Probabilistically Safe Avoidance of Dynamic Obstacles with Uncertain Motion Patterns

Brandon Luders, Georges Aoude, Joshua Joseph, Nicholas Roy, and Jonathan P. How

Abstract— **This paper presents a real-time path planning algorithm which can guarantee probabilistic feasibility for autonomous robots subject to process noise and an uncertain environment, including dynamic obstacles with uncertain motion patterns. The key contribution of the work is the integration of a novel method for modeling dynamic obstacles with uncertain future trajectories. The method, denoted as RR-GP, uses a learned motion pattern model of the dynamic obstacles to make long-term predictions of their future paths. This is done by combining the flexibility of Gaussian processes (GP) with the efficiency of RRT-Reach, a sampling-based reachability computation method which ensures dynamic feasibility. This prediction model is then utilized within chance-constrained rapidly-exploring random trees (CC-RRT), which uses chance constraints to explicitly achieve probabilistic constraint satisfaction while maintaining the computational benefits of sampling-based algorithms. With RR-GP embedded in the CC-RRT framework, theoretical guarantees can be demonstrated for linear systems subject to Gaussian uncertainty, though the extension to nonlinear systems is also considered. Simulation results show that the resulting approach can be used in real-time to efficiently and accurately execute safe paths.**

I. Introduction

To operate safely in stochastic environments, it is crucial for agents to be able to plan in real time in the presence of uncertainty. Indeed, the stochasticity of such environments often precludes the guaranteed existence of safe, collision-free paths. Instead, this work considers probabilistically safe planning, in which paths must be able to satisfy all constraints with a user-mandated minimum probability.

A major challenge in utilizing trajectory prediction algorithms is addressing the multiple sources of uncertainty in the environment, often classified between environment sensing (ES) and environment predictability (EP) [1]. Under this partition, ES uncertainties might be attributable to imperfect sensor measurements and/or incomplete knowledge of the environment, while EP uncertainties, the focus of this paper, consider the typically limited knowledge of the future state of the environment. While probabilistic planning frameworks can readily admit dynamic obstacles, such objects are often very difficult to model in real-world domains. For example, for a car to reliably traverse a busy intersection, it must have some understanding of how other vehicles typically cross that intersection. Even under the assumptions of perfect sensors and complete knowledge of the current environment, predicting long-term trajectories of other mobile agents remains a difficult problem.

In Ref. [2], the authors surveyed several existing techniques for long-term trajectory prediction, and concluded that pattern-based approaches, despite some existing limitations, are typically the most suitable solution. There are two main techniques that fall into this category: a) discrete state-space, and b) clustering-based [3]. In discrete state-space techniques, the motion model is typically based on learned Markov chains, while clustering-based techniques group previously-observed trajectories into clusters which are each represented by a trajectory prototype.

Both classes of pattern-based approaches have proven popular in solving long-term prediction problems for mobile agents [3], [4]. However, discrete state-space techniques can suffer from over-fitting or under-fitting problems due to space discretization issues, unlike clustering-based techniques. In our previous work [5], [6], we presented a Bayesian nonparametric approach to modeling motion patterns that is well-suited to modeling dynamic obstacles with unknown motion patterns. This nonparametric model, a mixture of Gaussian process (GP), generalizes well from small amounts of data and allows the model to capture complex trajectories as more data is seen. However, in practice, GPs suffer from two interconnected shortcomings: their high computational cost and their inability to embed static feasibility or vehicle dynamical constraints. To handle both problems simultaneously, recent work introduced the RR-GP algorithm, a trajectory prediction solution using Bayesian nonparametric reachability trees [2] built via rapidly-exploring random trees (RRTs) [7]. This algorithm improves on the original GP algorithm in both prediction accuracy and computation time, making it well suited for real-time applications in prediction problems with poorly understood trajectory patterns.

The algorithm presented in this paper has similarities with [4], which uses Gaussian processes to model moving obstacles in an RRT path planner. However, unlike this work, [4] relies solely on Gaussian processes, which can lead to less precise prediction, especially when available data is sparse. RR-GP also embeds dynamic feasibility and prior knowledge of the environment, leading to better results than GP-only approaches. Finally, the planner in [4] uses heuristics to assess the safety of generated paths, while the planner developed in this paper uses a principled approach to achieve probabilistic safety.

B. Luders, Ph. D. Candidate, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, luders@mit.edu

G. Aoude, Ph. D. Candidate, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, gaoude@mit.edu

J. Joseph, Ph. D. Candidate, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, jmjoseph@mit.edu

N. Roy, Associate Professor of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, nickroy@mit.edu

J. P. How, Richard C. Maclaurin Professor of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, jhow@mit.edu

This paper presents a real-time path planning framework which guarantees probabilistic feasibility for autonomous agents subject to both process noise and an uncertain environment, particularly dynamic obstacles with uncertain motion patterns. The planning framework chosen for this work is chance-constrained RRTs (CC-RRT) [8], based on the chance constraint formulation of [9] for linear systems subject to Gaussian uncertainty. CC-RRT uses chance constraints to evaluate the risk of constraint violation at each timestep, embedding uncertainty directly within the planner. This approach maintains the benefits of sampling-based algorithms, particularly the fast identification of feasible solutions for complex motion planning problems. While several apparoaches have been previously proposed for path planning with probabilistic constraints, our approach does not require the use of MILP/SOCP optimizations [9], [10] or particle-based approximations [11]–[13], each of which can severely limit real-time applicability.

The CC-RRT algorithm has been demonstrated to effectively and efficiently guarantee probabilistic feasibility, even in the presence of dynamic obstacles, as long as the future state distributions are known [8]. This is effectively integrated here with the RR-GP algorithm, which can provide a likelihood and state distribution for each possible behavior of a dynamic obstacle at each future timestep. The CC-RRT formulation is revisited for the case of dynamic obstacles modelled via RR-GP, showing that probabilistic feasibility can still be guaranteed. Simulation results demonstrate the effectiveness of this approach in enabling agents to avoid dynamic threats with high likelihood.

II. Problem Statement

Consider a discrete-time linear time-invariant (LTI) system with process noise,

$$x_{t+1} = Ax_t + Bu_t + w_t, \tag{1}$$
$$x_0 \sim \mathcal{N}(\hat{x}_0, P_{x_0}), \tag{2}$$
$$w_t \sim \mathcal{N}(0, P_{w_t}), \tag{3}$$

where $x_t \in \mathbb{R}^{n_x}$ is the state vector, $u_t \in \mathbb{R}^{n_u}$ is the input vector, and $w_t \in \mathbb{R}^{n_x}$ is a disturbance vector acting on the system; $\mathcal{N}(\hat{a}, P_a)$ represents a random variable whose probability distribution is Gaussian with mean $\hat{a}$ and covariance P_a. The i.i.d. random variables w_t are unknown at current and future time steps, but have the known probability distribution (3) ($P_{w_t} \equiv P_w \ \forall\ t$).

There are also constraints acting on the system state and input. These constraints are assumed to take the form

$$x_t \in \mathcal{X}_t \equiv \mathcal{X} - \mathcal{X}_{t1} - \cdots - \mathcal{X}_{tB}, \tag{4}$$
$$u_t \in \mathcal{U}, \tag{5}$$

where $\mathcal{X}, \mathcal{X}_{t1}, \ldots, \mathcal{X}_{tB} \subset \mathbb{R}^{n_x}$ are convex polyhedra, $\mathcal{U} \subset \mathbb{R}^{n_u}$, and the $-$ operator denotes set subtraction. The set $\mathcal{X}$ defines a set of time-invariant convex constraints acting on the state, while $\mathcal{X}_{t1}, \ldots, \mathcal{X}_{tB}$ represent B convex obstacles to be avoided. For each obstacle, the shape and orientation are assumed to be known, while the placement is uncertain. This is represented as

$$\mathcal{X}_{tj} = \mathcal{X}_j^0 + c_{tj}, \ \forall\ j \in \mathbb{Z}_{1,B}, \ \forall\ t, \tag{6}$$
$$c_{tj} \sim p(c_{tj}) \ \forall\ j \in \mathbb{Z}_{1,B}, \ \forall\ t, \tag{7}$$

where the $+$ operator denotes set translation and $\mathbb{Z}_{a,b}$ represents the set of integers between a and b inclusive. In this model, $\mathcal{X}_j^0 \subset \mathbb{R}^{n_x}$ is a convex polyhedron of known, fixed shape, while $c_{tj} \in \mathbb{R}^{n_x}$ is a possibly time-varying translation, represented by the probability distribution $p(c_{tj})$. This can be used to represent dynamic obstacles whose future state distributions are known, as is done in Section V.

The primary objective of the planning problem is to reach the goal region $\mathcal{X}_{\text{goal}} \subset \mathbb{R}^{n_x}$ in minimum time, while ensuring the constraints (4)-(5) are satisfied at each time step $t \in \{0, \ldots, t_{\text{goal}}\}$ *with probability of at least* p_{safe}. In practice, since there is uncertainty in the state, we assume it is sufficient for the distribution mean to reach the goal region $\mathcal{X}_{\text{goal}}$. A secondary objective may be to avoid some undesirable behaviors, such as proximity to constraint boundaries.

Note that we are assessing probabilistic feasibility of the constraints at each time step, rather than over the entire path. Because the uncertainty at each timestep is correlated, due to the dynamics (1)-(3), one *cannot* approximate the probability of path feasibility by assuming independence and multiplying the probabilities of feasibility at each timestep. Instead, assessment of path feasibility requires the evaluation of a complex nested integral, necessitating the use of approximate solutions even under the assumption of Gaussian uncertainty. While such approximations do exist [13], [14], they require significant computation for most problems of interest. For the applications being considered, real-time identification of feasible paths is paramount; thus in this work we focus on feasibility at each timestep, using the efficient assessment of risk afforded by CC-RRT [8]. Because the constraints are made more restrictive as the lower bound on probabilistic feasibility is increased, a positive correlation is expected between feasibility at each timestep and feasibility of the whole path; this is shown to be the case in Section VI.

A. Motion Pattern

We define a motion pattern as a mapping from locations to a distribution over velocities (trajectory derivatives).[1] Given an agent's current position (x_t, y_t) and a trajectory derivative $(\frac{\Delta x_t}{\Delta t}, \frac{\Delta y_t}{\Delta t})$, its predicted next position (x_{t+1}, y_{t+1}) is $(x_t + \frac{\Delta x_t}{\Delta t}\Delta t, y_t + \frac{\Delta y_t}{\Delta t}\Delta t)$. Thus, modeling trajectory derivatives is equivalent to modeling trajectories. In addition to being indepdendent of the lengths and discretizations of the trajectories, modeling motion patterns as flow fields also allows us to group trajectories sharing key characteristics. For example, a single motion pattern can capture all the paths that an agent might take from different starting points to a single ending location.

[1]The choice of Δt determines the scales at which we can expect to predict an agent's next position well, making the trajectory derivative more useful than instantaneous velocity.

B. Mixtures of Motion Patterns

Our finite mixture model defines a distribution over the ith observed trajectory t^i.[2] This distribution is written as

$$p(t^i) = \sum_{j=1}^{M} p(b_j) p(t^i | b_j). \tag{8}$$

where b_j is motion pattern j and $p(b_j)$ is its prior probability. Since we are interested in vehicles traveling along a known road network we can assume the number of motion patterns, M, is known *a priori.*

III. Motion Model

We define the motion model as the mixture of weighted motion patterns (Eq. 8). Each motion pattern is weighted by its probability and is modeled by a pair of Gaussian processes mapping (x, y) locations to distributions over trajectory derivatives $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$. This section provides a brief overview of the motion model; please consult [2], [6], [15] for more details.

A. Gaussian Process Motion Patterns

This section describes the model for $p(t^i|b_j)$ from Eq. 8, the probability of trajectory t^i given motion pattern b_j. This model is the distribution over trajectories expected for a particular mobility pattern. An example distribution over trajectories might be a linear model with Gaussian noise, of the form $x_{t+1} \sim \mathcal{N}(A_j x_t, \sigma_j)$. Unfortunately, this model is too simplistic to capture the dynamics of the variety of expected motion patterns. Another common approach, discrete Markov models, is ill suited to model mobile agents in the types of real-world domains of interest [6], [15]. They are inherently plagued by the decision of how to perform the state discretization.

To capture the variety of trajectories that may be encountered, an expressive representation (a fine discretization) is necessary, resulting in a model that requires a large amount of training data. In real-world domains where collecting a large data set is costly or impossible, these models can become prone to over-fitting. To prevent over-fitting, a coarser discretization may be used, but this then risks being unable to accurately capture the agent's dynamics.

This work uses Gaussian processes (GP) as the model for motion patterns; although they come at significant mathematical and computational cost, they provide a natural balancing between generalization in regions with sparse data and preventing under-fitting in regions of dense data [15], [16]. Observations of an agent's trajectory are discrete measurements from its continuous path through space. A GP [17] places a distribution over functions, serving as a non-parametric form of interpolation between these discrete measurements. Gaussian process models are extremely robust to unaligned, noisy measurements, and are well-suited for modeling the continuous paths underlying potentially non-uniformly sampled time-series samples.

[2]Throughout the paper a t with a superscript, such as t^i, refers to a trajectory, whereas a t without a superscript refers to a time value.

The first term inside the summation of Eq. (8), $p(b_j)$, is the prior probability of motion pattern b_j. Given an agent's trajectory t^i, the posterior probability of motion pattern is

$$p(b_j|t^i) \propto p(t^i|b_j)p(b_j), \tag{9}$$

where $p(t^i|b_j)$ is the probability of trajectory t^i under motion pattern b_j. This distribution, $p(t^i|b_j)$, is computed by

$$\begin{aligned} p(t^i|b_j) = \prod_{t=0}^{L^i} p\left(\frac{\Delta x_t}{\Delta t} \middle| x^i_{0:t}, y^i_{0:t}, \{t^k : z_k = j\}, \theta^{GP}_{x,j} \right) \\ \cdot\, p\left(\frac{\Delta y_t}{\Delta t} \middle| x^i_{0:t}, y^i_{0:t}, \{t^k : z_k = j\}, \theta^{GP}_{y,j} \right), \end{aligned} \tag{10}$$

where L^i is the length of trajectory i, z_k indicates the motion pattern trajectory t^k is assigned to, and $\theta^{GP}_{x,j}$ and $\theta^{GP}_{y,j}$ are the hyperparameters of the Gaussian process for pattern b_j.

A motion pattern's GP is specified by a set of mean and covariance functions. We describe the mean functions as $E[\frac{\Delta x}{\Delta t}] = \mu_x(x, y)$ and $E[\frac{\Delta y}{\Delta t}] = \mu_y(x, y)$, and implicitly set both of them to initially be zero everywhere (for all x and y) by our choice of parametrization of the covariance function. This encodes the prior bias that, without any additional knowledge, we expect the target to stay in the same place.

We denote the covariance function of the x-direction as $K_x(x, y, x', y')$, which describes the correlation between trajectory derivatives at two points, (x, y) and (x', y'). Given locations $(x_1, y_1, .., x_k, y_k)$, the corresponding trajectory derivatives $(\frac{\Delta x_1}{\Delta t}, .., \frac{\Delta x_k}{\Delta t})$ are jointly distributed according to a Gaussian with mean $\{\mu_x(x_1, y_1), .., \mu_x(x_k, y_k)\}$ and covariance Σ, where the cell $\Sigma_{ij} = K_x(x_i, y_i, x_j, y_j)$. In this work, we use the squared exponential covariance function

$$\begin{aligned} K_x(x, y, x', y') = \sigma_x^2 \exp\left(-\frac{(x - x')^2}{2{w_x}^2} - \frac{(y - y')^2}{2{w_y}^2} \right) \\ + \sigma_n^2 \delta(x, y, x', y'), \end{aligned} \tag{11}$$

where $\delta(x, y, x', y') = 1$ if $x = x'$ and $y = y'$ and zero otherwise. The exponential term above encodes that similar trajectories should make similar predictions and the length-scale parameters w_x and w_y normalize for the scale of the data. The σ_n-term represents within-point variation (e.g., due to noisy measurements); the ratio of σ_n and σ_x weights the relative effects of noise and influences from nearby points. We use $\theta^{GP}_{x,j}$ to refer to the set of hyperparameters σ_x, σ_n, w_x, and w_y associated with motion pattern b_j.[3]

For a GP over trajectory derivatives trained with tuples $(x_k, y_k, \frac{\Delta x_k}{\Delta t})$, the predictive distribution over the trajectory derivative $\frac{\Delta x}{\Delta t}^*$ for a new point (x^*, y^*) is given by

$$\begin{aligned} \mu_{\frac{\Delta x}{\Delta t}^*} &= K_x(x^*,y^*,X,Y) K_x(X,Y,X,Y)^{-1} \frac{\Delta X}{\Delta t}, \\ \sigma^2_{\frac{\Delta x}{\Delta t}^*} &= K_x(x^*,y^*,X,Y) K_x(X,Y,X,Y)^{-1} K_x(X,Y,x^*,y^*), \end{aligned} \tag{12}$$

where the expression $K_x(X, Y, X, Y)$ is shorthand for the covariance matrix Σ with terms $\Sigma_{ij} = K_x(x_i, y_i, x_j, y_j)$. The equations for $\frac{\Delta y}{\Delta t}^*$ are defined equivalently.

[3]We described the kernel for two dimensions, but it can be easily generalized to more.

IV. RR-GP Trajectory Prediction Algorithm

Section III outlined the approach of using GP mixtures to model mobility patterns. In practice, GPs suffer from two interconnected shortcomings: their high computational cost and their inability to embed static feasibility or vehicle dynamics constraints. Very dense training data may elevate this feasibility problem by capturing, in great detail, the environment configuration and physical limitations of the vehicle. Unfortunately, the computation time for predicting future trajectories using the resulting GPs would suffer significantly, rendering the motion model unusable for real-time application.

To handle both of these problems simultaneously, we developed RR-GP, a trajectory prediction solution using Bayesian nonparametric reachability trees [2]. RR-GP augments RRT-Reach [18], a reachability based method which creates dense, feasible trajectories from sparse samples using the GP mixture model (Section III). The developed approach is reviewed below.

A. High Level Architecture

Figure 1 shows the high level architecture of the RR-GP solution approach. RR-GP is based on two main components: 1) an intent predictor based on the GP mixture model (Section III) and 2) a trajectory generator based on the RRT-Reach algorithm. The intention predictor uses the history of sensor position measurements $(x_{0:t}, y_{0:t})$, along with a set of typical GP motion patterns b_j, $j \in \{1, \ldots, M\}$, to produce an intent distribution which is given to the trajectory generation component. The inputs to the trajectory generator are the aforementioned intent information, a dynamic model of the target vehicle, a map of the environment, and a sparse distribution of the future positions of the target vehicle using (12). The trajectory generator uses closed-loop RRT [19] to grow a separate tree of smooth trajectories for each motion pattern, thus embedding dynamical feasibility and collision avoidance. The resulting trees produce an improved probabilistic prediction of the future trajectories of the target vehicle. Note that to limit the scope of the problem to uncertainty in predictability, the sensors measurements are assumed to be noise-free, despite our motion pattern representation being robust to noisy measurements.

B. Multi-Tree RR-GP Algorithm

A key step of RR-GP is the expansion algorithm performed on each tree. It is illustrated in Figure 2 and detailed in [2]. This section presents the complete RR-GP algorithm, also called multi-tree RR-GP, because it extends the single-tree RR-GP expansion algorithm to handle multiple motion patterns for the target vehicle.

We denote the time duration of the prediction problem and the prediction time horizon as T and T_h, respectively; both are measured in seconds throughout the paper. The value of T_h is problem specific, and depends on the time length of the training data. For example, in a threat assessment problem for road intersections, T_h might be on the order of 5 seconds [20]. The RR-GP algorithm updates its measurement of the target vehicle every dt seconds; this value is chosen such that it ensures that the inner loop (lines 6-9) of the Algorithm 1 reaches completion before the next measurement update. Finally, the time step size for the low-level controller is equal to δt seconds. A low δt signifies a higher precision of the predicted trajectories, with a trade-off of slightly larger computation times.

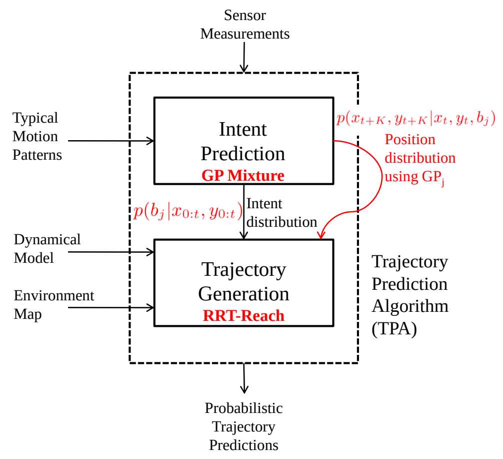

Sensor Measurements
Typical Motion Patterns
Intent Prediction
GP Mixture
$p(x_{t+K}, y_{t+K} | x_t, y_t, b_j)$
Position distribution using GP$_j$
$p(b_j | x_{0:t}, y_{0:t})$
Intent distribution
Dynamical Model
Environment Map
Trajectory Generation
RRT-Reach
Trajectory Prediction Algorithm (TPA)
Probabilistic Trajectory Predictions

Fig. 1. High-level architecture for RR-GP.

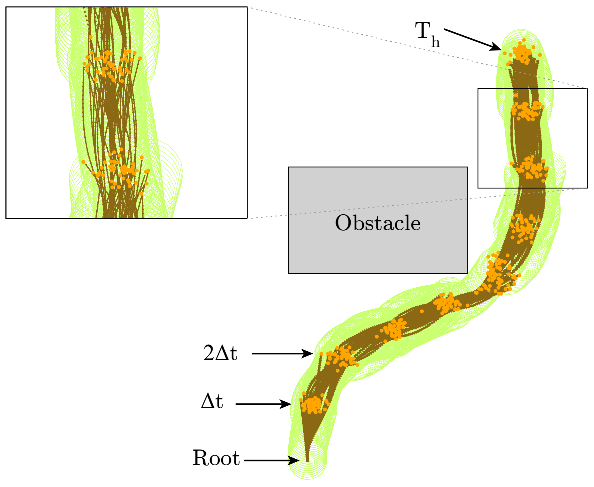


Fig. 2. Illustration of the single-tree RR-GP expansion algorithm. For more details about the algorithm, please refer to [2].

The inputs to the RR-GP algorithm are the set of GP motion patterns and the initial probability distribution over the motion patterns. This prior knowledge is typically proportional to the size of each motion pattern. In line 4, the position of the target vehicle is measured. Then, the probability that the vehicle trajectory belongs to each of the M motion patterns is updated. For each motion pattern (in parallel), line 7 grows a single-tree $\mathcal{T}_{GP}^{j}$ rooted at the current position of the target vehicle using the RR-GP single-tree algorithm [2]. In line 8, the means and variances of the predicted positions of the target vehicle $(\hat{x}(t), \hat{y}(t))$ are computed for each time step τ using position and time information of the nodes and edges of the single-tree output; note that $\tau \in [t + \delta t, t + 2\delta t, \ldots, t + T_h]$. This process can also be parallelized for the different motion patterns, since there is no

Algorithm 1 RR-GP, Multi-Tree Trajectory Prediction

```
1:  Inputs: GP motion pattern $b_j$; $p(b_j(0))\ \ \forall j \in [1,\dots,M]$
2:  $t \leftarrow 0$
3:  while $t < T$ do
4:      Measure target vehicle position $(x(t), y(t))$
5:      Update probability of each motion pattern $p(b_j(t)|x_{0:t}, y_{0:t})$ using Eq. (9)
6:      for each motion pattern $b_j$ do
7:          Grow a single $\mathcal{T}^j_{GP}$ tree rooted at $(x(t), y(t))$ using $b_j$ (Single-Tree RR-GP Algorithm)
8:          Using $\mathcal{T}^j_{GP}$, compute means and variances of predicted distribution $(\hat{x}_j(\tau), \hat{y}_j(\tau))$,
            $\forall\tau \in [t+\delta t, t+2\delta t, \dots, t+T_h]$
9:      end for
10:     $p(\hat{x}(\tau), \hat{y}(\tau)) \leftarrow \sum_j p(\hat{x}_j(\tau), \hat{y}_j(\tau))p(b_j(t)|x_{0:t}, y_{0:t})\ \ \forall\tau$
11:     $t \leftarrow t + dt$
12: end while
```

message passing between the RR-GP tree growth operations for each tree. Line 10 combines the position predictions from the single-tree RR-GP outputs into one distribution by incorporating the updated motion pattern probabilities b_j into the position distribution of the target vehicle. This computation is performed for all times τ, resulting in a probability distribution on the future trajectories of the target vehicle based on a mixture of GPs.

Since the RR-GP tree is grown at a higher rate compared to the original GP learning phase, the resulting distribution is generated at increments of $\delta t << \Delta t$. As shown in [2], the result is a significant improvement of the accuracy of the prediction compared to traditional GP algorithms, without a deterioration of the computation times, which makes RR-GP a suitable solution for real-time trajectory prediction.

V. CC-RRT WITH INTEGRATED RR-GP

This section integrates the RR-GP model for uncertain, dynamic obstacles established in previous sections into a probabilistic planning framework, CC-RRT, to identify trajectories which can probabilistically avoid those obstacles. First, the CC-RRT formulation is reviewed under the assumption that each obstacle's uncertainty is modeled by a single Gaussian; the simple extension to the multi-Gaussian formulation yielded by RR-GP follows.

A. Review of Online CC-RRT

In this section, the uncertainty of each obstacle is assumed to be represented by a single Gaussian:

$$c_{tj} \quad \sim \quad \mathcal{N}(\hat{c}_{jt}, P_{c_{jt}})\ \forall\ j \in \mathbb{Z}_{1,B},\ \forall\ t. \tag{13}$$

Given a sequence of inputs $u_0, \dots, u_{N-1}$, under the assumptions of linear dynamics and Gaussian uncertainty, the distribution of the state x_t (represented as the random variable X_t) can be shown to be Gaussian [9]:

$$P(X_t|u_0, \dots, u_{N-1}) \quad \sim \quad \mathcal{N}(\hat{x}_t, P_{x_t})\ \ \forall\ t \in \mathbb{Z}_{0,N},$$

where N is some time step horizon. The mean $\hat{x}_t$ and covariance P_{x_t} can be updated implicitly using the relations

$$\hat{x}_{t+1} \;=\; A\hat{x}_t + Bu_t\ \ \forall\ t \in \mathbb{Z}_{0,N-1}, \tag{14}$$

$$P_{x_{t+1}} \;=\; AP_{x_t}A^T + P_w\ \ \forall\ t \in \mathbb{Z}_{0,N-1}. \tag{15}$$

To ensure that the probability of collision with any obstacle on a given time step does not exceed $\Delta \equiv 1 - p_{\text{safe}}$, it is sufficient to show that the probability of collision with each of the B obstacles at that time step does not exceed Δ/B. [9] The jth obstacle is represented through the conjunction of linear inequalities

$$\bigwedge_{i=1}^{n_j} a_{ij}^T x_t < a_{ij}^T c_{ijt}\ \ \forall\ t \in \mathbb{Z}_{0,t_f}, \tag{16}$$

where n_j is the number of constraints defining the jth obstacle, and c_{ijt} is a point nominally (i.e. $c_{jt} = \hat{c}_{jt}$) on the ith constraint at time step t; note that a_{ij} is not dependent on t, since the obstacle shape and orientation are fixed. To avoid all obstacles, the system must satisfy B disjunctions of constraints at each time step,

$$\bigvee_{i=1}^{n_j} a_{ij}^T x_t \geq a_{ij}^T c_{ijt}\ \ \forall\ j \in \mathbb{Z}_{1,B},\ \forall\ t \in \mathbb{Z}_{0,N}. \tag{17}$$

For each obstacle – consider the jth one below – it is sufficient to not satisfy any one constraint in the conjunction (16). Thus, the probability of collision is *lower-bounded* by the probability of satisfying any single constraint:

$$P(\text{collision}) \quad \leq \quad P(a_{ij}^T X_t < a_{ij}^T c_{ijt})\ \ \forall\ i \in \mathbb{Z}_{1,n_j}. \tag{18}$$

To prove the probability of collision with the jth obstacle does not exceed Δ/B, it is thus sufficient to show that

$$\bigvee_{i=1}^{n_j} P(a_{ij}^T X_t < a_{ij}^T C_{ijt}) \leq \Delta/B, \tag{19}$$

where $C_{ijt} = c_{ijt} + (c_{jt} - \hat{c}_{jt})$ is a random variable.

Next, a change of variables is made to render the problem tractable for path planning algorithms. For the ith constraint of the jth obstacle at time step t apply the change of variable

$$V = a_{ij}^T X_t - a_{ij}^T C_{ijt}; \tag{20}$$

it can be shown [8] that the mean and covariance for V are

$$\hat{v} \;=\; a_{ij}^T \hat{x}_t - a_{ij}^T c_{ijt}, \tag{21}$$

$$P_v \;=\; \sqrt{a_{ij}^T (P_{x_t} + P_{c_{jt}}) a_{ij}}. \tag{22}$$

With this change of variables, the probabilistic constraint can be shown to be equivalent to a deterministic constraint, [9]

$$\begin{aligned} P(V < 0) \;&\leq\; \Delta/B \\ \Leftrightarrow\ \hat{v} \;&\geq\; \gamma \equiv \sqrt{2}P_v \text{erf}^{-1}\left(1 - 2\frac{\Delta}{B}\right), \end{aligned}$$

where erf$(\cdot)$ denotes the standard error function. Using this, the constraints (17) are probabilistically satisfied for the true state x_t if the conditional mean $\hat{x}_t$ satisfies

$$\bigvee_{i=1}^{n_j} a_{ij}^T \hat{x}_t \quad \geq \quad b_{ij} + \bar{b}_{ijt}\ \ \forall\ j \in \mathbb{Z}_{1,B},\ \forall\ t \in \mathbb{Z}_{0,N}, \tag{23}$$

where $\bar{b}_{ijt} = \gamma$ represents the amount of *deterministic* constraint tightening necessary to ensure *probabilistic* constraint satisfaction.

Whereas the traditional RRT algorithm incrementally grows a tree of states which are known to be feasible [7], the chance constrained RRT (CC-RRT) algorithm grows a tree of state distributions which are known to satisfy an upper bound on probability of collision. Furthermore, the CC-RRT algorithm can leverage a key property of the RRT algorithm – trajectory-wise constraint checking – by explicitly computing a bound on the probability of collision at each node, rather than simply satisfying tightened constraints for a fixed bound. In particular, the Online CC-RRT approach [8] leverages the relationship in (23) to compute the exact probability of satisfying each individual constraint for a given distribution $\mathcal{N}(\hat{x}, P_x)$ – an operation which is possible due to iterative constraint checking in the RRT algorithm. The key to the Online CC-RRT approach is the relationship

$$P(V < 0) = \frac{1}{2}\left(1 - \mathrm{erf}\left[\frac{\hat{v}}{\sqrt{2}P_v}\right]\right), \tag{24}$$

which has been shown [8] to be derived from (23). Again consider the ith constraint of the jth obstacle at time step t, using the change of variables (20). Let $\Delta_{ijt}(\hat{x}, P_x)$ denote the probability that this constraint is satisfied for a Gaussian distribution with mean $\hat{x}$ and covariance P_x; using (24),

$$\Delta_{ijt}(\hat{x}, P_x) = \frac{1}{2}\left(1 - \mathrm{erf}\left[\frac{a_{ij}^T\hat{x}_t - a_{ij}^T c_{ijt}}{\sqrt{2a_{ij}^T(P_{x_t} + P_{c_j})a_{ij}}}\right]\right) \tag{25}$$

Now define

$$\Delta_t(\hat{x}_t, P_{x_t}) \equiv \sum_{j=1}^{B} \min_{i=1,\ldots,n_j} \Delta_{ijt}(\hat{x}_t, P_{x_t}). \tag{26}$$

This term provides an upper bound on the probability of a collision with any obstacle at time step t [8]:

$$\begin{aligned} P(\text{collision}) &\leq \sum_{j=1}^{B} P(\text{collision with obstacle } j) \qquad (27)\\ &\leq \sum_{j=1}^{B} \min_{i=1,\ldots,n_j} P(a_{ij}^T X_t < a_{ij}^T C_{ijt})\\ &= \sum_{j=1}^{B} \min_{i=1,\ldots,n_j} \Delta_{ijt}(\hat{x}_t, P_{x_t}) = \Delta_t(\hat{x}_t, P_{x_t}) \end{aligned}$$

Thus, for a node/timestep with state distribution $\mathcal{N}(\hat{x}_t, P_{x_t})$ to be probabilistically feasible, it is sufficient to check that $\Delta_t(\hat{x}_t, P_{x_t}) \leq 1 - p_{\text{safe}}$.

The CC-RRT algorithm consists of two main routines, a tree expansion step which incrementally adds probabilistically feasible nodes to the tree, and an execution loop which periodically chooses the cost-minimizing path in the tree; see [8] for more details.

B. RR-GP Integration

Suppose the jth obstacle is one of the dynamic obstacles modelled using RR-GP (Section IV); that dynamic obstacle may follow one of $k = 1, \ldots, b$ possible behaviors. At each timestep t, and for each behavior k, the RR-GP algorithm provides a likelihood δ^k and Gaussian distribution $\mathcal{N}(\hat{c}_{jt}^k, P_{c_{jt}}^k)$ for the uncertainty distribution of the obstacle if following that behavior. Thus, the overall state distribution for this obstacle at timestep t is given by

$$c_{tj} \quad \sim \quad \sum_{k=1}^{b} \delta^k \mathcal{N}(\hat{c}_{jt}^k, P_{c_{jt}}^k). \tag{28}$$

At each timestep, the probability of collision with dynamic obstacle j can be written as a weighted sum of the probabilities of collision for the dynamic obstacle j under each behavior, using δ^k as the weights:

$$\begin{aligned} &P(\text{collision with obstacle } j)\\ &= \sum_{k=1}^{b} \delta^k P(\text{collision with obstacle } j, \text{behavior } k). \end{aligned}$$

Comparing with (27), a dynamic obstacle with uncertain behaviors can actually be modelled as b separate obstacles, each with its own Gaussian uncertainty, with the modification that each such term is weighted by δ^k. The existing CC-RRT framework can be used by treating each behavior's state distribution as a separate obstacle, so long as the resulting risk is scaled by δ^k. Thus the existing guarantees of probabilistic feasibility [8] still hold under this modification.

Finally, we note that the guarantees presented in this section only hold under the assumptions of linear dynamics and Gaussian uncertainty. Modifications are possible to alleviate these assumptions [14], though this may yield an increase in computational load (particularly if the uncertainty is non-Gaussian).

VI. Simulation Results

This section presents simulation results which demonstrate the effectiveness of the RR-GP algorithm in predicting the future behavior of an unknown, "hostile" vehicle, allowing the CC-RRT planner to design paths to safely avoid it. In these examples, only a single dynamic, uncertain obstacle is present; the approach can be extended to multiple dynamic, uncertain obstacles without further modification, and in fact has been shown to scale well under such conditions [8].

Examples for both linear and nonlinear dynamics are provided. As the probabilistic safety bound p_{safe} is increased at each timestep, the algorithm selects more conservative paths, which are more likely to reach the goal without colliding with any obstacles, but may require additional path length/time to do so. By placing the bound sufficiently high, however, the planner can design paths which consistently guide the host vehicle out of unsafe situations with high likelihood.

This section is organized as follows. First, the software infrastructure used to generate these results is explained. Two sets of simulation results then follow. The most thorough results consider a vehicle in an intersection scenario, where the agent must cross an intersection while avoiding another vehicle which is traveling on the wrong side of the road. In this case, the host vehicle is modeled as a double integrator, such that the theoretical CC-RRT results still hold. In the second scenario, the host vehicle, modelled using nonlinear car dynamics, must navigate across a complex obstacle field. In this case, the hostile vehicle may follow as many as six different behaviors.

A. Infrastructure

The path planning algorithms in this paper have been implemented using a multi-threaded, real-time Java application, executed on a 2.53GHz quad-core laptop with 3.48GB of RAM. The "hostile" vehicle's trajectories, both for GP training and simulated motion, were pre-generated by having a human operator manually drive the vehicle in simulation with a steering wheel, in a manner consistent with each behavior. The hostile vehicle dynamics are based on the iRobot Create skid-steered platform; a software wrapper imposes rate limits in acceleration and wheel speed differences, such that the vehicle emulates traditional automative dynamics at a maximum speed of 0.4 m/s. During each trial, one of these paths is randomly selected as the trajectory for the hostile vehicle.

B. Intersection Scenario

Consider a ground vehicle operating in a simple road network (Figure 3(a)). In this scenario, which assumes left-hand traffic, the objective of the host vehicle is to go straight through the intersection at bottom-center of Figure 3(a), reaching a goal location 6 meters ahead on the opposite side. However, to get there, the host vehicle must successfully avoid a hostile vehicle, which incorrectly believes that right-hand traffic rules are in effect and thus is traveling in the wrong lane. There are three possible behaviors for the hostile vehicle as it enters the intersection: (a) left turn, (b) right turn, and (c) straight.

The vehicle is modeled as a discrete, double integrator,

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ v^x_{t+1} \\ v^y_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ v^x_t \\ v^y_t \end{bmatrix} + \begin{bmatrix} \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} u^x_t \\ u^y_t \end{bmatrix} + \begin{bmatrix} w^x_t \\ w^y_t \end{bmatrix} \right),$$

with vehicle state $(x_{t+1}, y_{t+1}, v^x_{t+1}, v^y_{t+1})$ and timestep $dt =$ 0.2s, subject to avoidance constraints $\mathcal{X}$ (including velocity bounds) and input constraints $\mathcal{U} = \{(u^x, u^y) \mid |u^x| \leq 4, |u^y| \leq 4\}$. This choice of dynamics was made to ensure that the dynamics are still linear, such that the theoretical guarantees are still available [8]. To emphasize the impact of the hostile vehicle's uncertainty, the host vehicle's own dynamics are assumed deterministic. The vehicle is simulated and executed in closed-loop, using the controller

$$\begin{aligned} u^x_t &= -1.5(x_t - r^x_t) - 3(v^x_t - r^{v_x}_t), \\ u^y_t &= -1.5(y_t - r^y_t) - 3(v^y_t - r^{v_y}_t), \end{aligned}$$

where (r^x_t, r^y_t) is the reference position and $(r^{v_x}_t, r^{v_y}_t)$ is the reference velocity (0.35 m/s in the direction of the waypoint).

TABLE I
SIMULATION RESULTS, INTERSECTION SCENARIO

Method	p_{safe}	**% to Goal**[a]	**Path Duration, s**[b]	**Time per Node, ms**[c]	**Time per RR-GP, s**[d]
Naive	–	52%	17.22 (0.13%)	0.618	–
Nominal	–	70%	17.95 (4.39%)	0.753	–
CC-RRT	0.5	64%	17.97 (4.47%)	1.208	2.245
CC-RRT	0.8	72%	18.55 (7.85%)	1.255	2.206
CC-RRT	0.9	72%	19.01 (10.4%)	1.192	2.202
CC-RRT	0.99	92%	20.60 (19.8%)	1.183	2.173
CC-RRT	0.999	96%	20.63 (19.9%)	1.176	2.196

[a] Number of trials where system executed a path to goal without colliding with any obstacles.
[b] Percentage is average increase in path duration relative to minimal-time (obstacle-free) path, 17.2s. Only paths which reach goal are included.
[c] Cumulative time spent growing the CC-RRT tree, divided by number of nodes generated.
[d] Time spent in Algorithm 1.

In the RR-GP algorithm, $T_h = 10\text{s}$ and $\Delta t = 1\text{s}$, though the vehicle dynamics are simulated at 50 Hz. A total of 20 samples are attempted per Δt; each behavior's tree is terminated as infeasible once 300 samples are marked infeasible. RR-GP is called once every 2.5 seconds, each time giving Algorithm 1 at most 1.5 seconds to grow its trees. If the time limit is reached, the algorithm is terminated without reaching the time horizon T_h. A total of 350 trials were performed, consisting of 50 trials each for seven different algorithms:

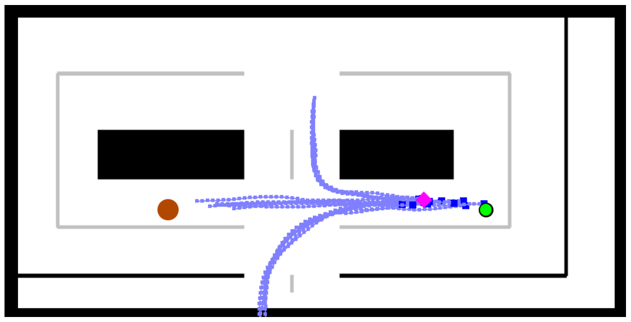
- Naive RRT: nominal RRT (no chance constraints) in which hostile vehicle is ignored entirely
- Nominal RRT: nominal RRT in which hostile vehicle is treated as a static obstacle at its most recent location
- CC-RRT (5 cases): CC-RRT using RR-GP with $p_{\text{safe}} = 0.5, 0.8, 0.9, 0.99,$ or 0.999

Each trial differs only in the path followed by the hostile vehicle and the random RRT samples; the sequence of hostile vehicle paths is consistent across all seven cases. Four quantities were measured and averaged across these trials: the percentage of trials in which the vehicle safely reaches the goal; the average duration of such paths; the average time to generate an RRT/CC-RRT tree node; and the average time per execution of Algorithm 1.

Table I presents the averaged results over the 50 trials for each case. The most immediate observation is the clear trade-off between overall path safety (in terms of percentage of trials which reach the goal) and average path duration when using CC-RRT. As p_{safe} is increased from 0.5 to 0.999, the percentage of safe trials steadily increases from 64% to 96%.[4] Furthermore, as p_{safe} is increased and the planner becomes more conservative, the average degree of suboptimality of the safe trajectories also increases.

In the current configuration, maintaining a safe trajectory in 100% of trials is likely unobtainable, as the time interval between RR-GP updates (2.5s) is sufficiently large that the planner cannot always react quickly enough to environmental changes. Subsequent work [21] has significantly reduced the

[4]Note that since p_{safe} is a bound on feasibility at each *timestep*, rather than over an entire *path*, it does *not* act as a bound on the percentage of paths which safely reach the goal. However, as discussed in Section II, a positive correlation between the two is expected.


(a) Intersection scenario

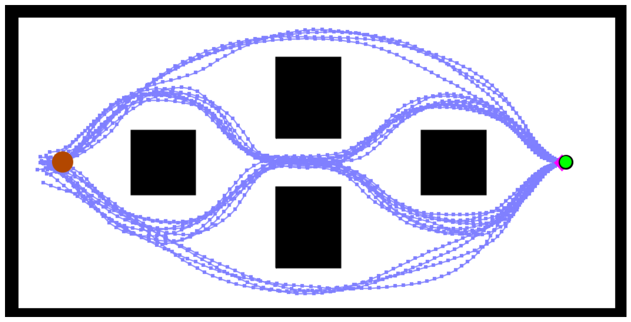

(b) Complex scenario

Fig. 3. Environments used for each scenario. The objective of the host vehicle (orange circle) is to reach the goal position (green circle) while avoiding all static obstacles (black) and the dynamic hostile vehicle (magenta diamond). The blue curves indicate the possible trajectories followed by the hostile vehicle. All objects are shown at true size; the grey lines are lane markings, which do not serve as constraints. Both environments are 11.2×5.5 m^2 in size.

runtime necessary to perform the RR-GP update operation, achieving runtimes (approximately 0.5s) which allow for a more rapid response and the possibility of all trials staying feasible.

Figure 4 sheds some light on how different values of p_{safe} affect the types of paths chosen by the planner. In this particular trial, the hostile vehicle crosses the intersection, a behavior the RR-GP is able to predict with high accuracy (indicated by the small $2-\sigma$ uncertainty ellipses) after 2 iterations. When $p_{\text{safe}} = 0.8$, the planner selects a path with the minimum perturbation needed to avoid the hostile vehicle (Figure 4(a)). After the third RR-GP update, the hostile vehicle's predicted path drifts slightly closer to the centerline; thus the host vehicle adds another small perturbation, barely missing the hostile vehicle as it passes (Figure 4(b)). On the other hand, when $p_{\text{safe}} = 0.999$, the host vehicle immediately tries to put as much distance as possible between itself and the hostile vehicle (Figure 4(c)). In realistic driving scenarios, the most desirable behavior is likely somewhere between these two extremes.

Using Naive RRT (i.e., CC-RRT as $p_{\text{safe}} \to 0$), we see that by ignoring the hostile vehicle, the time-optimal path is almost always achieved, but that a collision takes place in nearly half of all trials. The results for Nominal RRT are competitive with CC-RRT for lower values of p_{safe} (0.5 and 0.8). By treating the hostile vehicle as a static obstacle at its present location, the algorithm can quickly react to its movement; however, this interaction itself often leads to collisions (e.g., the host vehicle turns right to avoid the hostile vehicle, which is turning left). In general, nominal RRT is fundamentally limited in its ability to react to a dynamic vehicle, degrading its safety relative to CC-RRT.

Finally, we note that the average time to either generate an RRT node or call RR-GP is largely independent of p_{safe} for CC-RRT. There is a modest increase in average time per node when moving from nominal RRT algorithms to CC-RRT, though previous work has demonstrated that this increases scales well with environment complexity [8].

C. Complex Scenario

In the complex scenario (Figure 3(b)), as the hostile vehicle moves from right to left, it may display as many as six possible behaviors, corresponding to the sides on which it passes each obstacle. Furthermore, the hostile vehicle moves twice as quickly compared to the previous scenario (0.8 m/s), while the host vehicle's speed is only 0.5 m/s. The host vehicle and hostile vehicle are to swap positions; the host vehicle must move from left to right, while avoiding a collision with a faster-moving dynamic obstacle moving in the opposite direction.

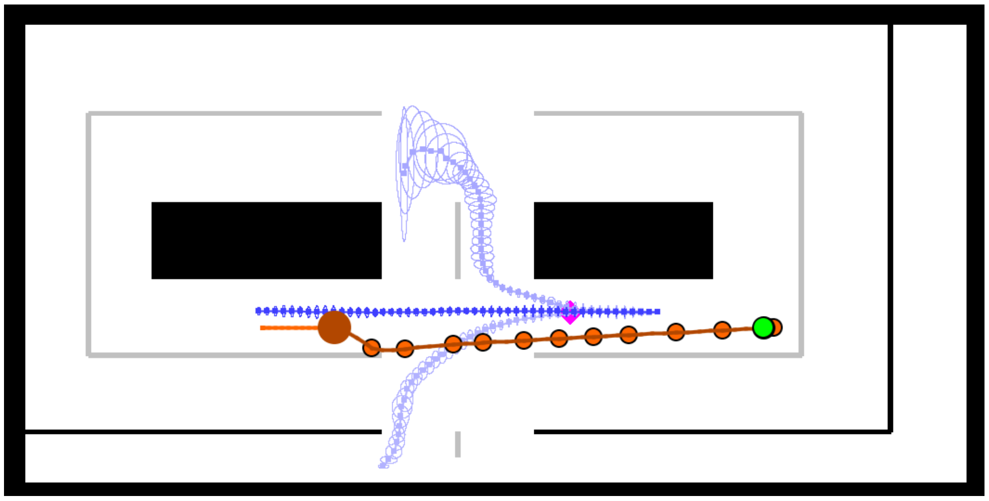
For this scenario, the host vehicle is modeled as nonlinear car dynamics with a fixed speed $v = 0.5$ m/s,

$$\begin{aligned} x_{t+1} &= x_t + (dt)v\cos\theta_t + w_t^x, \\ y_{t+1} &= y_t + (dt)v\sin\theta_t + w_t^y, \\ \theta_{t+1} &= \theta_t + (dt)\frac{v}{L_w}\tan\delta_t + w_t^\theta, \end{aligned}$$

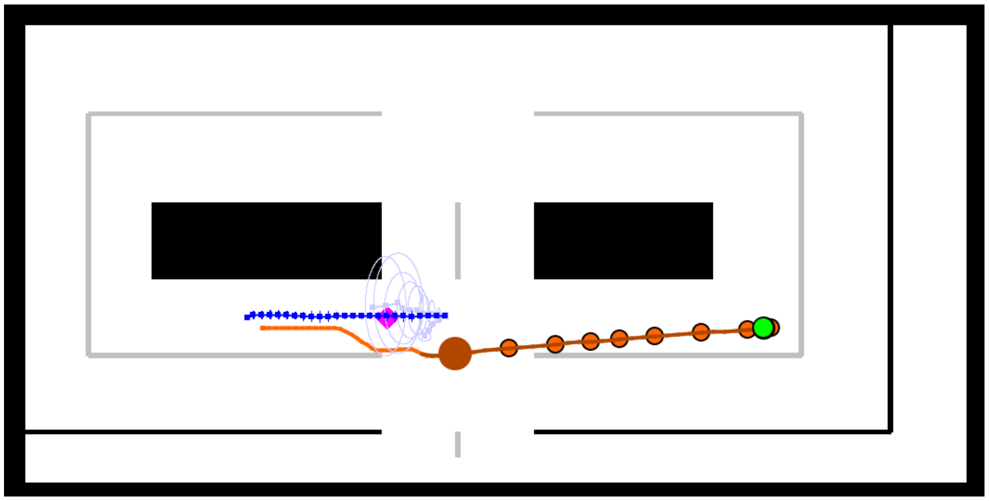
where $dt = 0.2$s, (x, y) is the vehicle position, θ is the heading, the wheel base $L_w = 0.2$ m, and $\delta_t \in [-\pi/4, +\pi/4]$ is the steering angle input. The vehicle is controlled in closed-loop using a nonlinear steering controller [22]. In this scenario, the vehicle must also contend with process noise, where the covariance on the disturbance $w_t = (w_t^x, w_t^y, w_t^\theta)$ is $P_w = \text{diag}(5\times10^{-5},\ 5\times10^{-5},\ 2\times10^{-4})$.

Using RR-GP for this scenario, the vehicle dynamics are simulated at 10 Hz rather than 50 Hz, and a total of 10 samples are attempted per Δt, rather than 20. The RR-GP algorithms are called once every 3.5 seconds, each time giving Algorithm 1 a total of 2.5 seconds to grow trees for each behavior.

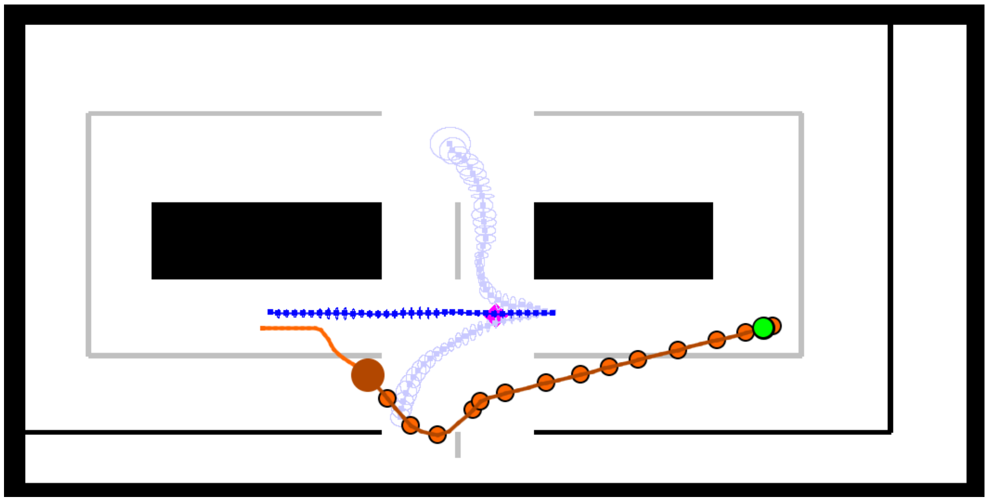
Figure 5 demonstrates the operation of CC-RRT using RR-GP for a typical trial. When the simulation is initialized, all possible behaviors are perceived as equally likely; the host vehicle selects a path aimed between the two center obstacles (Figure 5(a)). As the hostile vehicle moves downward, the three behaviors moving in that direction are assigned higher likelihoods, including two down the center path. As a result, the host vehicle's initial path selection becomes probabilistically infeasible, and thus the planner picks a new



(a) $p_{\text{safe}} = 0.8$, after 2 uses of RR-GP

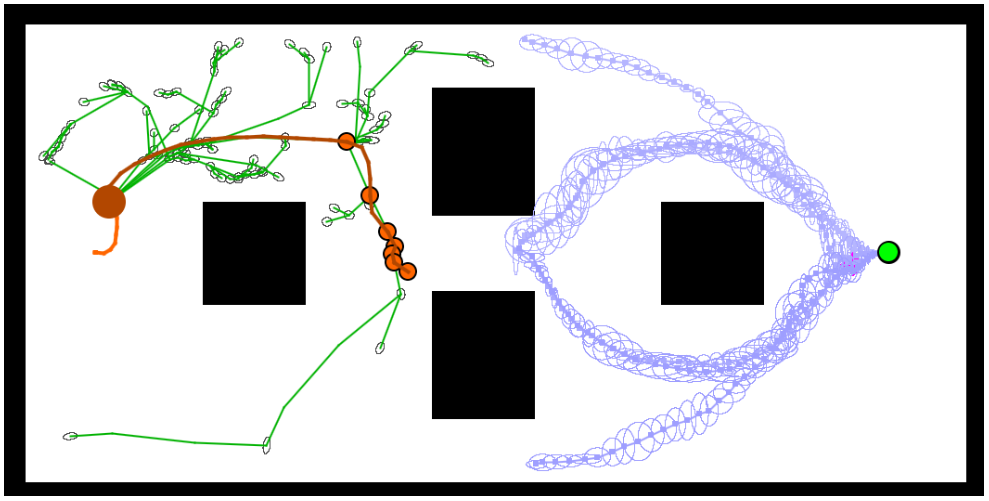


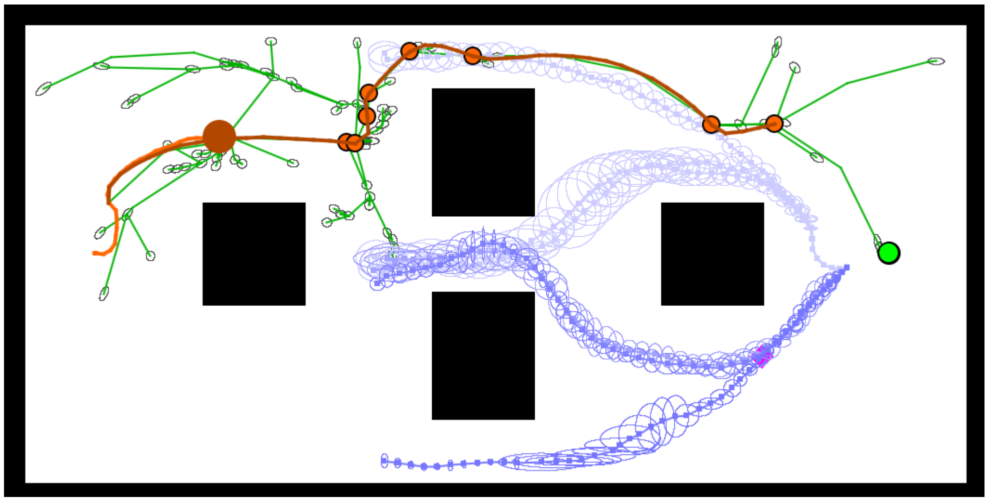
(b) $p_{\text{safe}} = 0.8$, after 3 uses of RR-GP



(c) $p_{\text{safe}} = 0.999$, after 2 uses of RR-GP

Fig. 4. Representative screenshots of the RR-GP and CC-RRT algorithms during Trial 5 of the intersection scenario, for two different values of p_{safe}. The host vehicle's path history and current path are in orange. The blue paths indicate the paths predicted by the RR-GP algorithm for each possible behavior, including $2-\sigma$ uncertainty ellipses; more likely paths are indicated with a brighter shade of blue.


(a) $t = 1$ s


(b) $t = 4$ s

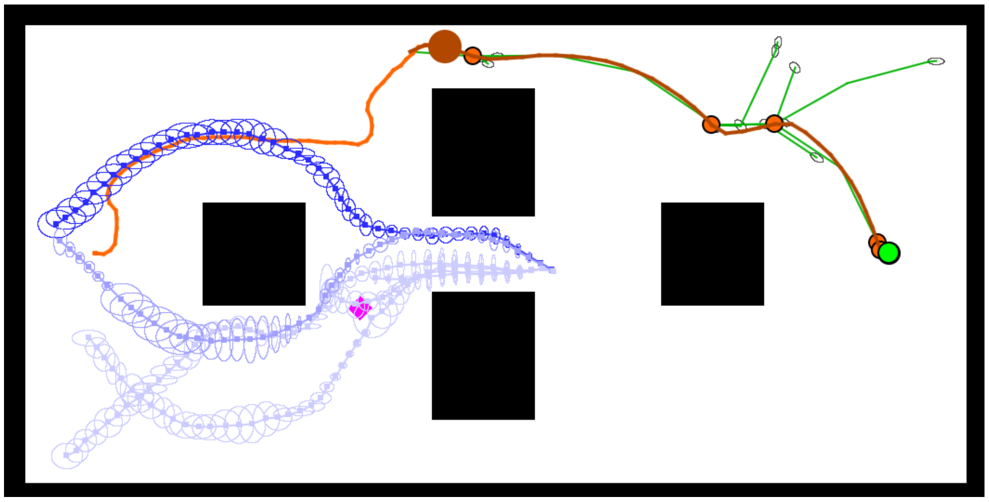

(c) $t = 7.5$ s

Fig. 5. Demonstration of the CC-RRT algorithm with RR-GP applied to a representative trial in the complex scenario. The CC-RRT tree is shown in green. Uncertainty ellipses for the host vehicle's process noise are indicated by $2 - \sigma$ uncertainty ellipses on the RRT tree.

path across the top of the environment, where the hostile vehicle is much less likely to appear (Figure 5(b)). This allows the host vehicle to safely reach the goal as the hostile vehicle continues to move to the left (Figure 5(c)). Note that due to the large time interval between RR-GP updates, it is possible for the RR-GP prediction to poorly match the vehicle's current position, as in Figure 5(c). As mentioned previously, subsequent updates to this work have reduced these update cycles to well under a second, allowing RR-GP to update more frequently based on the vehicle's current position [21].

VII. Conclusion

This paper introduced a real-time path planning framework for safe navigation of autonomous agents navigating in an environment populated with dynamic obstacles with uncertain motion patterns. The developed solution combined RR-GP, a Bayesian nonparametric reachability tree method [2], with chance-constrained rapidly-exploring random trees [8], resulting in a probabilistic path planning formulation that guarantees probabilistic feasability and is suitable for real-time implementation. Simulation results demonstrated the effectiveness of the developed approach in designing paths for vehicles to safely avoid the risk posed by a "hostile" vehicle with unknown future behavior.

References

[1] S. M. Lavalle and R. Sharma, "On motion planning in changing, partially-predictable environments," *International Journal of Robotics Research*, vol. 16, pp. 775–805, 1997.

[2] G. S. Aoude, J. Joseph, N. Roy, and J. P. How, "Mobile Agent Trajectory Prediction using Bayesian Nonparametric Reachability Trees," in *AIAA Infotech@Aerospace*, St. Louis, Missouri, March 2011.

[3] D. Vasquez, T. Fraichard, O. Aycard, and C. Laugier, "Intentional motion on-line learning and prediction," *Machine Vision and Applications*, vol. 19, no. 5, pp. 411–425, 2008.

[4] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2008, pp. 1056–1062.

[5] J. Joseph, F. Doshi-Velez, and N. Roy, "A bayesian nonparametric approach to modeling mobility patterns," 2010.

[6] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A bayesian nonparametric approach to modeling motion patterns," Tech. Rep., 2011.

[7] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep. 98-11, October 1998.

[8] B. Luders, M. Kothari, and J. P. How, "Chance constrained RRT for probabilistic robustness to environmental uncertainty," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, Toronto, Canada, August 2010 (AIAA-2010-8160).

[9] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *Proceedings of the IEEE American Control Conference*, 2006, pp. 2831–2837.

[10] G. C. Calafiore and L. E. Ghaoui, "Linear programming with probability constraints – part 1," in *Proceedings of the IEEE American Control Conference*, 2007.

[11] L. Blackmore, "A probabilistic particle control approach to optimal, robust predictive control," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.

[12] ——, "A probabilistic particle control approach to optimal robust predictive control," in *Proceedings of the IEEE American Control Conference*, 2007.

[13] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 502–517, 2010.

[14] B. Luders and J. P. How, "Probabilistic feasibility for nonlinear systems with non-gaussian uncertainty using rrt," in *Proceedings of the AIAA Infotech@Aerospace Conference*, 2011.

[15] J. Joseph, F. Doshi-Velez, and N. Roy, "A bayesian nonparametric approach to modeling mobility patterns," in *AAAI*, 2010.

[16] M. Tay and C. Laugier, "Modelling smooth paths using gaussian processes," pp. 381–390, 2008.

[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*, 2005.

[18] G. S. Aoude, B. D. Luders, D. S. Levine, and J. P. How, "Threat-aware Path Planning in Uncertain Urban Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010.

[19] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, September 2009.

[20] G. S. Aoude, B. D. Luders, D. S. Levine, K. K. H. Lee, and J. P. How, "Threat Assessment Design for Driver Assistance System at Intersections," in *IEEE Conference on Intelligent Transportation Systems*, Madeira, Portugal, September 2010.

[21] G. Aoude, B. Luders, J. Joseph, N. Roy, and J. P. How, "Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, 2011 (submitted).

[22] M. R. Walter, S. Karaman, E. Frazzoli, and S. Teller, "Closed-loop pallet manipulation in unstructured environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 2010.