Survivability: Measuring and Ensuring Path Diversity

Lawrence H. Erickson and Steven M. LaValle
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801 USA
{lericks4, lavalle}@uiuc.edu

***Abstract*— A novel criterion is introduced for assessing the diversity of a collection of paths or trajectories. The main idea is the notion of survivability, which measures the likelihood that numerous paths are obstructed by the same obstacle. This helps to improve robustness with respect to collision, which is an important challenge in the design of real-time planning algorithms. Efficient algorithms are presented for computing the survivability criterion and for selecting a subset of paths that optimize survivability from a larger collection. The algorithms are implemented and solutions are illustrated for four different systems. Chi-square tests are used to show uniform coverage obtained by using the computed paths in a simple breadth-first search. Random obstacle placement is used to show superior robustness of these primitives compared to uniform sampling of the control space.**

I. INTRODUCTION

For over a decade, planning algorithms have been hampered by the challenging issue of designing effective motion primitives. This is particularly an issue with nonholonomic and kinodynamic planning, in which it becomes important to efficiently overcome differential constraints while simultaneously searching for a collision-free trajectory. A discussion of this issue in the context of rapidly exploring random trees appears in [6]. For particular systems, the careful construction of effective primitives dramatically improved performance in [2], [3]. In much earlier work, an efficient nonholonomic planner was made by using Dubins curves as primitives [4]. Given the recent technological advances in autonomous vehicles, there has been substantial interest in real-time planning in cluttered, unpredictable environments. This has prompted the development of methods that design motion primitives for outdoor vehicle navigation [7], [10].

With all of the practical demonstrations that finding the "right" primitives dramatically helps in planning algorithms, we wonder what criterion should one optimize in choosing them? If this issue becomes well-understood, then it should be possible to automate the design of effective motion primitives as problems arise in new contexts.

The quest for a good collection of paths is related to basic sampling issues that have arrived in many contexts. For example, selecting points that optimize *discrepancy* is important for numerical integration [9]. Choosing samples in configuration space that reduce *dispersion* (radius of the largest empty ball) is desirable in optimization [9], [12] and motion planning [5]. An interesting extension of discrepancy to the space of paths, which is more closely related to the current paper, appears in [11].

Our paper is inspired by the recent works of Branicky, Knepper, and Kuffner, in which they introduce the notion of *path diversity* for designing a robust, efficient collection of paths or trajectories [1]. They discretize the environment into square segments and then greedily choose paths that minimize the number of squares that the paths hold in common. This makes important progress toward designing better primitives and improving our understanding; however, some issues remain. One limitation is that the definition depends on an arbitrary discretization of the state space into boxes. If the discretization is much coarser than the obstacles, then the method may report that two well-separated paths are likely to be blocked by the same obstacle. On the other hand, if the discretization is much smaller than the typical obstacle size, then two paths that might in fact be very close to each other could be reported as being well-separated.

In this paper, we introduce a new criterion, called *survivability*, which tries to assess the likelihood that other paths survive when a path in the collection is destroyed by an obstacle. The idea is to bombard each path by obstacles and characterize the collateral damage to other paths. Rather than fixing a particular resolution for obstacles, our survivability definition averages over all resolutions, thereby avoiding the discretization sensitivity in the path diversity definition from [1].

Section II introduces the mathematical definition of survivability, which is expressed in terms of integrals over all paths and obstacle sizes. Section III introduces an algorithm that computes the survivability in time $O(n^2 p \lg^2 p)$, in which n is the number of paths in the collection and p is the number of sample points chosen along each path to evaluate the integrals. Section IV then uses the algorithm to develop an $O(Nn^2 p \lg^2 p)$-time algorithm that selects a set of n paths from a larger collection of N paths by optimizing the computed survivability. The algorithms are implemented, and Section V presents experiments on four systems: a hovercraft, a double integrator, Dubins car, and a car pulling four trailers. The experiments show that primitives selected by optimizing survivability have better uniform exploration properties and robustness with respect to obstacles in comparison to primitives chosen by uniformly sampling the control space and primitives chosen at random.

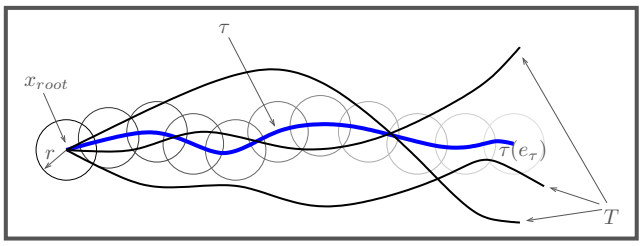

Fig. 1. An illustration of the inner integral of Equation 1. An obstacle of constant radius is dragged along τ.

II. Survivability

This section introduces a new measure, called *survivability*, for a collection of paths. Let $X \subset \mathbb{R}^n$ be a *state space*, which is assumed to be a typical manifold resulting from the *configuration space* of a robot, or possibly the *phase space*, which includes configuration and velocity variables. Let $\rho : X \times X \to [0, \infty)$ denote the distance metric on X.

Let $\tau : [0, e_\tau] \to X$ denote a continuous, finite-length, *path* parameterized with constant speed, in which $e_\tau > 0$ is the endpoint of the path domain. Let T be any finite collection of such paths, with the assumption that there is some fixed state x_{root} called the *root*, for which $\tau(0) = x_{root}$ for all $\tau \in T$. The domain endpoint e_τ may be different for each $\tau \in T$. The path parameter $s \in [0, e_\tau]$ may or may not correspond to time.

The main idea will be to evaluate paths based on whether they can "survive" an obstacle in X that blocks them. For an obstacle $\mathcal{O} \subset X$, a path is said to be *blocked* if there exists some $s \in [0, e_\tau]$ for which $\tau(s) \in \mathcal{O}$. Let *r-survivability* $\alpha(T, r, x) \in [0, 1]$ be the fraction of paths in T that are blocked by a ball of radius r centered at x. Since the choices of r and x are arbitrary, consider averaging over all possibilities. This yields the *survivability* $\sigma_\tau(T)$ of a path τ in the collection T, which is defined as

$$\sigma_\tau(T) = \frac{1}{e_\tau} \int_0^\infty \int_0^{e_\tau} \alpha(T, r, \tau(s)) ds dr. \tag{1}$$

Note that the upper bound of ∞ on the outer integral may be replaced by ℓ_{max}, the maximum length over all paths in T. All paths in T are blocked for radii larger then ℓ_{max} and therefore nothing more contributes to $\sigma_\tau(T)$.

The *survivability* of a set T of paths is defined by averaging (1) over all $\tau \in T$ to obtain:

$$\sigma(T) = \frac{1}{|T|} \sum_{\tau \in T} \sigma_\tau(T - \tau). \tag{2}$$

Note that because the limits of integration are independent of each other the integration order can be swapped.

Intuitively, $\sigma(T)$ attempts to measure the amount of collateral damage a single path τ is likely to do to the other paths in T if it is blocked by some obstacle. A path τ that has many intersections with other paths is problematic because an obstacle that blocks τ is likely to block the paths that intersect τ. Paths that almost intersect are also dangerous for the same reason. A path τ for which most of its image is close to x_{root} is extremely dangerous because an obstacle placed along it would likely block x_{root} and consequently all paths in T.

III. Calculating the Survivability

For very simple types of paths, such as straight lines originating from x_{root}, it is possible to compute $\sigma(T)$ analytically. However, as the dimension of the problem increases and the paths become more complicated, finding an exact value of $\sigma(T)$ becomes difficult or impossible. In these situations, we represent each path as a sequence of evenly-spaced points in X:

$$\tau \approx \hat{\tau} = (\tau(s_1), \tau(s_2), \tau(s_3), \ldots, \tau(s_n)), \tag{3}$$

for which $0 \leq s_1 < \cdots < s_n \leq e_\tau$.

Let $\hat{T}$ be the set in which each $\tau \in T$ is replaced by $\hat{\tau}$. The building block of computing survivability is Algorithm 1. It uses a subroutine called NEAREST$(x, \hat{\tau}_c)$ which returns the element in $\hat{\tau}_c$ that is closest to x.

Algorithm 1 ORDERED_PAIRWISE$(\hat{\tau}, \hat{\tau}')$

```
a ← 0
n is the number of points in τ̂
for i = 1 to n do
    x_c = NEAREST(τ(s_i), τ̂')
    a ← a + ρ(τ(s_i), x_c)
end for
Return a/n
```

Algorithm 1 approximately calculates (1) for $\sigma_\tau(T)$ with the condition that the only member of T is τ'. Since $\sigma_\tau(T)$ simply evaluates the fraction of surviving paths for different obstruction sizes and different points, it can be approximately computed by simply running Algorithm 1 for every $\hat{\tau}' \in \hat{T}$ and averaging over the results, giving Algorithm 2.

Algorithm 2 SINGLE_PATH_SURVIVABILITY$(\hat{\tau}, \hat{T})$

```
a ← 0
n ← |T̂|
for i = 1 to n do
    a ← a + ORDERED_PAIRWISE(τ̂, T̂_i)
end for
Return a/n
```

Algorithm 3 calculates (2) by iterating Algorithm 2 over all $\hat{\tau} \in \hat{T}$.

Algorithm 3 SURVIVABILITY$(\hat{T})$

```
a ← 0
n ← |T̂|
for i = 1 to n do
    a ← a + SINGLE_PATH_SURVIVABILITY(τ̂_i, T̂ − τ̂_i)
end for
Return a/n
```

Algorithm 3 executes quickly. If all of the paths are organized into kd-trees, which enable $O(n \lg n)$ construction time and $O(\lg n)$ lookup time, then, assuming that all paths have fewer than p points, Algorithm 1 takes $O(p \lg^2 p)$ time. There are two nested loops in Algorithms 2 and 3 that each run in $O(n)$ time; hence, the entire calculation runs in time $O(n^2 p \lg^2 p)$, in which n is the number of paths in T.

IV. Selecting the best paths

Suppose you are given a very large set of N paths T_{all}, and you want to select a small subset of n paths that maximize the survivability. Choosing the optimal set of n paths from a master set T_{all} of N paths would require checking every n-sized subset of T_{all}. Unfortunately, that would take $O(n!)$ time. However, if a subset T_{sub} of $n-1$ paths is provided, it is easier to find the best path to add to the subset. Simply running Algorithm 3 on the $(n-1)$-path subset augmented with every possible path would take $O(Nn^2 p \lg^2 p)$ time (assuming N is very large relative to n). Naively constructing an n-path subset step-by-step from one or two starting paths would therefore take $O(Nn^3 p \lg^2 p)$. However, adding a single path by completely recomputing Algorithm 3 for every possible added path ignores any previous computation that may have been done.

If Algorithm 3 has already been performed on T_{sub}, then the next best path to add can be found in linear time relative to n. Essentially, Algorithm 3 performs Algorithm 1 on every ordered pair of different paths in the input set. A set of size n has $n^2 - n$ ordered pairs of distinct elements. When a new path τ_{new} is added to the set, Algorithm 1 does not have to be executed on pairs that do not involve τ_{new}. If a new path τ_{new} is added to a set of n paths, there are $2n$ invocations of Algorithm 1 that involve P, meaning that a new path can be found in $O(n)$ time. Algorithm 4 calculates the value of SURVIVABILITY$(\hat{T}_{sub} + \hat{\tau}_{new})$, where $\hat{T}_{sub}$ is a pre-existing set of approximate paths, $\hat{\tau}_{new}$ is an approximate path to be added to the set, and $x =$SURVIVABILITY$(\hat{T}_{sub})$.

Algorithm 4 CHECK_SINGLE_PATH$(\hat{T}, \hat{\tau}_{new}, x)$

```
w_0 ← |T̂|(|T̂| − 1)
a ← w_0 x
for i = 1 to |T̂| do
  a ← a+ORDERED_PAIRWISE(T̂_i, τ̂_new)
  a ← a+ORDERED_PAIRWISE(τ̂_new, T̂_i)
end for
w_f ← (|T̂| + 1)|T̂|
a ← a/w_f
Return a
```

Building an n path set from 1 or 2 starting paths using this method would take $O(Nn^2 p \lg^2 p)$ time. This begs the question of how to select some constant number of starting paths that can be used as a nucleus upon which the rest of the set can be built. The easiest method is to declare the first member of the master set T_{all} to be the first member of the subset, as in [1]. This method is certainly quick, but it can be dangerous if the first path T_{all} is some extremely suboptimal path, such as one that closely circles the root for its entire length. A better method is to iterate through the master set some constant number of times K (we use $K = 3$) and look for the path that is most different from the one selected in the previous iteration, as demonstrated in Algorithm 5, which takes a master set $\hat{T}_{all}$ as input and returns two reasonable paths to use as a nucleus, along with their survivability value.

Algorithm 5 CHOOSE_2_START$(\hat{T}_{all})$

```
a ← 1
n ← |T̂|
for i = 1 to K do
  m ← −1
  b ← −1
  for j = 1 to n do
    x ←SURVIVABILITY(T̂_a + T̂_j)
    if x > m then
      m ← x
      b ← j
    end if
  end for
  if i ≠ K then
    a = b
  else
    Return T̂_a, T̂_b, and m
  end if
end for
```

In most models that have some sort of angular control, this method will quickly choose two paths that steer in opposite directions with high magnitude. However, this method is imperfect, and finding a superior way to choose some initial constant set is a subject of current research.

For certain extremely simple systems, such as the case where all motion primitives are all straight lines in the $p_x p_y$ plane emanating from x_{root}, this method will choose primitives that are as close as possible to uniform sampling. Notice the Van der Corput sequence-like behavior ([13]) of the algorithm as it selects more primitives in Figure 2.

V. Experiments

Two different types of experiments were performed. In the first set of experiments, a set of primitives was chosen to maximize survivability. These primitives were used to construct a tree in the workspace. The quality of these trees was gauged by a chi-squared measure, which rewards an even distribution of nodes and wide exploration, and punishes clumping of the nodes. A different robotic model was used in each experiment. In the second set of experiments, random obstacles were placed in the workspace, and the amount of surviving paths was measured. The primitives chosen to maximize survivability were compared to primitives chosen at random from the master set and primitives created by uniform sampling of the control space.

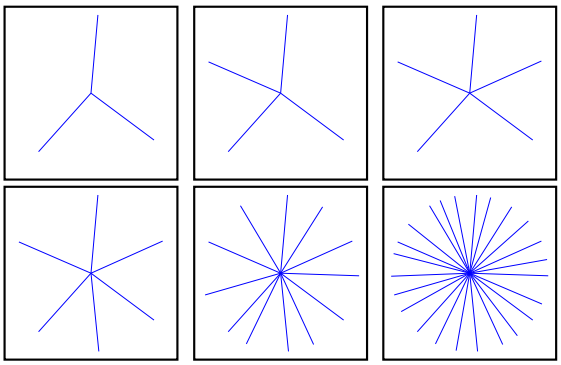

Fig. 2. The results of incrementally choosing 3, 4, 5, 6, 12, and 24 paths from an initial set of 100,000 linear paths. Two initial paths were chosen with Algorithm 5, and the other paths were chosen incrementally by iterating through the master set of paths while repeatedly calling Algorithm 4 to determine the most survivability-preserving path that was not already present in the subset. Every added path goes in the middle of the largest gap left from the previous iteration, producing a result that would be identical to uniform sampling if the proper primitives had been available from the input set.

A. Hovercraft

The hovercraft is a robot that can propell itself forward in the plane with the use of a large jet. However, since it hovers over the ground, it is not subject to friction, so the only way it can slow itself down is by rotating itself around and thrusting in a direction opposite to its movement. It is also equipped with two other jets that can be pointed in opposite directions to allow rotation.

The effectiveness of this method was tested by performing a chi-square test on a trees formed from primitives chosen with this algorithm, and comparing the results to both sets of randomly selected paths from the master set and sets of paths generated from uniformly sampled controls. The hovercraft model used in these tests has following state variables, x, y, v_x, v_y, and θ, denoting the x position, the y position, the x velocity, the y velocity, and the robot's orientation. It has two controls, u_a and u_ω. The state transition equation is

$$\begin{aligned} \dot{p}_x &= v_x \\ \dot{p}_y &= v_y \\ \dot{v}_x &= u_a \cos\theta \\ \dot{v}_y &= u_a \sin\theta \\ \dot{\theta} &= u_\omega. \end{aligned} \tag{4}$$

The value of u_a can range from 0 to 1, and the value of $\dot{\theta}$ (in radians per second) can range from $-.25$ to $.25$.

A set of 3000, 15-second long primitives were calculated from a non-moving initial state pointed in the $+x$ direction. Note that this means every path in the configuration has a different length in the state space. A weighted L_2 metric was used to calculate the distance between states x_1 and x_2 on the paths. Every set had 9 primitives in it. The environment was split into 9^5 buckets for the chi-squared calculation (every dimension was split into nine segments), and the sets of primitives were built into trees of depth 4 (for a total of 7381 nodes). Since the starting orientation, position, and velocity were all zero, and the trees were 60 seconds deep, the minimum and maximum reachable values for every $\mathbb{R}$ component are as follows:

dimension	minimum	maximum
p_x	-1125	1800
p_y	-1666	1666
v_x	-47.44	60
v_y	-57.72	57.72

Of course, not all combinations of these minimum and maximum values are actually reachable by the robot, but this is not very important when simply comparing the performance of primitive sets.

Two different uniform path sets were generated. The first one resulted from actual uniform sampling, so $u_a = 0$, $u_a = 1/2$, or $u_a = 1$. However, this creates a large number of points that simply sit at the origin, so a second uniform set was also generated where $u_a = 1/3$, $u_a = 2/3$, or $u_a = 1$. In both uniform sets, $u_\omega = -.25$, $u_\omega = 0$, or $u_\omega = .25$. Twenty randomly chosen sets of primitives were also tested. The chi-squared test results of three of the sets are shown individually, along with the average of the results of all twenty sets. All results are normalized so the result of the primitives calculated to maximize survivability is 1.000.

set	chi-squared result
calculated	1.000
random 1	5.532
random 2	2.142
random 3	11.289
20 random avg.	4.696
uniform 1	6.374
uniform 2	4.559

Figure 3 shows the results (in the $p_x - p_y$ plane) of building a depth 4 tree with three different primitive sets. The blue points represent the tree nodes, and the red lines are the edges. Note that the tree made from the primitives chosen to maximize the survivability have formed a tree that explores almost evenly in every direction. The uniformly sampled primitives (specifically, the uniform 2 set) explores far in the $+p_x$ direction, but little in the other directions. The randomly chosen set (the random 1 set) explores only a very small space.

B. Dubins Car

These experiments were also performed for more basic robotic models. The Dubins car has one input (the rate of the change in heading u_ω, which can vary from 0 to 1) and the state transition equation

$$\begin{aligned} \dot{p}_x &= \cos\theta \\ \dot{p}_y &= \sin\theta \\ \dot{\theta} &= u_\omega. \end{aligned} \tag{5}$$

These nine primitive sets were also drawn from a master set of 3000, were also 15 seconds long, and were also constructed into trees of depth 4. The path from the tree root to a leaf is 60 seconds long; the maximum and minimum

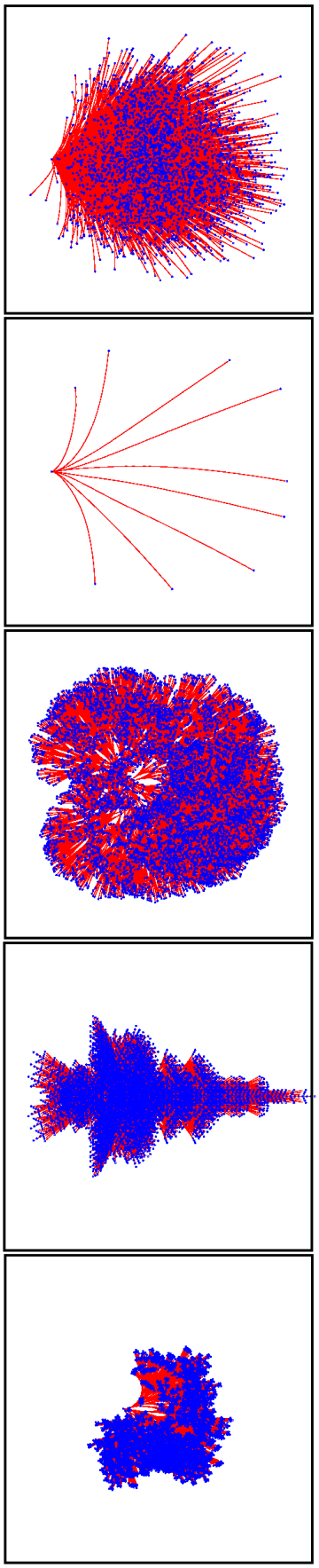

Fig. 3. [top] The entire 3000 path master set. [top middle] The primitives chosen to maximize survivability. [middle] The depth 4 tree made by the primitives chosen to maximize survivability. [bottom middle] The depth 4 tree made by uniform primitives. [bottom] The depth 4 tree made by randomly selected primitives.

reachable values for every $\mathbb{R}$ component are contained in the table below.

dimension	minimum	maximum
p_x	$-(60-\pi)$	60
p_y	$-(60-\pi/2)$	$60-\pi/2$

These chi-squared experiments divided the state space into 9^3 boxes. The normalized results are shown in the table below.

set	chi-squared result
calculated	1.000
random 1	1.412
random 2	1.783
random 3	1.623
20 random avg.	1.715
uniform	5.588

Figure 4 shows the distributions in the $p_x p_y$ plane of the points in the trees generated by primitives found to maximize survivability, the points generated by uniform sampling, and the points generated by the random 3 set. Notice that, despite the kinodynamic limitations of the Dubins car, the primitives chosen to maximize survivability search almost isotropically, while the uniform set clumps into certain branches. The random set has a distinct ditch near the top of its range.

C. Double Integrator

The double integrator has two state variables, p_x and v_x, and one input, u_a, which for these experiments varied from -1 to 1. The state transition equation is

$$\begin{aligned} \dot{p}_x &= v_x \\ \dot{v}_x &= u_a. \end{aligned} \tag{6}$$

These nine primitive sets were also drawn from a master set of 3000, were also 15 seconds long, and were also constructed into trees of depth 4. The path from the tree root to a leaf is 60 seconds long; the maximum and minimum reachable values for every dimension in the table below.

dimension	minimum	maximum
p_x	-1800	1800
v_x	-60	60

The state space was divided into 9^2 boxes for the chi-squared calculation. The normalized results are shown in the table below.

set	chi-squared result
calculated	1.000
random 1	7.471
random 2	4.835
random 3	3.877
20 random avg.	7.019
uniform	1.178

The primitives chosen to maximize survivability have excellent internal coverage. The uniform set produces a lattice, but many points lie on top of each other, which severely reduces the diversity of the point set and is a sign of poor internal coverage. The random set (random 2) in Figure 5 shows clear clumping in the upper right side, and it explores the space poorly.

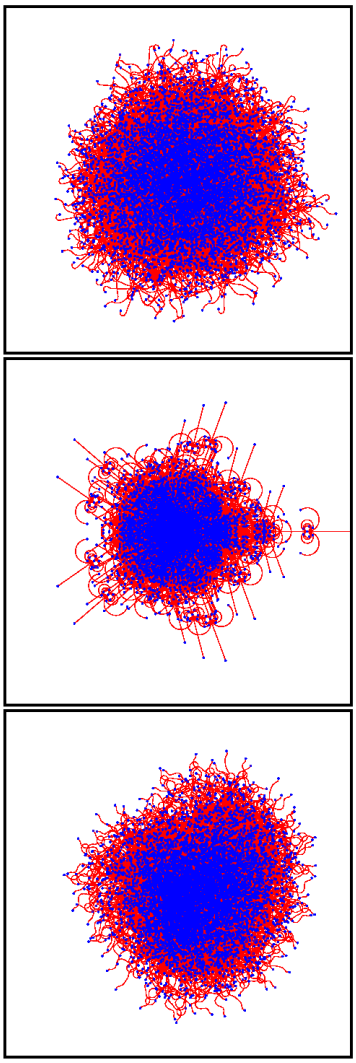

Fig. 4. [top] The depth 4 tree made by the primitives chosen to maximize survivability. [middle] The depth 4 tree made by uniform primitives. [bottom] The depth 4 tree made by randomly selected primitives.

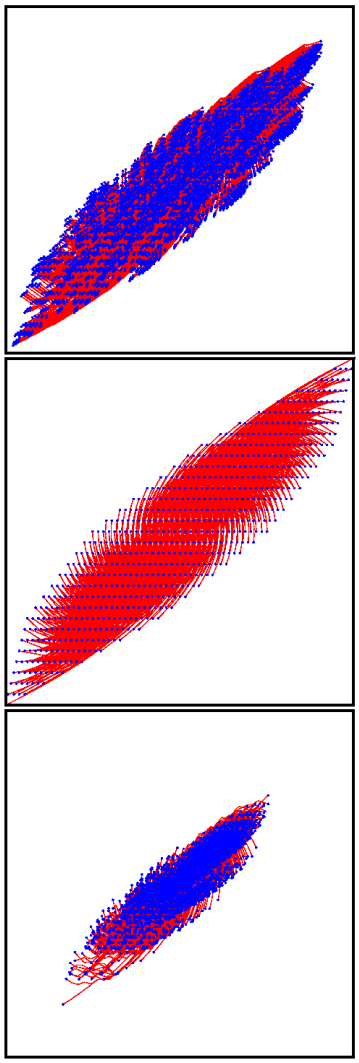

Fig. 5. [top] The depth 4 tree made by the primitives chosen to maximize survivability. [middle] The depth 4 tree made by uniform primitives. [bottom] The depth 4 tree made by randomly selected primitives.

D. Car Pulling Four Trailers

The car pulling four trailers has two inputs, u_v, the current velocity, and u_ϕ, the angle of the pulling car's front wheels, and seven state variables. For these tests, $-1 \leq u_v \leq 1$, and $-\pi/3 \leq u_\phi \leq \pi/3$. To construct the trees, 10 primitives were drawn from a pool of 3000, and trees of depth 3 were formed. There are also two constants, L, the front car length and d, the trailer hitch length, which were chosen to be $1/4$ and $3/4$ respectively. Our model is adapted from one presented in [8]. The state transition equation is

$$
\begin{aligned}
\dot{p}_x &= u_v \cos\theta_0 \\
\dot{p}_y &= u_v \sin\theta_0 \\
\dot{\theta}_0 &= \frac{u_v}{L} \tan u_\phi \\
\dot{\theta}_1 &= \frac{u_v}{d} \sin(\theta_1 - \theta_0) \\
\dot{\theta}_2 &= \frac{u_v}{d} \cos(\theta_0 - \theta_1) \sin(\theta_1 - \theta_2) \\
\dot{\theta}_3 &= \frac{u_v}{d} \cos(\theta_1 - \theta_2) \cos(\theta_0 - \theta_1) \sin(\theta_2 - \theta_3) \\
\dot{\theta}_4 &= \frac{u_v}{d} \cos(\theta_2 - \theta_3) \cos(\theta_1 - \theta_2) \cos(\theta_0 - \theta_1) \sin(\theta_3 - \theta_4).
\end{aligned}
\tag{7}
$$

The maximum and minimum reachable values for each $\mathbb{R}$ component are listed in the table below.

dimension	minimum	maximum
p_x	-44.55	45
p_y	-44.78	44.78

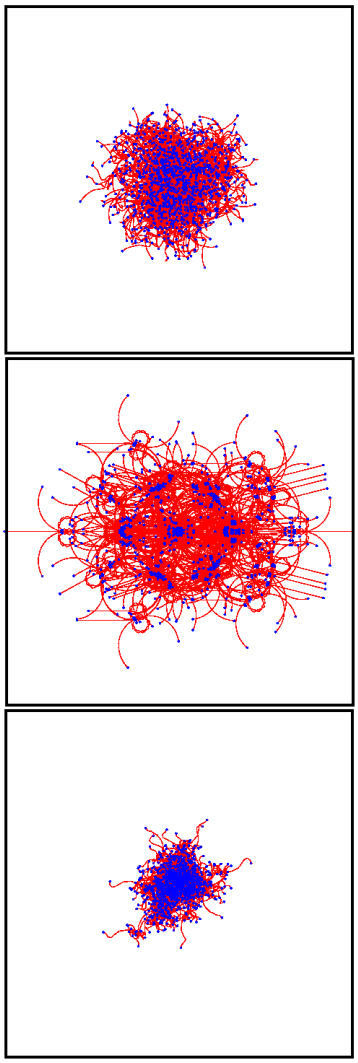

Fig. 6. [top] The depth 3 tree made by the primitives chosen to maximize survivability. [middle] The depth 3 tree made by uniform primitives. [bottom] The depth 3 tree made by randomly selected primitives.

For the chi-squared calculations, the state space was split into 7^7 regions. The normalized results are in the table below.

set	chi-squared result
calculated	1.000
random 1	1.558
random 2	1.553
random 3	1.305
20 random avg.	1.436
uniform	1.961

Note that Figure 6 only shows the robot's position in the $p_x p_y$ plane. The uniformly sampled primitives explore a wide area, but they have poor internal coverage, and many of the nodes are clustered into a few small areas. The calculated primitives have more evenly distributed internal coverage. As in the other tests, the randomly chosen primitives explore the space poorly.

Additionally, the uniform primitives explore some of the other dimensions poorly. Figure 7 shows the distribution of nodes in the $\theta_0\theta_4$ plane. From this view, it is easy to see the poor coverage of the uniform primitives and the superior coverage of the primitives that maximize survivability.

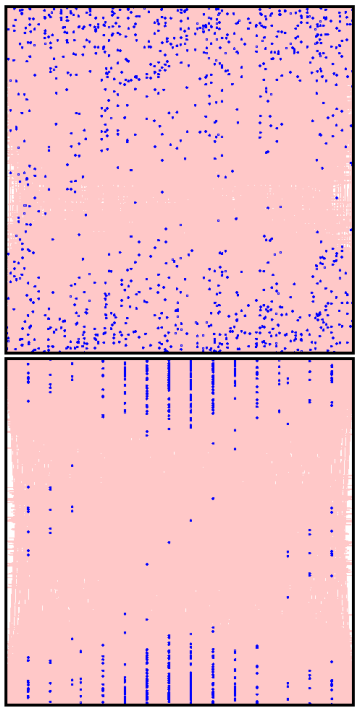

Fig. 7. Views from the $\theta_0\theta_4$ plane. [top] The depth 3 tree made by the primitives chosen to maximize survivability. [bottom] The depth 3 tree made by uniform primitives.

E. Random obstacle placement

These experiments involved placing a circular obstacle, with randomly chosen position and radius, into a workspace containing a set of 50 Dubins car paths and calculating the fraction of unblocked paths. Seven different sets were tested, one with primitives chosen to maximize survivability, one with primitives generated by uniform sampling of the controls, and five sets consisting of primitives chosen at random from the master set. Each set was tested with 5000 different obstacles, though a test was considered invalid unless at least one path in the set was blocked. The results of the tests are shown in the table below.

set	average fraction of surviving paths
calculated	0.560
random 1	0.504
random 2	0.501
random 3	0.513
random 4	0.500
random 5	0.489
uniform	0.499

About 25 paths remained unblocked on average in the random and uniform sets, compared to about 28 paths on average remaining unblocked in the set designed to maximize survivability, representing a 12% improvement.

VI. Conclusion

We have presented a new method of measuring path diversity and an algorithm that incrementally chooses the

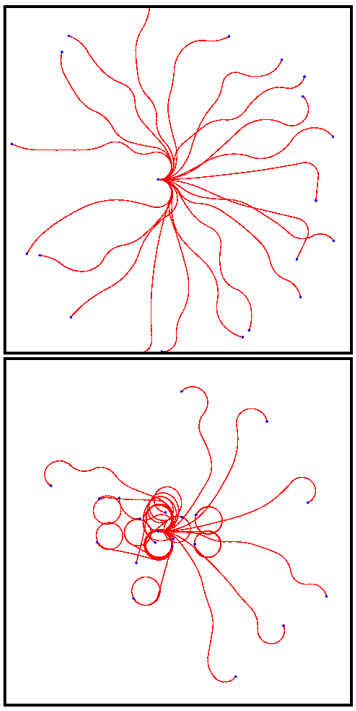

Fig. 8. [top] The Dubins car paths chosen to maximize survivability when the angular component is given a very low weight. [bottom] The Dubins car paths chosen to maximize survivability when the angular component is given a very high weight.

most diverse paths from a master set. However, there are still areas that require further exploration.

Most importantly, our definition of survivability is highly dependent on the distance metric used to define the ball in (1). Using different distance metrics will result in different optimal paths, as seen in Figure 8. It is highly probable that different metrics will be suited to different applications of the primitives. For example, if exploration of the $p_x p_y$ plane is the primary goal, then a metric that gives higher weights to the differences between p_x and p_y may be preferable. However, if the problem involves a small $p_x p_y$ space relative to the robot, but ensuring correct configuration is difficult (i.e. backing a truck with a trailer attached into a garage, where the workspace is the garage and a short driveway), then a metric that gives higher weights to the various $\mathbb{S}^1$ components may be more desirable. Finding the best metric for a given problem and robotic model is an open question.

Additionally, finding the correct number of primitives to choose given the differential constraints and the dimension of the state space is an important task. Choosing too few would unnecessarily restrict the robot's movement, but choosing too many would reduce the speed advantage granted by restricting the robot to a small number of primitives. It is unclear whether the number of primitives chosen should be based primarily on the dimension of the state space, control space, or some combination of the two.

A useful future experiment would be to use these primitives on a robot navigating through a previously mapped environment with additional unexpected obstacles. An example would be a robot attempting to drive through a valley. It may already have a path that it intends to follow, but perhaps some boulders are now obstructing parts of that path. It would have to quickly devise a plan to navigate around the boulders and get back on to its intended path. Since a path set with high survivability minimizes the chance that an obstacle blocking one path would also block the other paths in the set, it seems unlikely that a robot equipped with primitives that maximize survivability would be left without options when it encounters obstacles.

Acknowledgment

The authors are supported in part by NSF CISE grant 0535007.

References

[1] M. S. Branicky, R. A. Knepper, and J. J. Kuffner, "Path and trajectory diversity: Theory and algorithms," in *Proc. IEEE International Conference on Robotics and Automation*, 2008, pp. 1359–1364.

[2] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, Dec. 2005.

[3] J. Go, T. Vu, and J. J. Kuffner, "Autonomous behaviors for interactive vehicle animations," in *ACM SIGGRAPH Symposium on Computer Animation*, 2004.

[4] J.-C. Latombe, "A fast path planner for a car-like indoor mobile robot," in *Proceedings AAAI National Conference on Artificial Intelligence*, 1991, pp. 659–665.

[5] 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*, vol. 23, no. 7/8, pp. 673–692, July/August 2004.

[6] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A K Peters, 2001, pp. 293–308.

[7] D. Mateus, G. Avina, and M. Devy, "Robot visual navigation in semi-structured outdoor environments," in *Proc. IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 4691–4696.

[8] R. M. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

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

[10] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[11] S. Ramamoorthy, R. Rajagopal, Q. Ruan, and L. Wenzel, "Low-discrepancy curves and efficient coverage of space," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, New York, July 2006.

[12] A. G. Sukharev, "Optimal strategies of the search for an extremum," *U.S.S.R. Computational Mathematics and Mathematical Physics*, vol. 11, no. 4, 1971, translated from Russian, *Zh. Vychisl. Mat. i Mat. Fiz.*, 11, 4, 910-924, 1971.

[13] J. G. van der Corput, "Verteilungsfunktionen I," *Akademie van Wetenschappen*, vol. 38, pp. 813–821, 1935.