# Aggressive Moving Obstacle Avoidance Using a Stochastic Reachable Set Based Potential Field

Hao-Tien Chiang[1], Nick Malone[2], Kendra Lesser[3], Meeko Oishi[3], Lydia Tapia[2]

[1] Physics & Astronomy, University of New Mexico, Albuquerque, NM 87131, USA
[2] Computer Science, University of New Mexico, Albuquerque, NM 87131, USA
[3] Electrical & Comp. Eng., University of New Mexico, Albuquerque, NM 87131, USA

**Abstract.** Identifying collision-free trajectories in environments with dynamic obstacles is a significant challenge. However, many pertinent problems occur in dynamic environments, e.g., flight coordination, satellite navigation, autonomous driving, and household robotics. Stochastic reachable (SR) sets assure collision-free trajectories with a certain likelihood in dynamic environments, but are infeasible for multiple moving obstacles as the computation scales exponentially in the number of Degrees of Freedom (DoF) of the relative robot-obstacle state space. Other methods, such as artificial potential fields (APF), roadmap-based methods, and tree-based techniques can scale well with the number of obstacles. However, these methods usually have low success rates in environments with a large number of obstacles. In this paper, we propose a method to integrate formal SR sets with ad-hoc APFs for multiple moving obstacles. The success rate of this method is 30% higher than two related methods for moving obstacle avoidance, a roadmap-based technique that uses a SR bias and an APF technique without a SR bias, reaching over 86% success in an enclosed space with 100 moving obstacles that ricochet off the walls.

## 1 Introduction

Motion planning consists of finding a collision-free path from some start position to some goal position. In many applications, e.g., flight coordination, satellite navigation, and automated driving, the motion planning problem can be further complicated by moving obstacles, i.e. obstacles whose position changes over time during the planning process. Successful identification of valid, collision-free paths in environments with moving obstacles requires modification of the static planning problem to continuously re-evaluate plans, thus dynamically identifying valid trajectories given current and predicted obstacle positions.

Common approaches to solving the motion planning problem for dynamic obstacles include APF methods [1–4], tree based planners [5, 6], Probabilistic Roadmap Methods (PRMs) [7–10], and several variants which use heuristics [11,

12]. APF methods create a potential landscape and use gradient descent for navigation, plan locally, and can be dynamically reactive to unexpected obstacles. These methods generate an artificial potential in the robot's workspace, which repels the robot from obstacles and attracts the robot to the goal [13]. APF methods suffer from several well known drawbacks, most notably local minima traps and difficulty with narrow passages. However, recent work has improved upon and even eliminated some of these issues [1, 14]. The work in [4] has extended APFs to moving obstacles by considering the trajectories of the obstacles while computing the APF in a heuristic manner. In this paper, we present a method to generate potential fields that incorporates formal methods.

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 [15]. To solve problems in collision avoidance, the region in the relative state-space which constitutes collision is defined as the set of states the system should avoid [16, 17]. Unfortunately, the computation time for stochastic reachable sets (SR sets) is exponential in the dimension of the continuous state, hence assessment of collision probabilities with many simultaneously moving obstacles is not feasible. However, expensive SR sets can be computed offline and the result queried online. In prior work [7], we integrated SR sets into roadmap methods for dynamic path queries (SR-Query). We demonstrated highly successful path identification in environments with several moving obstacles, as compared to a roadmap-based approach that simply pruned invalid edges during dynamic path queries [10]. However, SR-Query was susceptible to ambushes by moving obstacles, due to limited reactivity and required navigation on the roadmap edges.

The method we propose here uses multiple SR sets, computed pairwise between the robot and each dynamic obstacle, to generate an APF for each obstacle. We then use the likelihood of collision with a given obstacle, computed a priori via the SR sets, to construct the repulsion field around obstacles. The repulsion fields are pre-computed offline and queried during the path planning phase. SR sets provide an accurate depiction of the collision probabilities between a robot and a moving obstacle. 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. Despite this limitation, the SR sets provide a more formal foundation for relating the collision probability to the repulsion field than other ad-hoc methods [4, 18, 19] because SR sets are computed based on relative robot-obstacle dynamics. While it is possible to use ad-hoc methods to generate a comparable repulsive potential field, the SR computation is a formal tool that more closely ties the repulsive potential field with the relative motion of the obstacles and robot.

Combining formal and ad-hoc methods provides several advantages over existing APF methods. First, the formal SR set provides an accurate representation of the collision probabilities, which is used to produce potential gradients which accurately reflect the collision probability. Second, the computation cost in low

dimensionality problems is lower than the roadmap method in [7]. Thus, the robot is more reactive and less prone to being ambushed by fast moving obstacles. Finally, our approach easily accommodates multiple obstacle scenarios, by combining multiple SR sets to generate approximate collision avoidance probabilities with many moving obstacles (which is impossible to compute through a single SR set that accounts for all obstacles simultaneously).

We demonstrate our method computationally on scenarios with up to three hundred stochastic dynamic obstacles. The APF with a SR bias can significantly improve the ability of the potential field landscape to reflect the heading and motion of obstacles. The success rate of our method is 30% higher than two related methods for moving obstacle avoidance: 1) our roadmap-based technique that uses a SR bias [7], and 2) an APF technique without a SR bias; with over 86% path success in an environment with 300 moving obstacles that ricochet off the walls. In addition, the common problem of local minima in APF is mitigated by a rapidly changing APF landscape produced by rapidly moving obstacles. Videos of the APF with a SR bias method can be viewed at
https://www.cs.unm.edu/amprg/Research/DO/.

While our results demonstrate that the APF-SR method outperforms comparable methods, we note two key limitations. First, the point-mass robot model is a simplification of actual robot motion. However, methods such as [1] and [14] exist, which extend APF methods to non-point robots. A more realistic robot model can be easily incorporated into the SR set calculation, but with additional computational cost. Second, we note that the SR set must be recalculated. One solution is to maintain a SR set database and to then match obstacle motion to sets as [20] does with funnel libraries. Neither of these limitations are insurmountable, and we maintain that the improved performance of the APF-SR method as compared to other approaches merits its use in many scenarios.

## 2 Related Work

APFs are a common approach to solving the path planning problem due to their simplicity, fast execution time, and applicability to several robotic problems, including unmanned aerial vehicles [2, 3], robot soccer [21], and mobile robots [1, 14, 22, 23]. For example, a recent APF method assigns non uniform repulsive bubbles around moving human obstacles to prevent robots from moving in front of a walking human [4]. Recent work has extended the APF method to account for cases in which the goal is not reachable due to obstacle proximity [1], and navigation in narrow passages is required [14]. Other recent work has focused on modification of the computation of the potential field through Fuzzy [23] and evolutionary [22] APFs. Another branch of work on APFs utilizes the repulsive and attractive concepts of APFs but also integrates another path planning method [24, 25]. For example, [24] uses a user defined costmap to influence node placement in a Rapidly exploring Random Tree (RRT) algorithm. The costmap dictates a repulsiveness or attractiveness factor for every region. Similarly, Navigation Fields [25] assign a gradient which agents follow and is used for crowd modeling.

A Hamilton-Jacobi-Bellman (HJB) formulation [26] allows for both a control input and a disturbance input to model collision-avoidance scenarios [27], [28] 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 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 [30], a similar approach is taken, but reachable sets are computed iteratively so that the robot can modify its actions. In [20], multiple obstacles that act as bounded, worst-case disturbances are avoided online, based on precomputed invariant sets.

An alternative approach is to calculate a SR set that allows for obstacles whose dynamics include stochastic processes. Discrete-time SR generates probabilistic reachable sets [15], based on stochastic system dynamics. In [16], the desired target set is known, but the undesired sets that the robot should avoid are random and must be propagated over time. In [17], a two-player stochastic dynamical game is applied to a target tracking application in which the target acts in opposition to the tracker.

## 3 Preliminaries

### 3.1 Obstacle Dynamics

We consider dynamic obstacles with one of two classes of trajectories with stochastic velocities. Each obstacle is represented as a two-dimensional point mass with state $\overline{x}^o = (x^o, y^o)$, that follows a straight-line or approximately constant-arc trajectory with stochastic velocity $w$, a discrete random variable that takes on values in $\mathcal{W}$ with probability distribution $p(w)$. However, more complex dynamics, e.g., ones that switch between straight-line and constant arc movements, can easily be incorporated. The obstacle dynamics discretized via an Euler approximation with time step $\Delta$ are

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

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

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

for constant-arc movement, with angular speed $w \in \mathcal{W}$, and radius $r \in \mathbb{R}^+$. The dynamics (2) approximate actual arc dynamics to maintain low dimensionality of the relative coordinate frame used in the calculation of the SR set.

The dynamics of both types of obstacle can be generalized to the form $\overline{x}^o_{n+1} = \overline{x}^o_n + \Delta f^o(w_n, n)$ with $f$ defined as appropriate by (1) or (2).

### 3.2 Relative robot-obstacle dynamics

We consider two models for the robot: 1) a holonomic point-mass model with state $\overline{x}^r = (x^r, y^r)$, and 2) a non-holonomic unicycle model with state $\overline{x}^r = (x^r, y^r, \theta^r)$. The holonomic model is defined as

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

with two-dimensional velocity control input $u = (u^x, u^y)$. The non-holonomic unicycle model is defined as

$$\begin{aligned} \dot{x}^r &= u^s \cos(\theta) \\ \dot{y}^r &= u^s \sin(\theta) \\ \dot{\theta}^r &= u^w \end{aligned} \tag{4}$$

with two-dimensional control input $u = (u^s, u^w)$, such that $u^s$ is the speed and $u^w$ is the angular velocity of the unicycle. Discretizing the robot dynamics (3) and (4) with time step $\Delta$ results in

$$\overline{x}^r_{n+1} = \overline{x}^r_n + \Delta u. \tag{5}$$

for the holonomic model and

$$\begin{aligned} x^r_{n+1} &= x^r_n + \Delta u^s_n \cos(\theta^r_n) \\ y^r_{n+1} &= y^r_n + \Delta u^s_n \sin(\theta^r_n) \\ \theta^r_{n+1} &= \theta^r_n + \Delta u^w_n \end{aligned} \tag{6}$$

for the unicycle model. We can generalize the robot dynamics to $\overline{x}^r_{n+1} = \overline{x}^r_n + \Delta f^r(u_n, \theta_n)$ where $\theta_n = 0$ for the holonomic case.

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

$$\tilde{x}_{n+1} = \tilde{x}_n + \Delta f^r(u_n, \theta_n) - \Delta f^o(w_n, n) \tag{7}$$

with $f^r(\cdot)$ as in (5) and (6), $f^o(\cdot)$ as in (1) and (2), and a *collision* is defined as

$$|\tilde{x}_n| \leq \epsilon. \tag{8}$$

Equation (7) describes 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 $\tilde{x}_n$, $u_n$ at time step $n$.

### 3.3 SR for Collision Avoidance

We generate collision avoidance probabilities by formulating a SR problem with the avoid set, $\overline{K}$, defined as the set of states in which a collision is said to occur (8). To avoid collision with the obstacle, the robot should remain within $K$, the complement of $\overline{K}$. The probability that the robot remains within $K$ over $N$ time steps, with initial relative position $\tilde{x}_0$, can be calculated using dynamic programming [31], introduced for the stochastic reachability problem in [15]. An abbreviated derivation for calculating the SR set follows, with details in [7]. As in [15], the SR set is generated via dynamic programming, iterated backwards in time from time $n = N$ to time $n = 0$.

$$V_N(\tilde{x}) = \mathbf{1}_K(\tilde{x}) \tag{9}$$

$$V_n(\tilde{x}) = \mathbf{1}_K(\tilde{x}) \int_{\mathcal{X}} V_{n+1}(\tilde{x}')\tau(\tilde{x}' \mid \tilde{x}, u, n)\, d\tilde{x}' \tag{10}$$

$$= \mathbf{1}_K(\tilde{x}) \sum_{w \in \mathcal{W}} V^*_{n+1}\left(\tilde{x} + \Delta f^r(u, \theta) - \Delta f^o(w, n)\right) p(w). \tag{11}$$

The value functions (9)-(11) make use of an indicator function $\mathbf{1}_K(x)$ that is equal to 1 if $x \in K$ and equal to 0 otherwise. The value function $V^*_0(\tilde{x}_0)$ at time $n = 0$ describes the probability of avoiding collision over $N$ timesteps when starting in some initial state $\tilde{x}_0$. The optimal control input $u$ to avoid collision is determined by evaluating

$$V^*_n(\tilde{x}) = \max_{u \in \mathcal{U}} \left\{ \mathbf{1}_K(\tilde{x}) \sum_{w \in \mathcal{W}} V^*_{n+1}\left(\tilde{x} + \Delta f^r(u, n) - \Delta f^o(w, n)\right) p(w) \right\}. \tag{12}$$

Figure 1a shows the SR set for a straight-line obstacle with a point mass holonomic robot. The peaks show a higher probability of collision when the robot is in line with the obstacle's 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, the SR set in Figure 1a took 1727.25 seconds to compute, over a horizon of $N = 30$ steps, with time step of length $\Delta = 1$ and a point mass holonomic robot. 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 (12) is the maximum probability of avoiding a collision, and a tight upper bound. For two obstacles with separately calculated avoidance probabilities $V^{*,1}_0(\tilde{x}^1_0)$, $V^{*,2}_0(\tilde{x}^2_0)$ (with relative position $\tilde{x}^i_0$ with respect to obstacle $i$), the probability of avoiding collision with *both* obstacles is

$$\mathbb{P}[B_1 \cap B_2] \leq \min\{V^{*,1}_0(\tilde{x}^1_0), V^{*,2}_0(\tilde{x}^2_0)\} \tag{13}$$

where $B_i$ corresponds to the event that the robot avoids collision with obstacle $i$. We therefore examine each collision avoidance probability individually, and the minimum over all obstacle robot pairs is the upper bound to the total collision

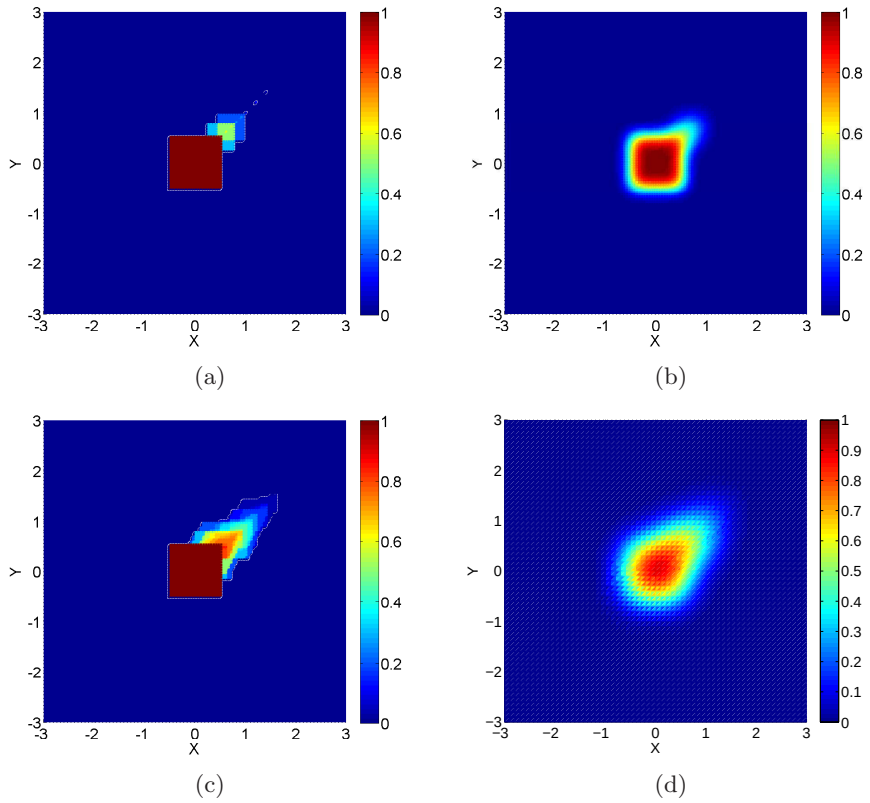



Fig. 1: SR sets for the same straight-line obstacles at origin with width and height = 1. The color represents probability of collision. (a) SR set with a holonomic robot. (b) Holonomic robot SR set after convolution with a Gaussian ($\sigma = 0.15$). (c) SR set with the unicycle robot. (d) Unicycle robot SR set after convolution with a Gaussian ($\sigma = 0.15$).

avoidance probability. While an upper bound provides no guarantee of safety, it can inform which paths are more likely, relative to other paths, to avoid collision. Since our focus is on finding paths with higher success rates, rather than theoretically guaranteed collision-free paths, the upper bound (13) is appropriate. Further discussion and the derivation of (13) is in [7].

## 4 Methods

In this section, we present a novel method for integrating SR sets with APF methods. To generate the obstacle gradients and the gradient to the goal with SR sets we must first modify the SR sets to accommodate APF, incorporate the SR sets into the gradient calculation, and then update the robot's control law.

**Algorithm 1** APF-SR

**Input:** obstacles $O$ with precomputed smoothed SR sets, robot $r$

1: **for** $t = 0$; $t < maxTime$; $t = t + \Delta$ **do**
2: **for** Obstacle $o \in O$ **do**
3: updateObstacle($t$,$o$,$o.w$,$o.p(w)$)
4: **end for**
5: $APF_{vector} = (0, 0)$
6: **for** Obstacle $o \in O$ **do**
7: **if** dist( $\overline{x}_n^o$, $\overline{x}_n^r$) $< d_{min}$ **then**
8: $APF_{vector} = APF_{vector} + o.getAPFGradient(\overline{x}_n^r)$
9: **end if**
10: **end for**
11: $APF_{vector} = APF_{vector}$+ goal-gradient
12: $u = calcControl(APF_{vector})$
13: $\overline{x}_{n+1}^r = \overline{x}_n^r + \Delta \cdot f^r(u, t)$
14: **end for**

One hurdle in using SR sets to inform the potential field is the possibility of non-smoothness in the optimal value for (12). In general, no guarantees of smoothness are possible. In fact, we find a marked discontinuity in the part of the SR set corresponding to a robot located just behind the obstacle (Figure 1). Since APF methods use a gradient as a warning that the robot is about to collide with an obstacle, we smooth the SR set by convolving the set with a Gaussian with $\mathcal{N}(\mu = 0, \sigma^2)$. Figure 1 shows the original SR set (1a) and the resulting set after convolution (1b). As expected, the discontinuity in Figure 1a from 0 to 1 at the obstacle boundary is smoothed in Figure 1b.

The main APF-SR algorithm, Algorithm 1, first updates the obstacle positions via the *updateObstacle* function (Line 3). Then Algorithm 1 calculates the APF gradient by summing the obstacle gradients, calculated in *getAPFGradient*, and the goal gradient (Lines 5-11), which is then used by *calcControl* to construct the control input $u$ (Line 12). Recall the APF gradient is the direction the robot should move in to avoid obstacles and reach the goal. Finally, the control law for the robot is updated with the control input $u$ (Line 13).

The *updateObstacle* function (Algorithm 2) uses the same dynamics used to calculate the SR sets. This algorithm updates the obstacle locations. At every sampling instant (time $T$ apart), Algorithm 2 evaluates a speed $w$ of the obstacle, based on the distribution $p(w)$ of possible speeds (Lines 2-9), and updates the obstacle dynamics with this speed (Line 10).

The APF gradient is calculated for all obstacles nearby the robot in the $getAPFGradient(\overline{x}_n^r)$ function. For every obstacle $o$, if $o$ is within distance $d_{min}$ query the potential field influence of $o$ on the robot. This gradient is calculated by first finding the smallest neighboring value, $p_{i,j}$, in the SR set from the robot's current relative position. The gradient is then calculated by the $2^{nd}$ order central finite difference centered at $p_{i,j}$. The gradient from each obstacle is then summed

**Algorithm 2** updateObstacle

**Input:** Time step $n$, sample interval $T$. obstacle $o$, velocities $w \in \mathcal{W} = \{w_1, w_2, ..., w_{n_W}\}$, probabilities $p(w)$

---

1: **if** $mod(n, T/\Delta) == 0$ **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: $\overline{x}^o_{n+1} = \overline{x}^o_n + \Delta \cdot f^o(o.w, t_n)$

together to produce a final gradient due to the obstacles. The *goal-gradient* is a small magnitude vector that constantly points toward the goal. The *goal-gradient* and the gradient due to the obstacles are summed together to get the final APF gradient, denoted $APF_{vector}$.

After the $APF_{vector}$ is calculated, the control input $u$ is calculated by the *calcControl($APF_{vector}$)* function. For the holonomic case $u = APF_{vector}$. However, for the non-holonomic case a heading and speed must be extracted from the $APF_{vector}$ to construct $u = (u^s, u^w)$. This is done by first setting $u^w$ to the maximum turn rate in the direction of the $APF_{vector}$, then setting $u^s$ to the maximum speed in the direction of the $APF_{vector}$. The maximum speed of the unicycle is the same as the maximum speed used in the SR calculation. Finally, $u$ is used to update the control law for the robot.

## 5 Experiments

We present three experiments of increasing difficulty. The first experiment (Section 5.1), evaluates the APF-SR method on 50 moving obstacles, with two different trajectories (straight-line and constant-arc) and a holonomic point robot. The second experiment (Section 5.2), shows the relationship between the number of obstacles, 50 to 300, and success rate for the proposed method, with a holonomic robot and ricocheting straight-line obstacles. When the ricocheting obstacles reach the environment boundary, they bounce off the wall with simple friction free reflective behavior (and do not leave the planning area). Finally, Section 5.3 evaluates the APF-SR method with a non-holonomic unicycle robot with 100 ricocheting straight-line obstacles. Note that since the SR calculation is computed once for each type of obstacle and robot dynamics, the offline computation time is not affected by the number of obstacles.

Our APF-SR method is compared to three methods: a simple Gaussian method with $\mathcal{N}(0, 0.15^2)$ [32], the same Gaussian method with $\mathcal{N}(0, 0.45^2)$, and

a roadmap based method (SR-Query) which also uses SR sets [7]. The Gaussian methods wrap a Gaussian potential field around the moving obstacle. The two Gaussian methods demonstrate that increasing the standard deviation can increase the success of the Gaussian method, but at the expense of making some paths infeasible due to the large repulsion area. The final method, SR-Query, builds a roadmap in the workspace by sampling valid configurations (nodes) and connecting these nodes with valid transitions (edges) thus constructing a graph. The SR-Query method updates the edge weights by querying the SR set of each moving obstacle which overlap with the roadmap. The edge is then assigned the worst probability of collision and a graph search algorithm is used to find the path with the lowest probability of collision. The robot travels along the edges and can only replan when it reaches a node. For the comparisons shown, the SR-Query uses a roadmap created by a uniform cell decomposition in the workspace, with 500 nodes and edges between all 8 cell neighbors.

For the APF-SR experiments, the SR set was convolved with a Gaussian with $\sigma = 0.15$. The $\sigma$ of the smoothing Gaussian has the same value as the smaller Gaussian comparison method (Gaussian $\sigma = 0.15$) to eliminate the smoothing done to the SR set as possible bias for APF-SR's success. The value $\sigma = 0.15$ worked well since larger values destroyed the shape of the SR set and smaller values did not provide enough smoothing. The value was chosen empirically by comparing $\sigma = 0.05$, $\sigma = 0.45$ and $\sigma = 0.15$.

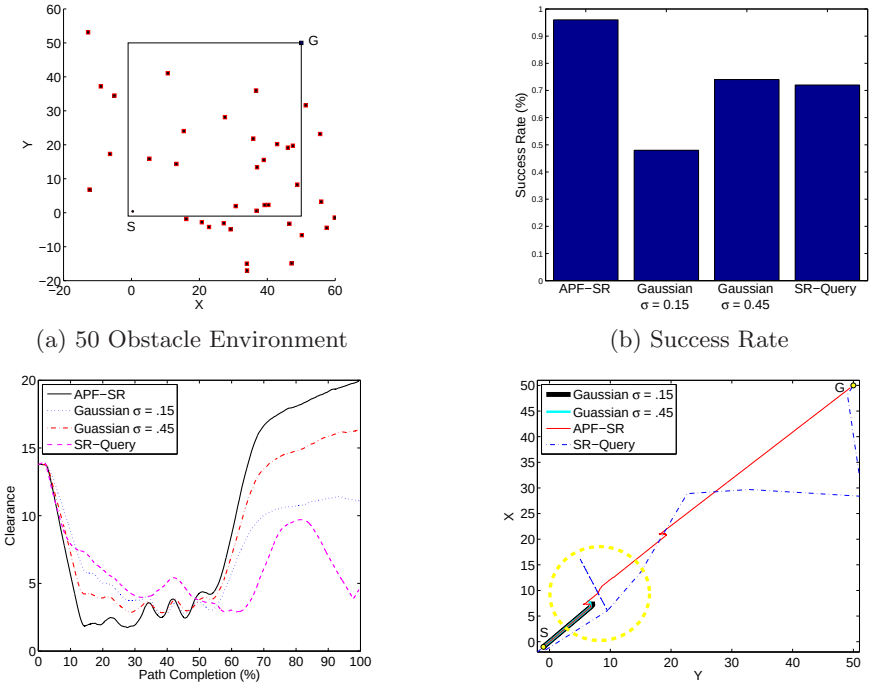
To generate the SR sets, the obstacles must have a known probabilistic velocity distribution. For all the experiments, the straight-line obstacles have stochastic velocities, $w = \{0.1, 0.2, 0.5, 0.7\}$, with corresponding probabilities $p(w) = \{0.3, 0.2, 0.3, 0.3\}$. Experiments with obstacles traveling along constant-arc trajectories have $w = \{\frac{.4}{20\pi}, \frac{.6}{20\pi}, \frac{.9}{20\pi}, \frac{1.2}{20\pi}\}$ and $p(w) = \{0.2, 0.2, 0.3, 0.3\}$ with radii 30, 40 and 50.

In Sections 5.1 and 5.2, the robot is holonomic with a maximum velocity of 0.36 units per second. In Section 5.3, the robot is a unicycle with a maximum velocity of 0.36 meters per second and maximum turn rate of $\frac{\pi}{5}$ radians per second. The other critical parameters are $d_{min} = 3m$, the *goal-gradient* is a vector with magnitude 0.1 in the direction of the goal, the robot makes a decision and moves every $\Delta = 0.01$ second, and the obstacle sampling interval is $T = 1$ second.

### 5.1 Comparison of holonomic robot with line and arc obstacles

The environmental setup is constant between all three methods. However, because the obstacles have stochastic velocity, multiple trials (100) are conducted and mean results presented. Each method is run with the same random seed. In these experiments, there are 25 constant-line obstacles and 25 constant-arc obstacles with stochastic velocities. Figure 2a shows the initial locations of the obstacles, as well as the start location (S) and goal location (G) of the robot.

Figure 2b shows the percentage of trials which reach the goal without collision. The APF-SR method has the highest success rate (95%); much higher than the next highest success rate (75%) via the Gaussian method with $\sigma = .45$.




(a) 50 Obstacle Environment

(b) Success Rate

(c) Clearance

(d) Example Paths

Fig. 2: 50 Obstacle Comparison: (a) The environment at $t = 0$. The obstacles start outside the environment boundaries and move towards the robot. (b) Percentage of trials which reach the goal without collision. (c) Distance from the nearest obstacle over the coarse of the trial. (d) Example of the paths for a single stochastic trial. The start is marked with a S and the goal with a G. Gaussian methods do not reach to goal due to a collision.

Hence, incorporating the formal SR set methods into the ad-hoc APF method provides a significant advantage. This advantage originates from the fact that the APF-SR method provides information about an obstacle's dynamics, enabling the robot to avoid an obstacle's path while maneuvering around all obstacles.

Of further interest is that the SR-Query method only achieves a 74% success rate in this experiment. This is because the robot using the SR-Query method is ambushed by obstacles while traversing an edge in the roadmap [7]. Recall that the SR-Query method only makes path planning decisions at nodes and is therefore vulnerable while traversing edges. However, the proposed APF-SR method does not suffer from this particular problem, making the proposed method more reactive to the moving obstacles. The APF-SR method makes path planning decisions at every timestep, while the SR-Query method only replans at nodes in the roadmap [7].

Figure 2c evaluates a second metric, clearance, which we define as the distance from the robot to the nearest obstacle averaged over all trials. The paths are normalized for comparison. The clearance of the APF-SR method is comparable to the clearance of the other methods. However, the shape of the potential field provides a more informed path through the obstacles. Figure 2d shows the difference in example paths for the four methods. The difference in the decisions can be seen in the yellow circle where the two Gaussian methods collide and stop. The Gaussian $\sigma = 0.15$ method makes a slight turn to avoid the obstacle and is then hit. While the Gaussian $\sigma = 0.45$ method makes a slightly more pronounced turn, it is still hit. However, the APF-SR method makes a much steeper turn due to the shape of the potential field, successfully avoids the moving obstacle, and reaches the goal.

### 5.2 Holonomic robot with ricocheting line obstacles

We compare the Gaussian method and the APF-SR method in challenging environments with 100 straight-line obstacles. Unlike in the previous experiment, the straight-line obstacles may ricochet off the walls defined in the 50 by 50 environment. This increases the difficulty of the problem as all obstacles are always present in the planning region. Figures 2a and 3a show the difference between the 50 obstacle and 100 obstacle experimental environments.

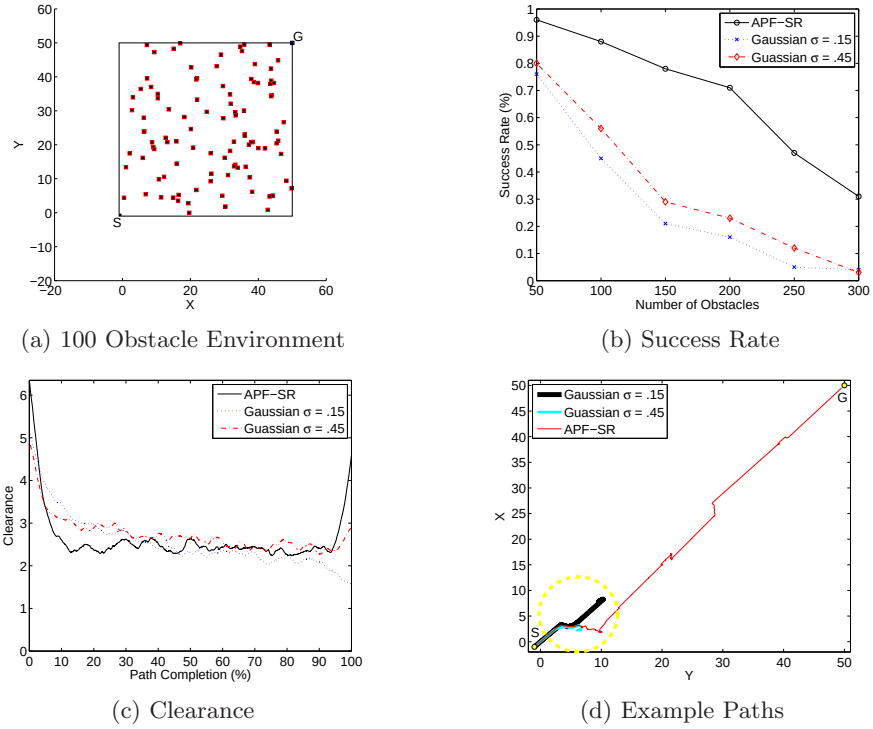
Figure 3b shows that the APF-SR method has a success rate of 86%, while the Gaussians have a success rate of at most 56%, for 100 obstacles. As expected, the success rate is lower than in the 50 obstacle experiment.

Since the clearances shown in Figure 3c for each of the three methods are comparable, the shape of the APF allows the robot to take more informed paths through the obstacles. Figure 3d shows the path differences, particularly evident inside the yellow circle, where the three methods follow very different paths. The APF-SR method takes the most evasive action and successfully avoids collision, whereas the other two methods fail.

As the number of obstacles increases from 50 to 300 (Figure 3b), the success rate decreases, as expected. However, the APF-SR method decreases at a slower rate and still has approximately 75% success rate with 200 obstacles, whereas the Gaussian methods have less than 25% success rate. By incorporating the SR sets, the APF-SR method can better avoid large numbers of obstacles. Further, the online execution of the APF-SR method is fast, scaling linearly with the number of obstacles. On a single core of an Intel 3.40 GHz CORE i7-2600 CPU with 8 GB of RAM, execution time is 0.0168ms per step for the 50 obstacle environment, and 0.0247ms per step for the 100 obstacle environment.

### 5.3 Non-holonomic Unicycle

In this experiment, the robot is modeled as a non-holonomic unicycle (4) and the obstacles follow straight-line trajectories. The robot can only turn at a rate of $\frac{\pi}{5}$ radians per second, which makes the problem more difficult than the holonomic




(a) 100 Obstacle Environment

(b) Success Rate

(c) Clearance

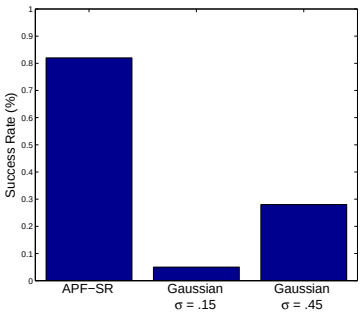
(d) Example Paths

Fig. 3: 50 - 300 Ricocheting Obstacles: (a) The 100 Obstacle environment at time $t = 0$. (b) Success rate with increasing number of obstacles. (c) Distance from the nearest obstacle normalized over the path (100 Obstacles). (d) Example of the paths for a single stochastic obstacle run. The start is marked with 'S' and the goal with 'G'. Gaussian-method paths do not reach to goal due to a collision.

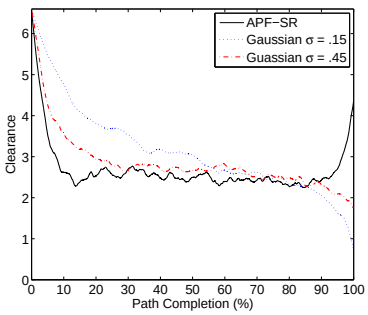
case. We also note that this problem cannot be solved with the SR-Query method presented in [7] without path modification for the non-holonomic constraints.

Figure 4a compares the APF-SR method and the Gaussian methods. The APF-SR method performs approximately 50% better than the next highest Gaussian method. Thus, the SR set allows the APF-SR method to make significantly better path planning decisions.

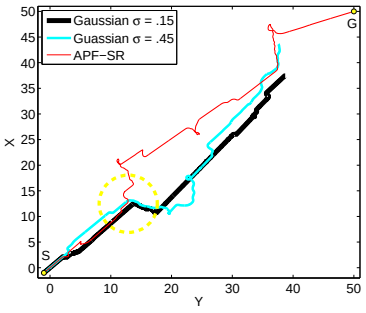
Figure 4b shows that clearance is comparable across all methods, indicating that the APF-SR method's repulsion fields produce more informed paths. Figure 4c shows an example of these paths for a single run. These paths differ more than the paths in the previous experiments. This is due to the limited ability of the robot to turn, and thus early differences in decisions result in large path differences later. For example, in the yellow circle in Figure 4c, the APF-SR method diverges from the Gaussian method early on due to the shape of the potential field constructed with the SR sets. These paths are more erratic than the holonomic robot's paths because the robot's turning ability is limited and




(a) Success Rate




(b) Clearance




(c) Example Paths

Fig. 4: 100 Obstacles with Unicycle Robot: (a) Percentage of trials which reach the goal without collision. (b) Distance from the nearest obstacle normalized over the path. (c) Paths Example. The start is marked with a S and the goal with a G. The paths for the Gaussian methods do not reach to goal due to a collision.

hence the robot must take more dramatic evasive motions to avoid the obstacles. The sharp direction changes are due to the unicycle changing velocity from positive to negative (or vice versa), which creates a sharp reversal.

## 6 Conclusion

The incorporation of the formal SR sets into the ad-hoc APF method provides the APF with a more accurate representation of the relative robot-obstacle dynamics, which leads to an increased success rate during path planning. The APF-SR method has a success rate at least 30% higher than other methods used for comparison. We also showed that this gain was due not to increased clearance from the obstacles, but rather to more informed path planning. The SR set informs the APF-SR algorithm of the direction and velocity of the obstacle, which is used to generate a repulsive potential that reflects the probability of collision. Hence the APF-SR algorithm can make informed planning decisions in the presence of multiple moving obstacles.

## References


1. Ge, S.S., Cui, Y.J.: New potential functions for mobile robot path planning. IEEE Trans. Robot. Automat. **16**(5) (2000) 615–620
2. Cetin, O., Kurnaz, S., Kaynak, O., Temeltas, H.: Potential field-based navigation task for autonomous flight control of unmanned aerial vehicles. Int. J. of Autom. and Cont. **5**(1) (2011) 1–21
3. Khuswendi, T., Hindersah, H., Adiprawita, W.: Uav path planning using potential field and modified receding horizon a* 3d algorithm. In: Int. Conf. on Electrical Eng. and Informatics (ICEEI). (2011) 1–6
4. Lam, C.P., Chou, C.T., Chiang, K.H., Fu, L.C.: Human-centered robot navigation towards a harmoniously human–robot coexisting environment. IEEE Trans. Robot. **27**(1) (2011) 99–112
5. Lee, H.C., Yaniss, T., Lee, B.H.: Grafting: a path replanning technique for rapidly-exploring random trees in dynamic environments. Advanced Robotics **26**(18) (2012) 2145–2168
6. Narayanan, V., Phillips, M., Likhachev, M.: Anytime safe interval path planning for dynamic environments. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). (2012) 4708–4715
7. Malone, N., Lesser, K., Oishi, M., Tapia, L.: Stochastic reachability based motion planning for multiple moving obstacle avoidance. In: Hybrid Systems: Computation and Control, HSCC (2014) 51–60
8. Van Den Berg, J., Ferguson, D., Kuffner, J.: Anytime path planning and replanning in dynamic environments. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2006) 2366–2371
9. Rodriguez, S., Lien, J.M., Amato, N.M.: A framework for planning motion in environments with moving obstacles. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). (2007)
10. Jaillet, L., Simeon, T.: A PRM-based motion planner for dynamically changing environments. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS). (2004)
11. Al-Hmouz, R., Gulrez, T., Al-Jumaily, A.: Probabilistic road maps with obstacle avoidance in cluttered dynamic environment. In: IEEE Intelligent Sensors, Sensor Networks and Information Processing Conf. (2004) 241–245
12. Bohlin, R., Kavraki, L.E.: Path planning using Lazy PRM. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2000) 521–528
13. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1) (1986) 90–98
14. Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J.: Path planning for autonomous vehicles in unknown semi-structured environments. Int. J. Robot. Res. **29**(5) (2010) 485–501
15. Abate, A., Prandini, M., Lygeros, J., Sastry, S.: Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. Automatica (2008) 2724–2734
16. Summers, S., Kamgarpour, M., Lygeros, J., Tomlin, C.: A stochastic reach-avoid problem with random obstacles. In: Proc. Int. Conf. Hybrid Sys.: Comp. and Cont. (HSCC). (2011) 251–260
17. Kamgarpour, M., Ding, J., Summers, S., Abate, A., Lygeros, J., Tomlin, C.: Discrete time stochastic hybrid dynamical games: Verification and controller synthesis. In: IEEE Conf. on Decision and Cont. (2011) 6122–6127

18. Valavanis, K.P., Hebert, T., Kolluru, R., Tsourveloudis, N.: Mobile robot navigation in 2-d dynamic environments using an electrostatic potential field. IEEE Trans. Sys., Man, Cybern. **30**(2) (2000) 187–196
19. Ge, S.S., Cui, Y.J.: Dynamic motion planning for mobile robots using potential field method. Auto. Robots **13**(3) (2002) 207–222
20. Majumdar, A., Tedrake, R.: Robust online motion planning with regions of finite time invariance. In: Algorithmic Foundations of Robotics. Springer (2013) 543–558
21. Weijun, S., Rui, M., Chongchong, Y.: A study on soccer robot path planning with fuzzy artificial potential field. In: Int. Conf. on Comput., Cont. and Industrial Eng. (CCIE). Volume 1. (June 2010) 386–390
22. Vadakkepat, P., Tan, K.C., Ming-Liang, W.: Evolutionary artificial potential fields and their application in real time robot path planning. In: IEEE Congress on Evolutionary Computation. Volume 1. (2000) 256–263
23. Song, Q., Liu, L.: Mobile robot path planning based on dynamic fuzzy artificial potential field method. Int. J. of Hybrid Info. Tech. **5**(4) (2012)
24. Jaillet, L., Cortés, J., Siméon, T.: Sampling-based path planning on configuration-space costmaps. IEEE Trans. Robot. **26**(4) (2010) 635–646
25. Patil, S., Van Den Berg, J., Curtis, S., Lin, M.C., Manocha, D.: Directing crowd simulations using navigation fields. Trans.on Visualization and Computer Graphics **17**(2) (2011) 244–254
26. Mitchell, I., Bayen, A., Tomlin, C.: A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. Trans. on Auto. Cont. (2005) 947–957
27. Margellos, K., Lygeros, J.: Hamilton-Jacobi formulation for reach-avoid problems with an application to air traffic management. Amer. Cont. Conf. (2010) 3045–3050
28. Gillula, J.H., Hoffmann, G.M., Haomiao, H., Vitus, M.P., Tomlin, C.J.: Applications of hybrid reachability analysis to robotic aerial vehicles. Int. J. Robot. Res. (2011) 335–354
29. Takei, R., Huang, H., Ding, J., 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) 323–329
30. Ding, J., Li, E., Huang, H., Tomlin, C.: Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA). (2011) 2160–2165
31. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Sci. (2005)
32. Massari, M., Giardini, G., Bernelli-Zazzera, F.: Autonomous navigation system for planetary exploration rover based on artificial potential fields. In: Dynamics and Cont. of Systems and Structures in Space (DCSSS). (2004) 153–162