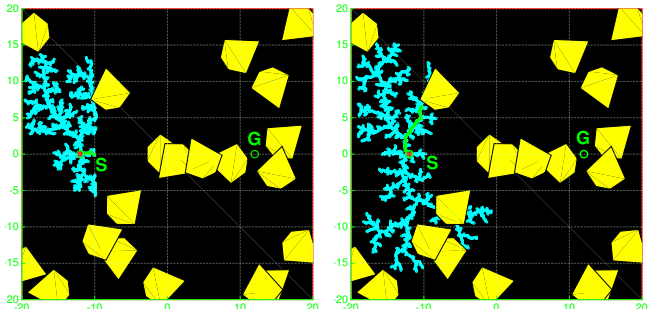
AMPRG Technical Report TR16-001

Dynamic Risk Tolerance: Motion Planning by Balancing Short-Term and Long-Term Stochastic Dynamic Predictions

Hao-Tien (Lewis) Chiang[1], Baisravan HomChaudhuri[2], Abraham P. Vinod[2], Meeko Oishi[2], Lydia Tapia[1]

***Abstract*— Identifying collision-free paths over long time windows in environments with stochastically moving obstacles is difficult, in part because long-term predictions of obstacle positions typically have low fidelity, and the region of possible obstacle occupancy is typically large. As a result, planning methods that are restricted to identifying paths with a low probability of collision may not be able to find a valid path. However, allowing paths with a higher probability of collision may limit detection of imminent collisions. In this paper, we present *Dynamic Risk Tolerance (DRT)*, a framework that dynamically evaluates risk tolerance, a function which is formulated as a time-varying upper bound on the acceptable likelihood of collision for a given path. DRT is implemented with forward stochastic reachable sets to predict the exact distribution of obstacles in a scalable manner over an arbitrarily long time window. In effect, DRT identifies actions that balance risks posed by both near and far obstacles. We empirically compare DRT to other state of the art methods that are capable of generating real-time solutions in highly crowded environments, and demonstrate the success rates for DRT that is 46% higher than the best performing comparison method, even in the most difficult problem tested.**



(a) (b)

Fig. 1: The robot (red asterisk) must navigate from S to G without colliding with 20 stochastically moving dynamic obstacles (yellow). (a) RRT (blue) employed by SES [8], a method with no dynamic risk tolerance, with 100,000 iterations. The tree accepts a node only if the probability of collision is less than 0.01. (b) Our method, DRT (blue), a Multi-Stage Risk Tolerating tree, with 25,000 total RRT iterations. The tree in (b) explores more area and can therefore extract longer paths (green curve) by considering risk tolerance that varies over the path.

I. Introduction

Navigation in crowded environments with dynamic, stochastic obstacles is a challenging problem in a variety of domains including robotics (swarms, micro-robotics, UAVs) and transportation systems (air traffic control, intelligent highways, autonomous vehicles in pedestrian-heavy environments). Obvious challenges include the density of obstacles, their dynamic movement, and the inherent uncertainty associated with any prediction of obstacle position. Over long time windows, the variance of the likelihood of obstacle position typically grows so large that vast regions have non-trivial likelihood, and the distribution can become uniform. Such uncertainty impairs existing planning methods, which rely on a static upper bound for *risk tolerance*, the acceptable likelihood of collision over a given path. Hence real-time solutions are required that can manage both the need for immediate, evasive actions as well as the need for long-term planning.

The notion of risk tolerance was originally posed in stochastic optimal control problems to navigate in uncertain but static environments [16]. However, in dynamic environments, a constant-valued risk tolerance is detrimental: a low risk tolerance means that while imminent collisions will be avoided, a path that is deemed feasible over the entire time window is extremely unlikely in crowded environments; conversely, a high risk tolerance means that while feasible paths will be returned, immediate evasive maneuvers may not be taken. We propose a solution for uncertain, dynamic environments that is based on dynamically adjusting the risk tolerance. Intuitively, risk tolerance should be low in the short-term to avoid imminent collisions, and high in the long-term to generate feasible paths. *Our proposed solution dynamically adjusts risk tolerance to meet both short-term and long-term objectives.*

We present *Dynamic Risk Tolerance (DRT)*, a framework that dynamically evaluates risk tolerance, a function which is formulated as a time-varying upper bound on the acceptable likelihood of collision for a given path. DRT is demonstrated with offline-computed forward stochastic reachable sets to predict the exact distribution of obstacles in a scalable manner over an arbitrarily long time window. In effect, DRT is capable of identifying a path in real-time that balance risks posed by both near and far obstacles. We empirically compare DRT to other state of the art methods that are capable of generating real-time solutions in highly crowded environments, and demonstrate a success rate for DRT that is 46% higher than the best performing state of the art comparison methods even in the most difficult problem tested. The enclosed video demonstrates the comparison between methods.

[1]Computer Science, University of New Mexico, MSC01 1130, 1 University of New Mexico, Albuquerque, NM 87131, USA

[2] Electrical & Computer Engineering, University of New Mexico, MSC01 1100, 1 University of New Mexico, Albuquerque, NM 87131, USA

II. Related Work

Many collision avoidance problems in aerospace and automotive applications [27], [23], [11], [15], [5] have been handled through backward reachable sets and exploit the relatively low dimensionality of the state-space. These approaches consider an optimal control framework, in which the worst-case realization of the obstacle (or disturbance) is presumed, and provide strict assurances of safety. Other approaches use forward reachable sets (whose formal relationship to backwards reachable sets is described in [28]) for obstacle avoidance in robotics [31], [10], [2], [22], automotive applications [2], and UAVs [22]. These methods are often based on an approach used by the robust control community [26], [3], [21] to evaluate the maximum set of states that the system can achieve in the presence of bounded disturbances. The resulting controllers often create very conservative solutions, particularly when the disturbance variance is large, and can lead to infeasible solutions for longer prediction horizons.

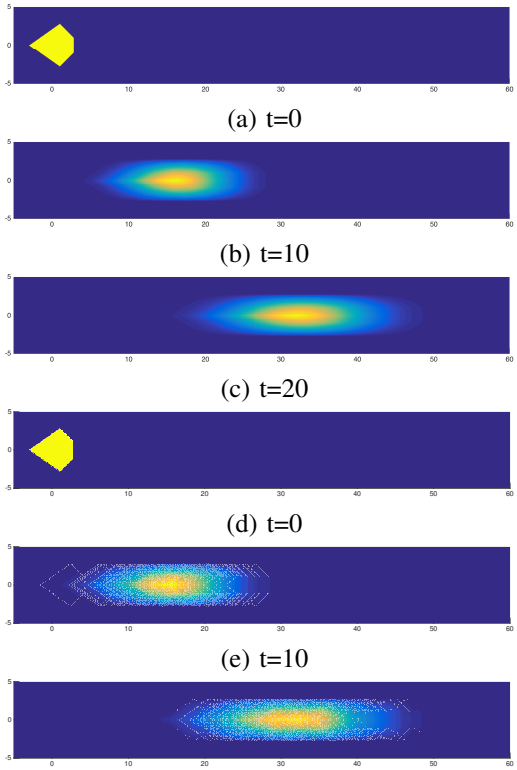
A variety of planning methods have been posed to prevent collisions in stochastic, dynamic environments. Researchers have incorporated velocity obstacles [13], the set of possible robot velocities that lead to collision, into planning methods that handle obstacles following an arbitrary trajectory [19] and stochastically moving obstacles [4]. However, computing the velocity obstacle online is prohibitively expensive for obstacles with complex geometry or stochastic dynamics. Our previous work [6], [7] used a backwards stochastic reachable set, computed for pairwise collisions only, to bias the repulsive potential in an artificial potential field-based method. However, the union of pairwise-computed backwards stochastic reachable sets can only provide an underapproximation of the likelihood of collision in environments with more than a single obstacle, and direct computation of the backwards reachable set for more than a single obstacle is prohibitively expensive. Stochastic Ensemble Simulation (SES) [8] attempts to circumvent this issue through a tree-based planner that uses of an offline or online [9] Monte Carlo simulation to find collision free paths. However, predicted collisions limit the algorithm's ability to find long-term paths in crowded environments. The resulting segments of short-term paths often leads the robot to an Inevitable Collision States (ICS) [14].

Several methods attempt to balance risks posed by both near and far obstacles. The method proposed in [1] considers possible future measurement results of obstacle location and the ability of the robot to replan. A directed roadmap weighted by the probability of collision considering the future measurements is constructed by sampling the compound probability distribution of obstacles at every time step. This method provides a point-based POMDP-like approach to exploit replanning and future predictions of obstacles [18]. The computation cost however, is exponential in the horizon of the compound probability distribution sampling and therefore it can be difficult to generate a long term path. Another method, [12], also leverages the fact that the robot can replan in the future and considers collision avoidance at the expected obstacle locations in the predicted horizon. This allows the planner to utilize long term predictions that may be critical to planning success. However, if the obstacle motion is highly stochastic, the assumption that the obstacle will end up at the average location has a high chance of leading the robot into an ICS.

III. Predictions and Dynamic Risk Tolerance

We first present a method for computing a prediction distribution for a stochastically moving obstacle. While this can be done with several techniques, we demonstrate a method for exactly predicting the future position distribution of moving obstacles by using Forward Stochastic Reachable (FSR) sets. Then, we pose an optimization problem for collision avoidance with a dynamic risk tolerance.

A. Forward Stochastic Reachable Sets



(a) t=0

(b) t=10

(c) t=20

(d) t=0

(e) t=10

(f) t=20

Fig. 2: Comparison between FSR obstacle prediction at various times in the future (a-c) and Monte Carlo predictions (d-f) with 500 particles. Computing the FSR took 2.91s. Monte Carlo predictions are accomplished as described in [8], and took 0.30s. A diamond shaped obstacle (described in Section V) is traveling to the right linearly with stochastic speed. As time progresses ((b-c) and (e-f)), the obstacle occupancy distribution (shown as a heat map) spreads for both the FSR and Monte Carlo predictions, respectively.

We consider obstacles with stochastic dynamics that are time-invariant and possibly nonlinear,

$$x^o_{t+1} = f^o(x^o_t, w^o_t) \tag{1}$$

with state $x_t^o \in \mathbb{R}^n$, disturbance $w_t^o \in \mathcal{W}$, an i.i.d. discrete random variable with a finite sample space $\mathcal{W}$ and probability mass function $p(w)$, and initial state x_0^o, without loss of generality.

We compute the Forward Stochastic Reach (FSR) set and the forward stochastic density function for a specific obstacle and a known time window T. The FSR set $\text{Reach}(t, \mathcal{I}) \subseteq \mathbb{R}^n$ characterizes the states that the obstacle could reach at time instant t with non-zero probability when $x_0^o \in \mathcal{I}$, a Borel set of $\mathbb{R}^n$, such that $\mathcal{I} \in \mathcal{B}(\mathbb{R}^n)$. For a given initial set $\mathcal{I} \in \mathcal{B}(\mathbb{R}^n)$ and a time instant $t \in \{1, 2, \ldots, T\}$, we define the forward stochastic density function, $\mathcal{F} : \{0, 1, \ldots, T\} \times \mathcal{B}(\mathbb{R}^n) \times \mathbb{R}^n \to [0, 1]$, recursively. With $\mathbb{P}\left\{x_t^o = z \middle| x_{t-1}^o = y\right\}$ as the probability that the obstacle is at location z at time t, conditioned on the obstacle's location at time instant $t-1$, we have

$$\mathcal{F}(t, \mathcal{I}, z) = \mathbb{P}\left\{x_t^o = z | x_0^o \in \mathcal{I}\right\} \tag{2a}$$

$$= \sum_{y \in \text{Reach}(t-1, \mathcal{I})} \mathbb{P}\left\{x_t^o = z \middle| x_{t-1}^o = y\right\} \mathcal{F}(t-1, \mathcal{I}, y)$$

$$= \sum_{y \in \text{Reach}(t-1, \mathcal{I})} \mathbb{P}\left\{w \mid z = f^o(y, w)\right\} \mathcal{F}(t-1, \mathcal{I}, y)$$

$$= \sum_{y \in \text{Reach}(t-1, \mathcal{I})} \sum_{v \in \{w \mid z = f^o(y,w)\}} p(v) \mathcal{F}(t-1, \mathcal{I}, y) \tag{2b}$$

$$\text{Reach}(t, \mathcal{I}) = \{z \in \mathbb{R}^n | \mathcal{F}(t, \mathcal{I}, z) > 0\} \tag{2c}$$

with $\mathcal{F}(0, \mathcal{I}, z) = 1, z \in \mathcal{I}$, and $\text{Reach}(0, \mathcal{I}) = \mathcal{I}$.

To compute the FSR sets, because the distribution is a probability mass function, we use a brute-force approach and discretize the state space with 0.05m resolution and propagate the probability values of each point in the set forward in time. More efficient computational methods can be exploited for certain classes of dynamics and distributions. For a 6m wide diamond shaped obstacle moving in a straight line with stochastic speeds, Figures 2(a), (b), and (c) show the FSR set and corresponding probability distribution at times 0s, 10s, and 20s, respectively. For comparison, Figures 2 (d), (e), and (f) show 500 particle predictions using Monte Carlo where the obstacle occupancy distribution is mapped onto a discretized workspace with the same resolution as FSR. Our proposed method, DRT planning, is general and can also use these approximate predictions.

To consider M moving obstacles, let $\mathcal{F}_i(t, x_i^o, \cdot)$ and $\text{Reach}_i(t, x_i^o)$ denote the probability density and the FSR set associated with the i^{th} obstacle when starting from $x_i^o \in \mathbb{R}^n$ at time t. Defining $\mathcal{E}_i = \{z \in \text{Reach}_i(t, x_i^o)\}$ as the event of a state z lying in the set $\text{Reach}_i(t, x_i^o)$, the probability of collision, that is, the probability that any one of the M obstacles occupies location z at time $t \in \{0, 1, \ldots, T\}$ is

$$\mathcal{G}(t, z) = \mathbb{P}(\cup_{i=1}^M \mathcal{E}_i). \tag{3}$$

Since the obstacles dynamics are independent, exact computation of (3) is possible via the inclusion-exclusion principle. However to improve computational time, we approximate (3) to second order.

$$\mathcal{G}(t, z) \approx \sum_{i=1}^{M} \mathcal{F}_i(t, x_i^o, z) - \sum_{i=1, i \neq j}^{M} \mathcal{F}_i(t, x_i^o, z) \mathcal{F}_j(t, x_j^o, z) \tag{4}$$

B. Dynamic Risk Tolerance

We presume the robot dynamics are given by

$$x_{t+1}^R = f^R(x_t^R, u_t), \tag{5}$$

with state $x_t^R \in \mathbb{R}^{n_r}$ and control input $u_t \in \mathcal{U} \subseteq \mathbb{R}^{p_r}$. For a path defined as a sequence of states $\{x_0^R, x_1^R, \ldots, x_T^R\} \in (\mathbb{R}^{n_r})^{T+1}$ to avoid collision with obstacles with a time-varying likelihood $P_{\text{accept}}(t; \tau)$, we require 1) $\mathcal{G}(t, x_t^R) \leq P_{\text{accept}}(t; \tau), \forall t = \{0, 1, \ldots, T-1\}$, and 2) that the terminal state lies in the goal set, $x_T^R \in \mathcal{X}_{\text{goal}}$.

This problem can be mathematically formulated as

Problem 1 (Dynamic risk tolerance problem):

$$\min_{\bar{u} \in \mathcal{U}^{T-1}} T \tag{6}$$

$$\text{subj. to} \begin{cases} x_T^R & \in \mathcal{X}_{\text{goal}} \\ x_{t+1}^R & = f^R(x_t^R, u_t), t \in \{0, 1, \cdots, , T-1\} \\ \mathcal{G}(t, x_t^R) & \leq P_{\text{accept}}(t; \tau), t \in \{0, 1, \cdots, T-1\} \end{cases} \tag{7}$$

where $P_{\text{accept}}(t; \tau) = \begin{cases} P_{\text{accept}}^{\text{const}} & \text{if } t \leq \tau \\ \alpha(t) & \text{if } t > \tau \end{cases}$.

The objective is to reach the goal $\mathcal{X}_{\text{goal}}$ in minimum time T with control input $\bar{u} = [u_1, u_2, ..u_T]$, such that the probability of collision of each node x_t^R is always less than $P_{\text{accept}}(t; \tau)$. Constraints arise due to robot dynamics and due to tolerance $P_{\text{accept}}(t; \tau)$ for risk. The optimal solution to Problem 1 is a path that minimizes time to reach while satisfying the constraints.

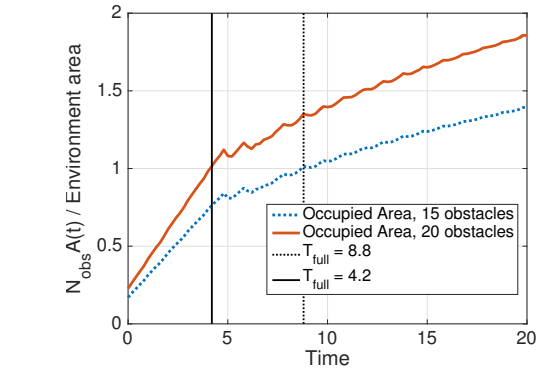
For $P_{\text{accept}}(t; \tau) = P_{\text{accept}}^{\text{const}}$, Problem 1 reduces to a straightforward (static) risk tolerance, typically chosen a priori to balance conflicting needs: short-term path safety requires low $P_{\text{accept}}^{\text{const}}$, but feasibility of paths in the long-term requires high $P_{\text{accept}}^{\text{const}}$, especially in the presence of a large number of stochastically moving obstacles. In contrast, the DRT function $P_{\text{accept}}(t; \tau)$ provides high likelihoods of collision avoidance in the short-term, but also to provide flexibility needed to find a feasible path over a longer horizon. That is, using the DRT function as the path acceptance threshold 1) enables the evaluation of longer paths, 2) maintains safety of the path in the short-term, and 3) avoids overly conservative solutions which limit path feasibility.

We propose several heuristic solutions for the choice of $P_{\text{accept}}(t; \tau)$ that use readily available domain knowledge, and investigate the value of τ and the form of the function $\alpha(t)$.

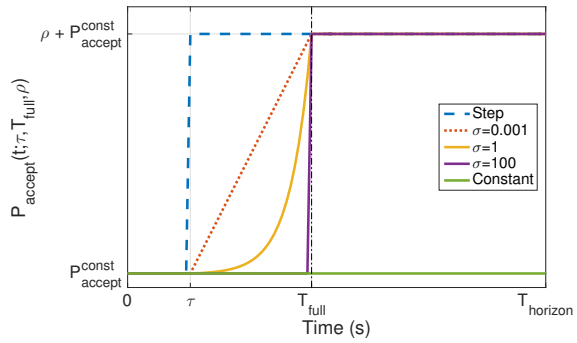
IV. Methods

A. Dynamic Risk Tolerance Function

We consider two classes of functions $\alpha(t)$ to describe the dynamic risk tolerance function $P_{\text{accept}}(t; \tau)$, as shown in Figure 3. While many possibilities exist, we select $\alpha(t)$ that are constructed with easily obtainable domain knowledge.



(a)



(b)

Fig. 3: (a) To heuristically determine a threshold at which no path will be feasible, we consider the area of possible occupancy of the obstacles in the environment without knowledge of their position. The upper limit on the normalized area occupied by 15 or 20 stochastically moving obstacles is shown, with a prediction of occupancy higher than $P^{\text{const}}_{\text{accept}} = 0.01$. For any time $t \geq T_{\text{full}}$, any path will eventually have a collision with probability greater than this limit. (b) Possible dynamic $\alpha(t)$ functions: step (blue) and exponential $e^{\sigma(t-\tau)}$, $\sigma \in \{1e-3, 1, 1e2\}$) (red, orange, and purple, respectively). Also shown is a constant valued $\alpha(t)$ (green). Domain specific properties, such obstacle density ρ and T_{full}, characterize $P_{\text{accept}}(t;\tau)$. For the environment shown in Figure 1, we select $\tau = 3$s, $P^{\text{const}}_{\text{accept}} = 0.01$, $\rho = 0.17$, and $T_{\text{full}} = 8.8$s in the above.

We first identify the time τ to transition to dynamic risk tolerance. Consider the following scenario. We define the area an obstacle occupies with probability greater than some value $P^{\text{const}}_{\text{accept}} \in (0, 1]$ as $A(t; P^{\text{const}}_{\text{accept}})$. If there are N_{obs} dynamic obstacles in the environment, the total area occupied by the obstacles, with probability greater than $P^{\text{const}}_{\text{accept}}$, is no greater than $N_{obs}A(t; P^{\text{const}}_{\text{accept}})$. As an example, Figure 3(a) shows $N_{obs}A(t; P^{\text{const}}_{\text{accept}})$, normalized by environment size, for various values of N_{obs} with $P^{\text{const}}_{\text{accept}} = 0.01$. In this plot, $N_{obs}A(t; P^{\text{const}}_{\text{accept}})$ exceeds the environment area at 8.8 seconds with $N_{obs} = 15$, and at 4.2 seconds with $N_{obs} = 20$. We define the time instant when $N_{obs}A(t; P^{\text{const}}_{\text{accept}})$ is equal to the area of the environment as T_{full}. For $t \geq T_{\text{full}}$, any path will eventually have a collision probability greater than $P^{\text{const}}_{\text{accept}}$.

Therefore, for $t \geq T_{\text{full}}$, we define an upper bound ρ which is the ratio of $N_{obs}A(0; P^{\text{const}}_{\text{accept}})$ to the total environment area. Essentially, we set the acceptable probability of collision to the portion of area actually occupied by the obstacles. Note that ρ and T_{full} can be computed numerically from analysis of predictions, i.e., FSR sets, offline.

The function $\alpha(t)$ that describes $P_{\text{accept}}(t;\tau)$ for $\tau < t < T_{\text{full}}$ can be any function that satisfies the boundary conditions. The best choice for $\alpha(t)$ may depend on problem specific properties, such as obstacle uncertainty growth rates (Figure 3(a)). We consider two classes of dynamic risk tolerance functions, shown in Figure 3(b): a step function $\mathbf{1}(t-\tau)$ and an exponential function $e^{\sigma(t-\tau)}$ with $\sigma = 0.001$ (similar to a linear function), $\sigma = 1$, and $\sigma = 100$. For comparison, a constant-valued function (Constant) is also shown.

Algorithm 1 growMultiStageTreeAndGetPath

Input: Precomputed FSR, Robot current state x^R, time T_{full}, current position of M obstacles $x^o = [x^o_1, ..., x^o_M]$
Output: [$\mathcal{P}$, $pathState$]

```
   /* τ-STAGE */
1: T = growInitialTree(x^R, FSR, maxIterTau, x^o)
2: if goal reaching nodes exist then
3:    return [goal reaching path, INIT_GOAL_REACHED]
4: else
   /* DRT STAGE */
5:    T = growRiskToleratingTree(x^R, FSR, T_full, maxIterRisk, x^o)
6:    [P, pathState] = getRiskToleratingPath(T)
7:    if pathState == RISK_TOLERATING then
8:       return [P, RISK_TOLERATING]
9:    else
   /* EMERGENCY STAGE */
10:      T = growEmergencyTree(x^R, FSR, maxIterEmrg, x^o)
11:      return [P, EMERGENCY] = getEmergencyPath(T)
12:   end if
13: end if
```

B. Multi-Stage Risk Tolerating Tree

Once the dynamic risk tolerance function is selected, a Multi-Stage Risk Tolerating tree is grown (Algorithm 1) in two stages, representing 1) $P_{\text{accept}}(t \leq \tau;\tau)$ and 2) $P_{\text{accept}}(t > \tau;\tau)$. A final stage is used when a path is not found from the first two stages. This is an emergency stage where any probability of collision is allowed for tree growth.

First, to find paths that ensure short-term path safety, the τ-stage, with $P_{\text{accept}}(t \leq \tau;\tau) = P^{\text{const}}_{\text{accept}}$, grows a tree with time step T_{step} in which all nodes in the tree have a low probability of collision (line 1, Algorithm 1). Since time of each node in the tree corresponds to a particular FSR set, Reach$(t, \cdot)$, the probability of collision can be evaluated for each node by simply querying the occupancy probability $\mathcal{G}(t, x)$, as in (4). Any efficient tree-based planner can be used to grow a tree from the robot's current position, where standard collision checking is replaced by querying

the probability of collision from the FSR sets. (We used RRT [20] in this paper.) To ensure short-term path safety, the τ-stage tree adds a tentative node if the probability of collision is less than $P_{\text{accept}}^{\text{const}}$. The tree building terminates if a path that reaches the goal exists or when the number of iterations reaches $maxIterTau$. Note that given a large $maxIterTau$, the tree grown in the τ-stage explores all possible state-time regions connected to the root with probability of collision $\leq P_{\text{accept}}^{\text{const}}$. Therefore, τ (the maximum time at which the probability of collision is $\leq P_{\text{accept}}^{\text{const}}$ for all nodes) can be defined for each *leaf* node of this tree. Algorithm 1 returns either the goal reaching path (line 3) or continues to the DRT stage.

The DRT stage, with $P_{\text{accept}}(\tau < t < T_{\text{full}}; \tau)$ or $P_{\text{accept}}(t \geq T_{\text{full}}; \tau)$, uses the DRT function $P_{\text{accept}}(t; \tau)$ (Figure 3) to find longer-term paths. In this stage, a risk tolerating tree is grown *from the leaves* of the tree grown in the τ-stage (GrowRiskToleratingTree, line 5). This tree utilizes the DRT function as the risk constraint. It adds a tentative node if the probability of collision is less than $P_{\text{accept}}(t; \tau)$, allowing the tree to explore regions with higher probability of collision. An example of this combined tree is shown in Figure 1(b). As compared to the tree grown in the τ-stage (e.g., Figure 1(a)), this tree explores a larger area thus enabling the extraction of long-term paths.

A path is extracted (line 6) by choosing the end node in this combined tree, which is at least $minPathTime_{RT}$ seconds away from the root. This ensures the path is a long-term path. The extracted path minimizes $maxCollProb + \epsilon$ distToGoal, where $maxCollProb$ is the maximum probability of collision of all nodes along the path and ϵ is a small greediness parameter that biases the extracted path to have low probability of collision and assures progression towards the goal. We use $maxCollProb$ instead of the accumulative probability of collision [17] since it is independent to the path step size T_{step}, i.e., it remains constant instead of approaching 1 as $T_{step} \to 0$.

If no path is found in the DRT stage, the algorithm enters the emergency stage, in which an emergency tree is grown on the previous tree (growEmergencyTree function, line 10). This tree adds tentative nodes with *any* probability of collision. A path that is at least $minPathTime_{EMRG}$ seconds long and minimizes $maxCollProb$ is extracted (getEmergencyPath, line 11).

The paths extracted from the Multi-Stage Risk Tolerating tree have the following properties. 1) Since the DRT stage is only triggered if a path cannot be identified with probability of collision less than $P_{\text{accept}}^{\text{const}}$, the resulting path avoids taking unnecessary risk and minimizes short-term collision probability. 2) The DRT function facilitates identification of a long-term path that balances progression toward the goal and probability of collision, thus reducing the chance of guiding the robot into an ICS.

C. DRT planning

The proposed motion planning framework, DRT planning is shown in Algorithm 2. DRT planning integrates Algorithm 1 and a set of replanning criteria, in order to identify paths and to utilize new obstacle observations and predictions.

Algorithm 2 DRT planning

Input: Precomputed FSR, Robot current state x^R, T_{full}, World simulation Time Step Δ_{sim}, max world simulation time $maxTime$

1: reGrowTree = true;
2: x^o = observeObstacles()
3: **for** $t = 0$; $t < maxTime$; $t = t + \Delta_{sim}$ **do**
4: **if** reGrowTree == true **then**
5: [$\mathcal{P}_{current}$, $pathState$] = growMultiStageTreeAndGetPath(x^R, x^o)
6: reGrowTree = false;
7: **end if**
8: x^R = x^R + Δ_{sim}· getAction($\mathcal{P}_{current}$)
9: [reGrowTree, x^o] = observeObstaclesAndCheckPath($\mathcal{P}$, x^R, t)
10: **if** t - lastGrowTrial >= $TrialTreePeriod$ && reGrowTree == false **then**
11: lastGrowTree = t;
12: [$\mathcal{P}_{trial}$, $pathState_{trial}$] = growMultiStageTreeAndGetPath(x^R, x^o)
13: $\mathcal{P}_{current}$ = chooseBetterPath($\mathcal{P}_{current}$, $\mathcal{P}_{trial}$, $pathState$, $pathState_{trial}$)
14: **end if**
15: **end for**

First, a Multi-Stage Risk Tolerating tree (Algorithm 1) is grown and a path is extracted (Algorithm 2, line 5) after observing the current position of all obstacles (line 2). The robot then executes the path $\mathcal{P}_{current}$ (line 8). Every time the robot reaches a node, the robot observes the obstacles and decides to find a new path (line 9, reGrowTree is set to true if a new path is required) if 1) the path has been fully executed, or 2) any node along the path in the near future has a probability of collision greater than $P_{\text{accept}}^{\text{const}}$ (within $checkPathLength$ seconds from the current instant).

In order to incorporate new obstacle observations and prediction, every few seconds ($TrialTreePeriod$), a trial path $\mathcal{P}_{trial}$ is extracted by growing another Multi-Stage Risk Tolerating tree (line 12). The trial path is constructed based on the new obstacle observation and is compared with the current path $\mathcal{P}_{current}$.

The trial path will replace the current path if it is expected to have a lower probability of collision along the path (line 13). This can happen if the trial path: is extracted from an earlier stage of the Multi-Stage Risk Tolerating tree, has a higher amount of time left to reach the last node with probability of collision lower than $P_{\text{accept}}^{\text{const}}$ (DRT stage), or has a smaller $maxCollProb$ (Emergency stage).

Note that DRT planning differs from the receding horizon control formalism [25] as the trial paths are identified every $trialTreePeriod$ (instead of every time step) and paths are not replaced unless they have lower collision probability

(instead of constant replacement). This is important since tree-based methods operate stochastically and different paths maybe returned for different runs.

V. Results and Discussion

We conducted experiments in a 40m by 40m environment (Figure 1(c)) with 5 to 20 fast moving obstacles. The obbstacles could have any convex geometry; we arbitrarily chose a 6m wide diamond shape to demonstrate our method's ability to handle obstacles with a convex, non-circular shape, which is known to be difficult for velocity obstacle based planners [30]. The obstacles are positioned randomly in the environment at the start time. We presume that the obstacles follow the stochastic dynamics (1) with $f^o(x_k, w_k) = x_k + \begin{bmatrix} w_k & \psi w_k \end{bmatrix}^T \cdot \Delta_{sim}$, with constant heading $\psi \in \mathbb{R}$ and stochastic speed $w_k \in \mathcal{W} = \{0.15, 0.90, 2.10, 3.00\}$ m/s with probability $p(w) = \{0.4, 0.1, 0.1, 0.4\}$, and obstacles can vary speed every 1s. The bimodal distribution and large speed differences of obstacle motion greatly increases the difficulty of planning, since the future position of obstacles is highly unpredictable. To maintain constant obstacle density, if the obstacle's center of mass hits a boundary, the obstacle is transported to the antipodal boundary with unchanged velocity. The holonomic point robot travels at a maximum speed of 1 m/s, 64% of the average speed of the obstacles and three times slower than any obstacle's maximum speed. Although the number of obstacles is much lower than compared to [6] and [8], the obstacles can travel much faster than the robot. In addition, the obstacles occupy a large portion of the planning domain (up to 23% in the 20 obstacle case as compared to 11% in the densest obstacle scenario in [8]).

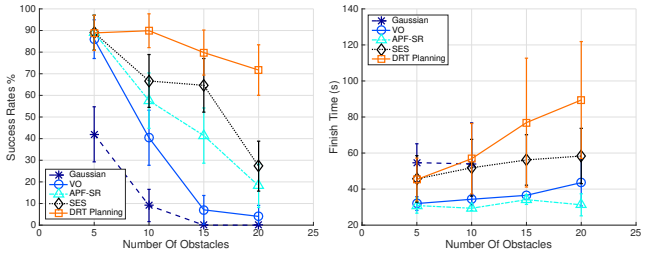
Our proposed method shares some parameters with other tree-based methods such as SES. We empirically determined these parameters for SES and applied them to DRT planning: Tree and FSR time resolution (T_{step} = 0.2s), greediness parameter (ϵ = 0.01), planning time horizon ($T_{horizon}$=20s), $P^{\text{const}}_{\text{accept}}$ = 0.01 and $checkPathHorizon$ = 2s. Parameters exclusive to our method are: Maximum RRT iterations ($maxIterTau$ = 10000, $maxIterRisk$ = 10000, $maxIterEmrg$ = 5000), minimum risk tolerating path time ($minPathTime_{RT}$ = 8s) and minimum emergency path time ($minPathTime_{EMRG}$ = 5s). Lastly, the world simulation resolution Δ_{sim} is set to 0.01s.

The FSR set offline computation was implemented in MATLAB and all planning methods were implemented in C++. All experiments were repeated 100 times and ran on a single core of an Intel i7-3720QM at 2.6GHz with 16GB of RAM. Uncertainty in success rates due the limited number of experiments is captured using the 99% confidence level derived from the central limit theorem while the variation in finish time is depicted by standard deviation.

A. Comparison to Existing Methods

Our first experiment is designed to compare our method with Stochastic Ensemble Simulation (SES) [8], Gaussian Artificial Potential Field (Gaussian) [24], Artificial Potential Fields biased by Stochastic Reachable sets (APF-SR) [6], and Velocity Obstacles (VO) [30].

We empirically determined the parameters that yield the highest navigation success rates for the comparison methods. SES uses the same parameters as the proposed method except for maxCD = 25000 (maximum number of RRT iterations), which is set to the same value as the total RRT iterations used in our proposed method. The Gaussian method uses $\mathcal{N}(0,3)$ as the repulsive potential. APF-SR has a goal bias (relative strength weight between the attractive potential and repulsive potential) of 0.01. VO was adapted from the RVO2 C++ code base implementation of the Optimal Reciprocal Collision Avoidance (ORCA) algorithm [29] to allow for single-agent collision avoidance, removing the reciprocal aspect of ORCA while maintaining many of ORCA's linear programming optimizations.



(a) (b)

Fig. 4: Success rates and finish times in environments with 5–20 stochastically moving obstacles. The finish time of Gaussian APF method is omitted for 15 and 20 obstacles since it failed to reach the goal. Averages are taken over collision-free (successful) runs.

Figure 4 shows that DRT planning has a 73% higher success rate while the best performing comparison method only has a 27% success rates in the crowded 20 obstacles environment. All methods, except Gaussian, perform well in the low obstacle density environments. SES performs well in medium obstacle density environments (10 and 15) compared to VO, APF-SR, and Gaussian, since it predicts the future position of obstacles with Monte Carlo simulations, much more accurately than a constant velocity approximation employed by VO, or the repulsive potential biased by SR sets computed for one obstacle used by APF-SR. However, in the most crowded environment (20 obstacles), SES also performs poorly, due primarily to the RRT online planner failing to find long-term paths and thus reaching ICS.

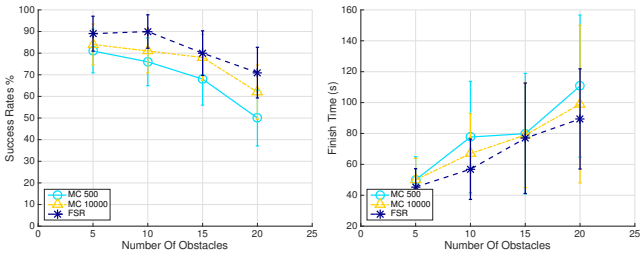
DRT planning has a computation time per planning step that is similar to SES and much slower than reactive methods (APF-SR, Gaussian, VO) (Table I). However, DRT finds paths with higher success rates and remains real-time capable (3.6ms per planning step) in the most crowded environment.

B. Impact of Prediction Quality

This experiment compares the obstacle prediction from FSR sets with those from Monte Carlo (MC) within DRT planning. The MC prediction has the same time resolution (T_{step}) as FSR and uses 500 and 10000 particles, respectively. The remaining parameters are those used previously.

Number of obstacles	DRT planning	SES	APF-SR	Gaussian	VO
5	3.7±1.3	1.7±1.2	0.016±.006	0.008±.005	0.006±.004
10	3.1±0.9	2.1±1.6	0.028±.009	0.016±.012	0.022±.015
15	3.2±0.8	2.9±2.3	0.034±.011	0.032±.019	0.017±.010
20	3.6±1.0	5.3±4.8	0.046±.016	0.027±.017	0.027±.017

TABLE I: Average computation time per planning step (milliseconds).



(a) (b)

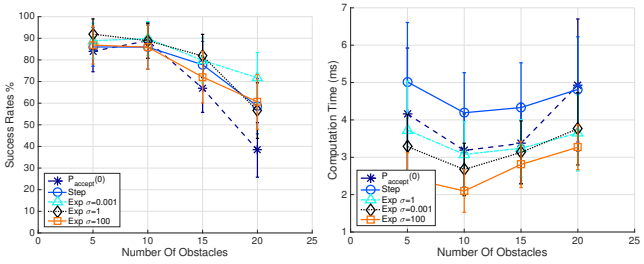
Fig. 5: Success rates and finish times using FSR and MC prediction with 500 particles and 10000 particles to predict future obstacle occupancy within DRT planning.

Figure 5(a) shows that, as expected, FSR has success rates consistently higher than MC predictions. In addition, MC simulations with low number of particles (MC 500) perform worse than ones with higher number of particles (MC 10000). Similarly with finish time (Figure 5(b)); lower finish times are associated with more accurate predictions, since unexpected obstacle motions cause the robot to replan.

This result demonstrates the impact of accurate predictions of obstacle's future position distribution on planning success. While FSR gives the exact position distribution and therefore is a valuable tool for planning, approximate predictions, such as MC approaches, can be applied as well.

C. Impact of Dynamic Risk Tolerance Heuristic

This experiment compares the performance of the various DRT functions proposed in Section IV-A with constant risk tolerance ($P_{\text{accept}}(t,\tau) = P_{\text{accept}}^{\text{const}}$). Note that the emergency tree (Algorithm 2, line 13) can still grow in all cases.



(a) (b)

Fig. 6: Comparison of planning success rates and computation time using constant, step, and exponential functions (as defined in Section IV-A) with various values σ.

Figure 6(a) shows all DRT functions have higher success rates than a constant risk tolerance in the crowded environments when T_{full} is less than $T_{horizon}$. In this case, we evaluated $T_{horizon} = 20s$ with T_{full} taking on values greater than 20, 20, 8.8, and 4.2s for environments with 5, 10, 15, and 20 obstacles, respectively. The best performing linear DRT function ($\sigma = 0.001$) has a success rate up to 30% higher than that with a constant risk tolerance, in the 20 obstacle environment. This suggests DRT is important to planning success in crowded environments, since it uses long term predictions to identify paths that avoid an ICS.

$P_{\text{accept}}^{\text{const}}$	Success rates (DRT)	Success rates (SES)
0	89±8%	50% ± 13%
0.01	83±10%	65% ± 12%
0.05	73±11%	63% ± 12%

TABLE II: Performance comparison between various $P_{\text{accept}}^{\text{const}}$. The environment has 15 stochastically moving obstacles.

Amongst DRT functions, the exponential that is almost linear (with σ=0.001) has the highest success rates, especially in crowded environments. This is likely due to the fact that the position uncertainty of our obstacle dynamics also grows roughly linearly with time (Figure 3(a)). Therefore, a linear-like function may balance between short term path safety and risk tolerance in order to plan a long term path.

Figure 6(b) indicates that a function that accepts risks early on (like the step function) has the highest computation time, likely due to an increase of replanning, since the path has a higher probability of collision and replanning is often required due to a potential collision is detected in the future (Algorithm 2, line 9).

D. Parameter Sensitivity Analysis

We conducted parameter sensitivity analysis on $P_{\text{accept}}^{\text{const}}$, the maximum probability of collision for $t = [0,\tau]$ for a node to be accepted in the tree, and for T_{step}, the time resolution in RRT. The environment has 15 dynamic obstacles.

A low value for $P_{\text{accept}}^{\text{const}}$ is often required to avoid short-term imminent collision, however, without DRT, this also hinders the planner's ability to find a long-term solution in order to avoid ICS. This tradeoff is reflected in Table II: the success rates of SES are highest at $P_{\text{accept}}^{\text{const}} = 0.01$. In contrast, the success rates of DRT planning increase as $P_{\text{accept}}^{\text{const}}$ decreases, with $P_{\text{accept}}^{\text{const}} = 0$ yielding the highest success rates. This suggests that the DRT bypasses the dilemma, resulting in much higher success rates (34% higher) and does not require the user to manually adjust $P_{\text{accept}}^{\text{const}}$.

Table III shows that success rate increases as T_{step} decreases. This is consistent with our observation that many collisions occur between nodes. Recall that DRT planning only evaluates the probability of collision at nodes of the Multi-Stage Risk Tolerating tree. A Multi-Stage Risk Tolerating tree with higher time resolution (lower T_{step}) reduces

T_{step} (s)	Success rates	Finish Time (s)
0.1	90±8%	79±40
0.2	83±10%	65±30
0.4	77±11%	64±25

TABLE III: Comparison of success rate and finish time for values of T_{step}. The environment has 15 stochastically moving obstacles.

the probability of collisions occurring between nodes, hence producing a higher success rate.

VI. Conclusion

We presented a framework for dynamic risk tolerance, based in dynamical systems and stochastic reachability. This framework is shown implemented in DRT planning that provides a time-varying bound on the acceptable likelihood of collision for a given path. We showed DRT planning with both exact predictions of obstacle locations through FSR sets and approximate predictions. Our experiments demonstrate a 46% higher success rate than the best performing comparison methods even in the most crowded environment tested.

VII. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number IIS-1528047, IIS-1553266 (Tapia, CAREER), CMMI-1254990 (Oishi, CAREER), and CNS-1329878. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] D. Althoff, B. Weber, D. Wollherr, and M. Buss. Closed-loop safety assessment of uncertain roadmaps. *Autonomous Robots*, 40(2):267–289, 2016.

[2] M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

[3] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[4] B.Kluge. Recursive agent modeling with probabilistic velocity obstacles for mobile robot navigation among humans. In *Proc. IEEE International Conference on Intelligent Robots Systems (IROS)*, pages 376–380, 2003.

[5] M. Chen, Q. Hu, C. Mackin, J. F. Fisac, and C. J. Tomlin. Safe platooning of unmanned aerial vehicles via reachability. In *IEEE Conference on Decision and Control (CDC)*, pages 4695–4701, 2015.

[6] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia. Aggressive moving obstacle avoidance using a stochastic reachable set based potential field. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2014.

[7] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2347–2354, 2015.

[8] H.-T. Chiang, N. Rackley, and L. Tapia. Stochastic ensemble simulation motion planning in stochastic dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3836–3843, 2015.

[9] H.-T. Chiang, N. Rackley, and L. Tapia. Runtime SES planning: Online motion planning in environments with stochastic dynamics and uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page To appear, 2016.

[10] C. F. Chung, T. Furukawa, and A. H. Goktogan. Coordinated control for capturing a highly maneuverable evader using forward reachable sets. In *International Conference on Robotics and Automation (ICRA), 2006*, pages 1336–1341, 2006.

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

[12] N. E. Du Toit and J. W. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2012.

[13] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[14] T. Fraichard and H. Asama. Inevitable collision statesa step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[15] J. H. Gillula, G. M. Hoffmann, H. Haomiao, M. P. Vitus, and C. J. Tomlin. Applications of hybrid reachability analysis to robotic aerial vehicles. *Int. J. Robot. Res.*, pages 335–354, 2011.

[16] V. A. Huynh, L. Kogan, and E. Frazzoli. A martingale approach and time-consistent sampling-based algorithms for risk management in stochastic optimal control. In *IEEE Conference on Decision and Control (CDC)*, pages 1858–1865, 2014.

[17] L. Janson, E. Schmerling, and M. Pavone. Monte Carlo motion planning for robot trajectory optimization under uncertainty. *arXiv:1504.08053*, 2015.

[18] H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, 33(3):255–272, 2012.

[19] F. Large, S. Sckhavat, Z. Shiller, and C. Laugier. Using non-linear velocity obstacles to plan motions in a dynamic environment. In *Control, Automation, Robotics and Vision*, pages 734–739, 2002.

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

[21] D. Limon, I. Alvarado, T. Alamo, and E. Camacho. Robust tube-based mpc for tracking of constrained linear systems with additive disturbances. *Journal of Process Control*, 20(3):248–260, 2010.

[22] Y. Lin and S. Saripalli. Collision avoidance for UAVs using reachable sets. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 226–235, 2015.

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

[24] M. Massari, G. Giardini, and F. Bernelli-Zazzera. Autonomous navigation system for planetary exploration rover based on artificial potential fields. In *Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) Conference*, 2004.

[25] J. Mattingley, Y. Wang, and S. Boyd. Receding horizon control. *IEEE Control Systems*, 31(3):52–65, 2011.

[26] D. Q. Mayne, M. M. Seron, and S. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[27] I. Mitchell, A. Bayen, and C. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *Transaction on Automatic Control*, pages 947–957, 2005.

[28] I. M. Mitchell. Comparing forward and backward reachability as tools for safety analysis. In *International Conference on Hybrid Systems: Computation and Control*, pages 428–443, 2007.

[29] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha. Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation. `http://gamma.cs.unc.edu/RVO2/`.

[30] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin. Interactive navigation of multiple agents in crowded environments. In *Symposium on Interactive 3D Graphics and Games*, pages 139–147, 2008.

[31] A. Wu and J. P. How. Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. *Autonomous Robots*, 32(3):227–242, 2012.