Stochastic Reachability Based Motion Planning for Multiple Moving Obstacle Avoidance

Nick Malone
Computer Science
University of New Mexico
Albuquerque, NM 87131, USA
nmalone@cs.unm.edu

Kendra Lesser
Electrical & Comp. Eng.
University of New Mexico
Albuquerque, NM 87131, USA
lesser@unm.edu

Meeko Oishi
Electrical & Comp. Eng.
University of New Mexico
Albuquerque, NM 87131, USA
oishi@unm.edu

Lydia Tapia
Computer Science
University of New Mexico
Albuquerque, NM 87131, USA
tapia@cs.unm.edu

ABSTRACT

One of the many challenges in designing autonomy for operation in uncertain and dynamic environments is the planning of collision-free paths. Roadmap-based motion planning is a popular technique for identifying collision-free paths, since it approximates the often infeasible space of all possible motions with a networked structure of valid configurations. We use stochastic reachable sets to identify regions of low collision probability, and to create roadmaps which incorporate likelihood of collision. We complete a small number of stochastic reachability calculations with individual obstacles a priori. This information is then associated with the weight, or preference for traversal, given to a transition in the roadmap structure. Our method is novel, and scales well with the number of obstacles, maintaining a relatively high probability of reaching the goal in a finite time horizon without collision, as compared to other methods. We demonstrate our method on systems with up to 50 dynamic obstacles.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous Vehicles*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control Theory, Dynamic Programming*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Stochastic Processes*

Keywords

Stochastic Reachability; Motion Planning;
Probabilistic Roadmaps

1. INTRODUCTION

One of the many challenges in designing autonomy for operation in uncertain and dynamic environments is the planning of collision-free paths. In applications such as search and rescue, coordinated sensing, collaborative monitoring, or automated manufacturing environments, a robot must traverse from a known start state to a goal state, in an environment that could contain many moving obstacles with stochastic dynamics. While theoretical solutions may be available via stochastic reachability, computational expense limits such an approach to a very small number of dynamic obstacles, depending on the model complexity of the robot and obstacle dynamics. Motion planning techniques provide a more computationally feasible alternative, depending on degrees of freedom of the robot, the nature of the environment, and the planning constraints. However, there is strong evidence that any complete planner will require exponential time in the number of DOFs of the robot [14], [11], [6].

In this paper, we present a novel, stochastic reachability based method to create probabilistic roadmaps that accommodate many moving obstacles that travel stochastically along straight line or arc trajectories. We use the likelihood of collision with a given object, computed a priori via stochastic reachability (SR), to inform the likelihood of collision along a given path. We demonstrate our method computationally on scenarios with up to 50 stochastic dynamic obstacles.

The robotic motion planning problem consists of finding a valid (collision-free) path for a robot from a start state to a goal state. One common solution to solving the planning problem is to use a roadmap, a network of valid configurations (nodes) and transitions between configurations (edges), that captures the topology of the collision-free space. Common approaches are cell-decomposition methods which place nodes at regular intervals [15], Probabilistic Roadmap Methods (PRMs) which place nodes probabilistically [14], and several variants which use heuristics to place nodes [2, 4]. However, planning in environments with dynamic obstacles remains a significant challenge.

Stochastic reachability analysis provides offline verification of dynamical systems, to assess whether the state of the system will, with a certain likelihood, remain within a desired subset of the state-space for some finite time, or avoid an undesired subset of the state-space [1]. To solve problems in collision avoidance, the region in the relative state-space which constitutes collision is defined as the set of states we wish the system to avoid [28, 13]. Unfortunately, the computation time for stochastic reachable sets (SR sets) is exponential in the dimension of the continuous state, making the assessment of collision probabilities with many simultaneously moving obstacles next to impossible (once the dynamics of each obstacle are incorporated into the state). However, while expensive, SR sets can be computed offline and the result queried online.

Our method combines multiple SR sets (computed pairwise between the robot and each dynamic obstacle), to generate appropriate weights associated with the edges in the roadmap. The SR sets are generated offline, computed individually for relative dynamics associated with each obstacle, and the results combined and queried at runtime by our algorithm. In an environment with multiple obstacles, the intersection of multiple SR sets clearly cannot provide a strict assurance of safety, since the reachable set is computed for one dynamic obstacle in isolation. However, such an approach can significantly improve the ability of the roadmap to reflect obstacle dynamics. Further, in simulation, we find that the SR - weighted roadmap is able to intelligently navigate in the presence of stochastic dynamic obstacles significantly more often than standard roadmap methods.

Our proposed combination of formal and ad-hoc methods has several advantages over existing moving obstacle solutions and over SR alone. First, at runtime, the method is fast since it does not have to make expensive collision detection calls and instead just queries the precomputed SR set. Second, it scales well with many obstacles. Furthermore, it provides a framework in which multiple SR sets can be combined to generate approximate collision avoidance probabilities with many moving obstacles, which would otherwise be impossible using a single SR set that accounts for all obstacles simultaneously. Finally, by using SR for the underlying collision probability calculation, the method provides an upper bound on the probability of collision avoidance, which can be used comparatively to select the best path.

Section 2 describes related literature in roadmaps with moving obstacles as well as in stochastic reachability for motion planning and collision avoidance. Section 3 presents the robot and obstacle dynamics, and known techniques for roadmap construction. Section 4 presents the computed stochastic reachable sets for collision avoidance with two types of stochastic dynamic obstacles, as well as our algorithm for roadmap construction that queries the stochastic reachable set. Section 5 describes our computational experiments, with two moving obstacles, and finally with 50 moving obstacles. Lastly, conclusions and directions for future work are offered in Section 6.

2. RELATED WORK

2.1 Roadmaps and moving obstacles

Several roadmap-based techniques including PRM variants have been developed to address planning in spaces with moving obstacles [23], [5], [30], [10], [24], [25]. Generally, these approaches adapt to moving obstacles using one of two approaches. The first category generates a roadmap with little obstacle information, and later filters paths at runtime with local obstacle information [23], [5]. These methods have low precomputation costs, but generally prove expensive during path selection. They start with an initial path that is collision free and incrementally modify the path to maintain a smooth, collision free path. These methods only rely on physical obstacle clearance by using protective bubbles to deform the path.

The second category approximates the environment and is cheap at runtime. These methods create an approximate roadmap and then use a heuristic approach to produce locally valid paths to avoid moving obstacles. These methods decrease runtime costs at the expense of path accuracy [24], [12]. In [24], a first stage constructs a dynamic roadmap that considers some obstacles and is shared across multiple moving robots. Then, in a second stage, a path is extracted by a single robot that is locally modified to account for neighboring robots (moving obstacles). Similarly, [30] repairs the existing roadmap when an obstacle makes an edge or group of edges invalid. The authors of [34] use a roadmap, but deform the edges around moving obstacles. The work in [2] trades off distance from the goal and the dynamic obstacles to path plan. Approaches in [32] and [22] utilize roadmap methods with heuristics to manage the moving obstacles, while [31] attempts to optimize the roadmap for moving obstacles under motion constraints.

2.2 Stochastic reachable sets

A Hamilton-Jacobi-Bellman (HJB) formulation [21] allows for both a control input and a disturbance input to model collision-avoidance scenarios [19], [9] for motion planning. The result of these reachability calculations is a maximal set of states within which collision between two objects is guaranteed (in the worst-case scenario), also known as the reachable set. The set which assures collision avoidance is simply the complement of the reachable set. In [29], reachable sets are calculated to assure a robot safely reaches a target while avoiding a single obstacle, whose motion is chosen to maximize collision, and the robot cannot modify its movements based on subsequent observations. In [8], a similar approach is taken, but with reachable sets computed iteratively so that the robot can modify its actions. In [17], multiple obstacles that each act as bounded, worst-case disturbances are avoided in an online fashion, based on precomputed invariant sets.

An alternative approach is to calculate an SR set that allows for obstacles whose dynamics include stochastic processes. Discrete-time SR generates probabilistic reachable sets [1], based on stochastic system dynamics. In [28], the desired target set is known, but the undesired sets that the robot should avoid are random and must be propagated over time. In [13] a two-player stochastic dynamical game is considered, and applied to a target tracking application in which the target acts in opposition to the tracker.

3. PRELIMINARIES

3.1 Obstacle Dynamics

We consider two representative types of dynamic obstacles, which have known trajectories but stochastic velocities. In particular, we consider straight-line and constant-arc trajectories, and presume that each obstacle is represented as a two-dimensional point mass. The obstacle dynamics are of the form $\dot{\overline{x}}^o = f(w, t)$, with obstacle state $\overline{x}^o = (x^o, y^o)$, and with w a discrete random variable that takes on values in $\mathcal{W}$ with probability distribution $p(w)$. We only consider a discrete random variable here for computational simplicity, although a continuous random variable could be introduced. The discretized obstacle dynamics (via an Euler approximation with time step Δ) are

$$\begin{aligned} x^o_{n+1} &= x^o_n + \Delta w_{n+1} \\ y^o_{n+1} &= \alpha x^o_{n+1} \end{aligned} \tag{1}$$

for straight-line movement, with speed $w \in \mathcal{W}$ and slope $\alpha \in \mathbb{R}$, and

$$\begin{aligned} x^o_{n+1} &= x^o_n + \Delta r\left(\cos(w_{n+1}(n+1)) - \cos(w_{n+1}n)\right) \\ y^o_{n+1} &= y^o_n + \Delta r\left(\sin(w_{n+1}(n+1)) - \sin(w_{n+1}n)\right) \end{aligned} \tag{2}$$

for constant-arc movement, with angular speed $w \in \mathcal{W}$.

3.2 Relative robot-obstacle dynamics

We presume a two-dimensional point-mass model for the robot with state $\overline{x}^r = (x^r, y^r)$ and dynamics in Cartesian coordinates

$$\begin{aligned} \dot{x}^r &= u_x \\ \dot{y}^r &= u_y \end{aligned} \tag{3}$$

with two-dimensional control input $u = (u_x, u_y)$ that is the velocity of the robot in both directions. While the obstacle is not trying to actively collide with the robot, its dynamics (1), (2) contain a stochastic component, which can be considered a disturbance that affects the robot's behavior. Discretizing the dynamics (3) using an Euler approximation with time step Δ results in

$$\overline{x}^r_{n+1} = \overline{x}^r_n + \Delta \cdot u. \tag{4}$$

A collision between the robot and the obstacle occurs when $|\overline{x}^r_n - \overline{x}^o_n| \leq \epsilon$ for some n and ϵ small. We construct a relative coordinate space that is fixed to the obstacle, with the relative state defined as $\tilde{x} = \overline{x}^r - \overline{x}^o$. Hence the dynamics of the robot *relative* to the obstacle are

$$\tilde{x}_{n+1} = \tilde{x}_n + \Delta u_n - \Delta f(w_n, t_n) \tag{5}$$

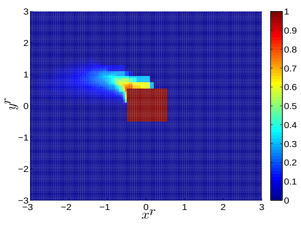
with $f(\cdot)$ as in (1) and (2), and a *collision* is defined as

$$|\tilde{x}_n| \leq \epsilon. \tag{6}$$

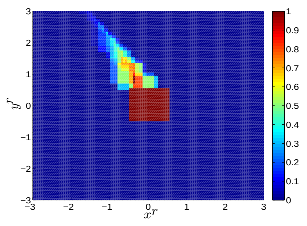
Using (5), we now have a dynamical system with state $\tilde{x} \in \mathcal{X}$, control input $u \in \mathcal{U}$ that is bounded, and stochastic disturbance w. Because $\tilde{x}_{n+1}$ is a function of a random variable, it is also a random variable. Its transitions are governed by a stochastic transition kernel, $\tau(\tilde{x}_{n+1} \mid \tilde{x}_n, u_n, n)$, that represents the probability distribution of $\tilde{x}_{n+1}$ conditioned on the known values $\tilde{x}_n$, u_n and time step n.

3.3 Roadmap Construction

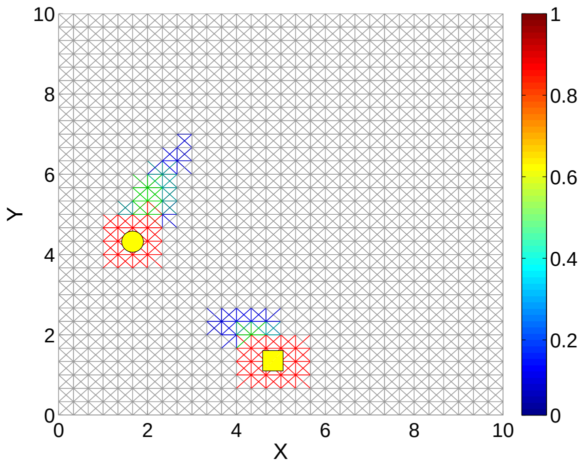
Roadmap-based techniques attempt to approximate the topology of the collision-free C-space, the space of all possible robot configurations (where a configuration completely specifies the location of every point on the robot) [16]. They work by building a graph in collision-free C-space through sampling collision-free robot configurations (*node generation*), connecting neighboring nodes with weighted edges if a collision-free transition exists (*node connection*), and then querying the resulting roadmap by finding a path to a goal configuration (*roadmap query*). Node generation can be done via several different methods, e.g., using a cell decomposition of the space [15], a uniform random distribution



(a) SR set for arc obstacle.



(b) SR set for line obstacle.



(c) Roadmap with edges weighted by SR sets

Figure 1: Two stochastic reachable sets for constant-arc and straight-line obstacles and their incorporation into a roadmap. (a) Stochastic reachable set that shows the likelihood of collision between the robot and an arc obstacle (in relative coordinates). (b) Stochastic reachable set for a straight-line obstacle with $\alpha = 1$. (c) Roadmap (Cartesian coordinates) with likelihood of collision indicated by edge color. The yellow circle (square) shows the line (arc) obstacle locations.

[14], obstacle boundaries [33], or visibility [26]. The utility of these various node generation methods varies with problem complexity. For example, cell decomposition methods are powerful, but their utility degrades with complex obstacle boundaries and in high-dimensional planning problems. On the other hand, uniform random placement, PRM, works well with high-dimensional problems, but has difficulty with obstacles that form tight narrow passages. Collision-free tests are often performed with static obstacles, and edge weights can be determined by several metrics of interest, e.g., distance [14], clearance from obstacles [18] [20], or other problem-specific measures [27].

4. METHODS

In this section, we present the novel methods for integrating SR sets with roadmap path extraction. First, we formulate the SR problem for collision avoidance with the straight-line and constant-arc dynamic obstacles. We then show how SR can be used to help build roadmaps that select a path that avoids multiple moving obstacles.

4.1 SR for Collision Avoidance

The SR problem can be formulated in the context of collision avoidance, where the probability of avoiding collisions within some finite time horizon is determined. The set $\overline{K}$ is defined as the set of states in which a collision is said to occur (6). To avoid collision with the obstacle, the robot should remain within K, the complement of $\overline{K}$. The probability that the robot will remain within K over N time steps, with initial relative position $\tilde{x}_0$, is given by

$$A_{\tilde{x}_0}^{\overline{u},N}(K) = \mathbb{P}\left[\tilde{x}_0, \ldots, \tilde{x}_N \in K \mid \tilde{x}_0, \overline{u}\right] \tag{7}$$

with $\mathbb{P}$ denoting probability and input sequence $\overline{u} = [u_0, u_1, \cdots, u_{N-1}]^T$.

Since $\mathbb{P}[x \in K] = \mathbb{E}[\mathbf{1}_K(x)]$, with $\mathbb{E}$ denoting expected value and $\mathbf{1}_K(x)$ denoting the indicator function defined as $\mathbf{1}_K(x) = 1$ for $x \in K$, and 0 otherwise, equation (7) can be rewritten as (see [1])

$$A_{\tilde{x}_0}^{\overline{u},N}(K) = \mathbb{E}\left[\prod_{n=0}^{N} \mathbf{1}_K(\tilde{x}_n) \mid \tilde{x}_0, \overline{u}\right], \tag{8}$$

since $\prod_{n=0}^{N} \mathbf{1}_K(\tilde{x}_n) = 1$ if $\tilde{x}_0, \ldots, \tilde{x}_N \in K$, and 0 otherwise.

Finally, instead of assuming a predetermined set of control inputs $\overline{u}$, we construct a state-feedback control input to maximize the likelihood of avoiding collision and to facilitate real-time control selection for motion planning. Equation (8) can then be reformulated as a stochastic optimal control problem.

$$A_{\tilde{x}_0}^{N}(K) = \max_{\pi \in \Pi} \mathbb{E}\left[\prod_{n=0}^{N} \mathbf{1}_K(\tilde{x}_n) \mid \tilde{x}_0\right] \tag{9}$$

Hence we define a policy $\pi = (\pi_0, \ldots, \pi_{N-1})$ with $\pi_n : \mathcal{X} \rightarrow \mathcal{U}$ and optimize (9) over all possible policies Π of this form. The resulting optimal policy π^* provides an upper bound on the probability of avoiding collision.

We implement a dynamic programming recursion [3], first introduced for the reachability problem in [1], to estimate the collision avoidance probability.

$$V_N(\tilde{x}) = \mathbf{1}_K(\tilde{x}) \tag{10}$$

$$V_n(\tilde{x}) = \mathbf{1}_K(\tilde{x}) \int_{\mathcal{X}} V_{n+1}(\tilde{x}')\tau(\tilde{x}' \mid \tilde{x}, u, n)\, d\tilde{x}' \tag{11}$$

Iterating (10), (11) backwards, the value function at time 0 provides the probability of avoiding collision,

$$V_0(\tilde{x}_0) = A_{\tilde{x}_0}^{N}(K). \tag{12}$$

The optimal control is determined by evaluating

$$V_n^*(\tilde{x}) = \sup_{u \in \mathcal{U}} \left\{ \mathbf{1}_K(\tilde{x}) \int_{\mathcal{X}} V_{n+1}^*(\tilde{x}')\tau(\tilde{x}' \mid \tilde{x}, u, n)\, d\tilde{x}' \right\} \tag{13}$$

which also returns the optimal policy π^*, with

$$\pi_n^*(\tilde{x}) = u_n = \arg\sup_{u \in \mathcal{U}} V_n^*(\tilde{x}). \tag{14}$$

Equation (13) can be simplified to

$$V_n^*(\tilde{x}) = \max_{u \in \mathcal{U}} \left\{ \mathbf{1}_K(\tilde{x}) \sum_{w \in \mathcal{W}} V_{n+1}^*\left(\tilde{x} + \Delta u - \Delta f(w, n)\right) p(w) \right\}. \tag{15}$$

Figure 1a shows the SR set for a constant-arc obstacle with radius $r = 5$, and probabilities $p(w) = \{0.2, 0.2, 0.3, 0.3\}$ associated with angular speeds $w \in \mathcal{W} = \left\{\frac{.4}{2\pi}, \frac{.6}{2\pi}, \frac{.9}{2\pi}, \frac{1.2}{2\pi}\right\}$. The slight curvature seen in the probability peaks corresponds to the obstacle trajectory. Similarly, Figure 1b shows the SR set for a straight-line obstacle with probabilities $p(w) = \{0.3, 0.4, 0.3\}$ associated with speeds $w \in \mathcal{W} = \{0.5, 0.7, 0.9\}$, and slope $\alpha = -1$. The peaks show higher probability of collision with the obstacle when the robot is in line with the obstacle trajectory. Intuitively, the closer the robot is to the obstacle, the higher the probability of collision.

On a single core of an Intel 3.40 GHz CORE i7-2600 CPU with 8 GB of RAM, Figure 1a took 1727.25 seconds to compute, over a horizon of $N = 30$ steps and a time step of length $\Delta = 1$. Figure 1b took 1751.87 seconds to compute, again with $N = 30$ and $\Delta = 1$. In both cases, we observed convergence in the stochastic reachable sets for $N > 5$ since the robot and obstacle traveled sufficiently far apart within this time frame.

With a single obstacle, $V_0^*(\tilde{x}_0)$ in (15) is the maximum probability of avoiding a collision, and hence a tight upper bound. For two obstacles with separately calculated avoidance probabilities $V_0^{*,1}(\tilde{x}_0^1)$, $V_0^{*,2}(\tilde{x}_0^2)$ (with relative position $\tilde{x}_0^i$ with respect to obstacle i), the probability of avoiding collision with *both* obstacles is

$$\begin{aligned} \mathbb{P}[B_1 \cap B_2] &= \mathbb{P}[B_1] + \mathbb{P}[B_2] - \mathbb{P}[B_1 \cup B_2] \\ &\leq \min\{\mathbb{P}[B_1], \mathbb{P}[B_2]\} \\ \mathbb{P}[B_1 \cap B_2] &\leq \min\{V_0^{*,1}(\tilde{x}_0^1), V_0^{*,2}(\tilde{x}_0^2)\} \end{aligned} \tag{16}$$

where B_i corresponds to the event that the robot avoids collision with obstacle i. We obtain an upper bound on the collision avoidance probability for two obstacles by taking the minimum of the individual avoidance probabilities. The same holds similarly for m obstacles. The minimum of the m individual avoidance probabilities provides an upper bound on the probability of avoiding collision with all m obstacles.

Lastly, we note that because we ultimately use the collision avoidance probabilities to determine routing choices on a roadmap, the true probabilities are of less interest than the relative probabilities at different locations. By generating an upper bound on the probability of avoiding collision with several moving obstacles, the robot can identify and travel along the path with the greatest upper bound. Further, if the obstacles are not so dense that avoiding the obstacle with the highest probability of collision implies a greater likelihood of avoiding all other obstacles as well, then this upper bound is fairly tight, and the robot can accurately identify the safest route through the roadmap.

4.2 SR Query

We now integrate SR sets (12) for straight-line and constant-arc obstacles into a pre-computed roadmap using techniques developed for static obstacles [15, 14]. Given a roadmap and an SR set for each moving obstacle, we identify paths that are likely to be free.

Algorithm 1 describes integration of the stochastic reachable sets into an existing roadmap via the roadmap query process. Although the SR calculation is performed offline, Algorithm 1 is intended to run in real time, using the information currently available to the robot (i.e. obstacle locations). Paths are extracted using Dijkstra's algorithm [7].

Algorithm 1 SR Query

Input: Obstacles O, $Roadmap$, Max time $T = N \cdot \Delta$
Output: boolean $Success$

```
 1: $nextNode = start$
 2: $previousNode = start$
 3: for $t_n = 0$; $t_n < T$; $n = n + 1$ do
 4:   for Obstacle $o \in O$ do
 5:     $updateObstacle(o)$
 6:   end for
 7:   if at(robot, nextNode) then
 8:     for each edge $e \in Roadmap$ do
 9:       $EdgeWeight = updateEdgeWeights(e, O)$
10:     end for
11:     $Path = Dijkstras(previousNode, GoalNode)$
12:     $nextNode = Path.next$
13:     $x^r_{n+1} = Path.next.getXVelocity()$
14:     $y^r_{n+1} = Path.next.getYVelocity()$
15:   end if
16:   $\overline{x}^r_{n+1} = interp(previousNode, nextNode, t_n)$
17: end for
```

However, to find paths of combined shortest distance and lowest probability of collision, the SR computation must be integrated into the roadmap edge weights. First, since the robot knows the location of each obstacle at the current time, the positions of the obstacles are updated to reflect their current locations. Second, we consider each node in the roadmap to be a waypoint. Updates of the roadmap weights are then performed at waypoints (see Algorithm 1, line 7). Updates consist of reweighting all edges (line 9), finding the path of lowest edge weight (line 11), querying the SR optimal control (14) to determine the robot's speed and resulting trajectory (lines 13 and 14), and traversing along that edge with the determined robot speed for the allotted time (line 15). If the robot is not at a waypoint, then it continues along the predetermined roadmap edge.

Two elements that are critical to the success of Algorithm 1 and atypical for probabilistic road maps are 1) updating of the obstacles, and 2) the subsequent effect on edge weights.

Regarding the first element, the likelihood of avoiding collision (12) and the optimal control (14) are evaluated over a discretized set of states, and are stored for use during run time for path planning. The algorithm propagates the location of the obstacles according to each obstacle's stochastic dynamics (1), (2). The stochastically determined obstacle speeds are chosen as per the randomization in Algorithm 2. The relative states are computed for every robot-obstacle pair in the environment.

Regarding the second element, edges define a transition between two configurations (see Section 3.3). These edges can be subdivided (often uniformly) into sets of discrete points defining the transition between configurations in the roadmap, and each point corresponds to a new intermediate configuration. The weight for a single edge is updated as in Algorithm 3. For each intermediate configuration associated with an edge, the relative distance to each obstacle is calculated and the probability of collision avoidance at the current relative distance is queried for each obstacle. The minimum of all avoidance probabilities is taken as the weight for

Algorithm 2 updateObstacle

Input: time $t_n = n \cdot \Delta$, obstacle o, velocities $w \in \mathcal{W} = \{w_1, w_2, ..., w_{n_W}\}$, probabilities $p(w)$

```
 1: if $mod(t_n, 1) == 0$ then
 2:   $s = rand(0, 1)$
 3:   for $index = 0$; $index < n_W$; $index$++ do
 4:     if $s \leq p(w)[index]$ then
 5:       $o.w = w[index]$
 6:       break
 7:     end if
 8:   end for
 9: end if
10: $\overline{x}^o_{n+1} = \overline{x}^o_n + \Delta \cdot f(o.w, t_n)$
```

Algorithm 3 updateEdgeWeight

Input: Edge e, Obstacles O

```
 1: $EdgeWeight = 0$
 2: for Configuration $c \in e$ do
 3:   $PROB = 1$
 4:   for Obstacle $o \in O$ do
 5:     $\tilde{x} = c - o$
 6:     $PROB = \min\{PROB, o.V^*_0(\tilde{x})\}$
 7:   end for
 8:   if $PROB < EdgeWeight$ then
 9:     $EdgeWeight = PROB$
10:   end if
11: end for
12: $e.Weight = \frac{1}{EdgeWeight}$
```

that configuration. This calculation is fast in comparison to standard collision detection methods whose computational complexity is defined by the number of polygons in the planning problem. The assigned edge weight is then the lowest probability of collision avoidance amongst all intermediate configurations for that edge, inverted for use in Dijkstra's algorithm (which finds minimum cost paths for graphs with nonnegative edge weights). Note that we presume the same time horizon N and time step Δ in Algorithm 1 as we do in the reachability calculations.

5. EXPERIMENTS

We evaluated our method on successful navigation in environments with several moving obstacles. Successful navigation is defined as the ability to find a path from a start state to goal state, without any collisions and within a specified time horizon. The stochastic reachable sets were computed in Matlab, and the SR Query was added to the Parasol Motion Planning Library (PMPL) from Texas A&M University. PMPL was also used to generate the initial roadmaps. Experiments were run on a single core of an Intel 3.40 GHz CORE i7-2600 CPU with 8 GB of RAM.

We compared our method (SR Query) to a Lazy-based method (Lazy) for moving obstacle avoidance [12]. The Lazy method updates the roadmap as obstacles move, by invalidating edges and nodes that are found to be in collision with the new position of the moving obstacles. This comparison

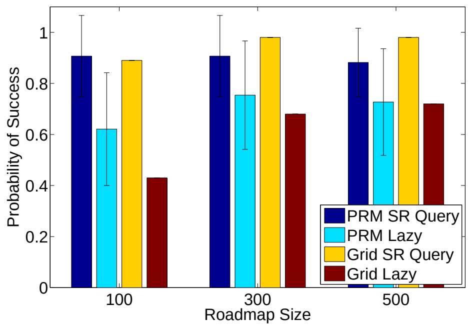
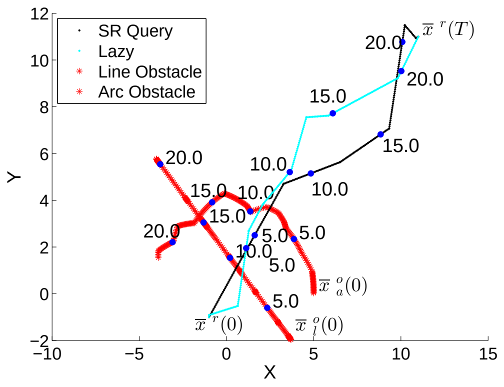
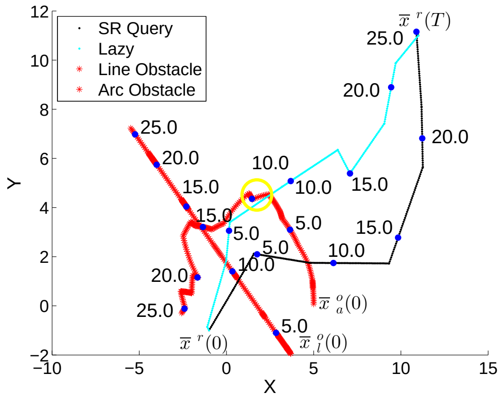


Figure 2: Comparison of SR Query and Lazy methods for the two dynamic obstacle experiment. Averaged likelihood of successfully traversing a collision-free path within the allotted time horizon for a given roadmap size, for Grid-based maps and PRM roadmaps. Note: Grid runs do not have error bars since there is only a single cell decomposition for a given roadmap size.



(a) Similar paths generated by SR Query and Lazy methods for two dynamic obstacle scenario.



(b) Different paths generated by SR Query and Lazy methods for two dynamic obstacle scenario.

Figure 3: Sample trajectories found by SR Query (black line) and Lazy (blue line) methods on a PRM roadmap with two obstacles (red lines) over $T = N\Delta = 30$ seconds. (a) Both methods found qualitatively similar paths, due to little obstacle interference. (b) SR Query and Lazy found very different paths, likely due to a near miss with one of the dynamic obstacles (yellow circle).

shows the accuracy gained by considering the probabilities of collision instead of just the obstacles' current locations.

Furthermore, we show the flexibility of the method by running experiments with node generation done with a uniform random distribution (PRM) [14] and with a regular cell decomposition (Grid) [15]. While cell decompositions can be ideal solutions, they are often infeasible for planning problems with several or complex static obstacles or of high dimensionality. In those cases, PRMs are often preferred. Since both types of roadmaps are treated the same way by the algorithm, we investigate how the topology of the roadmap can impact our method. In the Grid roadmaps, every node is connected with up to 8 adjacent neighbors. PRM roadmaps are constructed with uniform random sampling and each node is connected to its five closest neighbors.

5.1 Two Moving Obstacles

In this experiment, the robot navigates across a planning space while avoiding two dynamic obstacles that follow straight-line (1) and constant-arc (2) dynamics from initial conditions $\overline{x}_l^o(0)$, $\overline{x}_a^o(0)$. The robot's start state and goal state are at the opposite corners of a 20×20 planning space (Figure 3a). The obstacle trajectories are chosen to generate sufficient opportunities for conflict with the robot, and obstacles may exit the planning space.

In order to evaluate the performance of our algorithm, we constructed roadmaps of $|\mathcal{N}| = 100$, 300, and 500 nodes using the standard PRM method. For each map size, we used 10 random seeds to create 10 different PRM roadmaps. We also produced Grid roadmaps of size $\lfloor\sqrt{|\mathcal{N}|}\rfloor^2$ nodes, where $\mathcal{N}$ is the number of nodes in the corresponding PRM roadmap, to account for their square and unformly spaced node structure. We simulated 100 obstacle pair trajectories, resulting in $10 \times 100 = 1,000$ simulations for each map size. The success of the algorithm was measured by collision-free path completion (the robot reaching the target) within the given time horizon. To be conservative, we also declared instances in which the robot did not find a collision-free path within the allotted time horizon as unsuccessful. However, time horizons are only applicable for the SR Query method since the Lazy method is allowed to run until a path is found, no path exists, or a collision occurs. Each simulation was run for $T = 30$ seconds with a sampling interval of $\Delta = 1$ seconds ($N = 30$ time steps).

Figure 2 shows the effect of map size as well as the relative effectiveness of the two methods on PRM and Grid roadmaps in terms of the mean percentage of success. For the PRM roadmaps, SR Query was able to find successful paths 88% to 91% of the time, based on roadmap size. The error bars show how the randomized roadmap structures impact the success rate. In comparison, the Lazy method found successful paths 63% to 75% of the time. Unsuccessful runs of Lazy were due either to pruned nodes and edges that made traversal to the goal impossible, or direct collision with a moving obstacle. Error bars are not included for Grid due to the static map structure of a cell decomposition. In comparing Grid-based maps to PRM roadmaps, we find that the Grid-based maps produce better results for SR Query with larger map sizes, but poorer results for Lazy (for all map sizes). This is consistent with evidence that

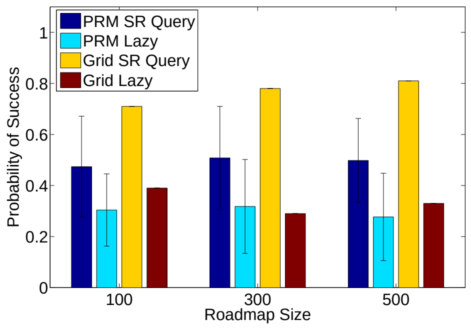


Figure 4: Comparison of SR Query and Lazy methods for the 50 dynamic obstacle experiment. Averaged likelihood of successfully traversing a collision-free path within the allotted time horizon for a given roadmap size, for Grid-based maps and PRM roadmaps. Note: Grid runs do not have error bars since there is only a single cell decomposition for a given roadmap size.

Grids perform as well or better than randomized roadmaps in environments without static obstacles [15]. In all cases (Grid-based or PRM roadmaps), the SR Query method performs between 15% and 45% better than the Lazy method.

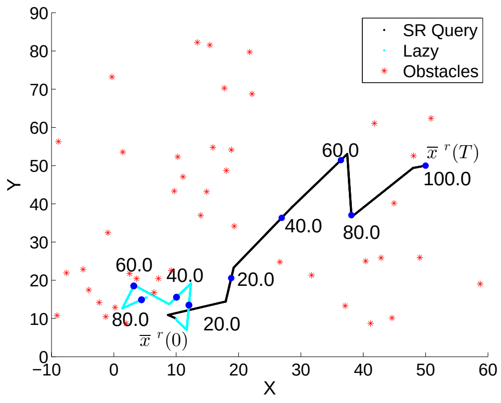
We further examine the paths selected by the two algorithms. In Figure 3a, the path generated via the SR Query method (black line) is fairly similar to the path generated via the Lazy method. The moving obstacles are shown in red, and time is indicated as labeled waypoints along each path. Both Lazy and SR Query methods follow the same path initially, but at around $t = 10$ seconds, the SR Query method identifies an incoming obstacle and moves the robot away from the obstacle. However, the Lazy method does not anticipate a possible collision, and so it does not change its path. In this case, the Lazy method allows the robot to barely pass in front of the obstacle. A similar near collision for the Lazy method is shown in Figure 3b. In this example, the paths for SR Query and Lazy are the same for the first 10 seconds. Again, SR Query anticipates an incoming obstacle and changes its path to avert a possible collision. The Lazy method generates a path for the robot that passes in front of the obstacle with very little clearance. This near miss is highlighted in Figure 3b inside the yellow circle.

5.2 Fifty Moving Obstacles

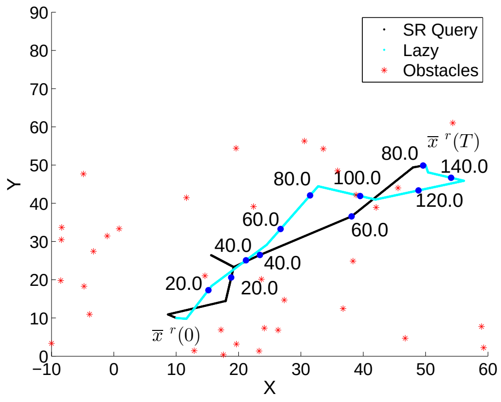
In this experiment, a robot navigates across a 60×60 planning space while avoiding 50 dynamic obstacles, O_i, $i \in \{1, \cdots, 50\}$. Twenty-five of the obstacles have straight-line dynamics (1), five each traveling along lines with $\alpha \in \{-1.5, -1, -0.5, +0.5, 1.0\}$, respectively. The other 25 dynamic obstacles have constant-arc dynamics (2), 10 each with radius $r = 50$, 10 with $r = 40$, and five with $r = 30$. The speeds and associated probabilities for each obstacle are as described in Section 4.1.

We constructed 10 each of the three roadmap sizes, as in Section 5.1. We again generated 100 obstacle trajectories, resulting in 1000 total simulations for each map size. Since obtaining a feasible path is more difficult with so many more obstacles, we increased the time horizon to $T = 100$ seconds.

Figure 4 shows the effect of map size as well as the average success rate of the two methods on PRM and Grid



(a) Fifty dynamic obstacles scenario in which Lazy method results in collision, SR Query method does not.



(b) Fifty dynamic obstacle scenario in which both methods successfully find collision-free paths.

Figure 5: Sample trajectories found by SR Query (black line) and Lazy (blue line) methods on a PRM roadmap with 50 obstacles shown at 80s into the simulation (red squares). Total simulation time $T = N\Delta = 100$s. (a) The Lazy method results in collision, but SR Query method successfully reaches the goal state without collision. (b) Sample trajectories (as in (a)), in which similar successful paths are found via both methods. *Movies of the 50 moving obstacle simulations are available at* `https://www.cs.unm.edu/amprg/Research/DO/`

roadmaps. As this is a significantly harder problem, the percentages of success are lower as compared to the two obstacle scenario in Figure 2. However, in all cases the SR Query method is at least 20% better than the Lazy method. Interestingly, the Grid-based solution is significantly more successful than the PRM-based method. This is likely due to the regular spacing of the roadmap nodes, which prevents long edges and allows the algorithm to make quicker replanning decisions. However, this advantage would not likely exist in more complex environments. As in Section 5.1, the error bars in Figure 4 indicate the significant impact of randomization in the PRM roadmap.

The 50 obstacle test in Figure 4 has lower success rates than the two obstacle test in Figure 2 because of two factors. First, finding a collision-free path is significantly harder with

50 obstacles as opposed to merely two. Second, the roadmap density, defined as the number of nodes per area of the planning space, is lower with 50 obstacles than with two obstacles. Since the roadmap sizes are the same, but the area increases from 20×20 to 60×60, the 50 obstacle tests have *lower* roadmap density. Lower density roadmaps force the robot to travel greater distances before path replanning (which occurs in Algorithm 1 at roadmap nodes), and consequently should have more collisions. However, relatively high success rates are evident for the SR Query methods, especially via Grid methods, likely due to the even distribution of nodes that allow for consistent replanning.

Figures 5a and 5b show two sample trajectories, one in which the SR Query method significantly outperforms the Lazy method, and another in which the two methods behave comparably. Movies of the full 50 obstacle simulation are available at
`https://www.cs.unm.edu/amprg/Research/DO/`.

6. CONCLUSIONS

We have successfully incorporated stochastic reachability into motion planning roadmaps, in order to develop a novel planning algorithm that accounts for stochastically moving obstacles. We combined SR sets for individual obstacles into a single planning solution, generating an upper bound on the total avoidance probability with several obstacles. We demonstrated our method on an example with 50 obstacles and on two methods of roadmap construction. To date, SR has never been applied to such large systems. By combining roadmaps with stochastic reachability, our algorithm significantly outperforms another existing roadmap-based method for moving obstacles. Future work includes exploring better ways of integrating multiple SR sets to generate tighter upper bounds on total collision avoidance probabilities when incorporated into the roadmap. We also plan to investigate higher dimensional systems with more complex obstacle movement, by exploring approximate SR calculations. We believe the incorporation of SR into roadmaps is a promising technique for motion planning under uncertainty, as demonstrated by our simulations.

7. ACKNOWLEDGMENTS

Malone is supported in part by the UNM Manufacturing Engineering Program supported by Sandia National Laboratories PO# 1110220. Lesser and Oishi are supported in part by National Science Foundation (NSF) Career Award CMMI-1254990 and NSF Award CPS-1329878. Tapia is supported in part by the National Institutes of Health (NIH) Grant P20RR018754 to the Center for Evolutionary and Theoretical Immunology.

8. REFERENCES

[1] Abate, A., Prandini, M., Lygeros, J., and Sastry, S. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica* (2008), 2724–2734.

[2] Al-Hmouz, R., Gulrez, T., and Al-Jumaily, A. Probabilistic road maps with obstacle avoidance in cluttered dynamic environment. In *Intelligent Sensors, Sensor Networks and Information Processing Conf.* (2004), IEEE, pp. 241–245.

[3] Bertsekas, D. P. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[4] Bohlin, R., and Kavraki, L. E. Path planning using Lazy PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2000), pp. 521–528.

[5] Brock, O., and Khatib, O. Elastic strips: Real-time path modification for mobile manipulation. *International Symposium of Robotics Research* (1997), 5–13.

[6] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., and Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[7] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik* (1959), 269–271.

[8] Ding, J., Li, E., Huang, H., and Tomlin, C. Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2011), pp. 2160–2165.

[9] Gillula, J. H., Hoffmann, G. M., Haomiao, H., Vitus, M. P., and Tomlin, C. J. Applications of hybrid reachability analysis to robotic aerial vehicles. *The Int. Journal of Robotics Research* (2011), 335–354.

[10] Hsu, D., Kindel, R., Latombe, J. C., and Rock, S. Randomized kinodynamic motion planning with moving obstacles. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)* (2000), pp. SA1–SA18.

[11] Hsu, D., Latombe, J.-C., and Kurniawati, H. Foundations of probabilistic roadmap planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)* (2006).

[12] Jaillet, L., and Simeon, T. A PRM-based motion planner for dynamically changing environments. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)* (2004).

[13] Kamgarpour, M., Ding, J., Summers, S., Abate, A., Lygeros, J., and Tomlin, C. Discrete time stochastic hybrid dynamical games: Verification and controller synthesis. In *Conf. on Decision and Control* (2011), pp. 6122–6127.

[14] Kavraki, L. E., Švestka, P., Latombe, J. C., and Overmars, M. H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.* (1996), 566–580.

[15] Lindemann, S. R., and LaValle, S. M. Current issues in sampling-based motion planning. In *Robotics Research*. Springer, 2005, pp. 36–54.

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

[17] Majumdar, A., and Tedrake, R. Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics*. Springer, 2013, pp. 543–558.

[18] Malone, N., Manavi, K., Wood, J., and Tapia, L. Construction and use of roadmaps that incorporate workspace modeling errors. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)* (to appear, 2013).

[19] Margellos, K., and Lygeros, J. Hamilton-Jacobi formulation for reach-avoid problems with an application to air traffic management. *American Control Conf.* (2010), 3045–3050.

[20] Missiuro, P. E., and Roy, N. Adapting probabilistic roadmaps to handle uncertain maps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2006), IEEE, pp. 1261–1267.

[21] Mitchell, I., Bayen, A., and Tomlin, C. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE transactions on automatic control* (2005), 947–957.

[22] Nieuwenhuisen, D., van den Berg, J., and Overmars, M. Efficient path planning in changing environments. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)* (2007), IEEE, pp. 3295–3301.

[23] Quinlan, S., and Khatib, O. Elastic bands: Connecting path planning and control. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (1993), pp. 802–807.

[24] Rodriguez, S., Lien, J.-M., and Amato, N. M. A framework for planning motion in environments with moving obstacles. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)* (2007), pp. 3309–3314.

[25] Shiller, Z., Large, F., and Sekhavat, S. Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2001), vol. 4, pp. 3716–3721.

[26] Simeon, T., Laumond, J.-P., and Nissoux, C. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics* (2000), 477–493.

[27] Song, G., Miller, S. L., and Amato, N. M. Customizing PRM roadmaps at query time. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2001), pp. 1500–1505.

[28] Summers, S., Kamgarpour, M., Lygeros, J., and Tomlin, C. A stochastic reach-avoid problem with random obstacles. In *Proc. Int. Conf. Hybrid Systems: Computation and Control (HSCC)* (2011), pp. 251–260.

[29] Takei, R., Huang, H., Ding, J., and Tomlin, C. Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2012), pp. 323–329.

[30] Van Den Berg, J., Ferguson, D., and Kuffner, J. Anytime path planning and replanning in dynamic environments. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2006), IEEE, pp. 2366–2371.

[31] van den Berg, J. P., Nieuwenhuisen, D., Jaillet, L., and Overmars, M. H. Creating robust roadmaps for motion planning in changing environments. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)* (2005), IEEE, pp. 1053–1059.

[32] Van Den Berg, J. P., and Overmars, M. H. Roadmap-based motion planning in dynamic environments. *IEEE Transactions on Robotics* (2005), 885–897.

[33] Wu, Y. An obstacle-based probabilistic roadmap method for path planning. Master's thesis, Department of Computer Science, Texas A&M University, 1996.

[34] Yoshida, E., and Kanehiro, F. Reactive robot motion using path replanning and deformation. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)* (2011), IEEE, pp. 5456–5462.