This paper appears in Robot Learning, 2026.
https://doi.org/10.55092/rl20260009

Paper Type | Received Day Mon Year; Accepted Day Mon Year; Published Day Mon Year
https://doi.org/10.55092/xxxx

Prediction-based RRT for Minimal Replanning in Dynamic Environments

Amr Marey[1]*, Qing Zhao[1] and Mahdi Tavakoli[1,2]

[1] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada

[2] Department of Biomedical Engineering, University of Alberta, Edmonton, Canada

* Correspondence author; E-mail: amarey@ualberta.ca

Highlights:

- Proposes Prediction-Based RRT* (PBRRT), which integrates obstacle motion prediction into RRT* for dynamic environments.
- Introduces a probabilistic feasibility cost function that favors paths less likely to be blocked by moving obstacles.
- Highlights trade-offs: PBRRT needs more sampling iterations initially, but improves safety and efficiency in dynamic planning.

Abstract: Rapidly-exploring Random Trees (RRT) have become a foundational tool for solving high-dimensional motion planning problems in both static and dynamic environments. In this paper, we introduce a novel RRT algorithm designed specifically for dynamic settings. Most existing RRT* variants for dynamic environments rely on the robot's current knowledge of obstacle positions and initiate replanning only after a collision risk is detected. This reactive strategy often leads to frequent replanning, which can be computationally expensive and result in longer paths. In contrast, our approach incorporates a general state prediction algorithm to estimate both current and future positions of moving obstacles. These predictions allow us to anticipate where obstacles are likely to appear in the configuration space and plan a motion that proactively avoids potential conflicts. Our RRT*-variant, Prediction-Based RRT* (PBRRT), modifies the traditional RRT* cost function to incorporate a measure of probabilistic feasibility. Simulation results demonstrate that PBRRT reduces path execution risk compared to other state-of-the-art algorithms.

Keywords: Dynamic Environment Motion planning, Prediction-based planning, Rapidly-exploring Random Trees (RRT), Autonomous robotics, Obstacle avoidance

1. Introduction

Motion planning in dynamic environments is a critical challenge in robotics and autonomous systems, where an agent must navigate safely and efficiently while addressing moving obstacles, changing terrain, and unpredictable elements. Unlike static environments, where paths can often be precomputed offline, dynamic settings demand real-time decision making, adaptability, and integration of predictive models. Key challenges include collision avoidance with moving obstacles, computational demands imposed by real-time constraints, uncertainty in predicting the motion of dynamic agents, and rapid replanning in response to unexpected changes. These challenges motivate the development of advanced algorithms, such as sampling-based planners, optimization methods, and machine learning techniques that enable robust and efficient navigation in dynamic environments [1].

Motion planning in dynamic environments leverages a diverse set of techniques to achieve real-time adaptability and reliable collision avoidance. Although sampling-based planners remain widely used, they are often complemented by other approaches that address the unique demands of dynamic settings. Optimization-based methods, such as Model Predictive Control (MPC), Covariant Hamiltonian Optimization for Motion Planning (CHOMP), and Stochastic Trajectory Optimization for Motion Planning (STOMP), refine motion plans by optimizing smoothness, safety, and feasibility [2, 3]. Graph-based planners, including A* variants such as D*-Lite and Hybrid A*, support efficient replanning in response to environmental changes [4, 5]. Artificial Potential Fields (APF) guide agents using attractive and repulsive forces, while Velocity Obstacles (VO) and Reciprocal Velocity Obstacles (RVO) facilitate collision avoidance in multi-agent scenarios by predicting future trajectories [6, 7]. Furthermore, machine learning techniques, such as deep reinforcement learning (DRL) and imitation learning, enable agents to make adaptive decisions based on learned behaviors and prior experience [8].

Accurate estimation of obstacle positions and velocities is critical for effective motion planning in dynamic environments. Predicting the future positions of moving obstacles can be approached using a range of techniques—from simple kinematic models, such as constant velocity or constant acceleration assumptions, to more advanced methods that account for uncertainty and complex dynamics [6]. Common approaches include filtering techniques such as the Extended Kalman Filter (EKF) and particle filters, which are well-suited for handling non-linear and stochastic motion models [9, 10]. In addition, data-driven methods, such as long short-term memory (LSTM) networks and multisensor fusion models, can learn complex motion patterns from data and offer greater flexibility when traditional models are inadequate [11, 12].

Rapidly-exploring Random Trees (RRT) [13] and its optimal extension, RRT* [14], are widely used sampling-based algorithms for motion planning in high-dimensional configuration spaces. RRT incrementally builds a search tree from a given start state by randomly sampling configurations and attempting to connect them, allowing it to efficiently find feasible paths in complex environments. However, RRT does not guarantee optimality. To address this limitation, RRT* introduces a rewiring step that iteratively improves the tree structure, enabling convergence to an optimal solution and ensuring asymptotic optimality over time [13, 14]. Despite their effectiveness, both algorithms assume a static and fully known environment, limiting their applicability in dynamic settings.

There are several variants in the literature that extend RRT* to dynamic environments. For instance, the Fuzzy-Kinodynamic RRT algorithm presented in [15] integrates fuzzy logic with kinodynamic constraints to enable unmanned aerial vehicles to adapt their paths in real time for obstacle avoidance in uncertain, changing scenarios. Furthermore, RT-RRT introduces a reverse tree guided strategy that constructs both a goal-rooted reverse tree and a forward tree from the robot's current state, facilitating rapid re-planning when unexpected obstacles disrupt the path [16]. Meanwhile, RRT-FND builds upon the RRT* framework by selectively repairing only the invalidated portions of its tree structure rather than rebuilding from scratch, thereby improving computational efficiency in dynamic environments [17]. Furthermore, RT-RRT* offers a real-time variant of RRT* that interleaves tree expansion, rewiring, and action execution to quickly update paths, making it well-suited for applications such as robotics and interactive gaming [18]. [19] presents another multi-query sampling-based motion planning algorithm for dynamic environments called AM-RRT*. The key idea is to augment the standard Euclidean distance with an assisting metric—such as diffusion distance—that better captures connectivity when line-of-sight is blocked, while still preserving probabilistic completeness and asymptotic optimality. The authors of AM-RRT* also propose a targeted rewiring strategy that focuses computation along the current path to the goal, significantly reducing re-planning time when goals change [19].

This paper proposes a prediction-based extension of RRT* for motion planning in dynamic environments, aiming to reduce the frequency of replanning events by proactively incorporating obstacle motion predictions. The proposed method generates an initial motion plan that is more resilient to dynamic obstacles, thereby minimizing the need for frequent replanning. This is achieved by introducing a probabilistic feasibility measure into the RRT cost function, prioritizing paths with a lower likelihood of future obstacle interference. Additionally, the approach integrates a multi-objective stochastic cost function that balances path feasibility with traditional optimization criteria such as travel distance, arrival time, and energy efficiency.

The rest of this paper is organized as follows: Section 2. presents the problem and the mathematical notation used in this paper. Section 3. discusses background information on RRT to provide the reader with the preliminary information needed to understand the proposed method. Section 4. presents the proposed dynamic environment RRT method and discusses the theory developed. Section 5. presents the simulation-based experimental results for the algorithms and compares them to other state-of-the-art RRT in dynamic environments. Section 6. discusses these results and outlines directions for future work. Finally, 7. concludes this paper.

2. Problem Definition and Notation

Let $\mathscr{X} \subseteq \mathbb{R}^n$ denote the configuration space of the robot. Furthermore, let $\mathscr{X}_{obs} \subseteq \mathscr{X}$ and $\mathscr{X}_{free} = \mathscr{X} \setminus \mathscr{X}_{obs}$ denote the obstacle and free configuration spaces. The state of the robot at time t is represented as $x[t]$. The start and goal states of the robot are denoted as $x_0 = x[t_0]$ and $x_N = x[t_f]$ respectively. $t_f \in \mathbb{R}_+$ denotes the arrival time of the path towards the point x_N, which is a free variable to be solved. Furthermore, let $\mathscr{T}$ denote the tree generated by a rapidly-exploring Random Tree (RRT) motion planning variant. V and E denote the vertices and edges of the tree respectively.

Let $\sigma : [0,1] \to \mathbb{R}^n$ be a continuous function that represents a path from x_0 to x_N such that $\sigma(0) = x_0$ and $\sigma(1) = x_N$. The tuple $(x_0, x_N, \mathscr{X}_{free})$ defines a path planning problem to move a robot from x_0 to x_N while avoiding obstacles. We let $\bar{\sigma}$ be the optimal path between x_0 and x_N in the case where there are no obstacles. This path is described by the notation $\overline{(x_0, x_N)}$.

Optimal motion planning considers finding a feasible path σ determined by the tuple $(x_0, x_N, \mathscr{X}_{free})$ that minimizes a cost function $J(\sigma)$. J is a monotonic, bounded, and strictly positive function for all collision-free paths. Furthermore, $J \equiv 0$ if and only if $\sigma(\theta) = \sigma(0) \forall \theta \in [0,1]$. Hence, the optimal path σ^* is defined as

$$\sigma^* = \arg\min_{\sigma} \{J(\sigma(\theta)) \mid \sigma(\theta) \in \mathscr{X}_{free} \forall \theta \in [0,1] \tag{1}$$

In a dynamic environment, the motion planning tuple needs a fourth argument for the arrival time t_f at the final state. Hence, the dynamic environment motion planning tuple is defined as $(x_0, x_N, \mathscr{X}_{free}, (t_0, t_f))$.

$\mathscr{X}_{obs}$ can be estimated by the function $\mathscr{Q} = f[\mathbf{x}, t]$. This function describes the obstacles the robot sensors detect at time t. In the case of a dynamic environment and the obstacles' motions are unknown to the robot agent, $\mathscr{Q}$ is a probabilistic time-varying function at each state. In the special case of a static fully known environment, $\mathscr{Q}$ is a deterministic time-invariant function. Furthermore, we denote the true state trajectory of the $i-$th dynamic obstacle as $\mathscr{Q}_{D_i}[t]$; similarly the states consisting of the static obstacles are denoted as $\mathscr{Q}_S$, such that $\mathscr{Q}_S = \mathscr{X} \setminus (\cup Q_{D,i})$. The estimated states of the dynamic obstacles are denoted as $\hat{\mathscr{Q}}[t]$. Also, let the number of separate dynamic obstacles be denoted by the positive integer k. The probability that the direct path from node x_{i-1} to node x_i is occupied by at least one obstacle at time t is denoted as $\mathrm{P}(x_{i-1}, x_i, \mathscr{X}_{free}, (t_0, t_f))$.

3. Background

*3.1. RRT**

The RRT* algorithm, shown in Algorithm 1 and originally presented in [14], is an optimal version of the RRT sampling-based algorithm. The environment is assumed to be static and fully known in both algorithms [13, 14].

RRT* repeatedly samples points from the configuration space and determines the node belonging to the tree that is nearest to the sampled point (lines 3,4). This node is incrementally steered slightly towards the sampled node to generate a new node in the tree (line 8) under the assumption that there is no obstacle at the location of this node (line 5). RRT* actively evaluates whether the new sample can serve as a more efficient intermediary for nearby nodes (within cost below R), updating the connections to lower the overall path cost (lines 9-14). Furthermore, the algorithm also checks if the nearby nodes can serve as a more efficient intermediary for the new sample (lines 17-23).

3.2. Obstacle State Estimation

The proposed approach assumes access to another model that can track and predict the most probable trajectory that the obstacles will follow (i.e., estimate $\mathrm{P}(x_0, x_1, \mathscr{X}_{free}, (t_0, t_f))$). There are decades of research work that discusses this. One can use the classical methods such as extended Kalman filters (EKF) and particle filters (PF) [9, 10]. However, one can also use recently proposed data-driven techniques for object

Algorithm 1: RRT* Algorithm presented in [14]

$(V,E) \leftarrow (\{x_0\},\varnothing)$;
for $i = 1,2,\cdots,n$ **do**
  $x_{rand} \sim \mathscr{X}$;
  $x_{nearest} = \text{Nearest}(\mathscr{T},x_{rand})$;
  $x_{steer} = \text{Steer}(x_{nearest},x_{rand})$;
  if $\overline{(x_{nearest},x_{steer})} \subset \mathscr{X}_{free}$ **then**
    $X_{near} \leftarrow \{x \in \mathscr{T} | J(x,x_{steer}) < R\}$;
    $V \leftarrow V \cup \{x_{steer}\}$;
    $(x_{min},J_{min}) \leftarrow \Big(x_{nearest}, J(x_0,x_{nearest}) + J(x_{nearest},x_{steer})\Big)$;
    for $x \in X_{near}$ **do**
      $J \leftarrow J(x_0,x) + J(x,x_{steer})$;
      if $\overline{(x,x_{steer})} \subset \mathscr{X}_{free}$ ***and*** $J < J_{min}$ **then**
        $(x_{min},J_{min}) \leftarrow (x,J)$;
      end
    end
    $E \leftarrow E \cup \{x_{min},x_{new}\}$;
    for $x \in X_{near}$ **do**
      $J \leftarrow J(x_0,x_{steer}) + J(x_{steer},x)$;
      if $\overline{(x_{steer},x)} \subset \mathscr{X}_{free}$ ***and*** $J < J(x_0,x_{steer})$ **then**
        $(x_{min},J_{min}) \leftarrow (x,J)$;
      end
      $E \leftarrow (E \setminus \{(x_{\min},x)\}) \cup \{(x_{steer},x)\}$
    end
  end
end

tracking such as LSTM neural networks or reinforcement learning techniques [11, 20]. Theoretically, the proposed algorithm presented in this paper can be applied alongside any of the object state estimation algorithms found in the literature.

4. Prediction Based RRT

4.1. Optimization Framework

Previous work on RRT in dynamic environments leverages the obstacles of the current state to plan. The contribution of this work is that the proposed algorithm, termed prediction-based RRT* (PBRRT), uses the current state of the obstacles and the estimated future states of the obstacles. However, the proposed algorithm has the following assumption.

Assumption 1 *We assume that the state trajectory of $\mathscr{Q}_{D_i}[t]$ is at least C^1 smooth such that future states of $\mathscr{Q}_{D_i}[t]$ can be estimated within some time horizon t_H.*

Assumption 1 describes that the trajectory of each dynamic obstacle should not exhibit abrupt changes in their state trajectory to allow us to estimate the future location of the dynamic obstacles in the near future. Furthermore, an object cannot suddenly appear unexpectedly. Hence,

$$||\hat{\mathscr{Q}}_{D_i}[t] - \mathscr{Q}_{D_i}[t]|| < \varepsilon \forall t \in [t_0, t_0 + t_H], i = 1, 2, ..k$$

Since the robot agent does not have complete knowledge of $\mathscr{X}_{free}$ in the future, the cost function associated with the path between two RRT nodes $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ is treated as a probabilistic quantity, hence we try minimize the expectation of J between two nodes such that:

$$E[J(x_i, x_{i+1})] = \mathrm{P}(x_i, x_{i+1}, \mathscr{X}_{free}, (t_i, t_{i+1}))M + (1 - \mathrm{P}(x_i, x_{i+1}, \mathscr{X}_{free}, (t_i, t_{i+1})))J_{RRT^*}(x_i, x_{i+1}). \quad (2)$$

where J_{RRT^*} is the cost between the two nodes had we considered using the optimal RRT* algorithm or one of its variants. M is a large positive constant value that penalizes the path if it collides with the probabilistic dynamic obstacle. Hence, the value of M should be much larger than the RRT* cost ($M > J_{RRT^*}$). Hence in order to minimize $E[J(x_i, x_{i+1})]$, the node rewiring process will favor node connections where $\mathrm{P}(x_0, x_1, \mathscr{X}_{free}, (t_0, t_f)) \to 0$ as

$$\begin{gathered} J_{RRT^*} < E[J(x_i, x_{i+1})] \forall \mathrm{P}(x_i, x_{i+1}, \mathscr{X}_{free}, (t_i, t_{i+1})) < P_{safe} \\ \text{and} \\ \arg\min_{\mathrm{P}(x_i, x_{i+1}, \mathscr{X}_{free}, (t_0, t_f))} E[J_{node}] = 0. \end{gathered} \quad (3)$$

$0 < P_{safe} < 1$ is a user-defined parameter that describes how close the path the agent takes will be to the dynamic obstacles. This will direct the motion planning algorithm to search for low-cost paths that are most likely feasible. However, if we assume the robot's current position is at node x_0 at time t_0, the dynamic obstacle function obtained from the robot sensors can only reliably estimate the future dynamic obstacles' positions within a short time horizon up to t_1 such that $t_1 - t_0 < t_H$. As t_1 increases, the estimated probabilities $\hat{\mathrm{P}}(x_0, x_1, \mathscr{X}_{free}, (t_0, t_f))$ begin to deviate from the true probabilities $\mathrm{P}(x_0, x_1, \mathscr{X}_{free}, (t_0, t_f))$.

Hence, we modify the total expected cost for a path yet to be traversed by a robot using a PBRRT tree as

$$\begin{aligned}\hat{E}[J_{path}] = \hat{E}[J(x_0, x_N)] &= \sum_{i=0}^{N-1} \hat{E}[J(x_i, x_{i+1})] \\ &= \sum_{i=0}^{N-1} \hat{\mathrm{P}}(x_i, x_{i+1}, \mathscr{X}_{free}, (t_i, t_{i+1}) | \mathbf{x}[t_i], \mathscr{P}(t_0)) \gamma^{\alpha} M + \\ &\qquad (1 - \gamma^{\alpha} \hat{\mathrm{P}}(x_i, x_{i+1}, \mathscr{X}_{free}, (t_i, t_{i+1}) | \mathbf{x}[t_0], \mathscr{P}(t_i))) J_{RRT^*}(x_i, x_{i+1}). \quad (4)\end{aligned}$$

where γ is the discounting factor constant such that $0 < \gamma < 1$. If the discount factor is closer to 0, the motion planner will gradually neglect uncertain collisions in the long horizon; if $\gamma = 1$, even these uncertain collisions in the long horizon are considered for planning. Next, $\alpha \in \mathbb{Z}_{\geq 0}$ is the number of node generations between the starting node and node x_i. As such, as the number of node generations increases (i.e. the planner is planning in the far future), the expected cost between two nodes will be

$$\lim_{k \to \infty} \hat{E}[J(x_i, x_{i+1})] = J_{RRT^*}(x_i, x_{i+1}). \quad (5)$$

The proposed PBRRT algorithm aims to plan a trajectory that minimizes the expected path cost function in (4).

4.2. Proposed Algorithm

The proposed algorithm consists of an initial planning loop, which is responsible for developing an initial plan at the start of the simulation; and an execution loop, which causes the robot agent to execute its path until an obstacle interferes with the planned path due to a probability of collision (PoC) that is greater than P_{safe}. At this point, the algorithm develops a new plan based on the initial tree developed and some new samples to update the tree to the current free configuration space.

4.2.1. Initial Planning Loop

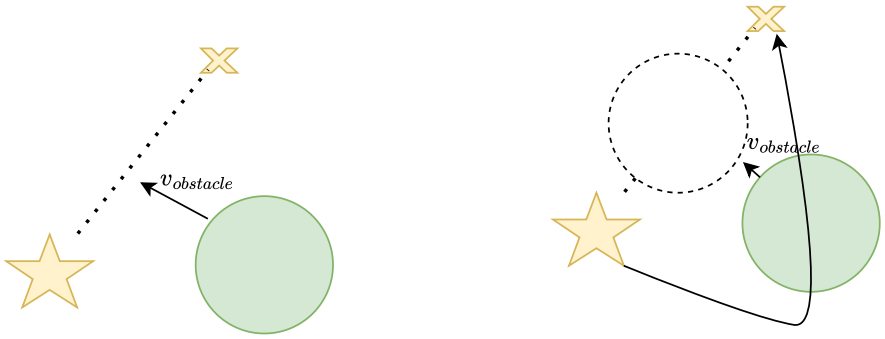
The initial planning loop, presented in Algorithm 2, has some similarities to the RRT* algorithm. However, instead of just considering whether we are colliding with any static obstacles when checking for collisions (Algorithm 1, lines 6, 12, and 19), the proposed algorithm checks the PoC between the agent and the dynamic obstacle (it is impossible to exactly determine as the obstacles' trajectories are unknown to the agent). If this collision probability is below some threshold P_{safe}, we proceed with the algorithm as in standard RRT*. However, the other major difference is that PBRRT uses the cost function presented in (4) to minimize instead of the regular RRT* cost metric. The resulting path is a smooth trajectory that is relatively close to the optimal path; however, the initially planned trajectory will not come too close to dynamic obstacles, as that heavily increases the cost function. This is sketched in Fig. 1.

4.2.2. Execution Loop

Once the robot has planned its initial path, it will begin to execute this motion plan until an obstacle interferes with this. This execution loop is shown in Algorithm 3. As the robot moves from its current

Algorithm 2: PBRRT Initial Loop

$(V,E) \leftarrow (\{(x_0,t_0)\},\varnothing)$;
for $i = 1,2,\cdots,n$ **do**
  $x_{rand} \sim \mathscr{X}$;
  $x_{nearest} = \text{Nearest}(\mathscr{T},x_{rand})$;
  $x_{steer} = \text{Steer}(x_{nearest},x_{rand})$;
  if $\overline{(x_{nearest},x_{steer})} \subset \mathscr{Q}_S$ *and* $\hat{P}(x_{nearest},x_{steer},\mathscr{X}_{free},(t_i,t)) < P_{safe}$ *for some* $t > t_i$ **then**
    $X_{near} \leftarrow \{x \in \mathscr{T} | \hat{E}[J(x_{nearest},x_{steer})] < R\}$;
    $V \leftarrow V \cup \{(x_{steer},t)\}$;
    $(x_{min},J_{min}) \leftarrow \left(x_{nearest},\hat{E}[J(x_0,x_{nearest})] + \hat{E}[J(x_{nearest},x_{steer})]\right)$;
    for $x \in X_{near}$ **do**
      $J \leftarrow \hat{E}[J(x_0,x)] + \hat{E}[J(x,x_{steer})]$;
      if $\overline{(x,x_{steer})} \subset \mathscr{X}_{free}$ ***and*** $\hat{P}(x,x_{steer},\mathscr{X}_{free},(t_i,t)) < P_{safe}$ ***and*** $J < J_{min}$ **then**
        $(x_{min},J_{min},t_{min}) \leftarrow (x,J,t)$;
      end
    end
    $E \leftarrow E \cup \{(x_{min},t_i),(x_{steer},t)\}$;
    for $x \in X_{near}$ **do**
      $J \leftarrow \hat{E}[J(x_0,x_{steer})] + \hat{E}[J(x_{steer},x)]$;
      if $\overline{(x_{steer},x)} \subset \mathscr{X}_{free}$ ***and*** $\hat{P}(x_{steer},x,\mathscr{X}_{free},(t_i,t)) < P_{safe}$ *and* $J < J(x_0,x_{steer})$ **then**
        $(x_{min},J_{min},t_{min}) \leftarrow (x,J,t)$;
      end
      $E \leftarrow (E \setminus \{(x_{\min},x)\}) \cup \{(x_{steer},x)\}$
    end
  end
end



(a) Node connection in other RRT* algorithms. The predicted position of the obstacle is not considered.

(b) Node connection in proposed RRT algorithm. The predicted position of the obstacle is considered.

Figure 1. Node connection comparison between proposed RRT algorithm and other RRT algorithms

position to the next node on the planned path, it removes previous nodes that it has passed from the tree alongside all of the node's descendants. Furthermore, it checks if the planned path has a high probability of not being interfered with by other obstacles; hence the PoC should remain low. If there is a high PoC, we update the probabilities of the existing tree and re-grow a new tree using the method discussed in the PBRRT initial loop section.

Mathematically, consider the planned path generated by the tree to be described by the set of tuples $S_{planned} \leftarrow \{(x_c, x_{c+1}), (x_{c+1}, x_{c+2}), ..., (x_{c+k-1}, x_{c+k})..., (x_{c+K-1}, x_{c+K}))\}$. Where x_c describes the current node location, and x_{c+k} describes the node in the tree that is in the k-th generation after x_c. K is the total number of nodes in $S_{planned}$. As soon as the agent executes a step such that its new current location is x_{c+1} (Line 2), the new planned path considered is $S_{planned} \leftarrow \{(x_{c+1}, x_{c+2}), (x_{c+2}, x_{c+3}), ..., (x_{c+K-1}, x_{c+K}))\}$. After executing a step in the planned path (Line 3), the parent node of where the robot is currently located is removed from the tree alongside any of its descendants that are not in $S_{planned}$ (Line 4). During this time-frame, the dynamic obstacles would have moved to new locations hence the current $S_{planned}$ may need to be modified and the PoC (Line 5). For the node in the k-th generation after the current location node (x_{c+k}), we determine if $\gamma^k \hat{P}(x_{c+k-1}, x_{c+k-1}) > P_{safe}$. If this condition is true for at least one of the nodes, then it implies that the current planned path is too risky to execute and hence a new planned path is needed. The replanning process (Line 7) is similar to the initial plan algorithm. However, we do not start from an empty tree. We have an existing tree; however, each sample in the tree needs to have its PoC value updated to match the current state of the configuration space. After the PoC values are updated, we can add new samples and connections to the tree similar to what is described in Algorithm 2.

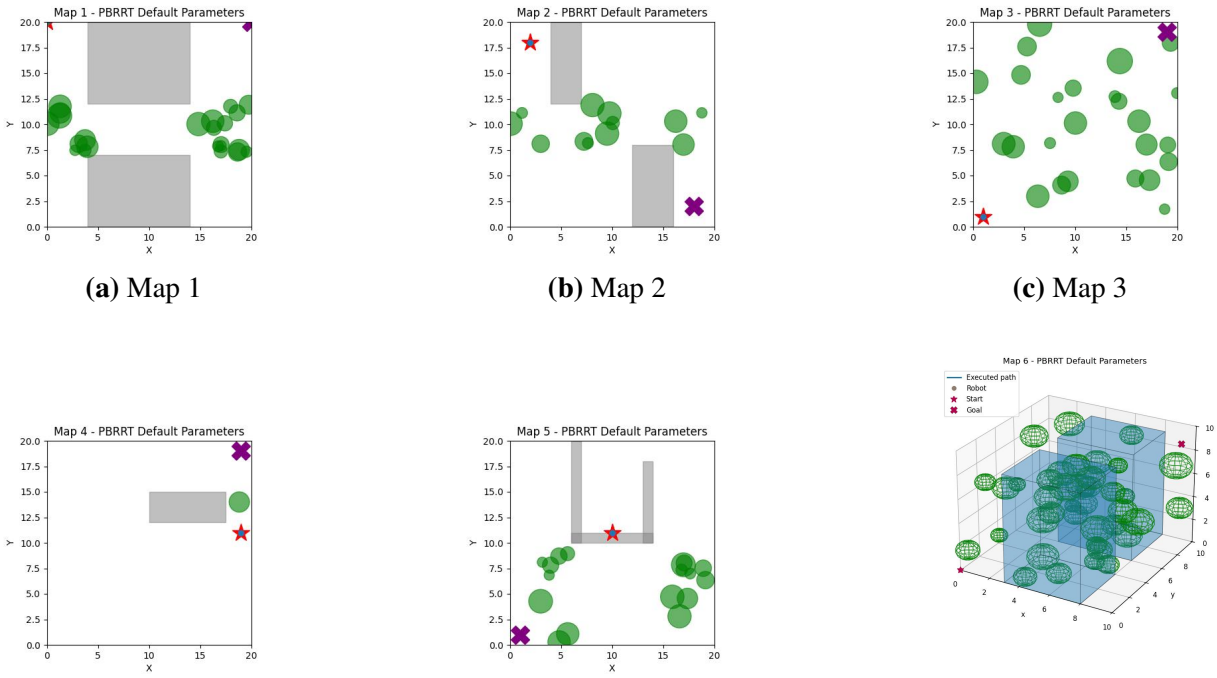
5. Simulation Experiments

The proposed algorithm is tested on five separate 2D maps and one 3D map using a point robot agent. Each 2D map is sized $[0, 20] \times [0, 20]$. The 3D map (Map 6) is sized $[0, 10] \times [0, 10] \times [0, 10]$. In the 2D

Algorithm 3: PBRRT Execution Loop

```
k = 0;
while k < K do
    ExecutePath();
    RemovePreviousNode&DescandantsfromTree();
    UpdateEnvironment();
    if γ^k P̂(x_{c+k-1}, x_{c+k-1}) > P_safe then
        Replan();
    end
    k = k+1 ;
end
```

maps, the static obstacles are non-circular while the dynamic obstacles are circular. Similarly, Map 6 has the dynamic obstacles to be shaped as spheres while the static obstacles are rectangular prisms. The main purpose of Map 6 is to show PBRRT working in higher dimensional planning spaces. The five 2D maps and the one 3D map are shown in Fig. 2. The default value for the parameters discussed in this paper are chosen based on manual tuning and are presented in Table 1. Section 5.3. provides ablation experiments that vary one of these parameters to study its effects. The simulations are set-up by default such that the robot begins executing a path as soon as PBRRT determines a feasible path. The authors set the dynamic obstacles move randomly to show the robustness of the algorithm against any type of obstacle motion. The videos showcasing the robots executing the planned paths for all experiments are provided in Section 8.1.2..



(a) Map 1 **(b)** Map 2 **(c)** Map 3

(d) Map 4 **(e)** Map 5 **(f)** Map 6

Figure 2. The six simulated maps. The start location is represented with the red star and the goal location is represented with the purple cross.

Table 1. Parameters used in Simulations.

Parameter	Value
η (Steering Parameter for RRT* [14])	0.15
P_{safe}	0.1
M	15
γ	0.9
σ^2	0.1

5.1. Obstacle Tracking Mechanism used in Simulation

As mentioned in Section 1. and 3., there is a multitude of ways to estimate and predict the location of moving obstacles. However, the authors decided to be general and focus on the planning aspect for the simulations as this is the main focus of the paper. In the simulations, we assume that the estimated position of a dynamic obstacle at any given time — denoted as $\hat{x}_{D_i}[t]$ — is

$$\hat{x}_{D_i}[t] = x_{D_i}[t] + \mathcal{N}(0, \sigma^2) \tag{6}$$

Where $x_{D_i}[t]$ denotes the true location of the dynamic obstacle and $\mathcal{N}(0, \sigma^2)$ denotes zero-mean Gaussian noise with variance σ^2. Details on how the PoC at each node is calculated are shown in Section 8..

The simulations are performed on an Ubuntu 24.04 desktop computer with an Intel Core i9-14900KF processing unit, 32 GB RAM, and a NVIDIA GeForce GTX 1660 Ti graphics card.

5.2. Comparison with other RRT Algorithms

PBRRT is tested on Maps 1-5 and compared with three other dynamic environment RRT algorithms - RRT*FND, RT-RRT*, and AM-RRT* [17, 18, 19]. The authors compare the executed path determined from these planning algorithms and determine the PoC at each timestep. To quantitatively determine how close the robot agent is to collision at each timestep, the distance to the nearest dynamic obstacle (DNDO) and the overall PoC are shown in Fig. 3 and Fig. 4 respectively. The end-to-end run-times for each algorithm, minimum value for nearest distance, and maximum value of probability of collision are shown in Table 2. Tables 3 and 4 report the replan counts and the initial plan node counts for PBRRT and RRT*FND across Maps 1-5. AM-RRT* and RT-RRT* are not included in this comparison as they execute motions prior to full planning the path and adjust the planned path at each timestep.

5.3. Ablation Experiments

Each one of the parameters listed in Table 1 is adjusted to showcase the effect of that parameter.

5.3.1. M Ablation Experiment

To showcase the effect of M on PBRRT, we re-ran Map 1 while adjusting M such that $M = 0$. Before executing the paths in this ablation study, the tree is allowed to grow to 10,000 samples to approximate the truly optimal path under an infinite number of samples. The PoC and DNDO graphs comparing this

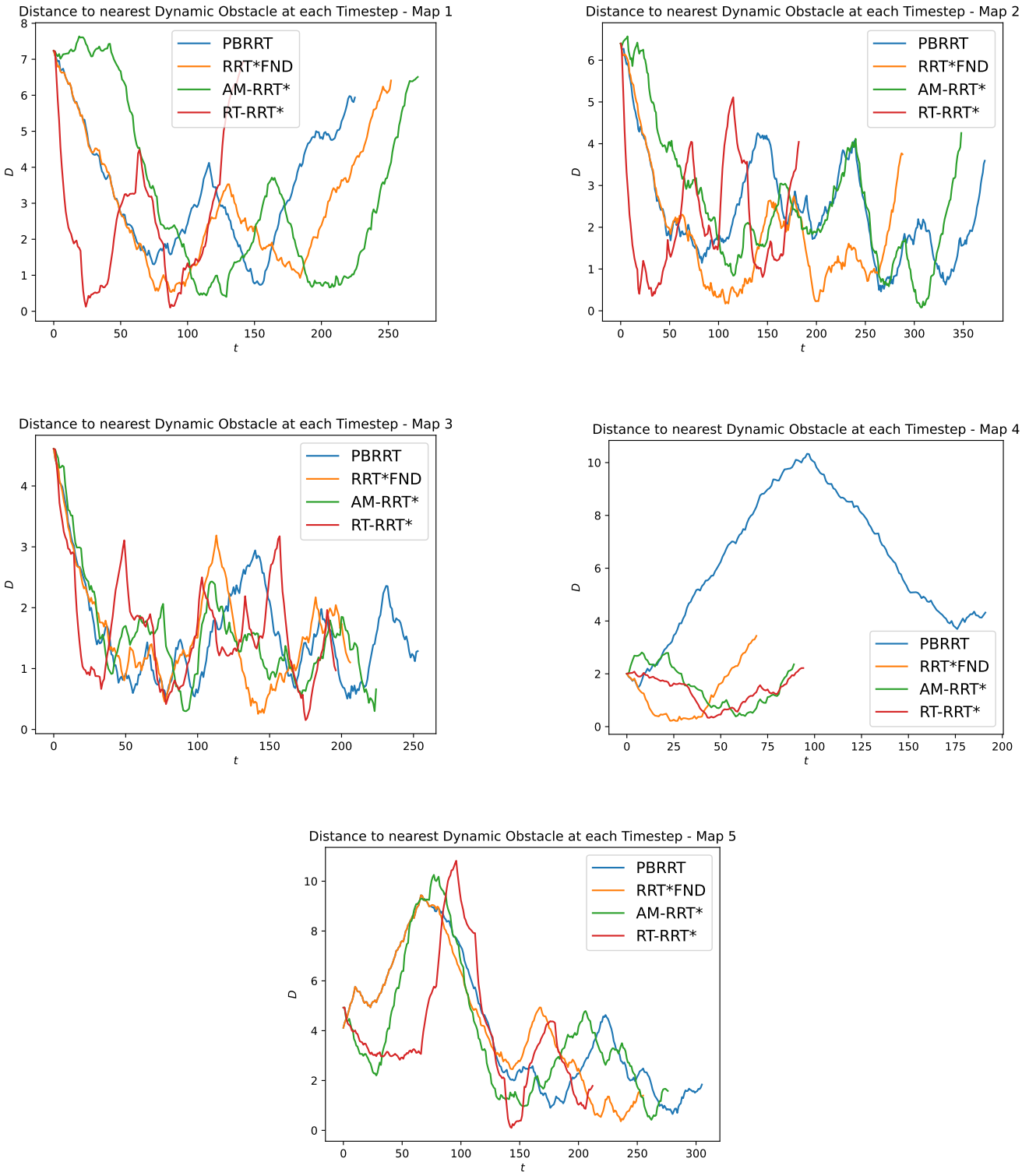


Figure 3. Distances to the nearest Dynamic Obstacles at each Timestep - Maps 1-5.

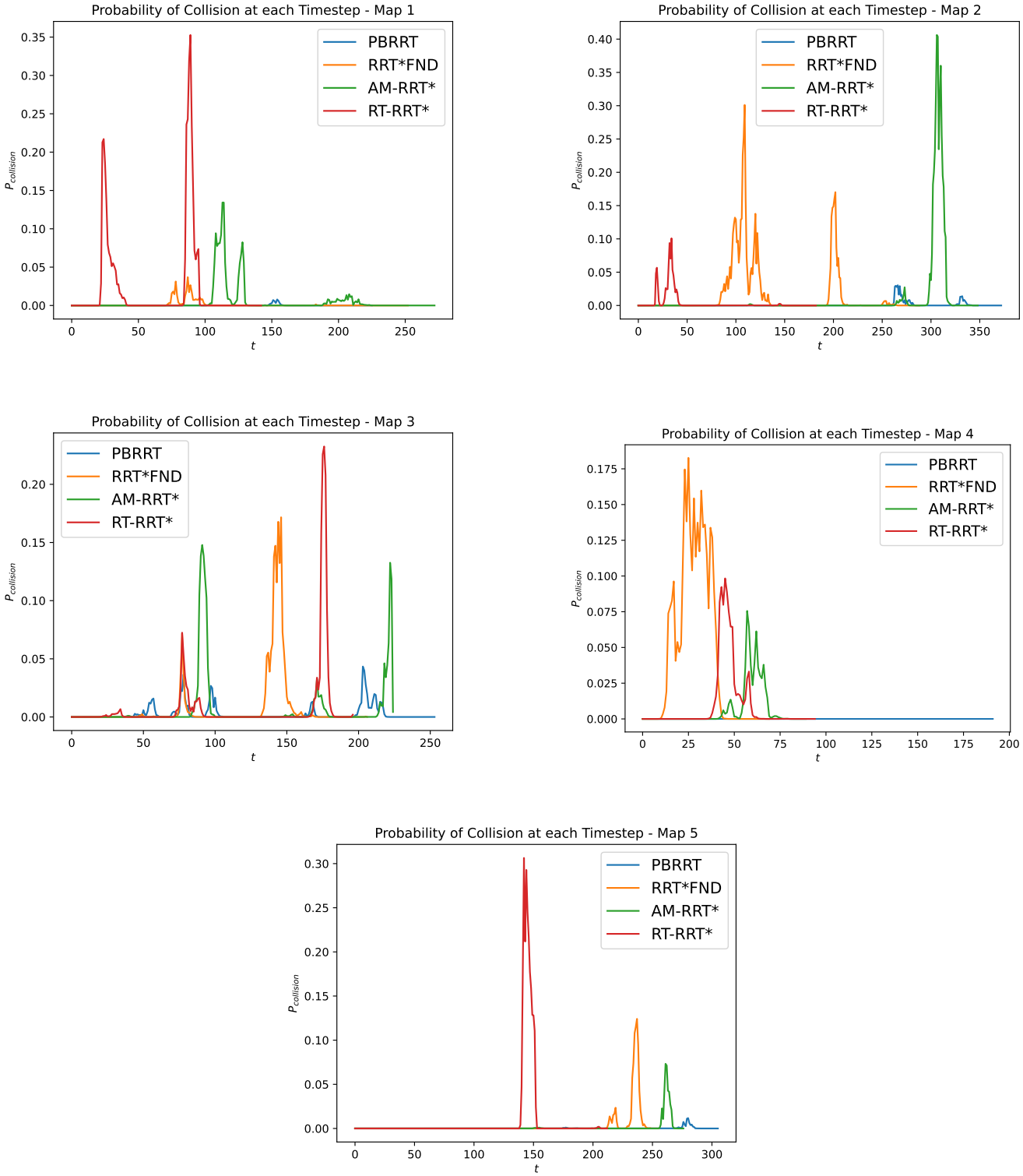


Figure 4. Probability of Collision at each Timestep - Maps 1-5.

Algorithm	T_{run}	P_{max}	D_{min}
PBRRT	11.54	0.01	0.73
RRT*FND	26.78	0.08	0.54
RT-RRT*	8.51	0.35	0.10
AM-RRT*	12.09	0.13	0.40

(a) Map 1

Algorithm	T_{run}	P_{max}	D_{min}
PBRRT	24.08	0.03	0.46
RRT*FND	52.05	0.30	0.17
RT-RRT*	10.41	0.10	0.36
AM-RRT*	12.95	0.41	0.08

(b) Map 2

Algorithm	T_{run}	P_{max}	D_{min}
PBRRT	47.25	0.04	0.51
RRT*FND	26.35	0.17	0.26
RT-RRT*	3.80	0.23	0.15
AM-RRT*	5.96	0.15	0.30

(c) Map 3

Algorithm	T_{run}	P_{max}	D_{min}
PBRRT	2.01	0.00	1.47
RRT*FND	2.28	0.18	0.22
RT-RRT*	3.82	0.10	0.33
AM-RRT*	5.54	0.08	0.38

(d) Map 4

Algorithm	T_{run}	P_{max}	D_{min}
PBRRT	18.60	0.01	0.65
RRT*FND	14.39	0.12	0.36
RT-RRT*	7.78	0.31	0.10
AM-RRT*	12.75	0.07	0.42

(e) Map 5

Table 2. End-to-end run-times (in seconds), maximum probability of collision, and minimum distance to nearest dynamic obstacle.

Table 3. Replans in PBRRT and RRT*FND for each map

Map	PBRRT	RRT*FND
1	0	5
2	4	16
3	20	6
4	0	2
5	1	2

alteration to the default setup are shown in Fig. 5. The robot agent had to do seven replans in the $M = 0$ scenario and five replans in the $M = 15$ scenario. This is expected as the case of $M = 15$ has a higher penalty for paths that will likely need to be replanned.

5.3.2. P_{safe} Ablation Experiment

To showcase the effect of P_{safe} on PBRRT, we re-ran Map 4 while adjusting P_{safe} such that $P_{safe} = 0.5$. The path is executed as soon as tree determines a path for this ablation study. The overall paths executed by both scenarios is illustrated in Fig. 6. The DNDO and PoC graphs are for the P_{safe} ablation study are shown in Fig. 7. In the scenario where $P_{safe} = 0.1$, there were zero replans after the initial plan and there were 3289 nodes in the tree when the initial plan was developed. However, in the scenario where $P_{safe} = 0.5$, there were five replans after the initial plan and there were 1865 nodes in the tree when the initial plan was developed.

The explanation for these results are as follows. In order to navigate from the start node to the end node, the agent can travel right of the static obstacle (the shorter distance according to the Euclidean metric) or the agent can travel left of the static obstacle (the longer distance according to the Euclidean metric). In the $P_{safe} = 0.1$, the expanding tree rejects nodes that have a 10% or higher PoC. These nodes are located on the right side of the static obstacle. Hence, the tree expansion favors the left side of the

Table 4. Number of samples needed for initial plan for each map

Map	PBRRT	RRT*FND
1	2874	3211
2	1347	1588
3	1066	1865
4	3289	1728
5	4514	4514

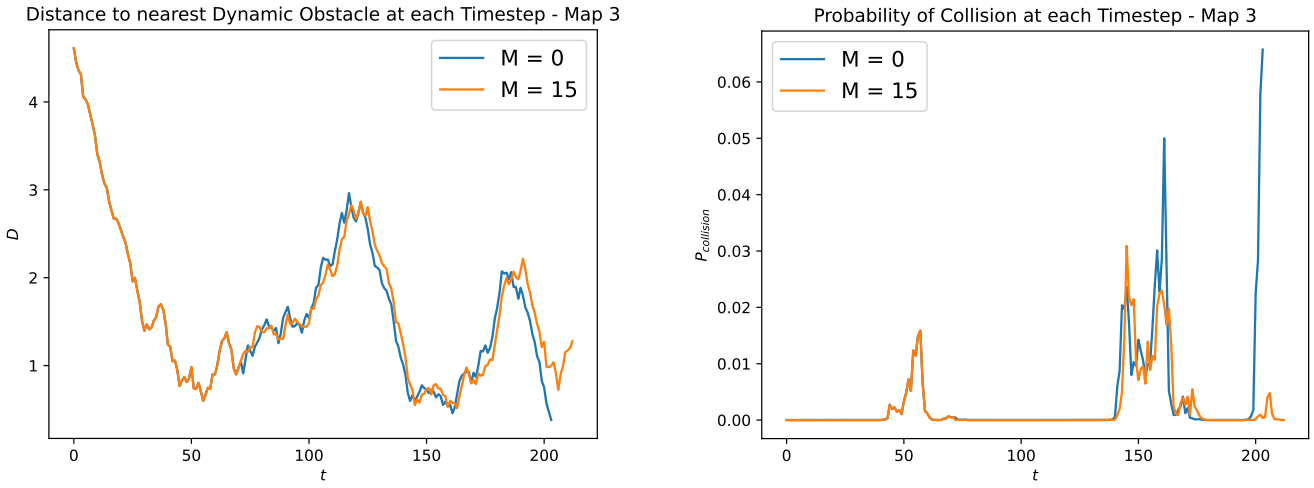
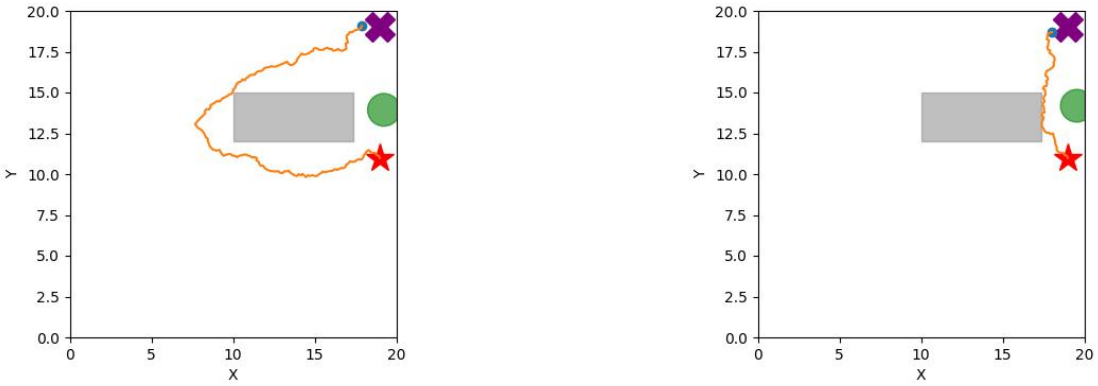


Figure 5. M ablation experiment results

static obstacle. However, when $P_{safe} = 0.5$, the percentage of nodes to the right of the static obstacle that are rejected decreases. Hence, a shorter path can be found quickly. This initial path is unlikely to be successful as the dynamic obstacle is likely to interfere with the planned path. Therefore, replanning is likely to occur.



(a) $P_{safe} = 0.1$ **(b)** $P_{safe} = 0.5$

Figure 6. Illustration of executed paths for P_{safe} ablation study.

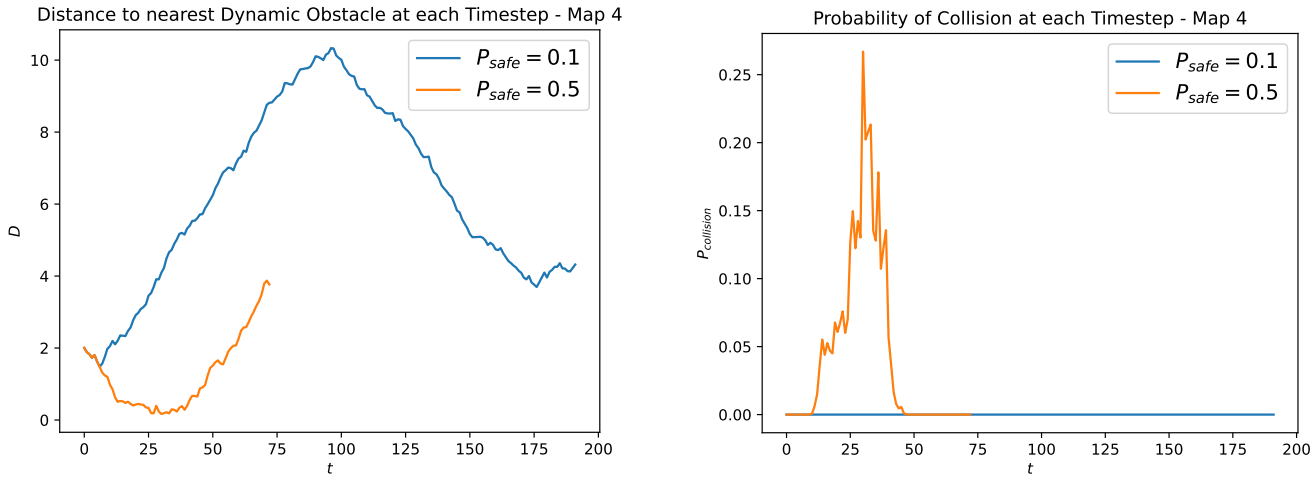


Figure 7. P_{safe} ablation experiment results

5.3.3. γ Ablation Experiment

To showcase the effect of γ on PBRRT, we re-ran Map 3 with $\gamma = 0.6$, 0.8, and 1. The DNDO and PoC graphs for the γ ablation study are shown in Fig. 8. When $\gamma = 0.6$, the robot agent needed to replan ten times while when $\gamma = 0.8$ or $\gamma = 1$, the robot agent only needed to replan three times after the initial plan.

These results are expected as when γ decreases, the agent is more nearsighted to dynamic obstacles. Hence, the agent is likely to have to replan when it comes near those dynamic obstacles that have not been provided much weight in the cost function. When $\gamma = 1$, the agent provides equal weight to nearby dynamic obstacles and far away dynamic obstacles. Hence, this explains the low number of replans and low PoC. In practice, the authors suggest using $\gamma = 1$, if one knows the entire future trajectory of the dynamic obstacles. Theoretically, there would be near zero replans in this hypothetical scenario.

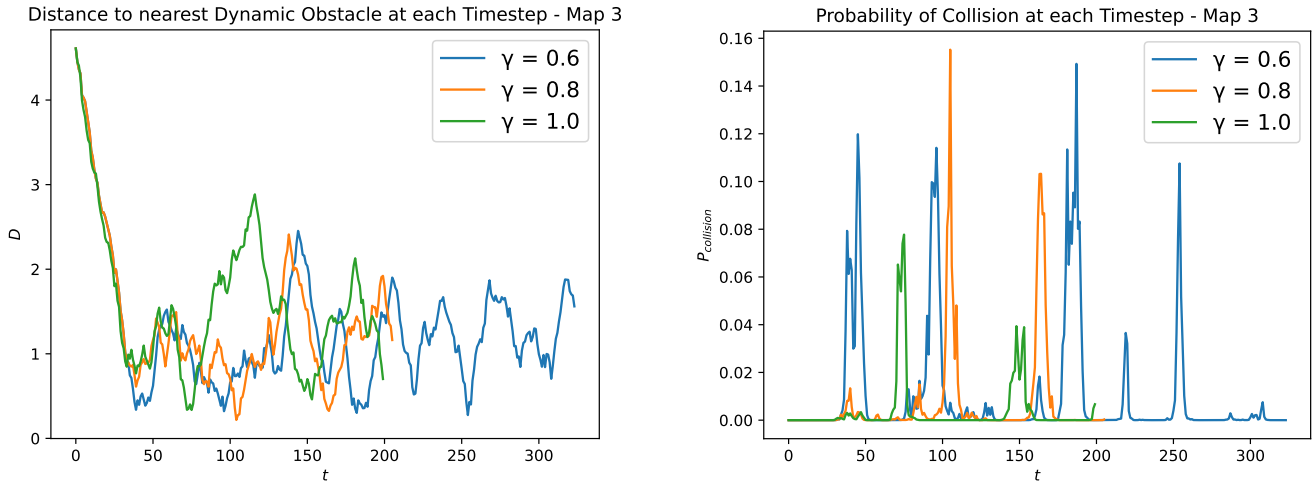


Figure 8. γ ablation experiment results

5.3.4. σ^2 Ablation Experiment

To show the effect of σ^2 on PBRRT, we re-ran Map 1 with $\sigma^2 = 0.1$, 0.5, 1.0, and 1.1. The DNDO graph for the σ^2 ablation study is provided in Fig. 9. The PoC graph is not provided here as the PoC for each of the executed path would be biased due to different a priori beliefs regarding the regions the dynamic obstacles may cover. When $\sigma^2 = 0.1$, there were no replans. When $\sigma^2 = 0.5$, there was one replan after the initial plan. When $\sigma^2 = 1.0$, there were five replans after the initial plan. When $\sigma^2 = 1.1$, there were 53 replans after the initial plan. The authors noted that when $\sigma^2 = 1.25$ or higher, PBRRT fails to return any path as the robot agent kept replanning indefinitely. The reason that the amount of replans increases as σ^2 increase is explained by the fact the robot agent is more uncertain about larger regions of the configuration space being free; hence it replans to avoid routing the planned path through these regions. With regards to Map 1, the agent struggles to pass through the region highlighted in blue in Fig. 10. If σ^2 is large, the entire region is marked off as too risky to travel through and hence the tree will not generate any nodes in that region. However, the robot agent must travel through this region in order to get to the goal region. This explains why, under large σ^2, PBRRT fails for Map 1.

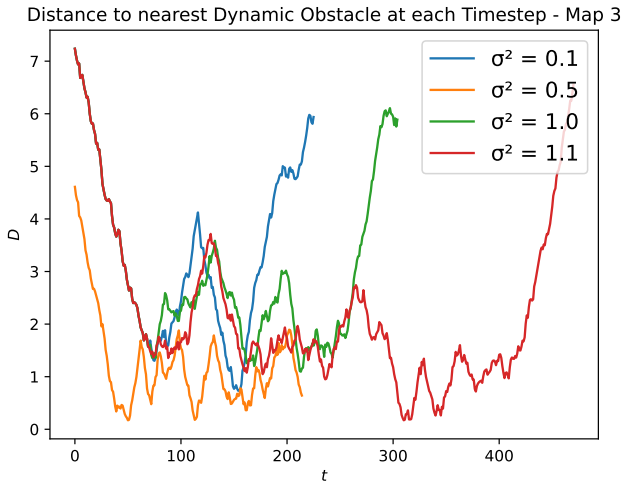


Figure 9. σ^2 ablation experiment results

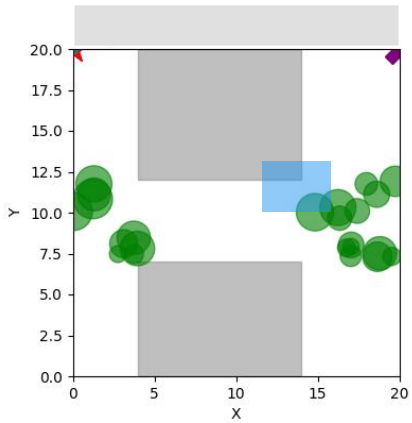


Figure 10. Region of difficulty for PBRRT under high σ^2

5.4. 3D Map Results

To demonstrate PBRRT working in higher dimensions, the authors developed Map 6. There are 45 spherical dynamic obstacles in Map 6 with radii ranging from 0.5 to 1 unit. The PBRRT DNDO and PoC graphs for Map 6 are provided in Fig. 11. The robot agent had to replan six times after the initial plan was developed. There were 6357 nodes in the tree when the initial plan was found. The end-to-end runtime for this scenario was 81.78 seconds. For illustration purposes, Fig. 12 shows the executed path from different angles.

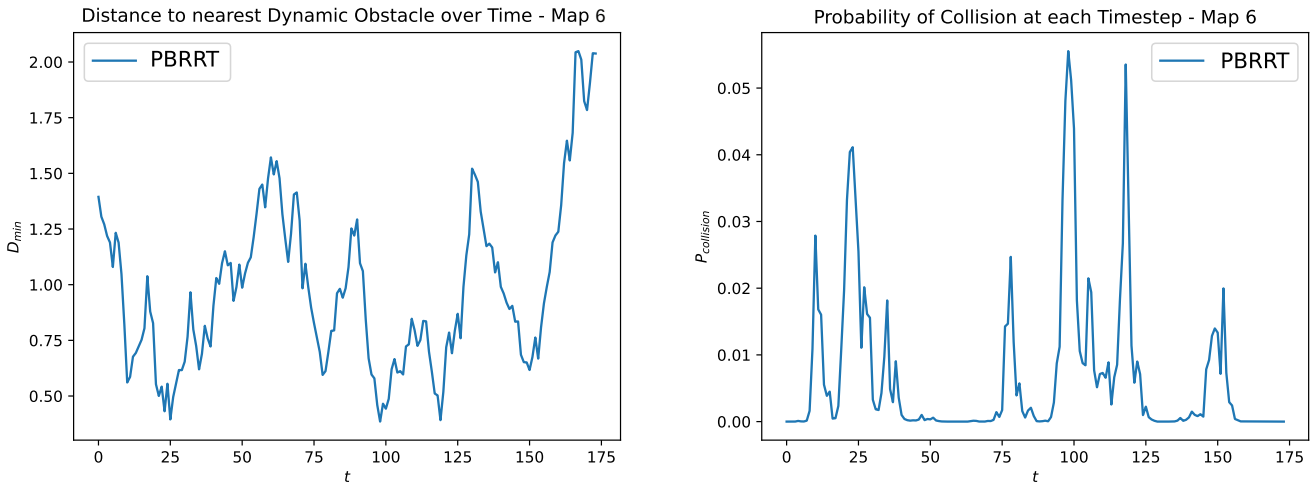


Figure 11. Map 6 DNDO and PoC graphs

6. Discussion and Future Work

The authors believe that although the proposed algorithm shows promise, there remains room for improvement. First, the planning and replanning times can be long due to the large number of collision-probability calculations required in dynamic environments. This process could potentially be parallelized using a graphics processing unit (GPU) to enable faster replanning in future work. Furthermore, the authors are interested in developing an approach that accounts for the previously planned path when replanning, in order to further reduce replanning time.

7. Conclusion

Prediction-based RRT* (PBRRT) is a novel RRT*-based algorithm for dynamic environments that focuses on predicted future obstacle positions. This work contributes to the growing field of predictive motion planning by integrating obstacle forecasting into sampling-based algorithms. The proposed approach is particularly useful for autonomous robots operating in highly dynamic environments, such as self-driving vehicles, warehouse robots, and assistive robotic systems. The main objective of the proposed algorithm is to minimize replanning attempts in dynamic environments. PBRRT adds a measure of probabilistic feasibility to the conventional RRT* cost function, avoiding potential dynamic

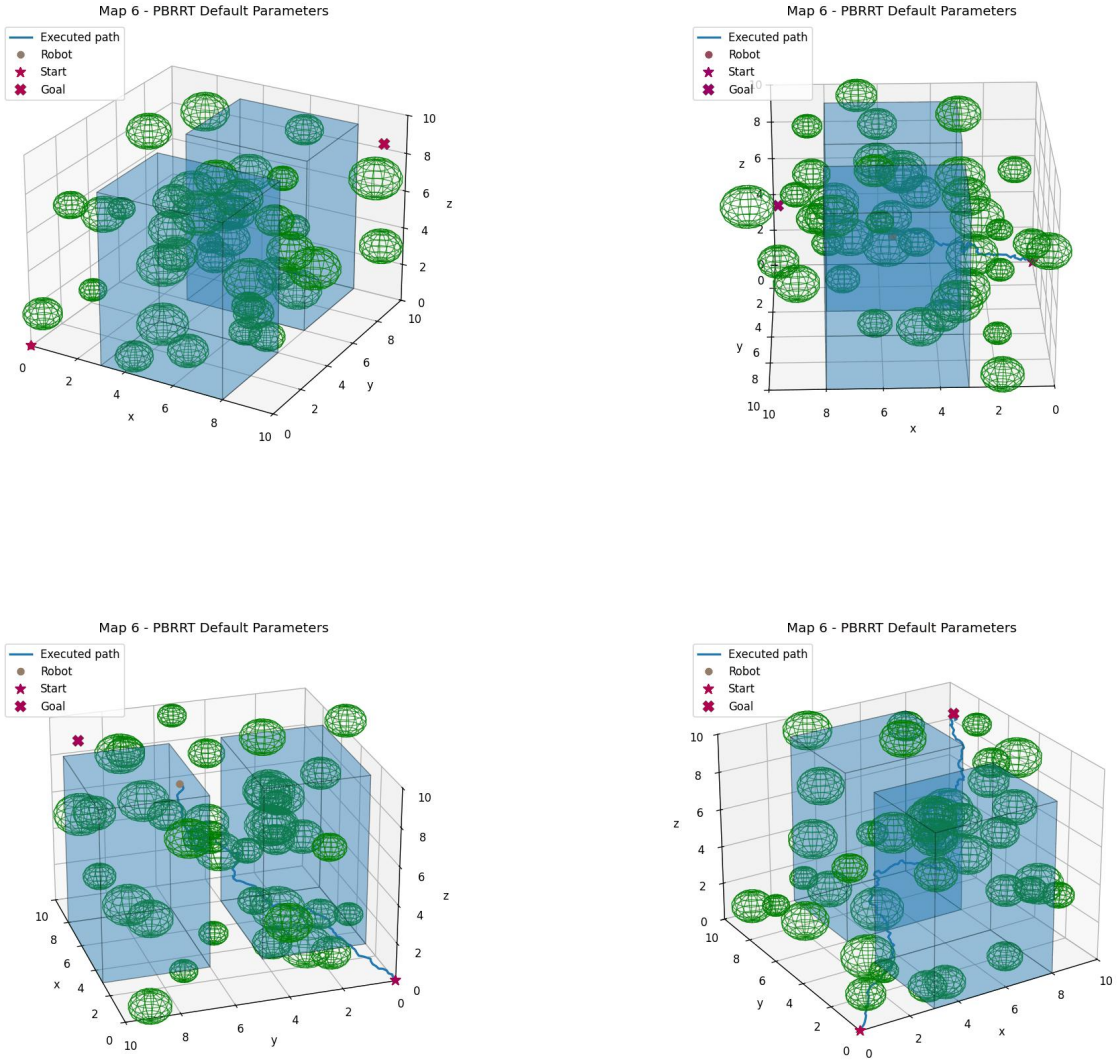


Figure 12. Snapshots of Map 6 executed path with PBRRT default parameters

obstacle collisions. Simulation results compare the proposed algorithm with other state-of-the-art motion-planning methods for dynamic environments [17, 18, 19] and show reduced replanning attempts and collision probability across all evaluated maps. Future research could explore extending PBRRT to higher-dimensional environments, integrating deep learning–based motion prediction, and optimizing computational efficiency for real-time deployment.

8. Appendix

8.1. Calculating the Probability of Collision in the Simulations

This section shows the mathematical details on how the PoC can be calculated based on the assumptions discussed in Section 5..

We define the following. Firstly, the estimated position of the k-th dynamic obstacle at discrete time i is $X_{D,k}[i] \sim \mathcal{N}(x_{D_k}[i], \sigma^2 I)$ (which is based on our assumptions in Section 5.1). The k-th dynamic obstacle has a radius of R_k. We assume the configuration of our point robot at discrete time i is $x[i]$.

8.1.1. Calculating the PoC from a Single Circular Dynamic Obstacle

Firstly, we consider the case where there is only one dynamic obstacle in the map. A collision is predicted to happen if the following equation is satisfied:

$$||X_{D,k}[i] - x[i]|| \leq R_k. \tag{7}$$

Hence, in our case calculating the PoC reduces to calculating $P(||X_{D,k}[i] - x[i]|| \leq R_k)$. Firstly, we define $Z_{D,k}[i]$ as:

$$Z_{D,k}[i] \coloneqq X_{D,k}[i] - x[i]. \tag{8}$$

Therefore

$$Z_{D,k}[i] \sim \mathcal{N}(x_{D,k}[i] - x[i], \sigma^2 I). \tag{9}$$

Next, we define $Y_{D,i}$ by scaling by σ:

$$Y \coloneqq \frac{Z_{D,k}[i]}{\sigma} \sim \mathcal{N}(\frac{x_{D,k}[i] - x[i]}{\sigma}, I). \tag{10}$$

Now we look at the squared norm:

$$\zeta \coloneqq ||Y||^2 = \frac{||X_{D,k}[i] - x[i]||^2}{\sigma^2}. \tag{11}$$

This random variable ζ has a non-central chi–square distribution [21] with n degrees of freedom and non-centrality parameter λ such that:

$$\lambda = \frac{||x_{D,k}[i] - x[i]||^2}{\sigma^2}. \tag{12}$$

Hence:

$$P(||X_{D,k}[i]-x[i]|| \leq R_k) = P(||X_{D,k}[i]-x[i]||^2 \leq R_k^2) = P(\sigma^2\zeta \leq R_k^2) = P(\zeta \leq \frac{R_k^2}{\sigma^2}) \\ = F_{\chi_d'^2}(\lambda)\left(\frac{R^2}{\sigma^2}\right). \tag{13}$$

Where $F\chi_d'^2(\lambda)$ is the cumulative distribution function (CDF) of the non-central chi-square distribution. We use the Scipy library to compute this CDF [22].

8.1.2. Calculating the PoC from Multiple Circular Dynamic Obstacles

Let's denote the event that $||X_{D,k}[i]-x[i]|| \leq R_k$ as A_k. If we have a map with Q dynamic obstacles, the probability that the robot will collide with at least one dynamic obstacle is [23]:

$$P\left(\bigcup_{k=1}^{Q} A_k\right) = \sum_{k=1}^{Q} P(A_k) - \sum_{k<j} P(A_k \cap A_j) + \cdots + (-1)^{Q+1} P\left(\bigcap_{k=1}^{Q} A_k\right). \tag{14}$$

Hence, we can use (14) to calculate the PoC at some node with configuration $x[i]$.

Supplementary data

The authors confirm that the supplementary data are available within this article.

Data availability statement

The implementation of the work and data generated in this study are available at https://github.com/amrmarey15/PBRRT.

Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors used generative AI tools only to improve language and readability. The authors take full responsibility for the content of the manuscript.

Acknowledgments

This research was supported by the Canada Foundation for Innovation (CFI), the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Canadian Institutes of Health Research (CIHR), Alberta Innovates (RES0062090), and the Government of Alberta's grant to Centre for Autonomous Systems in Strengthening Future Communities (RES0047163).

Author's contribution

Conceptualization, A.M., Q.Z. and M.T.; methodology, A.M., Q.Z. and M.T.; software, A.M.; validation, A.M.; formal analysis, A.M.; investigation, A.M., Q.Z. and M.T.; resources, Q.Z. and M.T.; data

curation, A.M.; writing—original draft preparation, A.M.; writing—review and editing, Q.Z. and M.T.; visualization, A.M.; supervision, Q.Z. and M.T.; project administration, Q.Z. and M.T.; funding acquistion, Q.Z. and M.T. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interests

No conflict of interests.

Ethical statement

Ethical Statement is not required.

References

[1] Zhao Z, Cheng S, Ding Y, Zhou Z, Zhang S, *et al.* A Survey of Optimization-Based Task and Motion Planning: From Classical to Learning Approaches. *IEEE/ASME Transactions on Mechatronics* 2024 pp. 1–27. 10.1109/TMECH.2024.3452509.

[2] Zucker M, Ratliff N, Dragan AD, Pivtoraiko M, Klingensmith M, *et al.* CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research* 2013 32(9-10):1164–1193. 10.1177/0278364913488805.

[3] Kalakrishnan M, Chitta S, Theodorou E, Pastor P, Schaal S. STOMP: Stochastic trajectory optimization for motion planning. In 2011 IEEE International Conference on Robotics and Automation. 2011 pp. 4569–4574. 10.1109/ICRA.2011.5980280.

[4] Koenig S, Likhachev M. D*lite. In Eighteenth National Conference on Artificial Intelligence, USA: American Association for Artificial Intelligence2002 p. 476–483.

[5] Dolgov D, Thrun S, Montemerlo M, Diebel J. Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments. *The International Journal of Robotics Research* 2010 29(5):485–501. 10.1177/0278364909359210.

[6] *Planning Algorithms*, Cambridge University Press2006.

[7] Vesentini F, Muradore R, Fiorini P. A survey on Velocity Obstacle paradigm. *Robot. Auton. Syst.* 2024 174(C). 10.1016/j.robot.2024.104645.

[8] Luo S, Schomaker L. Reinforcement Learning in Robotic Motion Planning by Combined Experience-based Planning and Self-Imitation Learning, 2023.

[9] Li X, Wang K, Wang W, Li Y. A multiple object tracking method using Kalman filter. In The 2010 IEEE International Conference on Information and Automation. 2010 pp. 1862–1866. 10.1109/ICINFA.2010.5512258.

[10] Jaward M, Mihaylova L, Canagarajah N, Bull D. Multiple object tracking using particle filters. In 2006 IEEE Aerospace Conference. 2006 pp. 8 pp.–. 10.1109/AERO.2006.1655926.

[11] Marinello N, Proesmans M, Van Gool L. TripletTrack: 3D Object Tracking using Triplet Embeddings and LSTM . In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Los Alamitos, CA, USA: IEEE Computer Society2022 pp. 4499–4509. 10.1109/CVPRW56347.2022.00496.

[12] Tian S, Duan M, Deng J, Luo H, Hu Y. MF-Net: A Multimodal Fusion Model for Fast Multi-Object Tracking. *IEEE Transactions on Vehicular Technology* 2024 73(8):10948–10962. 10.1109/TVT.2024.3375457.

[13] LaValle SM. Rapidly-exploring random trees : a new tool for path planning. *The annual research report* 1998 .

[14] Karaman S, Frazzoli E. Sampling-based Algorithms for Optimal Motion Planning. *CoRR* 2011 abs/1105.1186.

[15] Chen L, Mantegh I, He T, Xie W. Fuzzy Kinodynamic RRT: a Dynamic Path Planning and Obstacle Avoidance Method. In 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece2020 pp. 188–195. 10.1109/ICUAS48674.2020.9213964.

[16] Cui B, Cui R, Yan W, Wang Y, Zhang S. RT-RRT: Reverse Tree Guided Real-Time Path Planning/Replanning in Unpredictable Dynamic Environments. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates2024 pp. 5380–5387. 10.1109/IROS58592.2024.10802722.

[17] Adiyatov O, Varol HA. A novel RRT*-based algorithm for motion planning in Dynamic environments. In 2017 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan2017 pp. 1416–1421. 10.1109/ICMA.2017.8016024.

[18] Naderi K, Rajamäki J, Hämäläinen P. RT-RRT*: a real-time path planning algorithm based on RRT*. In Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, New York, NY, USA: Association for Computing Machinery2015, MIG '15 p. 113–118. 10.1145/2822013.2822036.

[19] Armstrong D, Jonasson A. AM-RRT*: Informed Sampling-based Planning with Assisting Metric. In 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China2021 pp. 10093–10099. 10.1109/ICRA48506.2021.9561604.

[20] Zhang D, Maei H, Wang X, Wang Y. Deep Reinforcement Learning for Visual Object Tracking in Videos. *CoRR* 2017 abs/1701.08936.

[21] Patnaik PB. The Non-Central χ^2 - and F-Distribution and their Applications. *Biometrika* 1949 36(1/2):202–232.

[22] Community TS. scipy.stats.ncx2 — SciPy v1.17.0 Manual, 2025. Accessed: 2026-02-22.

[23] *How to Count: An Introduction to Combinatorics and Its Applications*, Springer Cham2015, 1st edn. 10.1007/978-3-319-13844-2.