Toward a Deeper Understanding of Motion Alternatives via an Equivalence Relation on Local Paths

Ross A. Knepper   Siddhartha S. Srinivasa   Matthew T. Mason

Abstract

Many problems in robot motion planning involve collision-testing a set of local paths. In this paper, we propose a novel solution to this problem by exploiting the structure of paths and the outcome of previous collision tests. Our approach circumvents expensive collision tests on a given path by detecting that the entire geometry of the path has effectively already been tested on a combination of other paths. We define a homotopy-like equivalence relation among local paths to detect this condition, and we provide algorithms that (1) classify paths based on equivalence, and (2) circumvent collision testing on up to 90% of them. We then prove both correctness and completeness of these algorithms and provide experimental results demonstrating a performance increase up to 300% on the rate of path tests. Additionally, we apply our equivalence relation to the navigation problem in a planning algorithm that takes advantage of information gained from equivalence relationships among collision-free paths. Finally, we explore applications of path equivalence to several other mechanisms, including kinematic chains and medical steerable needles.

1 Introduction

Planning bounded-curvature paths for mobile robots is an NP-hard problem (Reif and Wang, 1998). Many nonholonomic mobile robots thus rely on hierarchical planning architectures (Allen et al., 2007; Kelly et al., 2006; Knepper and Mason, 2008; Marder-Eppstein et al., 2010), which decompose the problem into at least two layers: a slow global planner and fast local planner (Figure 1). We focus on the local planner (Algorithm 1 and Algorithm 2), which iterates in a tight loop searching through a set of paths and selecting the best path among them for execution. Within each loop, the planner tests many paths before making an informed decision. The bottleneck in path testing is collision checking (Sánchez and Latombe, 2002). In this paper, we introduce a novel approach that delivers a significant increase in path set collision-testing performance by exploiting the fundamental geometric structure of paths.

We introduce an equivalence relation intuitively resembling the topological notion of homotopy. Two paths are *path homotopic* if a continuous, collision-free deformation with fixed start and end points exists between them (Munkres, 2000). Like any path equivalence relation, homotopy partitions paths into equivalence classes. Different homotopy classes make fundamentally different choices about their route amongst obstacles. However, two constraints imposed by mobile robots translate poorly into homotopy theory: limited sensing and constrained action.

The robot may lack a complete workspace map, which it must instead construct incrementally from sensor data. Since robot perception is limited by range and occlusion, a robot's understanding of obstacles blocking its movement evolves as it moves. A variety of sensor-based planning algorithms have been developed to handle such partial information. Obstacle avoidance methods, such as potential fields (Khatib, 1985), vector field histograms (Borenstein and Koren, 1991), and the curvature-velocity method (Simmons, 1996), are purely reactive. The bug algorithm (Lumelsky

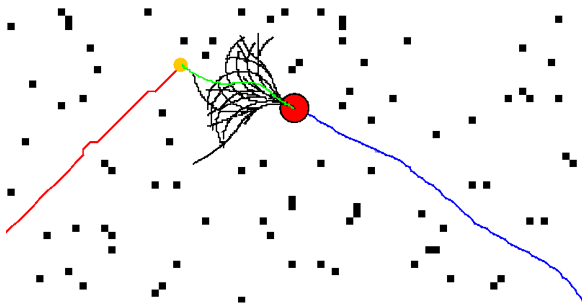

Figure 1: An example hierarchical planning scenario. The local planner's path set expands from the robot, at center, and feeds commands to the robot based on the best path that avoids obstacles (black squares). The chosen local path and corresponding global path (lower left) combine to form a proposed path to the goal.

and Stepanov, 1987), which generates a path to the goal using only a contact sensor, is complete in two dimensional spaces. A planner using the hierarchical generalized Voronoi graph, a roadmap with global line-of-sight accessibility (Choset and Burdick, 1995) achieves completeness in higher dimensions using range readings of the environment. Yu and Gupta (2000) propose a planner that iteratively constructs a probabilistic roadmap in response to partial sensed information about the world. Actions are selected on the basis of maximizing information gain for future plan steps. Our local planner resembles these algorithms in that it reacts to local obstacles while receiving global guidance about the direction to the goal.

If a robot is tasked to perform long-range navigation, then it must plan a path through unsensed regions. A low-fidelity global planner (i.e. one ignoring constraints) generates this path because we prefer to avoid significant investment in this plan, which will likely be invalidated later. Path homotopy, in the strictest sense, requires global knowledge of obstacles because homotopy equivalent paths must connect fixed start and goal points.

Relaxing the endpoint requirement of homotopy avoids reasoning about the existence of far-away, unsensed obstacles. In naively relaxing a fixed endpoint, our paths might be permitted to freely deform around obstacles, making all paths equivalent. To restore meaningful equivalence classes, we propose an alternate constraint based on path shape. Such path shape constraints stem from the nonholonomic motion constraints inherent to many mobile robots. Laumond (1986) first highlighted the importance of nonholonomic constraints and showed that feasible paths exist for a mobile robot with such constraints. Barraquand and Latombe (1993) created a grid-based planner that innately captures these constraints. LaValle and Kuffner (2001) proposed the first planner to incorporate both kinodynamic constraints and random sampling. In contrast to nonholonomic constraints, true homotopy forbids restrictions on path shape; two paths are equivalent if *any* path deformation exists between them. By restricting our paths to bounded curvature, we represent only feasible motions while limiting paths' ability to deform around obstacles. The resulting set of path equivalence classes is of immediate importance to the planner (Figure 2). The number of choices represented by these local equivalence classes relates to Farber's topological complexity of motion planning (2003).

Various planners have employed equivalence classes to reduce the size of the search space. In task planning, recent work has shown that equivalence classes of actions can be used to eliminate redundant search (Gardiol and Kaelbling, 2007). In motion planning, path equivalence often employs homotopy. A recent paper by Bhattacharya et al. (2010) provides a technique based on complex analysis for detecting homotopic equivalence among paths in 2D. Two papers employing

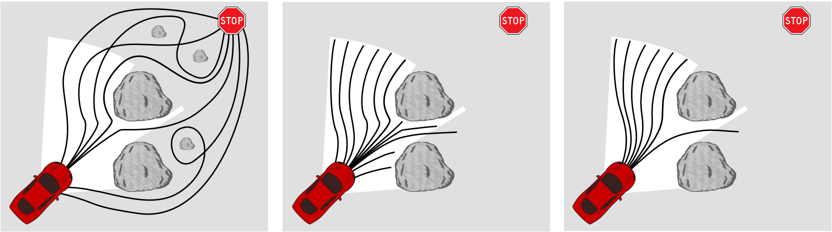

Figure 2: left: Paths from a few distinct homotopy classes between the robot and the goal. The distinctions between some classes require information that the robot has not yet sensed (the gray area is out of range or occluded). **middle:** With paths restricted to the sensed area, they may freely deform around visible obstacles. **right:** After restricting path shape to conform to motion constraints, we get a handful of equivalence classes that are immediately applicable to the robot.

equivalence classes to build probabilistic roadmaps (Kavraki et al., 1996) are by Schmitzberger et al. (2002) and Jaillet and Simeon (2008). The latter paper proposes the visibility deformation, a departure from true homotopic equivalence that restricts continuous deformation to line-of-sight visibility between paths. We propose here a different variation on homotopy. Not only do we restrict continuous deformation between paths, but we also fix path length to create a purely local path equivalence relation.

Our key insights are twofold: first, that this *local path equivalence* reveals shared outcomes among a set of paths; and second, that this relation enables a planner to reason collectively about such path sets instead of considering each path in isolation. These insights are based on the observation that two equivalent neighboring paths represent swept volumes of the robot that cover some common ground in the workspace. Between them lies a continuum of paths whose swept volumes are covered by the two bounding paths.

This paper makes several contributions relating to local path equivalence. We develop the mathematical foundations to detect equivalence relations among all local paths based on a finite precomputed path set. We utilize these tools to devise an efficient algorithm for detecting equivalence among a discretely sampled path set. By mapping local equivalence of discrete paths to the underlying continuum, we give an algorithm for implicitly collision-testing local paths, thus circumventing the usual expensive test. Furthermore, we propose a navigation algorithm for mobile robots that utilizes local path equivalence to improve overall execution success rates by understanding route alternatives and selecting routes most likely to permit safe execution.

1.1 Outline

The remainder of the paper is organized as follows. We provide an implementation of the basic classification algorithm in Section 2 and present the fast collision-testing technique. Next, Section 3 explores implications of local path equivalence to improving mobile robot navigation performance. Section 4 then delves into the theoretical foundations of our path equivalence relation. Section 5 provides some experimental results. Finally, Section 6 briefly explores several promising applications to other robotic systems, including kinematic chains such as manipulator arms, and steerable needles used in medical procedures.

2 Path Classification and Safety Algorithms

In this section, we present two new algorithms for path classification and implicit path collision-testing. We also borrow a path set generation routine from prior work. All of the algorithms presented here run in polynomial time with respect to path count. Throughout this paper, we use p to refer to a path and $\mathcal{P}$ to refer to a set of paths.

Definition 1 (path space)**.** *The* ***path space*** *is a metric space* $(\mathcal{P}, \mu)$ *in which the metric* μ *is used to measure the distance between a pair of paths in* $\mathcal{P}$*. Paths may vary in shape and length.* □

2.1 Path Set Generation

We use the greedy path set construction technique of Green and Kelly (2007). The algorithm iteratively builds a path set sequence $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$ by drawing paths from a densely-sampled source path set, $\mathcal{P}_N \subset \mathcal{X}$. $\mathcal{X}$, the continuum path space, is the set of all output paths corresponding to all possible control inputs. This set is never explicitly represented.

Instead, we compute the set $\mathcal{P}_N$, which can be created for any continuous system by finely discretizing the input parameters and varying them in combination. The requisite granularity of input discretization depends on characteristics of the particular system's mapping between input parameters and output paths. Note that in the construction of path set $\mathcal{P}_N$, no care need be given to uniformity of sampling among path shapes; with sufficient initial density, the Green-Kelly algorithm will provide approximate uniformity (based on some metric μ) at a range of sampling resolutions.

At step i, the Green-Kelly algorithm selects the path $p \in \mathcal{P}_N$ that minimizes the dispersion of $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{p\}$. Borrowing from Niederreiter (1992):

Definition 2 (dispersion)**.** *Given a bounded metric space* $(\mathcal{X}, \mu)$ *and a set* $\mathcal{P} = \{x_1, \ldots, x_n\} \in \mathcal{X}$, *the* ***dispersion*** *of* $\mathcal{P}$ *in* $\mathcal{X}$ *is defined by*

$$\delta(\mathcal{P}, \mathcal{X}) = \sup_{x \in \mathcal{X}} \min_{p \in \mathcal{P}} \mu(x, p)\,. \quad \square \tag{1}$$

The dispersion of $\mathcal{P}$ in $\mathcal{X}$ equals the radius of the biggest open ball in $\mathcal{X}$ containing no points in $\mathcal{P}$. By minimizing dispersion, we ensure that there are no large voids in path space. Thus, dispersion reveals the quality of $\mathcal{P}$ as an "approximation" of $\mathcal{X}$ because it guarantees that for any $x \in \mathcal{X}$, there is some point $p \in \mathcal{P}$ such that $\mu(x, p) \leq \delta(\mathcal{P}, \mathcal{X})$. Note that all paths in $\mathcal{X}$ are of fixed length and share a start state. This condition is sufficient to assure that $\mathcal{X}$ is bounded for a wide variety of path metrics.

The Green-Kelly algorithm generates a sequence of path sets $\mathcal{P}_i$, for $i \in \{1, \ldots, N\}$, that has monotonically decreasing dispersion. In seeking a path to execute, the local planner algorithm (Algorithm 1) searches paths in this order, thus permitting early termination while ensuring that a low-dispersion set of paths is collision tested. Note that although the source set $\mathcal{P}_N$ is of finite size—providing a lower bound on dispersion at runtime—it can be chosen with arbitrarily low dispersion in $\mathcal{X}$ a priori.

2.2 Path Classification

We next present Algorithm 3, which classifies collision-free members of a path set. The Hausdorff metric is central to the algorithm. Intuitively, this metric returns the largest amount of separation between two paths in the workspace. From Munkres (2000):

$$\mu_H(p_i, p_j) = \inf_{\varepsilon}\{p_i \subset (p_j)_\varepsilon \text{ and } p_j \subset (p_i)_\varepsilon\}, \tag{2}$$

```
Algorithm 1 Local_Planner_Algorithm($w$, $x$, $h$, $\mathcal{P}$)
Input: $w$ – a costmap object;
    $x$ – initial state;
    $h$ – a heuristic function for selecting a path to execute;
    $\mathcal{P}$ – a fixed set of paths
Output: Moves the robot to the goal if possible
while not at goal and time not expired do
    $\mathcal{P}_{free} \leftarrow Test\_All\_Paths(w, x, \mathcal{P})$
    $j \leftarrow h.Best\_Path(x, \mathcal{P}_{free})$
    $Execute\_Path\_On\_Robot(j)$
    $x \leftarrow Predict\_Next\_State(x, j)$
end while
```

```
Algorithm 2 $\mathcal{P}_{free} \leftarrow$Test_All_Paths($w$, $\mathcal{P}$)
Input: $w$ – a costmap object;
    $\mathcal{P}$ – a fixed set of paths
Output: $\mathcal{P}_{free}$, the set of paths that passed collision test
$\mathcal{P}_{free} \leftarrow \emptyset$
while time not expired and untested paths remain do
    $p \leftarrow Get\_Next\_Path(\mathcal{P})$
    $collision \leftarrow w.Test\_Path(p)$                // collision is Boolean
    if not collision then
        $\mathcal{P}_{free} \leftarrow \mathcal{P}_{free} \cup \{p\}$       // non-colliding path set
    end if
end while
return $\mathcal{P}_{free}$
```

where $(p)_r$ denotes dilation of p by r: $\{t \in \mathbb{R}^2 \colon \|t_p - t\|_{L2} \leq r \text{ for some } t_p \in p\}$. Note that μ_H satisfies all properties of a metric (Henrikson, 1999). We similarly define a normalized Hausdorff metric as

$$\mu_H(p_i, p_j, d) = \inf_{\varepsilon}\{p_i \subset (p_j)_{\varepsilon d} \text{ and } p_j \subset (p_i)_{\varepsilon d}\}. \tag{3}$$

These two Hausdorff metric notations are equivalent for now, but we exploit the normalized format in Section 6, where we replace the constant d with a function of path length. For our fixed path set generated by Green-Kelly and a given d, we precompute each pairwise path metric value of (3) and store them in a lookup table for rapid online access.

Algorithm 3 performs path classification on a set of paths that have already tested collision-free at runtime. We form an *equivalence graph* $G = (V, E)$ in which node $v_i \in V$ corresponds to path p_i. Edge $e_{ij} \in E$ exists, joining nodes v_i and v_j, when this relation holds:

$$\mu_H(p_i, p_j, d) \leq 1, \tag{4}$$

where d is the diameter of the robot. This condition is true when two paths are separated by at most one robot diameter. Taking the transitive closure of this relation, two paths p_a and p_b are equivalent if nodes v_a and v_b are in the same connected component of G (Figure 3).

In effect, this algorithm constructs a probabilistic roadmap (PRM) in the path space instead of the conventional configuration space. A query into this PRM tells whether two paths are equivalent.

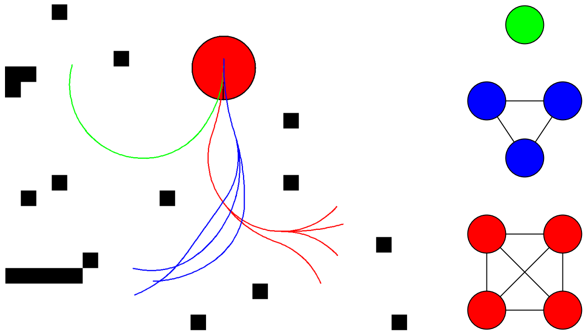

Figure 3: A simple path set, in which obstacles (black) eliminate colliding paths. The collision-free path set has three equivalence classes (red, green, and blue). In the corresponding graph representation, at right, adjacent nodes represent proximal paths. Connected components indicate equivalence classes of paths.

As with any PRM, a query is performed by adding two new graph nodes v_s and v_g corresponding to the two paths. We attempt to join these nodes to other nodes in the graph based on (4). The existence of a path connecting v_s to v_g indicates path equivalence.

2.3 Implicit Path Safety Test

There is an incessant need in motion planning to accelerate collision-testing, which may take up to 99% of total CPU time (Sánchez and Latombe, 2002). During collision-testing, the planner must verify that a given swath is free of obstacles.

Definition 3 (swath)**.** *A **swath** is the workspace area of ground or volume of space swept out as the robot traverses a path.* ☐

Definition 4 (safe)**.** *A path is **safe** if its swath contains no obstacles.* ☐

In testing many swaths of a robot passing through space, most planners effectively test the free workspace many times by testing overlapping swaths. We may test a path **implicitly** at significant computational savings by recalling recent collision-testing outcomes and circumventing new collision tests whenever possible. We formalize the idea in Algorithm 4, which is designed to be invoked from Algorithm 2, line 4 in lieu of the standard path test routine.

The implicit collision-test condition resembles the neighbor condition (4) used by Algorithm 3, but it has an additional "Is_Between" check, which indicates that the swath of the path under test is fully covered by two collision-free neighboring swaths. The betweenness trait can be precomputed and stored in a lookup table. Given a set of safe paths, we can quickly discover whether any pair covers the path under test. By precomputing eligible paths in the path set and efficiently tracking collision-free paths, the planner promises to realize significant performance gains, producing several times as many paths per second as conventional collision-test algorithms.

3 Route Selection

Thus far, we have shown that local path equivalence can produce significantly more collision-free paths per unit time than traditional collision testing. However, an earlier result (Knepper and Mason, 2008) found surprisingly that if a planner is given additional safe paths, its performance could actually decline. Given a coarsely-sampled path set $\mathcal{C}$ and a densely-sampled path set $\mathcal{D}$, we believe this effect is due to the fact that $\mathcal{D}$ is expected to contain a more optimal path than $\mathcal{C}$ that

Algorithm 3 $D \leftarrow$Equivalence_Classes($\mathcal{P}_{free}$, d)

```
Input: $\mathcal{P}_{free}$ – a set of safe, appropriate paths; $d$ – the diameter of the robot
Output: $D$ – a partition of $\mathcal{P}_{free}$ into equivalence classes (a set of path sets)
Let $G = (V, E) \leftarrow (\emptyset, \emptyset)$
$D \leftarrow \emptyset$                    // Partition of paths into classes (represented by a set of sets)
for all $p_i \in \mathcal{P}_{free}$ do                    // This loop discovers adjacency
    $V.add(p_i)$                    // Add a graph node corresponding to path $p_i$
    for all $p_j \in V \setminus \{p_i\}$ do
        if $\mu_H(p_i, p_j, d) \leq 1$ then
            $E.add(i, j)$                    // Connect nodes $i$ and $j$ with an unweighted edge
        end if
    end for
end for
$\mathcal{S} \leftarrow \mathcal{P}_{free}$                    // Unclassified paths
while $\mathcal{S} \neq \emptyset$ do                    // This loop finds the connected components
    $\mathcal{C} \leftarrow \emptyset$                    // Next connected component
    $p \leftarrow$ a member of $\mathcal{S}$
    $\mathcal{L} \leftarrow \{p\}$                    // List of nodes to be expanded in this class
    while $\mathcal{L} \neq \emptyset$ do
        $p \leftarrow$ a member of $\mathcal{L}$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{p\}$                    // Commit $p$ to class
        $\mathcal{S} \leftarrow \mathcal{S} \setminus \{p\}$
        $\mathcal{L} \leftarrow (\mathcal{L} \cup V.neighbors(p)) \cap \mathcal{S}$
    end while
    $D \leftarrow D \cup \{\mathcal{C}\}$
end while
return $D$
```

approaches closer to obstacles. In particular, $\mathcal{D}$ is likely to find risky narrow corridors that $\mathcal{C}$ might miss entirely. Therefore, we must establish that the additional safe paths can in fact be put to use in increasing navigation performance. We use path equivalence to help achieve this goal.

Since the local planner has a limited horizon, the resulting planned route is a concatenation of paths from the local and global planners. Only the local paths are feasible to execute directly on the robot, so the local planner must replan at regular intervals to allow continued progress. Thus, Algorithm 1 outputs a sequence of paths, the concatenation of which forms the true route. At the end of each replan cycle, the planner executes the beginning of the route representing the or least cost to the goal, a heuristic known as *Best_Path*. Using this heuristic, traditional hierarchical planners produce strongly goal-directed behavior that comes with two drawbacks: temporal incoherence and excessive obstacle proximity.

3.1 The Temporal Incoherence Problem

Temporal incoherence occurs because *Best_Path* does not generate consistent behavior between replan cycles, meaning that there is no deliberate process to maintain certain decisions throughout navigation. Often in hierarchical planning (Allen et al., 2007; Kelly et al., 2006), the ultimate route executed by the local planner algorithm is an emergent behavior because the planner lacks any continuity of intent between consecutive replan cycles. We propose local path equivalence as a means

Algorithm 4 $b \leftarrow$Test_Path_Implicit(p, w, $\mathcal{S}$, d)

```
Input: $p$ is a path to be tested
Input: $w$ is a costmap object          // used as a backup when path cannot be implicitly tested
Input: $\mathcal{S}$ is the set of safe paths found so far
Input: $d$ is the diameter of the robot
Output: $b$ – Boolean indicating whether path is safe
for all $p_i, p_j \in \mathcal{S}$ such that $\mu_H(p_i, p_j, d) \leq 1$ do
    if $p.Is\_Between(p_i, p_j)$ then                  // $p$'s swath has been tested previously
        $s_f \leftarrow p.Get\_End\_Point()$
        $collision \leftarrow w.Test\_Point(s_f)$      // endpoint may not be covered by swaths
        return $collision$
    end if
end for
return $w.Test\_Path(p)$                             // Fall back to explicit path test
```

of representing such continuity. In choosing a sequence of local paths, local planners implicitly also select a sequence of equivalence classes. This observation provides another perspective in which to view local path equivalence: based on information available to the local planner within a given replan cycle, the planned routes of all equivalent paths are homotopic. We propose a new algorithm to improve navigation performance by explicitly considering continuity within each replan cycle.

In general, we would like each replan cycle to select a new path that closely resembles the previous path, but such is not always the case. In prior work (Knepper and Mason, 2009), we proposed increasing the chance of such an outcome with the *Best_Path* heuristic by preserving the unexecuted remainder of the previous path as a **continuation**, which is considered along with the ordinary path set within subsequent cycles. Even so, on some occasions, consecutive replan cycles may switch equivalence classes, thus selecting a new planned route. *Best_Path* does not distinguish between classes, so such switches may happen arbitrarily often. Frequent switching is typically associated with perception noise. In especially noisy systems or where two planned routes are about equally costly, the planner may rapidly alternate between routes, thus effectively following an unplanned and undesirable path directly towards the obstacle separating the two routes.

3.2 The Obstacle Proximity Problem

Obstacle proximity, the second drawback incurred by *Best_Path*, risks the safety of the robot in cases of outside disturbance or internal prediction error. From a planning perspective, nearby obstacles also substantially reduce the quantity and diversity of safe paths available in subsequent replan cycles.

Two related approaches to the problem of decreasing robot proximity to obstacles have been in use for many years. The first approach involves "growing" the obstacles using a hard buffer (Buhmann et al., 1995), which runs the risk of closing off narrow openings. This problem is partially ameliorated by making the obstacle growth-radius vary in proportion to robot speed.

The second approach involves placing a soft buffer around each obstacle in the form of a gradient of elevated cost (Thorpe, 1984), such that cost varies inversely with obstacle proximity. Although this approach does not eliminate options from consideration, it is difficult to predict how a given cost function will affect decisions between corridors.

The drawback of both approaches is that they couple two distinct decisions: which route (equivalence class) to follow, and how to proceed (which path in the class) along that route. These de-

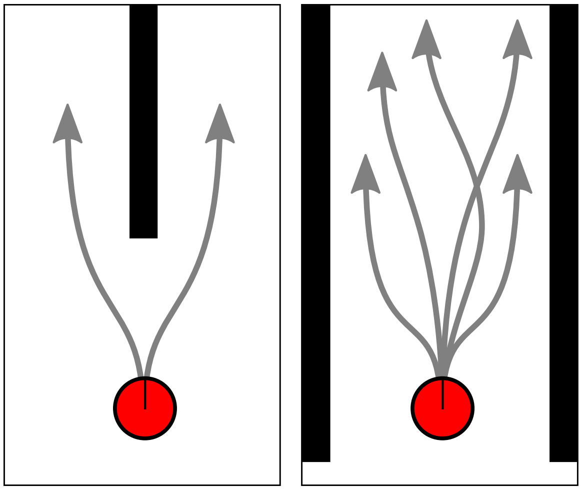

Figure 4: A navigating robot faces both discrete decisions (left) about which corridor to follow and continuous optimization (right) over where in the corridor to drive. A planner or controller should be able to consider these choices separately.

cisions are of qualitatively different character because continual fine-tuning is possible throughout the traverse of a corridor, but the choice of corridor to be traversed requires a discrete decision that soon becomes irreversible (Figure 4).

We introduce a new multi-stage path selection algorithm that separates these two decisions, thus allowing them to be weighed individually and traded off against one another. This process in turn improves planning and control flexibility, increasing continuity of plans, and retaining goal-directedness. Through application of a set of rules based on path equivalence (applied both within and across replan time steps), the algorithm selects paths for execution that guide the robot sufficiently far from obstacles while moving consistently towards the goal.

3.3 Logical Succession Path Relation

Having already demonstrated the value of path equivalence in a single replan cycle, we now introduce a relation on path equivalence classes to detect logical succession across multiple replan cycles.

Definition 5 (logical succession)**.** ***Logical succession*** *is a strict partial ordering among equivalence classes* $\mathcal{A} \rhd \mathcal{B}$ *such that some paths* $p_A \in \mathcal{A}$ *and* $p_B \in \mathcal{B}$ *exist for which* $\mu_H(p_A, p_B, d) \leq 1$ *and* $\mathcal{A}$ *was generated in an earlier replan time step than* $\mathcal{B}$*.* □

This definition establishes that two paths covering largely the same terrain but produced by different replan cycles can be said to follow the same route. The definition assumes a small time increment between replan cycles, such that little ground is covered in the interim. For larger steps, the definition would instead need to compare the end of p_A to the start of p_B. Of course, the pairwise logical succession property can be precomputed for our fixed path set in order to optimize performance.

In considering a new path for execution, logical succession provides a powerful tool for a planner to distinguish between paths that represent major and minor alterations to the prior plan. Suppose the planner just executed path p_i at time step $t-1$. We may initially choose to consider at time t only those paths p_j such that $[p_i] \rhd [p_j]$, where $[p]$ describes the equivalence class containing p. This restriction provides continuity of plan. Often, each equivalence class has only one logical successor

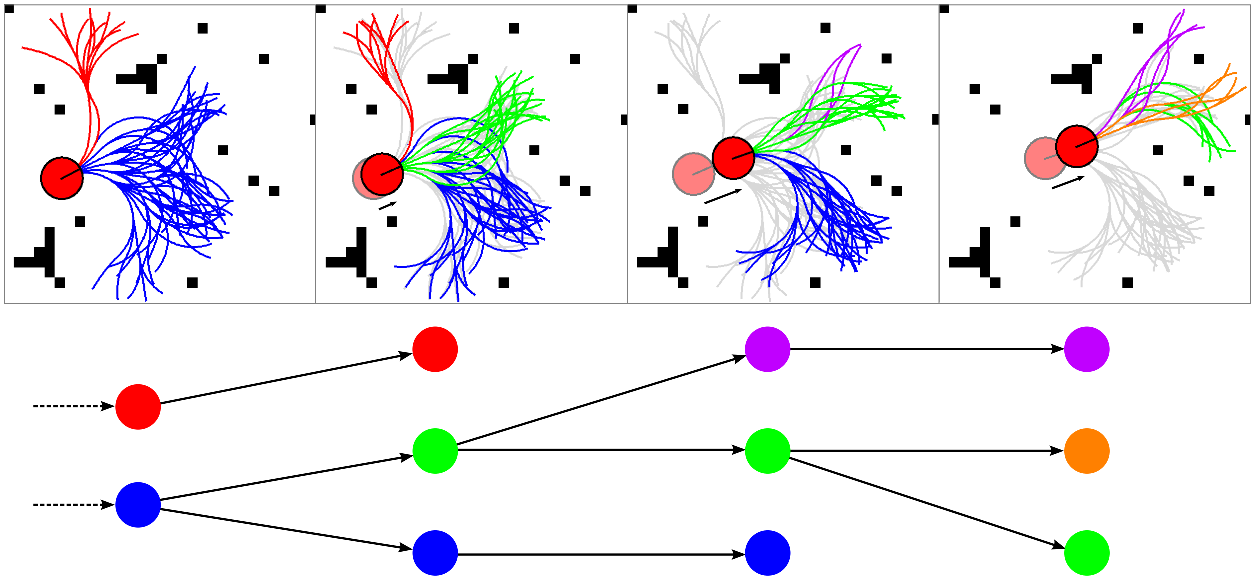

Figure 5: Between replan cycles, safe paths are associated by a logical succession of equivalence classes. This strict partial ordering relation is represented by a directed acyclic graph. Graph edges represent the relationship $\mathcal{P}_1 \rhd \mathcal{P}_2$. Graph node colors (and matching equivalence class path colors) are conserved in consecutive replan cycles only for the largest logical successor class. Between cycles, we may detect a termination, split, merge, or continuation of the previous equivalence classes. By preferring the logical successor to the previously-commanded path, subsequent path selections give better performance.

at the following replan time step. However, merges and splits may occur at critical points along the robot's traverse (Figure 5). When the planner detects a split, it is important to select the branch that maximizes success, given the locally-available information.

3.4 Multistage Path Selection Algorithm

We introduce the multistage local planner algorithm (Algorithm 7) to make principled path selections that trade off among the issues of logical succession, safety, and estimated path length to the goal. At a high level, the algorithm consists of two stages. Stage one selects for consideration a subset of $\mathcal{P}_{free}$ comprising one or more equivalence classes in order to ensure progress, safety, and consistency. Stage two selects from among the chosen subset one path for execution that trades off safety and cost of the path, while retaining goal-directedness.

3.4.1 Stage One: Solving the Decision Problem

In generally preferring to execute a new path that is a logical successor to the previously chosen equivalence class, we largely eliminate sensitivity to noisy perception data. Two exceptions arise in which the algorithm will not execute a logical successor path. First, if a non-successor equivalence class predicts a significantly lower cost to the goal, then the planner switches classes on the assumption that the magnitude of the change exceeds that of likely perception noise. Second, we allow the algorithm to consider broader alternatives—whether more or less costly—if all logical successor classes terminate or become *narrow*.

Definition 6 (narrow, wide)**.** *A **narrow** equivalence class contains few paths. We employ path count as a proxy for the measure of a corridor in path space. Thus, a low path count indicates little space to maneuver the robot through a narrow corridor. Non-narrow classes are called **wide**.* □

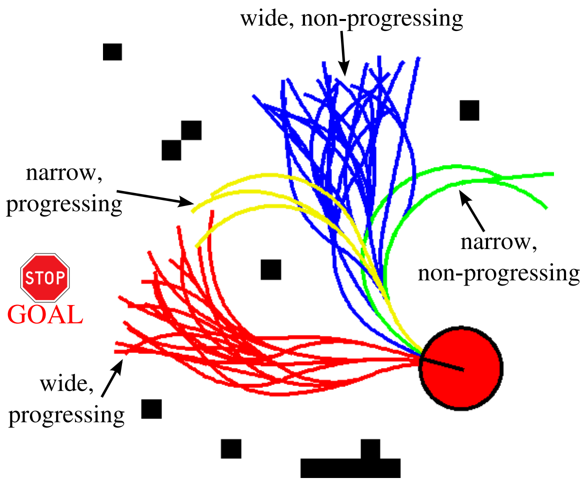

Figure 6: These four equivalence classes are annotated as to whether each is narrow or wide and progressing or non-progressing. Width is measured by the number of paths in the class as a fraction of all surviving paths, whereas progressivity describes whether its paths make progress towards the goal at left.

We define a constant fraction, MIN_PATH_THRESH, which adaptively selects the cutoff in corridor width as a percent of the number of paths in $\mathcal{P}_{free}$. Thus, the more densely we sample the space of paths, proportionately more paths are required to constitute a wide corridor. Even when densely sampling the path space, a highly cluttered environment may eliminate all but a few paths through collision with obstacles. In such a case, a passage containing relatively few paths may still be to be considered "wide" in comparison to others with fewer paths.

This concept of wide and narrow corridors closely resembles that of Borenstein and Koren (1991). Their vector field histogram represents obstacle density projected down to one dimension corresponding to heading. Sparse regions of the histogram indicate corridors, but due to the projection, only the component of corridor width perpendicular to that projection is recorded. Our approach to corridor detection and width estimation is more general since it closely approximates the full capabilities of the robot and is not limited to a particular obstacle configuration or observational perspective.

Although Algorithm 7 displays a preference for wide logical successor classes, it strictly selects only progressing paths within a class for consideration.

Definition 7 (progressing)**.** *A **progressing** path is one for which both of the following two points are nearer to the goal than is the current robot position, according to the global planner:*

1. *the point one replan time step in the future, and*
2. *the end point of the local path.* ☐

Figure 6 illustrates equivalences classes that are wide, narrow, progressing, and non-progressing. The progressing property is often shared by all paths in an equivalence class, but certain large classes in the absence of clutter can have mixed progressivity. By executing only progressing paths, we ensure that the robot monotonically approaches the goal, thus guaranteeing termination. Furthermore, by eliminating non-progressing paths during stage one, we are free to ignore goal-directedness in stage two while still guaranteeing progress.

When testing progressivity in a real implementation, it may be preferable to consider only criterion 2, a path's endpoint, and ignore the next step. Recognizing that curvature-constrained

Algorithm 5 $(\mathcal{W},\mathcal{N}) \leftarrow$Divide_Wide_Narrow($C$, t)

```
Input: $C$ – candidate set of classes; $t$ – threshold size of class
Output: $\mathcal{W}$ – set of paths in wide classes; $\mathcal{N}$ – set of paths in narrow classes
$\mathcal{W} \leftarrow \emptyset$
$\mathcal{N} \leftarrow \emptyset$
for all $\mathcal{C} \in C$ do
    if $|\mathcal{C}| > t$ then
        $\mathcal{W} \leftarrow \mathcal{W} \cup \mathcal{C}$        // Paths in wide classes
    else
        $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{C}$        // Paths in narrow classes
    end if
end for
$\mathcal{W} \leftarrow Cull\_Nonprogressing\_Paths(\mathcal{W})$        // Omit paths that move robot away from the goal
$\mathcal{N} \leftarrow Cull\_Nonprogressing\_Paths(\mathcal{N})$
return $(\mathcal{W},\mathcal{N})$
```

local paths need more space to maneuver than global grid paths, this relaxation provides the planner additional safety and flexibility when navigating around sharp corners, at the expense of termination guarantees.

In our implementation, Algorithm 5 is used to eliminate nonprogressing paths and divide the rest according to the narrow/wide dichotomy. Algorithm 7 uses it as a helper function in establishing an order of preference in selecting $\mathcal{S}$, the set of paths for consideration. The net order of preference is:

1. All wide, progressing, logical successor classes
2. Any wide, progressing class
3. All narrow, progressing, logical successor classes
4. Any narrow, progressing class
5. Return failure

After choosing a preliminary $\mathcal{S}$, we must check if the planner has found a highly suboptimal subset of paths; the algorithm compares the best path in $\mathcal{S}$ against the best path in $\mathcal{P}_{free}$. A difference above a certain SCORE_THRESH provokes a mid-course correction. Such a switch of equivalence class should be a rare event.

Note that we are making a choice with global implications based on unreliable information from a low-fidelity global planner. Lacking detailed knowledge of the complete path to the goal, we instead consider a calculation based solely on the statistics of this environment's average obstacle density, which predict that a narrow corridor "pinch point" should occur periodically at some frequency during traversal. Given a distance remaining to reach the goal, we can estimate an expected number of risky narrow corridors remaining. SCORE_THRESH should be chosen so that the decreased risk (stemming from the shorter path length) of getting stuck in a future narrow corridor outweighs the immediate risk involved in the current route change, which may itself jump to a narrower corridor.

Algorithm 6 $p \leftarrow$Optimize_Path(x, h, e, p)

```
Input: $x$ – initial state;
    $h$ – a heuristic function for selecting a path to execute;
    $e$ – equivalence object
    $p$ – seed path for optimization
Output: Return a path similar to $p$ but safer
repeat
    $\mathcal{N} \leftarrow e.Get\_Neighbors(p) \cup \{p\}$
    $p \leftarrow h.Farthest\_Obstacle\_Path(x, \mathcal{N})$    // Select path in set farthest from nearest obstacle
until $p$ converges or $p.obstacle\_proximity > \frac{3}{2}$ robot_diameter
return $p$
```

3.4.2 Stage Two: Solving the Optimization Problem

After establishing a final set of candidate paths $\mathcal{S}$, the algorithm moves on to stage two, which selects a single path for execution. Initially, it finds the greedy *Best_Path* option in $\mathcal{S}$, but this path may come unsafely close to an obstacle. Within the equivalence class containing the shortest path, the subroutine *Optimize_Path* performs a local, gradient-descent-type optimization in path space by traversing the equivalence graph. This optimization, which generates a soft safety buffer around obstacles, seeks to maximize the distance to the one nearest obstacle as described in Algorithm 6.

In especially wide corridors, the robot should be free to follow a reasonably short path, so the obstacle proximity penalty decays to zero beyond 1.5 robot diameters. The proximity penalty function is only defined with respect to the one nearest obstacle, so in a narrow corridor the penalty is locally minimized by the path most nearly following the center of the corridor, thus maximizing both safety and future planning options. Note that the algorithm will follow even an extremely narrow corridor, provided that the route represents the best means to progress towards the goal.

Ultimately, the algorithm we describe here improves on the original local planner algorithm by executing an action that is safe, maximizes future planning/control options, and remains consistent across replan cycles, all while retaining goal-directedness.

4 Foundations

In this section, we establish the foundations of an equivalence relation on path space based on continuous deformations between paths. We then provide correctness proofs for our algorithms for classification and implicit collision-testing.

We are given a kinematic description of paths. All paths are parametrized by a common initial pose, common fixed length, and individual curvature function. Let $\kappa_i(s)$ describe the curvature of path i as a function of arc length, with $\max_{0 \leq s \leq s_f} |\kappa_i(s)| \leq \kappa_{max}$. Typical expressions for κ_i include polynomials, piecewise constant functions, and piecewise linear functions. The robot motion produced by control i is a *feasible* path given by

$$\begin{bmatrix} \dot{\theta}_i \\ \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} \kappa_i \\ \cos\theta_i \\ \sin\theta_i \end{bmatrix}. \tag{5}$$

Definition 8 (feasible)**.** *A **feasible** path has bounded curvature (implying at least C^1 continuity) and fixed length. The set $\mathcal{F}(s_f, \kappa_{max})$ contains all feasible paths of length s_f and curvature $|\kappa(s)| \leq \kappa_{max}$.* □

Algorithm 7 $(p, \mathcal{L}) \leftarrow$Multistage_Local_Planner_Algorithm(w, x, h, e, $\mathcal{P}$, $\mathcal{L}$)

```
Input: $w$ – a costmap object;
    $x$ – initial state;
    $h$ – a heuristic function for selecting a path to execute;
    $e$ – equivalence object;
    $\mathcal{P}$ – a fixed set of paths;
    $\mathcal{L}$ – equivalence class of path selected in prior call (initially $\emptyset$)
Output: $p$ – a path progressing safely toward the goal;
    $\mathcal{L}$ – equivalence class of $p$
$\mathcal{P}_{free} \leftarrow Test\_All\_Paths(w, x, \mathcal{P})$    // May invoke implicit path test
$b \leftarrow h.Best\_Path(x, \mathcal{P}_{free})$    // Greedy shortest path
// Stage 1: select equivalence classes for consideration; trade off succession and corridor width
if $\mathcal{L} \neq \emptyset$ then    // Compute successor path candidates
    $C \leftarrow e.Get\_Logical\_Successor\_Classes(\mathcal{L})$    // Returns a set of classes
    $(\mathcal{W}_s, \mathcal{N}_s) \leftarrow Select\_Classes(C, \text{MIN\_PATH\_THRESH} \times |\mathcal{P}_{free}|)$
else
    $(\mathcal{W}_s, \mathcal{N}_s) \leftarrow (\emptyset, \emptyset)$
end if
$E \leftarrow e.Compute\_Equivalence\_Classes(\mathcal{P}_{free})$
$(\mathcal{W}, \mathcal{N}) \leftarrow Select\_Classes(E, \text{MIN\_PATH\_THRESH} \times |\mathcal{P}_{free}|)$    // Non-successor classes
if $\mathcal{W}_s \neq \emptyset$ then
    $\mathcal{S} \leftarrow \mathcal{W}_s$    // Consider the set of wide successor classes if some exist
else if $\mathcal{W} \neq \emptyset$ then
    $\mathcal{S} \leftarrow \mathcal{W}$    // Prefer any wide, progressing class over a narrow successor
else if $\mathcal{N}_s \neq \emptyset$ then
    $\mathcal{S} \leftarrow \mathcal{N}_s$    // Narrow successors are better than other narrow classes
else if $\mathcal{N} \neq \emptyset$ then
    $\mathcal{S} \leftarrow \mathcal{N}$    // Last resort: take any path
else
    return failure
end if
$p \leftarrow h.Best\_Path(x, \mathcal{S})$
if $p.score - b.score > \text{SCORE\_THRESH}$ then
    $\mathcal{S} \leftarrow \mathcal{P}_{free}$    // Jump equivalence classes to a significantly shorter route
end if    //
// Stage 2: Select one path from the set, trading off path length with safety
$p \leftarrow h.Best\_Path(x, \mathcal{S})$
$p \leftarrow Optimize\_Path(x, h, e, p)$    // Find safe enough path in selected path set
$\mathcal{L} \leftarrow e.Class\_Of(p)$
return $(p, \mathcal{L})$
```

As an aside, one may also consider piecewise C^1 continuous paths, such as paths with cusps following by the motion model of Reeds and Shepp (1990). Such paths may be equivalent provided that each follows the same sequence of forward and backward motions. Then, a conservative means of testing equivalence of the whole path is to perform pairwise tests on each C^1-continuous path segment. Thus, the following proofs may assume full C^1 continuity.

4.1 Properties of Paths

In this section, we establish a small set of conditions under which we can quickly determine that two paths are equivalent. We constrain path shape through two dimensionless ratios relating three physical parameters. We may then detect equivalence through a simple test on pairs of paths using the Hausdorff metric.

These constraints ensure a continuous deformation between neighboring paths while permitting a range of useful actions. Many important classes of action sets obey these general constraints, including the line segments common in RRT (LaValle and Kuffner, 2001) and PRM (Kavraki et al., 1996) planners, as well as constant curvature arcs. Figure 1 illustrates a more expressive action set (Knepper and Mason, 2008) that adheres to our constraints.

The three related physical parameters are: d, the diameter of the robot; s_f, the length of each path; and r_{min}, the minimum radius of curvature allowed for any path. Note that $1/r_{min} = \kappa_{max}$, the upper bound on curvature. For non-circular robots, d reflects the minimal cross-section of the robot's swath sweeping along a path. We express relationships among the three physical quantities by two dimensionless parameters:

$$v = \frac{d}{r_{min}} \qquad\qquad w = \frac{s_f}{2\pi r_{min}}.$$

We only compare paths with like values of v and w. Figure 7 provides some intuition on the effect of these parameters on path shape. Due to the geometry of paths, only certain choices of v and w are appropriate.

Definition 9 (appropriate path). *An* ***appropriate path*** *is a feasible path conforming to appropriate values of v and w from the proof of Lemma 2. Figure 7 previews the permissible values.*

When the condition in (4) is met, the two paths' swaths overlap, resulting in a continuum of coverage between the paths. This coverage, in turn, ensures the existence of a continuous deformation, as we show in Theorem 1, but first we formally define a continuous deformation between paths.

Definition 10 (continuous deformation). *A* ***continuous deformation*** *between two safe, feasible paths p_i and p_j in $\mathcal{F}(s_f, \kappa_{max})$ is a continuous function $f\colon [0,1] \to \mathcal{F}(s_f^-, \kappa_{max}^+)$, with s_f^- slightly less than s_f and κ_{max}^+ slightly more than κ_{max}. $f(0)$ is the initial interval of p_i, and $f(1)$ is the initial interval of p_j, both of length s_f^-.* ☐

Definition 11 (equivalent). *We write $p_i \sim p_j$ to indicate that a continuous deformation exists between paths p_i and p_j, and they are therefore* ***equivalent***. ☐

The length s_f^- depends on v and w, but for typical values, s_f^- is fully 95–98% of s_f. For many applications, this is sufficient, but an application can quickly test the remaining path length if necessary. Nearly all paths $f(c)$ are bounded by curvature κ_{max}, but it turns out that in certain geometric circumstances, the maximum curvature through a continuous deformation is up to $\kappa_{max}^+ = \frac{4}{3}\kappa_{max}$.

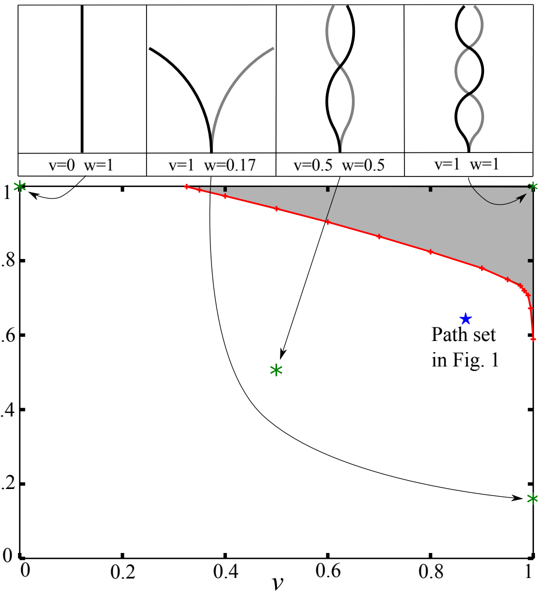

Figure 7: At top: several example paths combining different values of v and w. Each path pair obeys (4). The value of v affects the "curviness" allowed in paths, whereas w affects their length.
At bottom: this plot, generated numerically, approximates the set of appropriate choices for v and w. The gray region at top right must be avoided, as we show in Lemma 2. Such choices would permit an obstacle to occur between two safe paths that obey (4). A path whose values fall in the white region is called an *appropriate path.*

Definition 12 (guard paths)**.** *Two safe, feasible paths that define a continuous deformation are called **guard paths** because they protect the intermediate paths.* ☐

In the presence of obstacles, it is not trivial to determine whether a continuous deformation is safe, thus maintaining equivalency. Rather than trying to find a deformation between arbitrary paths, we propose a particular condition under which we show that a bounded-curvature, fixed-length, continuous path deformation exists,

$$\mu_H(p_1, p_2, d) \leq 1 \implies p_1 \sim p_2. \tag{6}$$

This statement, which we prove in the next section, is the basis for Algorithm 3 and Algorithm 4. The overlapping swaths of appropriate paths p_1 and p_2 cover a continuum of intermediate swaths between the two paths. The equivalence relation, of which (6) detects local instances, is a proper equivalence relation because it possesses each of three properties:

- **reflexivity**. $\mu_H(p, p, d) = 0$; p is trivially deformable to itself.
- **symmetry**. The Hausdorff metric is symmetric.
- **transitivity**. Given $\mu_H(p_1, p_2, d) \leq 1$ and $\mu_H(p_2, p_3, d) \leq 1$, a continuous deformation can be constructed from p_1 to p_3 passing through p_2.

4.2 Equivalence Relation

We now prove (6); that is, we show that shape constraints indicated by v and w combined with Hausdorff distance constraints are sufficient to ensure the existence of a continuous deformation

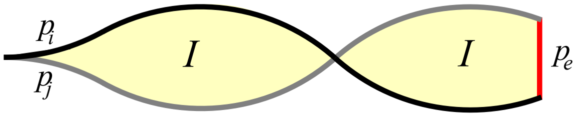

Figure 8: Paths p_i, p_j, and p_e form boundary B. Its interior, I, contains all paths in the continuous deformation from p_i to p_j. The set of paths in I illustrates the betweenness trait described in Section 2.3.

between two neighboring paths. Our approach to the proof will be to first describe a feasible continuous deformation, then show that paths along this deformation are safe.

Given appropriate guard paths p_i and p_j with common origin, let p_e be the shortest curve in the workspace connecting their endpoints without crossing either path (p_e may pass through obstacles). The closed path $B = p_i + p_e + p_j$ creates one or more closed loops (the paths may cross each other). By the Jordan curve theorem (Munkres, 2000), each loop partitions $\mathbb{R}^2$ into two sets, only one of which is compact. Let I, the interior, be the union of these compact sets with B, as in Figure 8.

Definition 13 (between)**.** *A path p_c is **between** paths p_i and p_j if $p_c \subset I$.* ☐

Lemma 1. *Given appropriate paths $p_i, p_j \subset \mathcal{F}(s_f, \kappa_{max})$ with $\mu_H(p_i, p_j, d) \leq 1$, a path sequence exists in the form of a feasible continuous deformation between p_i and p_j.*

Proof. We provide the form of a continuous deformation from p_i to p_j such that each intermediate path is between them. With t a workspace point and p a path, let

$$\gamma(t,p) = \inf_{\varepsilon} t \in (p)_\varepsilon \tag{7}$$

$$g(t) = \begin{cases} [0,1] & \text{if } \gamma(t,p_i) = \gamma(t,p_j) = 0 \\ \left\{\frac{\gamma(t,p_i)}{\gamma(t,p_i)+\gamma(t,p_j)}\right\} & \text{otherwise,} \end{cases} \tag{8}$$

where $g(t)$ is a set-valued function to accommodate intersecting paths. Each level set $g(t) = c$ for $c \in [0,1]$ defines a weighted generalized Voronoi diagram (GVD) forming a path as in Figure 9. We give the form of a continuous deformation using level sets $g^{-1}(c)$; each path is parametrized starting at the origin and extending for a length s_f^- in the direction of p_e.

Let us now pin down the value of s_f^-, the length of intermediate paths p_c. Every point t_i on p_i forms a line segment projecting it to its nearest neighbor t_j on p_j (and vice versa). Their collective area is shown in Figure 10. Equation (4) bounds each segment's length at d. s_f^- is the greatest value such that no intermediate path of length s_f^- departs from the region covered by these projections.

For general-shaped generators in $\mathbb{R}^2$, the GVD forms a set of curves meeting at branching points (Sampl, 2001). In this case, no GVD cusps or branching points occur in any intermediate path. Since $d < r_{min}$, no center of curvature along either guard path can fall in I (Blum, 1967). Therefore, each level set defines a unique path through the origin.

Each path's curvature function is piecewise continuous and everywhere bounded. A small neighborhood of either guard path approximates constant curvature. A GVD curve generated by two constant-curvature sets forms a conic section (Yap, 1987). Table 1 reflects that the curvature of p_c is everywhere bounded with the maximum possible curvature being bounded by $\frac{4}{3}\kappa_{max}$. For the full proofs, see (Knepper et al., 2010a). Thus, each intermediate path p_c is a feasible path. ☐

Lemma 2. *Given safe, appropriate guard paths $p_i, p_j \in \mathcal{F}(s_f, \kappa_{max})$ separated by $\mu_H(p_i, p_j, d) \leq 1$, any path $p_c \subset \mathcal{F}(s_f^-, \frac{4}{3}\kappa_{max})$ between them is safe.*

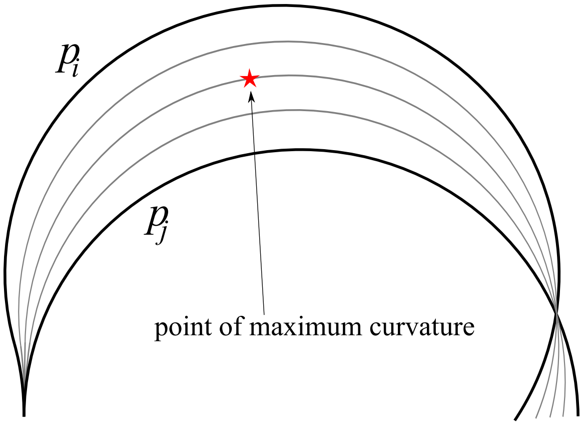

Figure 9: In a continuous deformation between paths p_i and p_j, as defined by the level sets of (8), each path takes the form of a weighted GVD. Upper bounds on curvature vary along the deformation, with the maximum bound of $\frac{4}{3}\kappa_{max}$ occurring at the medial axis of the two paths.

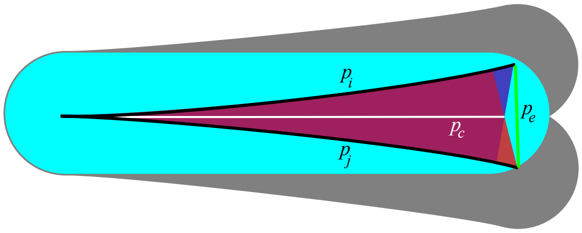

Figure 10: Hausdorff coverage (overlapping shapes in center) is a conservative approximation of swath coverage (gray). The Hausdorff distance between paths p_i and p_j is equal to the maximum-length projection from any point on either path to the closest point on the opposite path. Each projection implies a line segment. The set of projections from the top line and bottom line each cover a solid region between the paths. These areas, in turn, cover a slightly shorter intermediate path p_c, in white, with its light-colored swath. This path's length, s_f^- is as great as possible while remaining safe, with its swath inside the gray area.

Proof. We prove this lemma by contradiction. Assume an obstacle lies between p_i and p_j. We show that this assumption imposes lower bounds on v and w. We then conclude that for lesser values of v and w, no such obstacle can exist.

Let $sl(p,d) = \{t \in \mathbb{R}^2, t_p = nn(t,p) \colon \overline{t_p t} \perp p \text{ and } \|t - t_p\|_{L2} \leq \frac{d}{2}\}$ define a conservative approximation of a swath, obtained by sweeping a line segment of length d with its center along the path. $\overline{t_p t}$ is the line segment joining t_p to t and $nn(t,p)$ is the nearest neighbor of point t on path p. The two swaths form a safe region, $U = sl(p_i,d) \cup sl(p_j,d)$.

Suppose that U contains a hole, denoted by the set h, which might contain an obstacle. Now, consider the shape of the paths that could produce such a hole. Beginning with equal position and heading, they must diverge widely enough to separate by more than d. To close the loop in U, the paths must then bend back towards each other. Since the paths separate by more than d, there exist two open intervals $p_i^h \subset p_i$ and $p_j^h \subset p_j$ surrounding the hole on each path such that (at this point) $p_i^h \not\subset (p_j)_d$ and $p_j^h \not\subset (p_i)_d$. To satisfy (4), there must exist later intervals $p_i^e \subset p_i$ such that $p_j^h \subset (p_i^e)_d$ and likewise $p_j^e \subset p_j$ such that $p_i^h \subset (p_j^e)_d$, as in Figure 11a.

How long must a path be to satisfy this condition? Consider the minimum length solution to this problem under bounded curvature. For each point $t \in p_j^h$, the interval p_i^e must intersect the open disc $D = \text{int}((t)_d)$, as in Figure 11b. Since p_j^h grows with the width of h, and p_i^e must intersect

Table 1: Conic sections form the weighted Voronoi diagram. κ_1 and κ_2 represent the curvatures of the two guard paths, with κ_1 the lesser magnitude. Let $\kappa_m = \max(|\kappa_1|, |\kappa_2|)$. For details, see (Knepper et al., 2010a).

Type	Occurrence	Curvature bounds of intermediate paths
line	$\kappa_1 = -\kappa_2$	$\|\kappa\| \leq \kappa_m$
parabola	$\kappa_1 = 0, \kappa_2 \neq 0$	$\|\kappa\| \leq \kappa_m$
hyperbola	$\kappa_1\kappa_2 < 0, \kappa_1 \neq -\kappa_2$	$\|\kappa\| \leq \kappa_m$
ellipse	$\kappa_1\kappa_2 > 0$	$\|\kappa\| < \frac{4}{3}\kappa_m$

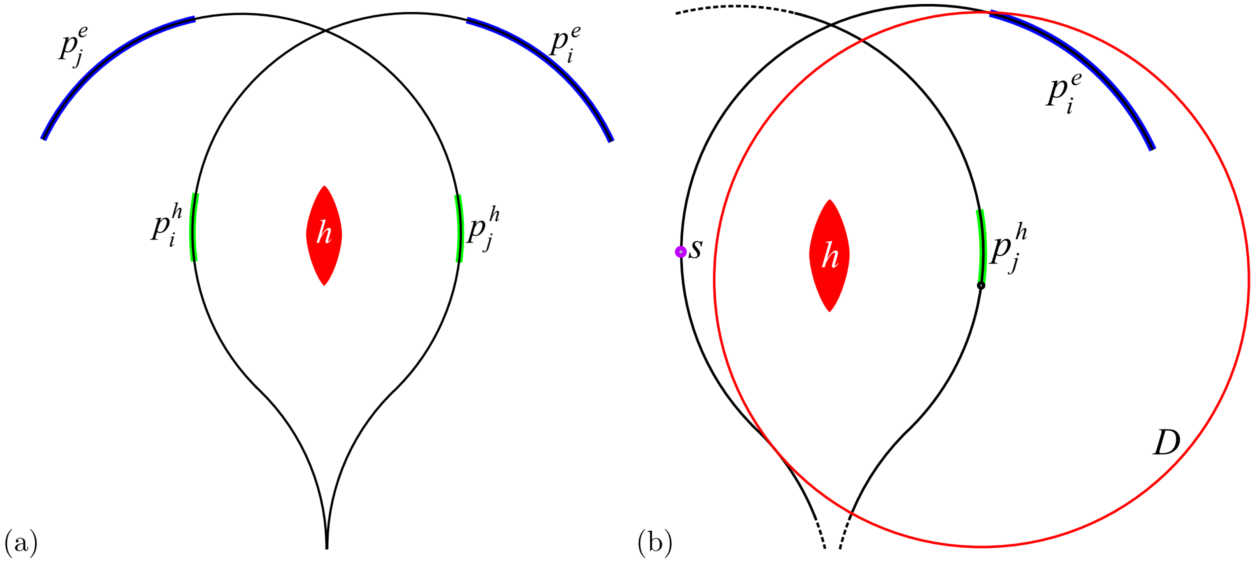

Figure 11: (a) With bounded curvature, there is a lower bound on path lengths that permit a hole, h, while satisfying (4), indicated by p_i^e, the blue highlight. Shorter path lengths ensure the existence of a safe continuous deformation between paths. (b) We compute the maximal path length that prevents a hole using Vendittelli's solution to the shortest path for a Dubins car. Starting from the dot marked s, we find the shortest path intersecting the circle D of radius r_{min}. The interval p_i^e illustrates path lengths permitting a hole to exist. Shorter paths leave some part of p_j^h uncovered.

all of these open neighborhoods D, the path becomes longer with larger holes. We will therefore consider the minimal small-hole case.

Vendittelli et al. (1999) solve the shortest path problem for a Dubins car to reach a line segment. We may approximate the circular boundary of D by a set of arbitrarily small line segments. One may show from this work that given the position and slope of points along any such circle, the shortest path to reach its boundary (and thus its interior) is a constant-curvature arc of radius r_{min}. In the limit, as v approaches one and the size of h approaches zero, the length of arc needed to satisfy (4) approaches $\pi/2$ from above, resulting in the condition that $w > 0.48$. Thus, for $w \leq 0.48$ and $v \in [0, 1)$, p_c is safe. For smaller values of v, D shrinks relative to r_{min}, requiring longer paths to reach, thus allowing larger values of w as shown in the plot in Figure 7.

We have shown that there exist appropriate choices for v and w such that (4) implies that U contains no holes. Since U contains the origin, any path $p_c \in I$ emanating from the origin passes through U and is safe. □

Theorem 1. *Given safe, appropriate guard paths* $p_i, p_j \in \mathcal{F}(s_f, \kappa_{max})$*, and given* $\mu_H(p_i, p_j, d) \leq 1$*,* *a safe continuous deformation exists between p_i and p_j.*

Proof. Lemma 1 shows that (8) gives a continuous deformation between paths p_i and p_j such that each intermediate path $p_c \subset I$ is feasible. Lemma 2 shows that any such path is safe. Therefore, a continuous deformation exists between p_i and p_j. This proves the validity of the Hausdorff metric as a test for path equivalence. $\square$

By chaining together continuous deformations between neighboring paths, we can demonstrate that a continuous deformation exists between any pair of paths within an equivalence class by following the correct sequence of edges of the equivalence graph. This property holds for any paths in our discretely sampled set. It also applies for any other pair of paths satisfying the shape constraints, provided that the discrete sampling is sufficiently dense. The existence of a sufficiently dense path sampling is the subject of the next section.

4.3 Resolution Completeness of Path Classifier

In this section, we show that Algorithm 3 is resolution complete. Resolution completeness commonly shows that there exists a sufficiently high discretization of each dimension of the search space such that the planner finds a path exactly when one exists in the continuum space. We instead show that there exists a sufficiently low dispersion sampling in the infinite-dimensional path space such that the approximation given by Algorithm 3 has the same connectivity as the continuum safe, feasible path space.

Let $\mathcal{F}$ be the continuum feasible path space and $\mathcal{F}_{free} \subset \mathcal{F}$ be the set of safe, feasible paths. Using the Green-Kelly algorithm, we sample offline from $\mathcal{F}$ a path sequence $\mathcal{P}_N$ of size N. At runtime, using Algorithm 2, we test members of $\mathcal{P}_N$ in order to discover a set $\mathcal{P}_{free} \subset \mathcal{P}_N$ of safe paths.

The following lemma is based on the work of LaValle et al. (2004), who prove resolution completeness of deterministic roadmap (DRM) planners, which are PRM planners that draw samples from a low-dispersion, deterministic source. Since we use a deterministic sequence provided by Green-Kelly, the combination of Algorithm 2 and Algorithm 3 generates a DRM in path space.

Lemma 3. *For any given configuration of obstacles and any path set $\mathcal{P}_N$ generated by the Green-Kelly algorithm, there exists a sufficiently large N such that any two paths $p_i, p_j \in \mathcal{P}_{free}$ are in the same connected component of $\mathcal{F}_{free}$ if and only if Algorithm 3 reports that $p_i \sim p_j$.*

Proof. LaValle et al. (2004) show that by increasing N, a sufficiently low dispersion can be achieved to make a DRM complete in any given C-Space. By an identical argument, given a continuum connected component $\mathcal{C} \subset \mathcal{F}_{free}$, all sampled paths in $\mathcal{C} \cap \mathcal{P}_N$ are in a single partition of $\mathcal{P}_{free}$. If q is the radius of the narrowest corridor in $\mathcal{C}$, then for dispersion $\delta_N < q$, our discrete approximation exactly replicates the connectivity of the continuum freespace. $\square$

Lemma 4. *Under the same conditions as in Lemma 3, there exists a sufficiently large N such that for any continuum connected component $\mathcal{C} \subset \mathcal{F}_{free}$, Algorithm 2 returns a $\mathcal{P}_{free}$ such that $\mathcal{P}_{free} \cap \mathcal{C} \neq \emptyset$. That is, every component in $\mathcal{F}_{free}$ has a corresponding partition returned by Algorithm 3.*

Proof. Let B_r be the largest open ball of radius r in $\mathcal{C}$. When $\delta_N < r$, B_r must contain some sample $p \in \mathcal{P}_N$. Since $\mathcal{C}$ is entirely collision-free, $p \in \mathcal{P}_{free}$. Thus, for dispersion less than r, $\mathcal{P}_{free}$ contains a path in $\mathcal{C}$. $\square$

There exists a sufficiently large N such that after N samples, $\mathcal{P}_N$ has achieved dispersion $\delta_N < \min(q, r)$, where q and r are the dispersion required by Lemmas 3 and 4, respectively. Under such conditions, a bijection exists between the connected components of $\mathcal{P}_{free}$ and $\mathcal{F}_{free}$.

Theorem 2. *Let* $D = \{\mathcal{D}_1 | \ldots | \mathcal{D}_m\}$ *be a partition of* $\mathcal{P}_{free}$ *as defined by Algorithm 3. Let* $C = \{\mathcal{C}_1 | \ldots | \mathcal{C}_m\}$ *be a finite partition of the continuum safe, feasible path space into connected components. A bijection* $f : D \to C$ *exists such that* $\mathcal{D}_i \subset f(\mathcal{D}_i)$*.*

Proof. Lemma 3 establishes that f is one-to-one, whereas Lemma 4 establishes that f is onto. Therefore, f is bijective. This shows that by sampling at sufficiently high density, we can achieve an arbitrarily good approximation of the connectedness of the continuum set of collision-free paths in any environment. ☐

Finally, we move on to show that we can detect path safety while circumventing a collision test.

Theorem 3. *A path interval* p_c *may be implicitly tested safe if it is between paths* p_i *and* p_j *such that* $\mu_H(p_i, p_j, d) \leq 1$ *and a small region at the end of* p_c *has been explicitly tested.*

Proof. By Lemma 2, the initial interval of p_c is safe because its swath is covered by the swaths of the guard paths. Since the small interval at the end of p_c has been explicitly tested, the whole of p_c is collision-free. ☐

5 Evaluation

We present some experimental results involving equivalence class detection and implicit path collision testing. All tests were performed in simulation on planning problems of the type described in previous work (Knepper and Mason, 2008). Briefly, planning problems consist of a combination of an environment and planning query. Environments were randomly generated within a room measuring twenty meters on a side, in which 10 cm diameter obstacles were randomly positioned until reaching the desired workspace obstacle coverage fraction. A planning query asks the 0.412 m diameter Nomad Scout robot to navigate from a start to a goal configuration, each chosen randomly so as to be separated by 14 m. Finally, candidate problems were rejected if a 10 cm resolution 8-connected grid planner was unable to solve the planning problem. Note that this grid planner also serves as global guidance to the local planner. Unless otherwise stated, the local planner receives 0.1 sec per replan cycle. Each reported result is an average over 100 runs on a fixed set of different planning problems.

During navigation, the local planner is permitted to run for a fixed amount of time within each replan cycle before executing its chosen path. A variable number of paths will be tested each cycle depending on factors such as obstacle clutter and implicit path testing. In some experiments, we vary the planning time allotted, whereas other experiments explore the effects of obstacle density.

5.1 Classification Performance Overhead

Path classification imposes a computational overhead due to the cost of searching for neighboring collision-free paths. Collision rate in turn relates to the density of obstacles in the environment. Figure 12 shows that the computational overhead of our classification implementation is nearly 20% in an empty environment but drops to 0.3% in dense clutter. It is in precisely such high-clutter environments that the usefulness of classification is maximized since two arbitrary paths are less likely to be equivalent amongst many obstacles. We now proceed to weigh these and other benefits of path classification against its costs.

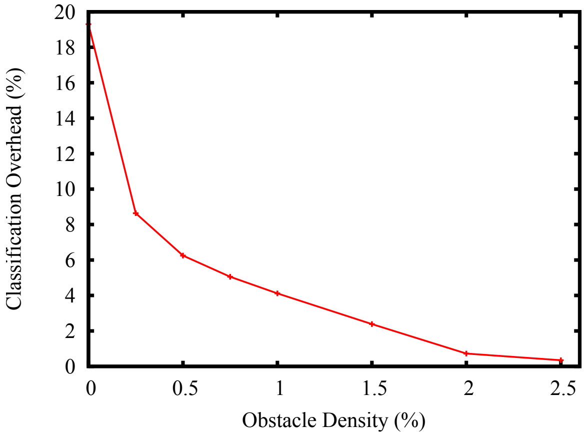

Figure 12: Path classification overhead is minimized in exactly those densely-cluttered problems where its contribution is most valuable. In this plot, a constant time of 0.1 secs is allotted to collision test and classify paths at a range of obstacle densities. Note that with such a fixed deadline, the planner finds more safe paths in lower-density scenarios. If a fixed quantity of *safe* paths must be generated regardless of elapsed time, this curve becomes significantly flatter.

5.2 Collision Testing

Regardless of obstacle density, implicit collision-testing more than compensates for the overhead of path classification. In comparing collision test algorithms, the baseline collision tester is among the most efficient for a rigid body moving along a path. The algorithm performs successive static collision tests at positions along the path. In order to determine the interval between tests, the algorithm computes the distance to the nearest obstacle and moves that distance (or less) along the path. The algorithm thus simultaneously guarantees correctness (never missing a collision) and efficient performance (computing a small number of static collision tests). Note that this same idea can also be extended to multibody kinematic chains (Schwarzer et al., 2004).

Figure 13 shows the effect of implicit path testing on total paths tested in the absence of obstacles. We compare the implicit collision-tester of Algorithm 4 against traditional explicit collision-testing. As the time allotment for testing paths increases, the number of paths collision-tested under the traditional algorithm increases linearly at a rate of 8,300 paths per second. With implicit testing, the initial test rate over small time allotments (thus small path set sizes) is over 22,500 paths per second. The marginal rate declines over time due to the aforementioned overhead, but implicit path testing still maintains its speed advantage until the entire 2,401-member path set is collision-tested. Note that this result occurs in the empty world case, where overhead is most severe.

Figure 14 presents implicit collision-testing performance in the presence of clutter. As obstacle density increases, we expect overhead to drop, but it simultaneously becomes more difficult to satisfy the necessary conditions for implicitly testing a path. Fixing the replan rate at an intermediate value of 10 Hz, we see that implicit path evaluation maintains an expected advantage across all navigable obstacle densities. In high clutter, this advantage is statistically less clear, yet implicit path testing still outperforms explicit path testing with over 90% confidence at the maximum tested obstacle density.

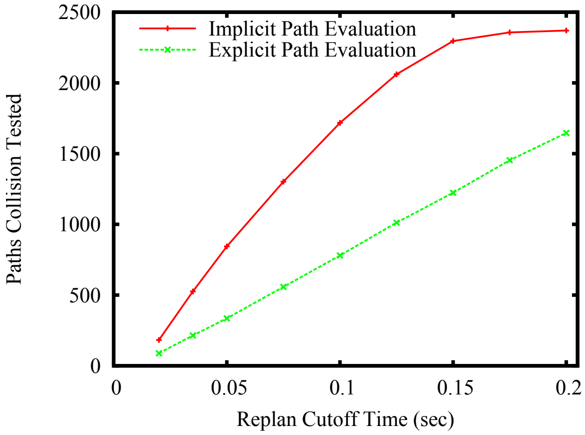

Figure 13: Paths tested per time-limited replan step in an obstacle-free environment. Path testing performance improves by up to 3x with the algorithms we present here. Note that an artificial ceiling curtails performance at the high end due to a maximum path set of size 2,401.

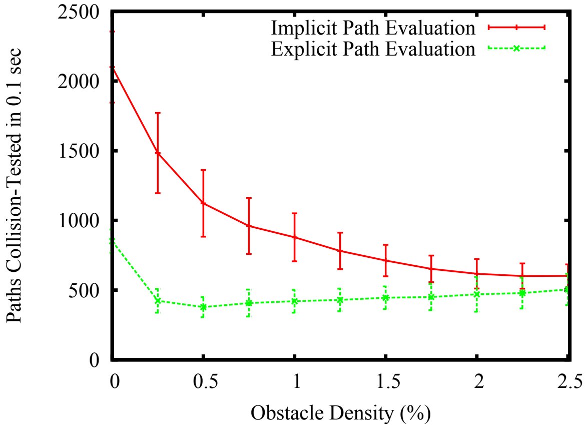

Figure 14: Paths tested per 0.1 second time step at varying obstacle densities. Implicit collision-testing allows significantly more paths to be tested per unit time. Even in extremely dense clutter, implicit path testing considers an extra six paths on average. The right edge of the graph represents the maximum density at which environments remain navigable. Error bars indicate 95% confidence.

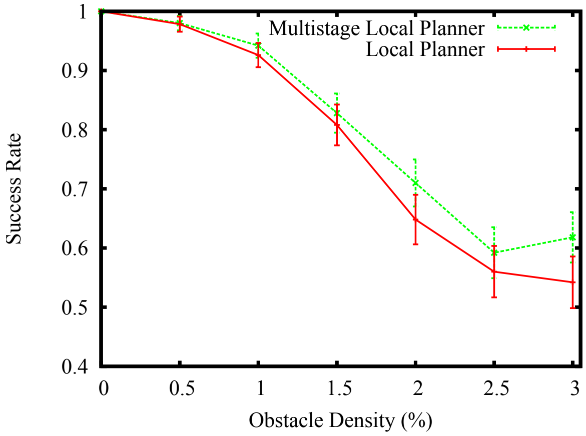

Figure 15: Improvement in planner success rate at solving queries. In high clutter, Algorithm 7 performs significantly better than Algorithm 1 at successfully completing navigation tasks.

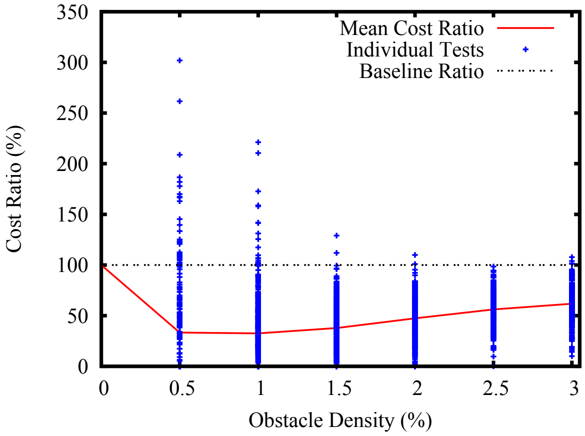

Figure 16: Ratio of *Best_Path* cost to multistage local planner algorithm cost. Lower is better. The cost of an individual path from either planner represents the path integral of the reciprocal of obstacle proximity. The overall mean cost improvement (solid line) is about a factor of two, meaning that during navigation the robot stays twice as far from obstacles, on average. The scatter plot (blue crosses) shows the individual results for the 1838 experiments in which both planners successfully reached the goal.

5.3 Route Selection

We tested the multistage local planner algorithm (Algorithm 7) over a variety of environments at a range of obstacle densities in order to evaluate the effects on planner performance of awareness of local equivalence classes. Experimental setup was the same as in Section 5, except that here we tested the planner on 500 different planning problems for each obstacle density. Replan cycle times for these experiments is 0.1 sec.

Figure 15 shows success rate for the local planner algorithm (greedy) and multistage local planner algorithm (equivalence aware). The latter produces a statistically significant improvement of 7.6% at solving planning problems in dense clutter. Path length increases only negligibly, and despite the extra path length, we find a decreased path cost, expressed as

$$c(p) = \int_p \frac{ds}{od(p(s))}, \tag{9}$$

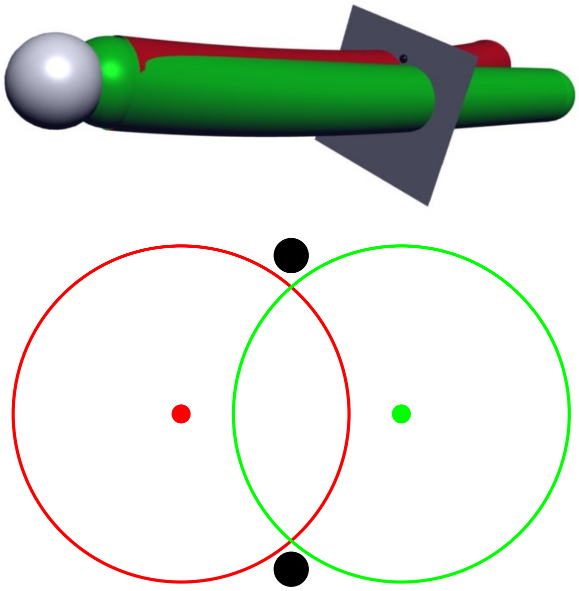

Figure 17: Two paths of a spherical robot (gray) in a 3D workspace are insufficient to establish equivalence. The bottom figure depicts a cross-sectional slice through the swept volumes of the paths. Although two paths may be separated by less than one robot diameter, obstacles (black) may still prevent a continuous deformation between the paths.

where $od(s)$ is the distance to the nearest obstacle from the given point along the path. Figure 16 shows the change in cost between the two planners. The overall mean change, shown with the solid line, indicates that the robot stays about twice as far from obstacles. We compared performance on each test problem separately so the individual data-points (shown with blue crosses) represent cost improvement normalized to the difficulty of the problem.

In contrast to the overall trend, a small fraction of the experiments showed cost worsening. In roughly 2% of cases, skewed heavily towards the less dense environments, obstacle proximity increased with the equivalence-aware planner. We attribute this phenomenon to the interplay between choice and lookahead distance. In more open environments, the increased choice makes it more tempting to stray from a coherent plan. At a different lookahead distance, the planner may have understood its choices better, but there are of course significant costs to increased lookahead. Making this tradeoff dynamically presents an interesting opportunity for future work.

6 Extensions and Future Work

Having thus far examined our equivalence relation in the case of a mobile robot in the plane, we now briefly turn to more complex systems and applications. Each system considered in this section has at least one of two attributes: motion in three dimensions, and internal articulation (i.e. it can change shape).

6.1 A Rigid Body in 3D

First, we address a rigid body in a 3D workspace, such as an aircraft or spacecraft. Let us suppose this rigid body takes the form of a sphere (or can be approximated as one). Two different paths, no matter how close together, are never sufficient to establish path equivalence, as they are in two

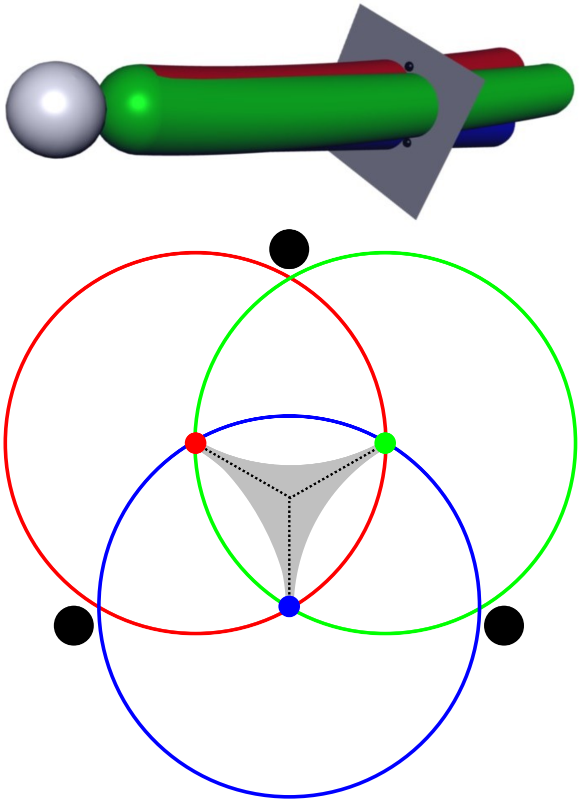

Figure 18: Three paths are required to establish path equivalence in 3D. The swept volumes of the three paths are shown. To establish equivalence, we require that the Hausdorff distance between each pair of paths must not exceed the robot's radius. Given such proximity, one path may be continuously deformed to another by following the dotted lines, without risk of intersecting the black obstacles. Upon inspection, one might think that a looser bound could be found because the radius-circles do not meet perpendicularly in this condition. In the limiting case as depicted, the path point and radius-circle intersections are collinear. Any relaxation of the inter-path distance would result in a case where the obstacles could squeeze into the corners at the intersection points and prevent a continuous deformation from occurring. Implicit collision testing may be performed on any fourth path discovered to be entirely inside the gray star-shaped region at center.

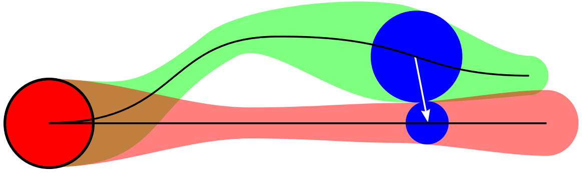

Figure 19: Mobile robot paths with variable radius. The nearest point on the opposite path depends on both the position and radius of each point along the path.

dimensions. Figure 17 illustrates this fact.

In three dimensions, three paths are equivalent if each of their pairwise Hausdorff separations is not more than the radius of the robot. Figure 18 depicts such a configuration of three paths and also shows how to construct a continuous deformation between any pair. As in the 2D case, with three paths in 3D we can perform implicit collision testing on a variety of paths. The region in which a fourth path must reside to implicitly declare it collision-free is shown in Figure 18.

One who is familiar with topology might question the value of path equivalence in three dimensions because ordinary bounded obstacles do not induce additional homotopy classes. However, this is where local path equivalence really shines. Since the planner has only local knowledge, it cannot distinguish between a finite, long, skinny obstacle and an infinite, skinny obstacle. For all practical purposes, the finite obstacle might as well be infinite. Given that the hierarchical planner has no knowledge of paths that perform an end-run around the object, such choices are sufficiently costly that the partition of local paths into separate classes represents a more accurate depiction of the available choices than the single class supplied by a true homotopy relation.

In the application of local path equivalence to 3D problems, one concern arises in the relatively uncluttered nature of most 3D environments. Such environments put the classification algorithm at the left end of the graph in Figure 12—a high-overhead situation. However, the situation is not as bad as it first appears. The data in that figure was gathered with a very efficient 2D collision tester. By comparison, collision testing in 3D is significantly more complex and costly, whereas the complexity of the classification algorithm increases slightly. Classification becomes proportionately much more efficient in three dimensions, thus the gain from avoiding explicit path tests in 2D (Figure 13 and Figure 14) becomes more dramatic in 3D.

6.2 Variable-Size Robots in 2D

We begin by introducing a variant of the basic 2D mobile robot in which the robot's size—still approximated as a disc—varies as a function of path length. This scenario occurs in the mobile manipulation problem, in which a mounted manipulator arm may extend out beyond the robot's own footprint. For example, the elastic strips of Brock and Khatib (2002), when projected onto the floor, closely resemble this variable-width robot. A radius function $r_i(\ell)$ expresses the robot's radius with respect to point $p_i(\ell)$ along path p_i. We introduce the concept of proportional dilation, in which the path width grows in accordance with its radius function:

$$(p)_{r_i} = \{t \in \mathbb{R}^2 \colon \|p(\ell) - t\|_{L2} \le r_i(\ell) \text{ for some } \ell \in I_L\}, \tag{10}$$

where $I_L = [0, L]$, with L the path length. Now a pair of nearby points on two neighboring paths may possess distinct diameters, as in Figure 19, thus giving rise to a new normalized Hausdorff metric,

$$\begin{aligned}\mu^v_H(p_i,p_j,r_i,r_j) = \max(&\underset{\varepsilon}{\operatorname{argmin}}(\forall \ell_a \in I_L, \exists \ell_b \in I_L\colon (p_i(\ell_a))_{\varepsilon r_i(\ell_a)} \cap (p_j(\ell_b))_{\varepsilon r_j(\ell_b)} \neq \emptyset),\\ &\underset{\varepsilon}{\operatorname{argmin}}(\forall \ell_a \in I_L, \exists \ell_b \in I_L\colon (p_j(\ell_a))_{\varepsilon r_j(\ell_a)} \cap (p_i(\ell_b))_{\varepsilon r_i(\ell_b)} \neq \emptyset)).\end{aligned} \tag{11}$$

Intuitively, this variant of the Hausdorff metric finds the minimal scale factor for the two paths' radius functions such that no gap remains between the two paths following proportional dilation. Following dilation, each point on either path is replaced by a disc. In order to ensure the above condition, each disc on one path must intersect some disc on the opposite path.

Given some appropriate constraints on the shape of paths as well as their radius functions, the equivalence relation on paths of variably-sized mobile robots then follows directly from the fixed-diameter case:

$$\mu^v_H(p_1,p_2,r_1,r_2) \leq 1 \implies p_1 \sim p_2. \tag{12}$$

Here, curvature bounds must apply to the boundary of the variable-diameter robot swath in addition to the path itself. However, in the case of a mobile manipulator, where the reach of an arm is on the same order of magnitude as the mobile base diameter, this distinction is rarely critical.

6.3 Articulated Robots

We also apply path classification to the trajectories of manipulator arms in 3D. At a high level, the situation closely parallels the 2D mobile robot case we present in Section 2. Two paths, separated by at most the diameter of the arm, are equivalent under certain shape and proximity constraints. In contrast to a 2D rigid body, the medial axis of the arm sweeps out a 2D manifold or sheet in the workspace (Figure 20 and Figure 21), so points along our path are now parametrized by (s,ℓ), where $s \in I_S = [0, s_f]$, a distance along the arm's motion in the configuration space, and $\ell \in I_L = [0, L]$, a distance along the axis of the arm from base to end-effector. Thus, $p(s,\ell)$ corresponds to a particular workspace location along the arm's axis while the arm is in a certain configuration.

A function $r(\ell)$ describes the radius of a disc circumscribing a cross-section of the arm at a point along its length. Note that radius is now a function of arm length rather than trajectory position. Note also that the disc is now normal to the arm axis, whereas in the 2D shape-changing robot, it is coplanar with the path. Though nearly identical to (10), we define proportional dilation in the context of a two-parameter path function,

$$(p)_{r_i} = \{t \in \mathbb{R}^2\colon \|p(s,\ell) - t\|_{L2} \leq r_i(\ell) \text{ for some } s \in I_s \text{ and } \ell \in I_L\}. \tag{13}$$

As in Section 6.2, we define a variant of the Hausdorff distance in the context of arm paths,

$$\begin{aligned}\mu^a_H(p_i,p_j,r_i,r_j) = \max(&\underset{\varepsilon}{\operatorname{argmin}}(\forall (s_a l \ell_a), \exists (s_b,\ell_b)\colon (p_i(s_a,\ell_a))_{\varepsilon r_i(\ell_a)} \cap (p_j(s_b,\ell_b))_{\varepsilon r_j(\ell_b)} \neq \emptyset),\\ &\underset{\varepsilon}{\operatorname{argmin}}(\forall (s_a,\ell_a), \exists (s_b,\ell_b)\colon (p_j(s_a,\ell_a))_{\varepsilon r_j(\ell_a)} \cap (p_i(s_b,\ell_b))_{\varepsilon r_i(\ell_b)} \neq \emptyset)),\end{aligned}$$

where $s_a, s_b \in I_S$ and $\ell_a, \ell_b \in I_L$. Note that this is a conservative expression for the distance separating two arm trajectories.

In considering the possibility that an obstacle divides two arm trajectories, the semantics of the application come into play. For example, objects in human spaces do not levitate, so in the absence of highly dynamic objects such as a thrown ball, we may relax the tight constraints imposed by by

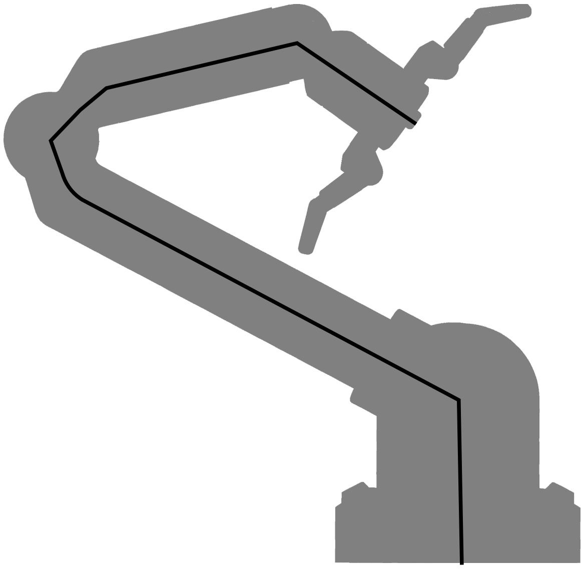

Figure 20: A kinematic chain such as a robot arm (gray) can be likened to a mobile robot path with varying radius as in Figure 19. When executing a trajectory, the arm's medial axis (black) sweeps out a two-dimensional sheet with varying radius in the spatial dimension and fixed radius in the temporal dimension.

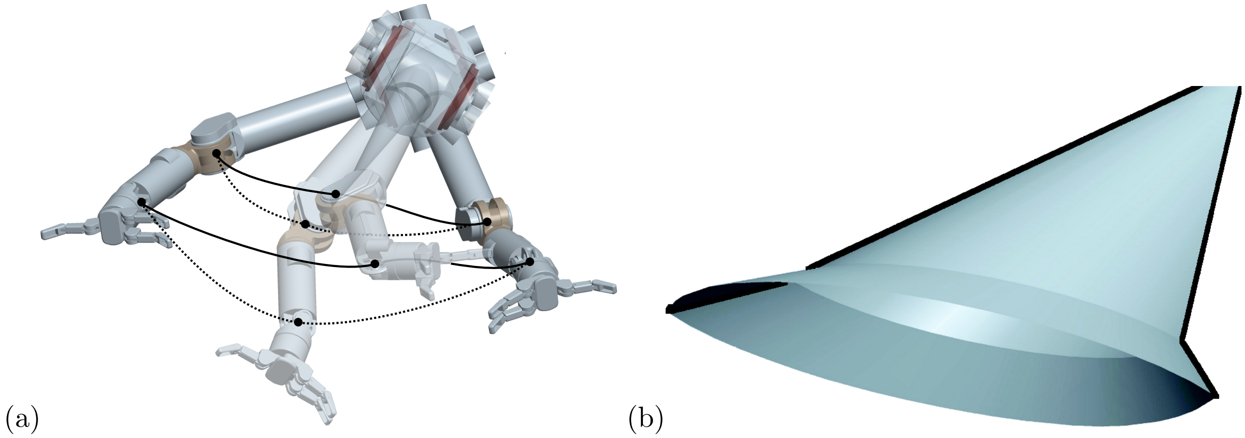

Figure 21: When moving an arm between two configurations, many trajectories are possible. Each trajectory traces a unique path through the workspace. (a) For two different trajectories represented by solid and dotted lines, the paths of the elbow and wrist are shown. (b) Two sheets correspond to two distinct arm trajectories. These 2D manifolds embedded in a 3D workspace share many properties with the 1D mobile robot paths embedded in a 2D workspace discussed earlier.

μ_H^a. Instead, the two arm trajectories need only to completely surround a pocket of space, meaning that their end-effector trajectories and end states overlap.

Next, we address constraints on arm path shape and length. Such concepts are inherently much more nebulous than their mobile-robot equivalents due to the arm's articulation—especially for arms with revolute joints. It is difficult to pin down general, meaningful constraints on arm path length and shape.

In principle, a useful measure of path length could be obtained by computing swept volume of the arm. After all, in the case of a rigid body mobile robot, all swaths of a given length that do not cross over themselves have equal length. In the case of manipulator arms (especially those with revolute joints), many useful motions do involve swept volumes that "cross over themselves," so an alternate formulation of path length is needed. Just as the length of a mobile robot path is found by integrating velocity, the length of an arm path may be found by integrating a form of velocity as well. Given an arm path p_i executed with unit C-Space speed, let $v_i(t,\ell)$ be the workspace velocity of point $p_i(t,\ell)$ along the axis of the arm at time t. We propose two alternative path length measures:

$$L_{mean}(i) = \frac{1}{L}\int_0^L \int_0^{s_f} v_i(s,\ell)\ ds\ d\ell \tag{14}$$

$$L_{max}(i) = \int_0^{s_f} \max_{\ell \in I_L} v_i(s,\ell)\ ds \tag{15}$$

In the case of a rigid body under arbitrary motion in $\mathbb{R}^3$, the mean and max path length are always related by a factor between one and two. For an articulated chain, the factor may be greater.

We now move on to address path shape constraints for an arm. This issue is both complex and mechanism-dependent. In previous work (Knepper et al., 2010b), we utilized path sets comprising straight lines in C-Space. Of course, such "straight" trajectories can involve arbitrarily high curvature of some point on the arm within the workspace. Given a sampling of such paths dense enough to establish path equivalence, it is not clear that the planner would often discover multiple equivalence classes. It would therefore be left to a given manipulation application to further constrain path sampling to a set of tasks useful for a given problem.

A few approaches worth exploring further include bounding the energy consumed in executing a given path (after subtracting out torques associated with gravity compensation), and retraction-based approaches. In the latter case, we propose to compare paths by utilizing retraction-like reductions in dimensionality of a search space, such as those proposed by Choset and Burdick (1995); Sun and Lumelsky (1991). Under this reduction, any given path in the free configuration space maps to a path in a one-dimensional set, which is the deformation retract of the freespace. We can then employ such retracts as a graph-like roadmap and compare only paths whose corresponding graph paths are similar. For general articulated systems, this approach raises some challenges of its own, such as the fact that in three or more dimensions, these one-dimensional retracts are necessarily either not connected or have extra loops not corresponding to topological features of the original freespace (Choset and Rizzi, 2005).

Despite these challenges, our path equivalence relation holds promise in the domain of motion planning for articulated robots, such as manipulator arms. For instance, even lacking constraints on path shape, it is possible to utilize μ_H^a to accelerate collision testing. Although arms pose greater challenges for satisfying the necessary conditions on proximity and betweenness, the cost of each collision test is significantly higher for articulated robots than it is for rigid bodies, so the gains remain potentially significant.

6.4 Steerable Needles

Steerable needles are long, flexible, hollow needles with a bevel tip. Such needles can be used in medical procedures to reach soft tissue anatomical features that would otherwise be inaccessible due to obstructing anatomy of a hard (bone) or delicate (nerve, artery) nature. During insertion, the bevel tip causes the needle to follow a constant curvature path. Steerable needles are interesting from a motion planning perspective because they are underactuated, having only two velocity controls. One may vary the rate of insertion and the rate of rotation about the axis of the needle. This twisting motion alters the plane in which the needle bends (Webster III et al., 2006).

During needle motion, uncertainty is introduced to the path primarily in the form of a random variable added to the needle's curvature. This uncertainty derives from the complexities of interaction with human tissue. Instantaneously, the uncertainty in position increases in the direction of the vector normal to the needle within the plane of curvature. Meanwhile, the instantaneous control inputs act primarily in vectors along the needle and normal to the plane of curvature. By duty-cycling the needle, steering control can be applied to cancel out errors in curvature, thus the needle can track arbitrary paths of bounded curvature (Minhas et al., 2007).

Planners have been proposed for needle steering to account for uncertainty in both motion (Alterovitz et al., 2008) and localization (van den Berg et al., 2010). It should be possible to extend many steerable needle planners using path equivalence. The equivalence algorithms in this paper assume that any path under consideration may be chosen deterministically. Thus, the distribution of paths is purely a function of control inputs. In the steerable needles context, the distribution among paths is generated by a combination of control and uncertainty. Consequently, we must find equivalence between entire groups of paths across a probability distribution instead of between individual paths. Thus, steerable needles constitute both a theoretical extension and a promising application for future work.

7 Conclusion

In this paper, we propose an equivalence relation on local paths based on the following constraints: fixed start position and heading, fixed length, and bounded curvature. We describe an algorithm for easily classifying paths using the Hausdorff distance between them. Path classification is a tool that permits collective reasoning about paths, leading to more efficient collision-testing.

We present a variety of applications for local path equivalence, including implicit path testing to accelerate collision testing. In this area, we demonstrate performance increases of over 300% in paths collision-tested per unit time. In addition to the strict implicit path-testing formalism we present, many generalizations are possible such as implicitly testing paths of varying length.

Of course, we must be able to make use of all those additional paths, so our second application is in the area of navigation. Equivalence classes are a powerful tool to distinguish between discrete decision problems and continuous optimization problems and to balance choices between the two. In navigation, we show an increase in navigation success of up to 7.6% for one class of long-range planning problem. Meanwhile, our navigation algorithm reduces by half the cost of obstacle proximity, indicating that these paths are significantly safer.

Finally, we explore some extensions and real-world applications. Local path equivalence, when applied to three-dimensional planning problems, is actually more meaningful in a hierarchical planning context than is true homotopy because it reflects choices that are locally discernible. We discuss the strengths and weaknesses of local path equivalence applied to three-dimensional kinematic chains. The path equivalence formulation seems particularly well-suited to the bevel-tipped steerable needle problem, since that mechanism is naturally curvature-constrained.

Surveying these applications, we conclude that local path equivalence is a formalism that provides valuable tools to address motion planning problems involving hierarchical planning, collision testing, and navigation for a variety of two- and three-dimensional systems.

8 Acknowledgments

The authors thank Matthew Tesch, Laura Lindzey, Alberto Rodriguez, Mehmet Doğar, and M. Jenae Lowe for valuable comments and discussions. Thanks extend also to the anonymous reviewers for their thoughtful and supportive feedback.

9 Funding

This work is sponsored by the Defense Advanced Research Projects Agency. This work does not necessarily reflect the position or the policy of the Government. No official endorsement should be inferred.

References

T. Allen, J. Underwood, and S. Scheding. A path planning system for autonomous ground vehicles operating in unstructured dynamic environments. In *Proc. Australasian Conference on Robotics and Automation*, 2007.

R. Alterovitz, M. Branicky, and K. Goldberg. Motion planning under uncertainty for image-guided medical needle steering. *International Journal of Robotics Research*, 27(11–12):1361–1374, 2008.

J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2-3-4):121–155, 1993.

S. Bhattacharya, V. Kumar, and M. Likhachev. Search-based path planning with homotopy class constraints. In *Proc. National Conference on Artificial Intelligence*, 2010.

H. Blum. A transformation for extracting new descriptors of shape. In Weiant Whaters-Dunn, editor, *Proc. Symposium on Models for the Perception of Speech and Visual Form*, pages 362–380, Cambridge, Mass., 1967. MIT Press.

J. Borenstein and Y Koren. The vector field histogram—fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research*, 21(12):1031–1052, December 2002.

J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S. Thrun. The mobile robot rhino. *AI Magazine*, 16(2):31–38, 1995.

H. Choset and J. Burdick. Sensor based planning, part I: The generalized Voronoi graph. In *Proc. International Conference on Robotics and Automation*, pages 1649–1655, 1995.

H. Choset and A. A. Rizzi. Topology in motion planning. In *Proc. International Symposium on Robotics Research*, August 2005.

M. Farber. Topological complexity of motion planning. *Discrete & Computational Geometry*, 29 (2):211–221, 2003.

N. H. Gardiol and L. P. Kaelbling. Action-space partitioning for planning. In *Proc. National Conference on Artificial Intelligence*, Vancouver, Canada, 2007.

C. Green and A. Kelly. Toward optimal sampling in the space of paths. In *Proc. International Symposium of Robotics Research*, Hiroshima, Japan, November 2007.

J. Henrikson. Completeness and total boundedness of the Hausdorff metric. *MIT Undergraduate Journal of Mathematics*, 1, 1999.

L. Jaillet and T. Simeon. Path deformation roadmaps: Compact graphs with useful cycles for motion planning. *International Journal of Robotics Research*, 27(11–12):1175–1188, 2008.

L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *Proc. International Conference on Robotics and Automation*, pages 566–580, 1996.

A. Kelly, A. Stentz, O. Amidi, M. W. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. M. Vallidis, and R. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research*, 25(1):449–483, May 2006.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proc. International Conference on Robotics and Automation*, St. Louis, USA, March 1985.

R. A. Knepper and M. T. Mason. Empirical sampling of path sets for local area motion planning. In *Proc. International Symposium of Experimental Robotics*, Athens, Greece, July 2008.

R. A. Knepper and M. T. Mason. Path diversity is only part of the problem. In *Proc. International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

R. A. Knepper, S. S. Srinivasa, and M. T. Mason. Curvature bounds on the weighted Voronoi diagram of two proximal paths with shape constraints. Technical Report CMU-RI-TR-10-25, Robotics Institute, Carnegie Mellon University, 2010a.

R. A. Knepper, S. S. Srinivasa, and M. T. Mason. Hierarchical planning architectures for mobile manipulation tasks in indoor environments. In *Proc. International Conference on Robotics and Automation*, Anchorage, USA, May 2010b.

J. P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *Intelligent Autonomous Systems, An International Conference*, Amsterdam, The Netherlands, December 1986.

S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7/8):673–692, July/August 2004.

V. Lumelsky and A. Stepanov. Automaton moving admist unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *Proc. International Conference on Robotics and Automation*, May 2010.

D. S. Minhas, J. A. Engh, M. M. Fenske, , and C. N. Riviere. Modeling of needle steering via duty-cycled spinning. In *Proc. Conference of the IEEE EMBS*, 2007.

J. R. Munkres. *Topology.* Prentice Hall, Upper Saddle River, NJ, 2000.

H. Niederreiter. *Random Number Generation and Quasi-Monte-Carlo Methods.* Society for Industrial Mathematics, Philadelphia, 1992.

J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

J. Reif and H. Wang. The complexity of the two dimensional curvature-constrained shortest-path problem. In *Third International Workshop on Algorithmic Foundations of Robotics*, pages 49–57, June 1998.

P. Sampl. Medial axis construction in three dimensions and its application to mesh generation. *Engineering with Computers*, 17(3):234–248, 2001.

G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *International Journal of Robotics Research*, 21(1):5–26, 2002.

E. Schmitzberger, J.L. Bouchet, M. Dufaut, D. Wolf, and R. Husson. Capture of homotopy classes with probabilistic road map. In *Proc. International Conference on Intelligent Robots and Systems*, October 2002.

F. Schwarzer, M. Saha, and J.C. Latombe. Exact collision checking of robot paths. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, pages 25–41, 2004.

R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. International Conference on Robotics and Automation*, Minneapolis, Minnesota, April 1996.

K. Sun and V. J. Lumelsky. Motion planning for three-link robot arm manipulators operating in an unknown three-dimensional environment. In *Proc. Conference on Decision and Control*, 1991.

C. E. Thorpe. Path relaxation: Path planning for a mobile robot. In *Proc. National Conference on Artificial Intelligence*, pages 318–321, 1984.

J. van den Berg, S. Patil, R. Alterovitz, P. Abbeel, and K. Goldberg. Lqg-based planning, sensing, and control of steerable needles. In *The Ninth International Workshop on the Algorithmic Foundations of Robotics*, Singapore, December 2010.

M. Vendittelli, J. P. Laumond, and C. Nissoux. Obstacle distance for car-like robots. *IEEE Transactions on Robotics and Automation*, 15:678–691, 1999.

R. J. Webster III, J. S. Kim, N. J. Cowan, G. S. Chirikjian, and A. M. Okamura. Nonholonomic modeling of needle steering. *International Journal of Robotics Research*, 25(5–6):509–525, 2006.

C. K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete & Computational Geometry*, 2:365–393, 1987.

Y. Yu and K Gupta. An information theoretic approach to view point planning for motion planning of eye-in-hand systems. In *International Symposium on Robotics*, pages 306–311, 2000.