadmap using only the physical space obstacles we refer the reader to [5].

7 The 2-axle car & Reeds-Shepp paths

The techniques described in this paper are currently being applied to the planar two-axle car. Planning for this system exhibits the same basic difficulties inherent in more complex nonholonomic systems, and the shortest paths for the planar car have been studied extensively. In fact Reeds and Shepp have completely characterized the shortest paths between any two configurations of the planar car, and an algorithm exists for computing these paths [20]. Briefly, the shortest paths consist of five or fewer path segments, each segment being a straight line or a turn with a radius of curvature equal to the minimum turning radius of the car. Enumerating over the various possibilities such as turn direction and order of segments produces forty-eight possible path types. Currently each of these path types must be "tried" when searching for the shortest paths between configurations s and g. Although not all forty-eight types can move the car between a given s and g, in general many of them can and the only way to find the shortest is to try (compute) them all. An interesting problem is to try to initially reduce the set of forty-eight paths to a smaller set of candidates to avoid computing so many paths, but this problem is not the focus of this research.

For our planner the important consequence of Reeds and Shepp's work is that the SFP metric is computable for the planar car. We received a subroutine for computing Reeds-Shepp paths from Michel Taix, who wrote the C code as part of a complete planner for car-like robots [16]. Figure 7 shows the level curves of the ρ'_R lookup table for the car-like robot (in generating this table the car was assumed to be a point, although this assumption is not necessary). Intuitively it should be easier for the car to move forward or backward (along the x-axis) than sideways (along the y-axis), and this intuition is verified by the level curves. Note the level curves become more and more circular as the distance increases, justifying the claim that at large distances the Euclidean distance closely approximates the SFP distance.

8 Generalizing to other systems

The planar two-axle car is a fairly well understood nonholonomic system, at least as far as shortest paths

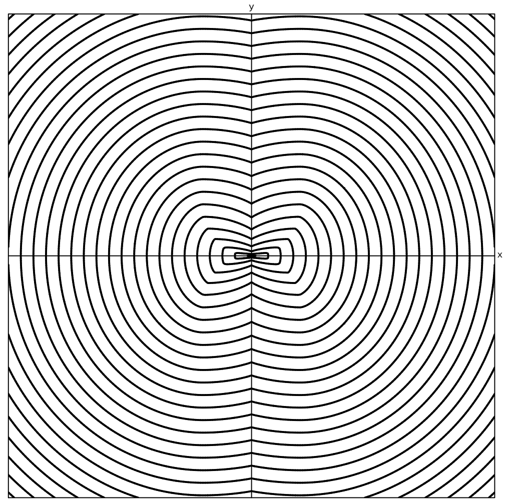

Figure 7: *Level curves of ρ'_R for the two-axle planar car. The region shown is a 10 $\times$ 10 square centered at the origin, with contours 0.5 units apart.*

are concerned. Furthermore there is a clear idea of what paths of low complexity should look like; they should be short and also have as few reversals as possible. For more complex nonholonomic systems the shortest paths between configurations may not be known, and the idea of what constitutes a simple path may also be unclear. We now discuss generalizing our algorithm to such systems.

8.1 Relaxing the SFP assumption

For an aircraft type robot finding the shortest paths between two configurations (positions and orientations in $\mathbb{R}^3$) is still an open problem. Planning for such systems using the skeleton method is still possible with a slight modification. Even when the shortest paths are not known, a "library" of paths can be found which are able to steer the system (non-optimally) between any two configurations. For example, the path library for the aircraft would include various looping paths for bringing the aircraft to the same point in space but with a different orientation.

Using this smaller set of library paths, an algorithm can be constructed for finding a path between any two configurations. Using rigid body motions to bring the robot to a home configuration makes it necessary to find paths only between the home configuration(s) and points in a compact set about the origin in physical space. Let B be a ball about the origin. Moving the robot from a home configuration to a point in B may involve a path which temporarily passes outside of B. However, a larger ball $B' \supset B$ can be found such that a path from a home configuration to any point in B never passes outside of B'. Note that the paths must come from the pre-defined library of paths.

If the library of paths consists of shortest paths, then the ball B' coincides exactly with the ball B. In the more general setting when the library paths are not necessarily shortest paths B' is strictly larger than B. The metric for constructing the skeleton should be based on the B' balls, since obstacle avoidance can only be guaranteed by insuring the B' ball lies completely in free space. When moving along the skeleton, the B balls are used. A path between skeleton jump points p_i and p_{i+1} will avoid obstacles if the point p_{i+1} lies within the B ball centered at point p_i. All of this rests on the fact that the paths between jump points will come from the same library of paths used to construct the distance table for the robot. Any choice of library paths is valid, as long as the same ones are used in the table building and planning stages. Of course the planner will be too conservative if the B' balls are much larger than the B balls, keeping the robot farther than necessary from the obstacles. A judicious choice of library paths will result in smaller discrepancy between B' and B balls, resulting in a better planner. A library consisting of shortest paths is optimal in this sense.

8.2 Defining path complexity in general

Defining path complexity for the planar car is fairly straightforward. In this case path complexity is an increasing function of the arc length of the path and the number of reversals along the path. However this example also shows that the concept of what makes a desirable (simple) path is dependent on the particular system. In the case of the car, there are two control inputs: the forward velocity and the steering angle of the front wheels. Changing the sign of the forward velocity input results in a reversal for the car, which is to be avoided. Yet changing the sign of the steering angle only causes the car to move from (say) a left turn path segment to a right turn path segment. Qualitatively, this latter occurrence is not as undesirable as a change of direction. A path containing several left-right switches seems better than a path of the same length which contains several forward-backward switches. Hence it seems that path complexity is closely tied to the specific system at hand.

Nonetheless, it is still possible to define some more general measure of path complexity. Lafferriere and Sussman have shown that a nilpotentizable system can be steered between two configurations by following a finite number of subpaths, where only one of the control inputs is non-zero over any one subpath [11]. The number of subpaths is bounded by some constant M for the particular system. For a nonholonomic robot representable as a nilpotentizable system, one can thus choose the path library to comprise all paths which are composed of M or fewer such path segments. The number of subpaths between successive jump points on the skeleton will then be bounded, and the algorithm will generate simple paths under this notion of complexity.

Acknowledgement: We would like to thank Michael Taix for the code to compute Reeds-Shepp Paths.

References

[1] J. Barraquand and J-C. Latombe. On nonholonomic mobile robots and optimal maneuvering. In *4th International Symposium on Intelligent Control*, Albany, NY, 1989.

[2] A. Bellaiche. Lower bounds on path complexity with obstacles. Workshop on nonholonomic planning, Toulouse, France, July 1991. Paris VII.

[3] A. Bellaiche, J-P. Laumond, and P. Jacobs. Controllability of car-like robots and complexity of the motion planning problem with non-holonomic constraints. In *International Symposium on Intelligent control*, Bangalore, India, 1991.

[4] J. F. Canny. *The Complexity of Robot Motion Planning*. M. I. T. Press, Cambridge, 1988.

[5] John F. Canny and Ming C. Lin. An opportunistic global path planner. In *International Conference on Robotics and Automation*, 1990.

[6] S. Fortune and G. Wilfong. Planing constrained motion. In *STOCS*, pages 445–459, Chicago, IL, May 1988. Association for Computing Machinery.

[7] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *International Conference on Robotics and Automation*, pages 2–7. IEEE, May 1989.

[8] P. Jacobs and J. Canny. Robust motion planning for mobile robots. In *International Conference on Robotics and Automation*. IEEE, 1990.

[9] P. Jacobs, J-P. Laumond, and M. Taix. A complete iterative motion planner for a car-like robot. In *Journees Geometrie Algorithmique*, INRIA, 1990.

[10] P. Jacobs, J-P. Laumond, M. Taix, and R. Murray. Fast and exact trajectory planning for mobile robots and other systems with non-holonomic constraints. Technical Report 90318, LAAS/CNRS, Toulouse, France, September 1990.

[11] G. Lafferriere and H. J. Sussman. Motion planning for controllable systems without drift: A preliminary report. Technical Report SYSCON-90-04, Rutgers Center for Systems and Control, June 1990.

[12] J-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[13] J-P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *Intelligent Autonomous Systems*. North Holland, 1987.

[14] J-P. Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *International Joint Conference on Artificial Intelligence*, pages 1120–1123, 1987.

[15] J-P. Laumond. Nonholonomic motion planning versus controllability via the multibody car system example. Technical Report STAN-CS-90-1345, Department of Computer Science, Stanford University, October 1990. (preprint).

[16] J-P. Laumond, M. Taix, and P. Jacobs. A motion planner for car-like robots based on a mixed global/local approach. In *IEEE International Workshop on Intelligent Robots and Systems*, Japan, 1990.

[17] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[18] R. M. Murray and S. S. Sastry. Grasping and manipulation using multifingered robot hands. In R. W. Brockett, editor, *Robotics: Proceedings of Symposia in Applied Mathematics, Volume 41*, pages 91–128. American Mathematical Society, 1990.

[19] R. M. Murray and S. S. Sastry. Steering nonholonomic systems using sinusoids. In *IEEE Control and Decision Conference*, 1990.

[20] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2), 1990.