Mobile Agent Trajectory Prediction using Bayesian Nonparametric Reachability Trees

Mobile Agent Trajectory Prediction using Bayesian Nonparametric Reachability Trees

Georges S. Aoude[*], Joshua Joseph[†], Nicholas Roy[‡], Jonathan P. How[§]

This paper presents an efficient trajectory prediction algorithm that has been developed to improve the performance of future collision avoidance and detection systems. The main idea is to embed the inferred intention information of surrounding agents into their estimated reachability sets to obtain a probabilistic description of their future paths. More specifically, the proposed approach combines the recently developed RRT-Reach algorithm and mixtures of Gaussian Processes. RRT-Reach was introduced by the authors as an extension of the closed-loop rapidly-exploring random tree (CL-RRT) algorithm to compute reachable sets of moving objects in real-time. A mixture of Gaussian processes (GP) is a flexible nonparametric Bayesian model used to represent a distribution over trajectories and have been previously demonstrated by the authors in a UAV interception and tracking of ground vehicles planning scheme. The mixture is trained using typical maneuvers learned from statistical data, and RRT-Reach utilizes samples from the GP to grow probabilistically weighted feasible paths of the surrounding vehicles. The resulting approach, denoted as RR-GP, has RRT-Reach's benefits of computing trajectories that are dynamically feasible by construction, therefore efficiently approximating the reachability set of surrounding vehicles following typical patterns. RRT-GP also features the GP mixture's benefits of providing a probabilistic weighting on the feasible trajectories produced by RRT-Reach, allowing our system to systematically weight trajectories by their likelihood. A demonstrative example on a car-like vehicle illustrates the advantages of the RR-GP approach by comparing it to two other GP-based algorithms.

Nomenclature

x_t	x-position at time t
y_t	y-position at time t
$\hat{x}_t$	Predicted x-position at time t
$\hat{y}_t$	Predicted y-position at time t
M	Number of motion patterns
t^i	Trajectory i
b_j	Motion pattern j
z_j	Indicator variable for trajectory j
θ^{GP}	Gaussian process hyperparameters
σ_x	Exponential weighting Gaussian process hyperparameter
σ_n	Noise weighting Gaussian process hyperparameter
w_x	x-direction length-scale Gaussian process hyperparameter
w_y	y-direction length-scale Gaussian process hyperparameter
K_x	x-direction covariance function
K_y	y-direction covariance function
Σ	Covariance matrix
$\mathcal{T}_{GP}$	RR-GP single tree
T_h	Time horizon of prediction
δt	Controller time period
dt	Measurement time interval
Δt	Gaussian process sampling time interval

[*]Ph.D. Candidate, Dept. of Aeronautics and Astronautics, MIT; Student Member AIAA gaoude@mit.edu
[†]Ph.D. Candidate, Dept. of Aeronautics and Astronautics, MIT jmjoseph@mit.edu
[‡]Associate Professor of Aeronautics and Astronautics, MIT nickroy@mit.edu
[§]Richard C. Maclaurin Professor of Aeronautics and Astronautics, MIT; Associate Fellow AIAA jhow@mit.edu

θ_t Vehicle heading
a_t Vehicle input acceleration
δ_t Vehicle input steering
Subscript
t Time
j Motion pattern index
x Parameter associated with the x-direction
y Parameter associated with the y-direction
Superscript
i Trajectory index

I. Introduction

Future collision avoidance (CA) and conflict detection (CD) systems must handle increasingly complex scenarios including both unmanned and manned vehicles interacting in uncertain and possibly hostile environments. A key challenge is inferring threats or dangers posed from observations of the surrounding agents by predictions of their future motion. For example, future advanced driver assistance systems (ADAS) could include new risk assessment algorithms for intersections, one of the most dangerous road scenarios due to the difficulty in predicting incoming drivers' trajectories.[1] Another example is the Next Generation Air Transportation System (NextGen), which has the goal of increasing the safety and capacity of air transport operations but will require the development of new conflict detection algorithms that can tolerate tighter separation distances and an increasing number of aircraft flying in the National Air Space.[2]

A key component for such threat assessment systems is a trajectory prediction algorithm (TPA) that estimates the future states of the *target vehicles*; these vehicles move in the vicinity of the *host vehicle* that has the threat assessment system onboard. The output of the TPA along with the knowledge of the current path of the host vehicle are fed into a threat assessment algorithm, which computes an estimate of the threat incurred by the host vehicle if it follows its intended path. A collision warning system uses the output of the threat assessment algorithm to warn the host vehicle about the risk of its path, while a collision mitigation system takes over the control of the vehicle and apply an escape maneuver that reduces the threat of collision.

A major challenge in trajectory prediction algorithms is caused by the different sources of uncertainties in dynamic environments. They can be classified into two different categories of uncertainties: 1) environment sensing (ES), and 2) environment predictability (EP).[3] While the ES type deals with the uncertainty due to the imperfection of sensor measurements or incomplete knowledge of the environment configuration, the EP type is concerned with the uncertainty in the future state of the environment. The focus of this work is on the EP uncertainty. Even with the assumption of the existence of perfect sensors and complete knowledge of the current state environment, predicting long-term trajectories of mobile agents remains very difficult. For instance, when approaching a road intersection, an agent could follow different sets of paths corresponding to different underlying intents (e.g., right turns versus straight paths). In addition to traversing its paths with varying speeds and headings. Modelling these variations is impossible when only relying on perfect state information.

Modelling the evolution of target vehicles can be classified into three main categories: 1) worst-case, 2) pattern based, and 3) dynamic based approaches. In the worst-case approach, the target vehicle is assumed to be actively trying to collide with the host vehicle.[4,5] The predicted trajectory of the target vehicle is the solution of a differential game where the target vehicle is modelled as a pursuer and the host vehicle as an evader. This solution is conservative so it is typically limited to short time-horizons in collision warning/mitigation problems to keep the level of false positives below a reasonable threshold.[6] In the pattern based approach, the vehicles are assumed to move according to typical patterns across the environment. These patterns are learned by observing the targets in the environment. There are two main techniques that fall under this category: a) discrete state-space, and b) clustering based.[7] In the discrete state-space technique, the motion model is typically based on Markov chains: the object state evolves from one state to another according to a learned transition probability.[8] In the clustering based technique, previously observed trajectories are grouped into different clusters. Each cluster is then represented by one trajectory prototype.[9] Given a partial path, prediction is then performed by finding the most likely cluster, or computing a probability distribution over the the different clusters. Both pattern based techniques have been popular in solving long-term prediction problems for mobile agents.[7] Finally, the dynamic based approach predicts the motion of the vehicle by typically estimating the current state of the vehicle and propagating it in the future assuming a fixed mode of operation. This prediction

typically uses a continuous Bayes filters such as the Kalman Filter or its variations.[10] A popular extension in the target tracking literature is the Interacting Multiple Model Kalman Filter (IMM-KF). It uses available observations to update a bank of Kalman Filters and then find the filter closest to the current mode of operation of the vehicle.[11] Since they are only based on the dynamic properties of the vehicle, dynamic based approaches are mainly suitable for short-term motion prediction. They typically perform poorly in long term prediction of trajectories due to their inability to model future changes in control inputs or take into consideration other external factors such as obstacles.

Since the focus of this work is on predicting long-term trajectories of vehicles navigating in cluttered environments, we take the pattern-based approach and adopt the clustering-based technique. Although learning a dynamic obstacle motion pattern model from training data reduces the need for expert knowledge, we must still choose a class of models for the motion patterns. If we were to choose a model class that is too simple, such as linear trajectories, we would be unable to capture the variety of motion patterns that we may encounter. On the other hand, if we were to choose an extremely sophisticated model class with many parameters, we would need large amounts of data which can be costly or impossible to collect in real world domains.[12] In our previous work,[12,13] we showed that a Bayesian nonparametric approach to modeling motion patterns 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. This paper augments the GP predictions with a new approach based on closed-loop rapidly-exploring random trees (CL-RRT)[14] to generate dynamically feasible and collision-free trajectories that improve the accuracy of trajectory prediction results.

We first describe our statement of the problem in Section II followed by the motion model in Section III. Section IV combines the motion model with the CL-RRT algorithm to create our RR-GP algorithm. Results showing the performance of RR-GP are presented in Section V in a simulated environment with a human-driven target vehicle and Section VI offers concluding remarks.

II. Problem Statement

We focus on the problem of predicting the position of a target vehicle moving according to a set of motion patterns in an obstacle filled environment. Our goal is to develop an efficient mechanism for prediction, suitable for real-time applications while achieving high long-term prediction accuracy. We envision two main potential applications for this work. The first is for autonomous vehicle navigation where our work would be embedded in a probabilistic path planning collision avoidance algorithm. Our second potential application is to form the basis of a threat assessment algorithm for collision warning detection. We are primarily motivated by the road network intersection threat assessment problem,[15] where we compute the threat of dangerous vehicles approaching an intersection for driver collision warning systems.

Figure 1 shows the high level architecture of our solution. The input of the Trajectory Prediction Algorithm (TPA) is sensor measurements of the target vehicle, and its output is a distribution over future trajectories of the target vehicle. The first component of the TPA is the Intention Predictor that uses the current and previous sensor measurements, along with a set of typical motion patterns, to produce an intent distribution which is given to the trajectory generation component. Using the intent information, a dynamics model of the target vehicle, and a map of the environment, the trajectory generator outputs a set of predicted trajectories with different probabilities of occurrence. To limit the scope of the problem to uncertainty in predictability (EP), the sensors measurements are assumed to be noise-free despite our motion pattern representation being robust to noisy measurements. Additionally, the map of the environment, typically a road network, is assumed to be available a priori. The next section introduces the motion model used in the intent predictor.

III. Motion Model

A. Motion Pattern

We define a *motion pattern* as a mapping from locations to a distribution over velocities (trajectory derivatives).[a] 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 blind to the lengths and discretizations of the trajectories, modeling motion patterns as flow fields rather than single paths also allows us to group trajectories sharing key characteristics: for example, a single motion pattern can

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

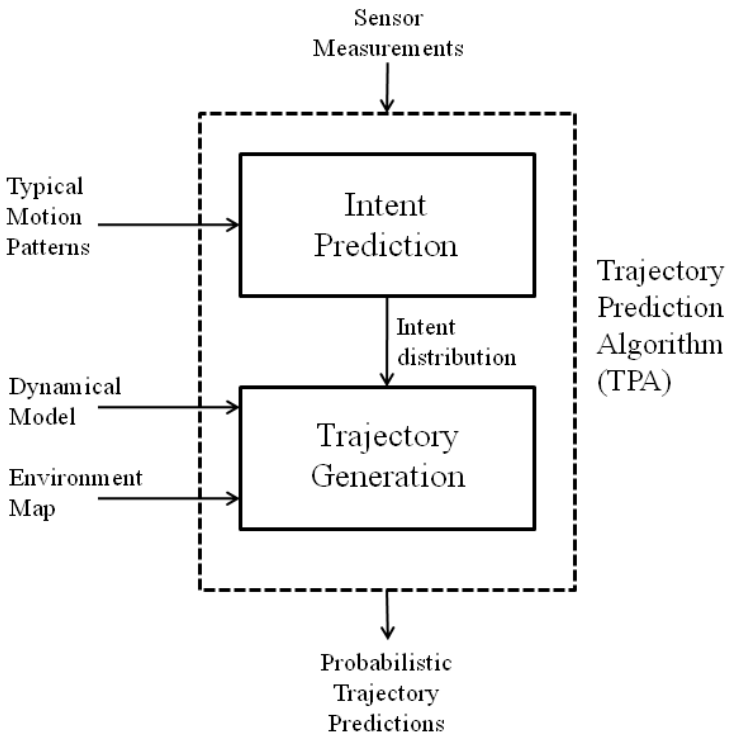


Figure 1: Architecture of the Trajectory Prediction Algorithm (TPA).

capture all the paths that an agent might take from different starting points to a single ending location.

B. Mixtures of Motion Patterns

A finite mixture model with M motion patterns $b_1, b_2, \ldots, b_M$ first assigns a prior probability for each pattern $p(b_1), p(b_2), \ldots, p(b_M)$. Given these prior probabilities, the probability of the i^{th} observed trajectory t^i under the mixture model[b] is

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

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*.

We define the *motion model* as a mixture of weighted motion patterns (Eq. (1)). Each motion pattern is weighted by its probability and is modeled as a pair of Gaussian processes mapping (x, y) locations to distributions over trajectory derivatives $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$. Our motion model has been previously presented,[13,16] and we reproduce it in Sections III-C and III-D for the convenience of the reader. This work assumes the trajectories are labeled with which motion pattern each belongs to for only the training data. This is a reasonable assumption for several domains of interests (e.g., road intersections) where the road network is known and the different set of patterns is easily distinguishable during a preprocessing step.

This section describes our model for $p(t^i|b_j)$ from Eq. (1), the probability of trajectory t^i given motion pattern b_j. This model is the distribution over trajectories we expect for a mobility pattern. This model is the distribution over trajectories we expect for a mobility pattern. An example distribution over trajectories may 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 and would be unable to capture the dynamics of the variety of motion patterns we expect. Another common approach is a discrete Markov model. Markov models are ill suited to model mobile agents in the types of real-world domains of interest.[13,16] They are inherently plagued by the decision of the state discretization. To capture the variety of trajectories that we may encounter, an expressive representation (a fine discretization) is necessary, resulting in a model that requires a large

[b]Note that throughout the paper a t with a superscript, such as t^i, refers to a trajectory and a t without a superscript is a time value.

amount of training data. In real-world domains where collecting a large data set is costly or impossible these models then become extremely prone to over-fitting. To prevent over-fitting a coarser discretization may be used but would then be unable to accurately capture the agent's dynamics. Our model for motion patterns are Gaussian processes (GP) and 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.

C. Gaussian Process Motion Patterns

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 of the agent's locations.

The first term inside the summation of Eq. (1), $p(b_j)$, is the prior probability of motion pattern b_j. Given an agent's trajectory t^i, the posterior probability of motion pattern is

$$p(b_j|t_i) \propto p(t_i|b_j)p(b_j) \tag{2}$$

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

$$p(t^i|b_j) = \prod_{t=0}^{L^i} p\left(\frac{\Delta x_t}{\Delta t}\middle| x^i_{0:t}, y^i_{0:t}, \{t^k : z_k = j\}, \theta^{GP}_{x,j}\right) \cdot p\left(\frac{\Delta y_t}{\Delta t}\middle| x^i_{0:t}, y^i_{0:t}, \{t^k : z_k = j\}, \theta^{GP}_{y,j}\right) \tag{3}$$

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 motion 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 $\Sigma_{ij} = K_x(x_i, y_i, x_j, y_j)$. In this work, we use the standard squared exponential covariance function

$$K_x(x, y, x', y') = \sigma_x^2 \exp\left(-\frac{(x - x')^2}{2{w_x}^2} - \frac{(y - y')^2}{2{w_y}^2}\right) + \sigma_n^2 \delta(x, y, x', y') \tag{4}$$

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 (each motion pattern has a separate set of hyperparameters).[c]

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

$$\begin{aligned} \mu_{\frac{\Delta x}{\Delta t}^*} &= K_x(x^*,y^*,X,Y)K_x(X,Y,X,Y)^{-1}\frac{\Delta X}{\Delta t} \\ \sigma^2_{\frac{\Delta x}{\Delta t}^*} &= K_x(x^*,y^*,X,Y)K_x(X,Y,X,Y)^{-1}K_x(X,Y,x^*,y^*) \end{aligned} \tag{5}$$

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 equivalent to those above, using the covariance K_y.

D. Estimating Future Trajectories

As summarized in Eq. (5), our Gaussian process motion model places a Gaussian distribution over trajectory derivatives $(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})$ for every location (x, y). While this provides us with a distribution over the agent's next location, for

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

longer term predictions we need to use this distribution over next position to produce a distribution over the future trajectory. Unfortunately, we cannot compute the distribution over future trajectories in close form, so instead, we will draw trajectory samples and use these sampled trajectories as a trajectory distribution.

To sample a trajectory from a current starting location (x_0, y_0), we first sample a trajectory derivative $(\frac{\Delta x_0}{\Delta t}, \frac{\Delta y_0}{\Delta t})$ to allow us to calculate its next location (x_2, y_2). Then starting from (x_1, y_1), we sample a trajectory derivative $(\frac{\Delta x_1}{\Delta t}, \frac{\Delta y_1}{\Delta t})$ to allow us to calculate its next location (x_2, y_2) and so on. We repeat this process until we have sampled a trajectory of length L. The entire sampling procedure is then repeated from the current location (x_0, y_0) multiple times to get a sampling of future trajectories the agent may take. Given a current location (x_t, y_t) and a given motion pattern b_j we can sample K time steps in the future using

$$
\begin{aligned}
& p(x_{t+K}, y_{t+K} | x_t, y_t, b_j) \\
&= \prod_{k=0}^{K-1} p(x_{t+k+1}, y_{t+k+1} | x_{t+k}, y_{t+k}, b_j) \\
&= \prod_{k=0}^{K-1} p\left(\frac{\Delta x_{t+k+1}}{\Delta t}, \frac{\Delta y_{t+k+1}}{\Delta t} \middle| x_{t+k}, y_{t+k}, b_j \right) \\
&= \prod_{k=0}^{K-1} p\left(\frac{\Delta x_{t+k+1}}{\Delta t} \middle| x_{t+k}, y_{t+k}, b_j \right) p\left(\frac{\Delta y_{t+k+1}}{\Delta t} \middle| x_{t+k}, y_{t+k}, b_j \right) \\
&= \prod_{k=0}^{K-1} \mathcal{N}\left(x_{t+k+1}; \mu_{j, \frac{\Delta x_{t+k+1}}{\Delta t}}, \sigma^2_{j, \frac{\Delta x_{t+k+1}}{\Delta t}} \right) \mathcal{N}\left(y_{t+k+1}; \mu_{j, \frac{\Delta y_{t+k+1}}{\Delta t}}, \sigma^2_{j, \frac{\Delta y_{t+k+1}}{\Delta t}} \right) \qquad (6)
\end{aligned}
$$

where the parameters from the Gaussian distribution are calculated using Eq. (5). Unfortunately, when used online we will not know the trajectory's motion pattern b_j. Given the past observed trajectory $(x_0, y_0), ..., (x_t, y_t)$, we can calculate the distribution K time steps in the future by combining Eq. (1) and Eq. (6). Formally,

$$
\begin{aligned}
& p(x_{t+K}, y_{t+K} | x_{0:t}, y_{0:t}) \\
&= \sum_{j=1}^{M} p(x_{t+K}, y_{t+K} | x_{0:t}, y_{0:t}, b_j) p(b_j | x_{0:t}, y_{0:t}) \qquad (7) \\
&= \sum_{j=1}^{M} p(x_{t+K}, y_{t+K} | x_t, y_t, b_j) p(b_j | x_{0:t}, y_{0:t}) \qquad (8)
\end{aligned}
$$

where $p(b_j | x_{0:t}, y_{0:t})$ is the probability of motion pattern b_j given the observed portion of the trajectory and $p(x_{t+K}, y_{t+K} | x_t, y_t, b_j)$ is given by Eq. (6). Note that we are able to go from Eq. (7) to Eq. (8) because we are assuming that given b_j, the trajectory's history provides no additional information about the future location of the agent. We refer the read to our previous work[13,16] for further discussion and analysis of the motion model implementation and performance.

IV. RR-GP Trajectory Prediction Algorithm

Section III outlined the approach of using GP mixtures to model mobility patterns and its benefits over other models. However, in practice, GPs suffer from two interconnected shortcomings: their high computational cost and their inability to embed static feasibility or vehicle dynamical constraints. Since GPs are based on statistical learning, they are unable to model prior knowledge of road boundaries, static obstacle location, or dynamic feasibility constraints (e.g., minimum turning radius). 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 applications.

To handle both of these problems simultaneously, we develop a specific implementation of the TPA architecture (Section II), denoted as RR-GP. Our approach augments RRT-Reach, a reachability based method which creates dense, feasible trajectories from sparse samples, with the GP mixture model. RRT-Reach was introduced by the authors as an extension of the closed-loop rapidly-exploring random tree (CL-RRT) algorithm to compute reachable sets of moving

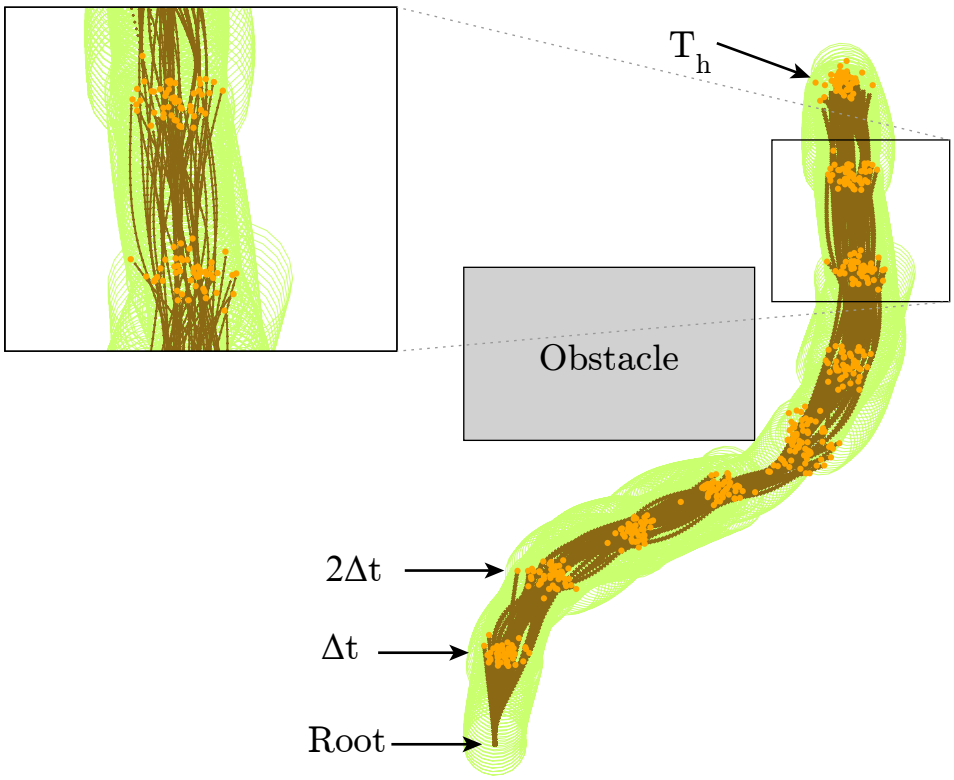


Figure 2: Simple RR-GP Illustration ($M = 1$). RR-GP grows a tree (in brown) using GP samples (orange dots) sampled at Δt intervals for a given motion pattern or intent b. The green circle represents the actual size of the target vehicle. The resulting tree provides a distribution of predicted trajectories of the target at $\delta t \ll \Delta t$ increments.

objects in real-time.[18] RRT-Reach was also successfully used in a novel threat assessment module for driver assistance systems at intersections.[15]

RR-GP is based on two main components: 1) a rapidly-exploring random tree (RRT), and 2) a GP-based mobility model. For each GP-based pattern b_j, $j \in \{1, \ldots, M\}$ as defined in Section III, and the current position of the vehicle we are trying to predict its future path, denoted as the target vehicle, the RR-GP uses an RRT-based technique to grow a separate tree of trajectories that follows b_j while embedding dynamical feasibility and collision avoidance to them. More specifically, it is based on the closed-loop RRT (CL-RRT) algorithm,[14] which grows a tree by randomly sampling points toward dynamically feasible trajectories are simulated. CL-RRT was successfully used by the MIT team in the 2007 DARPA Grand Challenge.[19] CL-RRT samples inputs to a controller rather than the vehicle itself, allowing the generation of smoother trajectories more efficiently than traditional RRT algorithms. Unlike the original CL-RRT approach, the RR-GP tree is not used to create paths leading to a goal location, but instead each tree is grown toward regions corresponding to a learned mobility pattern b_j of the target vehicle (See Figure 2). RR-GP is also different from the RRT-Reach algorithm in that it does not approximate the complete reachability set of target vehicle, which is a conservative approach that can be useful when no information regarding typical patterns is provided. Note that throughout this section we sometimes refer to b_j as intent, since each mobility pattern corresponds to a different intentional motion pattern.

A. Single Tree RR-GP Algorithm

In RR-GP, a dynamical model of the target vehicle is assumed to be available. Since we are interested in car-like vehicles, we adopt the standard bicycle model.[20] The parameters of the model are assumed to be available but its inputs are unknown; they represent the commands that the target vehicle applies to follow trajectories generated by RR-GP. The target vehicle inputs are approximated by the outputs of a pure-pursuit (PP) low level controller that computes a sequence of commands towards samples generated from the GP. The PP controller has been shown to generate smooth paths similar to those driven by road drivers, the target agents of interest.[14] Note that the RR-GP algorithm can be

Algorithm 1 RR-GP, Single Tree Expansion

1: Initialize tree $\mathcal{T}_{GP}$ with node at $(x(t), y(t))$; $gp \leftarrow$ GP motion pattern b; $t_{GP} \leftarrow t + \Delta t$; $K \leftarrow 1$; $n_{\text{success}} \leftarrow 0$; $n_{\text{infeas}} \leftarrow 0$
2: **while** $t_{GP} - t \leq T_h$ **do**
3: Take sample x_{samp} from gp, K time steps in the future using Eq. (6), and variance heuristics if necessary
4: Among nodes added at $t_{GP} - \Delta t$, identify N nearest to x_{samp} using distance heuristics
5: **for** each nearest node, in the sorted order **do**
6: Extend $\mathcal{T}_{GP}$ from nearest node using propagation function until it reaches x_{samp}
7: **if** propagated portion is collision free **then**
8: Add sample to $\mathcal{T}_{GP}$ and create intermediate nodes as appropriate **break**
9: Increment successful connection count n_{success}
10: **else**
11: Increment infeasible connection count n_{infeas} and if limit is reached **goto** line 18
12: **end if**
13: **end for**
14: **if** n_{success} reached desired target **then**
15: $t_{GP} \leftarrow t_{GP} + \Delta t$; $K \leftarrow K + 1$; $n_{\text{success}} \leftarrow 0$; $n_{\text{infeas}} \leftarrow 0$
16: **end if**
17: **end while**
18: **return** $\mathcal{T}_{GP}$

easily modified to handle other dynamical models and low level controllers. Every time Algorithm 1 is called, a tree denoted as $\mathcal{T}_{GP}$ is initialized with a root node at the current target vehicle position $(x(t), y(t))$. Variable t_{GP}, which is used to for time bookkeeping in the expansion mechanism of $\mathcal{T}_{GP}$, is also initialized to the current time $t + \Delta t$. Variable gp is initialized to the learned GP mobility pattern b. Finally, variable K specifying the number of time steps in the GP sampling is initialized to 1. At each step, only nodes belonging to the time bucket $t_{GP} - \Delta T$ are eligible to be expanded. They correspond to the nodes added at to the previous t_{GP}. Initially, the root node is the only node added to time bucket corresponding to initial time t. To grow $\mathcal{T}_{GP}$, first a sample x_{samp} is taken from the environment (line 3) at time $t_{GP} + \Delta t$, or equivalently K time steps in the future using Eq. (6). The nodes nearest to this sample, which were added in the previous step (i.e., belonging to time bucket $t_{GP} - \Delta t$), are identified for tree expansion in terms of some distance heuristics (line 4). An attempt is performed to connect the nearest node to the sample using a straight line reference path; this path is then tracked using the propagation function of a PP controller for steering and a Proportional-Integral controller for velocity tracking.[14] The actual resulting path is dynamically feasible and is checked for collisions. Since the $\mathcal{T}_{GP}$ is trying to generate typical trajectories that the target vehicle would follow, only a simulated trajectory that reaches the sample without a collision is kept, and its corresponding node is added to the tree (line 8). Intermediate nodes may be inserted occasionally to help faster future expansion. The new node and intermediate nodes are added to the current t_{GP} time bucket. When the total number of successful connections n_{success} is reached, t_{GP} is incremented by Δt and K is incremented by 1 (line 15).

Several heuristics have been used in Algorithm 1. RR-GP keeps tracks of the total number of unsuccessful connections n_{infeas} at each iteration. When n_{infeas} reaches some predetermined first threshold, the variance of the GP for the current iteration is temporary grown to capture a broader class of paths (line 3). This heuristic is typically useful to generate feasible trajectories when GP samples are close to obstacles. For example, in Figure 2, variance of gp was increased at $t_{GP} = t + 5\Delta t$. But if n_{infeas} reaches a second and final predetermined threshold, RR-GP "gives up" on growing the tree, and returns $\mathcal{T}_{GP}$. This situation usually happens when the mobility pattern b has a low likelihood and is generating a large number of GP samples in infeasible areas of the environment. The nearest node selection chooses between a cost-to-go metric and a path optimization metric that is based on estimated total path length.[21] Nodes that result in an infeasible trajectory are marked as unsafe and avoided in future node selections. These heuristics, along with the time bucket logic, facilitate efficient feasible trajectory generation in RR-GP.

The position and time of the nodes and edges of the resulting RR-GP produce a fine distribution of the target vehicle future positions. Since the RR-GP tree is grown at a higher rate compared to the original GP learning phase, the resulting distribution is generated at $\delta t << \Delta t$ sec increments δt is the low-level controller rate. The result is a significant improvement of the accuracy of the prediction without a deterioration of the computation times (Section V).

B. Multi-Tree RR-GP Algorithm

This section introduces the Multi-Tree RR-GP Algorithm that extends Algorithm 1 to handle multiple motion

Algorithm 2 RR-GP, Multi-Tree Trajectory Prediction

1: Inputs: GP motion pattern b_j; $p(b_j(0))\ \ \forall j \in [1, \ldots, M]$
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. (2)
6: **for** each motion pattern b_j **do**
7: Grow a single $\mathcal{T}_{GP}^j$ tree rooted at $(x(t), y(t))$ using b_j (Algorithm 1)
8: Using $\mathcal{T}_{GP}^j$, compute means and variances of predicted distribution $(\hat{x}_j(\tau), \hat{y}_j(\tau))$, $\forall \tau \in [t + \delta t, t + 2\delta t, \ldots, t + T_h]$
9: **end for**
10: $p(\hat{x}(\tau), \hat{y}(\tau)) \leftarrow \sum_j p(\hat{x}_j(\tau), \hat{y}_j(\tau)) \times p(b_j(t)|x_{0:t}, y_{0:t})\ \ \forall \tau$ using Eq. (8)
11: $t \leftarrow t + dt$
12: **end while**

patterns for the target vehicle. We assume that the length of the prediction problem is T s. We also assume that the prediction time horizon is T_h s. 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 will typically be in the order of 3 to 5 seconds.[15] The RR-GP algorithm updates its measurement of the target vehicle every dt s. This value is chosen such that it ensures that the inner loop (lines 6-9) of the Algorithm 2 reaches completion before the next measurement update. Finally, the time period of the low-level controller is equal to δt s. A low δt signifies a higher precision of the predicted trajectories, but can slightly affect computation times. In our problem, a suitable range of δt is 0.02s to 0.1s.

The input of the Algorithm 2 is a set of GP motion patterns and the initial probability distribution over the motion patterns. This prior knowledge is typically proportional to the size of the each motion patterns. In line 4, the position of the target vehicle is measured. Then, the probability that the vehicle trajectory belong to each of the M motion patterns is updated. Recall that we assume the measurements are noise-free. For each motion pattern (in parallel), line 7 grows a single-tree RR-GP rooted at the current position of the target vehicle using Algorithm 1. Then, 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 where $\tau \in [t + \delta t, t + 2\delta t, \ldots, t + T_h]$ using position and time information of the nodes and edges of the single-tree output. This process can be parallelized for the different motion patterns since there is no message passing between each the different single RR-GP tree growth operation. Line 10 combines the position prediction from the different single-tree RR-GP output 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 time τ where $\tau \in [t + \delta t, t + 2\delta t, \ldots, t + T_h]$, resulting in probabilistic distribution of the future trajectories of the target vehicle that is based on a mixture of GPs.

Figure 3 shows the trajectories generated by RR-GP in a scenario involving six different motion patterns. One pattern consists of the vehicle navigating above all obstacles from the top. Another intent corresponds to navigating above the first obstacle, then in between the middle obstacles, and finally below the last obstacle. The mobility patterns were learned using synthetic training data. Section V demonstrates of the Multi-Tree RR-GP Algorithm on a simplified example that highlights the benefits the developed approach and allows for a detailed comparison of its performance versus to other GP-based algorithms.

V. Simulated Environment with Human-Driven Target Vehicle

To highlight the advantages of the RR-GP algorithm discussed in Section IV, Algorithm 2 is applied on an example scenario consisting of a single target vehicle traveling in an environment with a single fixed obstacle. Recall that our purpose behind augmenting the motion model's predictions with the RRT component was to both overcome the performance loss due to having to use sparse training data for real-time planning as well as only allowing feasible trajectory predictions. This example was constructed to compare the performance of the RR-GP approach against two comparison algorithms, both using GP mixtures but with one given sparse training data and one given dense training data. We compare them in terms of both accuracy and computation time. The intuition is that augmenting the GP mixture model with the CL-RRT algorithm allows us to achieve similar runtime to the sparse GP mixture while maintaining (or even exceeding) the performance of the dense GP mixture. Note that although we refer to one of the models as sparse-GP, our RR-GP algorithm is trained with the identical data and is therefore equally "sparse".

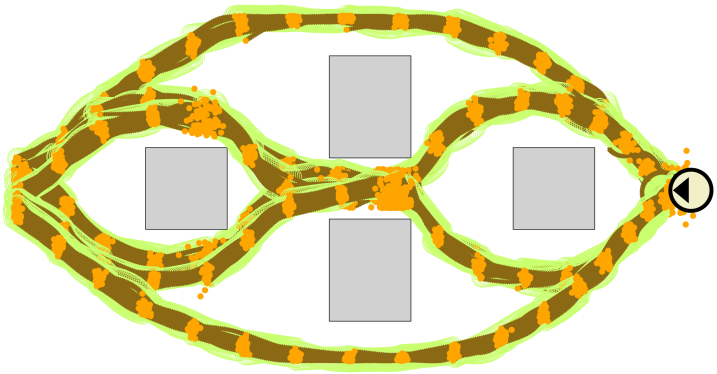


Figure 3: Multiple Tree RR-GP Illustration. RR-GP grows six trees corresponding to six different motion patterns of the target vehicle. A likelihood value is also computed for each tree. The trees and the likelihood values are updated in real-time with every new observation of the position of the target vehicle.

A. Setup

The trajectories were manually generated by driving a car-like vehicle in a simulated urban environment described in Ref. 18. The vehicle uses the iRobot Create software platform[22] with a skid-steered vehicle modified in software to emulate the traditional automotive steering obeying the standard bicycle model

$$\begin{aligned} \dot{x} &= v\cos(\theta), \quad & \dot{y} &= v\sin(\theta), \\ \dot{\theta} &= \frac{v}{L}\tan(\delta), \quad & \dot{v} &= a, \end{aligned} \tag{9}$$

where (x, y) is the rear axle position, v is the forward speed, θ is the heading, L is the wheelbase, a is the forward acceleration, and δ is the steering angle (positive counter-clockwise). The state of the vehicle is $s = (x, y, \theta, v) \in \mathcal{S}$, while the input is $u = (\delta, a) \in \mathcal{U}$, including the constraints $a_{\min} \leq a \leq a_{\max}$ and $|\delta| \leq \delta_{\max}$, where $a_{\min} = -0.7$ m/s^2, $a_{\max} = 0.4$ m/s^2, and $\delta_{\max} = 0.6$ rad. The vehicle follows two motion patterns, i.e., $M = 2$ in Eq. (1). Starting from its initial location, the vehicle either drives to the left of the obstacle or to its right. A total of 28 (21 left, 7 right) trajectories were driven and data was collected at 50 Hz (Figure 4).

Given the training trajectories, two GP motion patterns were learned according to Eqs. (2), (3), (4), and (5). Both our RR-GP algorithm and sprase-GP use data that was downsampled to 1Hz. While ideally we would have liked to run dense-GP with the 50 Hz data, 5 Hz is as dense as we realistically were able to test due to computational limitations. Note that the 5 Hz data is still easily considered dense given the dimensions of the scenario and the maximum speed of the vehicle. The test data consists of 42 withheld trajectories (24 left, 18 right) generated in the same manner as the training data. In testing, the algorithms received simulated measurements from the test trajectories at one second intervals, i.e., $dt = 1s$ in Algorithm 2. For each time step, sparse-GP, dense-GP, and RR-GP are run using the current state of the target vehicle over a time horizon $T_h = 9s$. In the RR-GP implementation, the control time step is $\delta t = 0.02$ s, while the GP samples are produced at $\Delta t = 1$ s. We follow Ref. 23 in calculating prediction error as the root mean square (rms) difference between the true position (x, y) and mean predicted position $(\hat{x}, \hat{y})$. The mean is computed using Eq. 8 for the sparse-GP and dense-GP techniques, and the multi-tree probabilistic distribution for the RR-GP approach. The predictions errors are averaged for all the 42 trajectories at each time step, and summarized in the following section.

B. Simulation Results

In this section we first highlight some features of the results for the RR-GP algorithm that were obtained in our scenario and then compare the prediction accuracy and computation time of RR-GP, sparse-GP, and dense-GP.

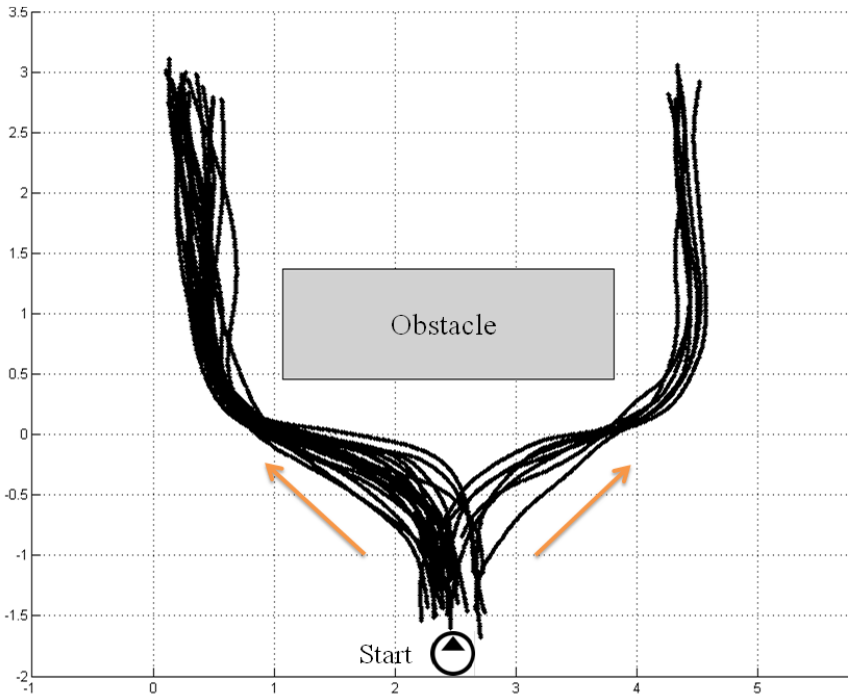


Figure 4: Training trajectories generated in the simulated road environment from the two motion patterns.

Figure 5 demonstrates the dynamic feasibility and collision avoidance features of the RR-GP approach on a test trajectory that belongs to the left motion pattern. At time $t = 0$ s (Figure 5a) the target vehicle is pointing upwards and the RR-GP outputs two trees using Algorithm 2, one for each GP motion pattern. At current time $t = 1$ s (Figure 5b) the vehicle has moved upwards and slightly rotated left. As in every step, RR-GP updates the likelihoods (Figure 7) of each motion pattern and generates two new dynamically feasible trees. After receiving this second measurement, Figure 5b shows the updated RR-GP output trees. Simply due to the forward movement and slight left rotation, our algorithm is starting to find more feasible left trajectories than right trajectories (this can be seen by comparing the trajectories as they pass the bottom left and bottom right corner of the obstacle).

Finally, at time $t = 2$ s (Figure 5c) the vehicle has more clearly turned to the left; the RR-GP algorithm returns an incomplete right tree which signals to our algorithm that all connections to grow the tree further along the right motion pattern failed. This desired behavior as the trajectory did, in fact, go left. Note that Figures 5a, 5b, and 5c show how the predicted left trees are dynamically feasible and only grown to collision free regions which is key to better predictions.

Figure 6 shows the performance at different times of the scenario of the three algorithms in terms of prediction the error between the true value and the predicted mean position of the target vehicle. At the start, when time $t = 0$ s (Figure 6a), the three plots have very close performance. This is expected since the starting position of the target vehicle in the testing phase are almost identical to those in the training phase leading to mostly collision-free predictions from the comparison approaches samples. This leads to sparse-GP, dense-GP, and RR-GP generating similar predictions. The magnitude of the prediction error grows to around 2.7 m, which is due to the ambiguity of the target's intention at the start (See Figure 7).

After one second has elapsed (Figure 6b), the vehicle has moved to a new position and the likelihood values of each motion pattern have been updated (Figure 7). The likelihood value of the wrong motion pattern has slightly decreased, leading to lower errors for all three algorithms. More interestingly, the error of the RR-GP is lower than both the Sparse-GP and Dense-GP starting $t = 5$ s which is typically around the time the target first encounters the obstacle. In fact, the error of the RR-GP algorithm at $t = 9$ s is 7% lower than the sparse-GP and 11% lower than the dense-GP when predicted from only two observations. This error can be explained by the difficulty of the comparison methods have in accurately predicting trajectories around the corners of the obstacles. This can further be seen by the jump in the difference in the prediction errors between RR-GP and the comparison algorithms between $t = 5$ s and $t = 6$ s

Table 1: Average Computation Times (over 42 runs)

Algorithm	Avg. Computation Time (s)
Sparse-GP (1 Hz)	0.2
Dense-GP (5 Hz)	53.5
RR-GP	0.6

where the vehicle predicted trajectories are close to the corners. Our RR-GP algorithm, by having collision avoidance and dynamically feasibility embedded in it, handles this situation which leads to lower prediction error. After $t = 6$ s, the difference in error remains, keeping the RR-GP performance on top.

Generally, after three seconds have elapsed, the probability of the correct motion pattern has approached 1 (Figure 7), which explains the decreased level of prediction error among the three algorithms (Figure 6c). The dense-GP slightly outperforms both sparse-GP (1Hz) and RR-GP between $t = 3$ s and $t = 4$ s but falls significantly after $t = 4$ s. From $t = 5$ s to $t = 9$ s RR-GP performs the best. In fact, the error of the RR-GP algorithm at $t = 9$ s is 6% lower than the sparse-GP and 34% lower than the dense-GP.

Table 1 summarizes the average computation times per iteration of the three algorithms over the 42 testing paths. The sparse-GP has the lowest computation time with our RR-GP algorithm performing only slightly worse. Out of the 0.6 s RR-GP spent, 0.4 s was spent in the RR-GP tree generation while the remaining 0.2 s was used propagating the GP samples. On the other hand, the Dense-GP had a computation time is significantly worse than the other two approaches. This is expected since GPs scale poorly with the training data size and data of this density renders the GP mixture model useless for any real-time implementation.

In summary, this simulated environment with a human-driven target vehicle showed that the RR-GP algorithm performed consistently better than the sparse-GP and dense-GP in the long-term prediction of the target vehicle. When the approaches compared in this section are used in combination with a planner in this type of domains, we often desire to take actions at frequencies approaching 20 Hz. It is important to highlight that the RR-GP can output predict trajectories at arbitrarily high output frequency, which is significantly higher than both the sparse-GP (1Hz) and the dense-GP (5Hz). While we could augment the comparison approaches with some form of interpolation technique, our approach systematically guarantees collision avoidance and dynamic feasibility of the trajectory predictions. Another feature of the RR-GP algorithm is its efficient computation time (Table 1), which is low enough to be suitable for real-time implementation in collision warning systems or probabilistic path planners for autonomous systems.

VI. Conclusion

In this work we presented an approach to improve the location predictions of agents with uncertain intentions for collision avoidance and conflict detection systems. To accomplish this we combined the CL-RRT algorithm with a GP mixture model. The CL-RRT algorithm provided us with a method of incorporating vehicle constraints and world obstacles into trajectory predictions of target agents. The GP mixture model is a nonparameteric Bayesian model that can capture the distribution over an extremely wide range of possible target agent motion patterns. Our RR-GP algorithm, by leveraging these approaches, ensured that the predicted trajectories are both feasible by construction and probabilistically weighted by the statistical model learned from the training data.

Our approach is well suited for problems in continuous domains with complex, poorly understood trajectory patterns. We demonstrated a substantial improvements in runtime and prediction accuracy over a dense-GP approach that required a significantly larger amount of training data. Additionally, we showed a significant improvement in accuracy at the cost of only a small increase in runtime performance in comparison to the sparse-GP approach.

In future work we plan to demonstrate our algorithm's effectiveness on real-world intersection data. We are confident in our algorithm's ability to handle real-world data since the GPs naturally cope with noisy measurements, allowing our algorithm with no modifications, to be applied there. Secondly, we plan to augment our target agent trajectory prediction algorithm with an RRT-based planner.[18] Many trajectory prediction algorithms suffer from being either overly conservative or too simplistic, as discussed in Section I, due to a lack of a target motion model. We can overcome this problem by using our RR-GP algorithm and allowing the training data to describe how the target agents behave.

References

[1]Fuerstenberg, K. C., "New European Approach for Intersection Safety - The EC-Project INTERSAFE," *Proceedings of IEEE Intelligent Transportation Systems*, 2005, pp. 432–436.

[2]Erzberger, H. and Paielli, R., "Concept for next generation air traffic control system." *Air Traffic Control Quarterly*, Vol. 10, No. 4, 2002, pp. 355–378.

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

[4]Miloh, T. and Sharma, S., *Maritime collision avoidance as a differential game*, Institut fur Schiffbau der Universitat Hamburg, 1976.

[5]Lachner, R., "Collision avoidance as a differential game: Real-time approximation of optimal strategies using higher derivatives of the value function," Vol. 3, 2002, pp. 2308–2313.

[6]Kuchar, J. and Yang, L., "A review of conflict detection and resolution modeling methods," *Intelligent Transportation Systems, IEEE Transactions on*, Vol. 1, No. 4, 2002, pp. 179–189.

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

[8]Zhu, Q., "Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation," *Robotics and Automation, IEEE Transactions on*, Vol. 7, No. 3, 2002, pp. 390–397.

[9]Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S., "Learning motion patterns of people for compliant robot motion," *The International Journal of Robotics Research*, Vol. 24, No. 1, 2005, pp. 31.

[10]Sorenson, H., *Kalman filtering: theory and application*, IEEE, 1985.

[11]Mazor, E., Averbuch, A., Bar-Shalom, Y., and Dayan, J., "Interacting multiple model methods in target tracking: a survey," *Aerospace and Electronic Systems, IEEE Transactions on*, Vol. 34, No. 1, 2002, pp. 103–123.

[12]Joseph, J., Doshi-Velez, F., and Roy, N., "A Bayesian Nonparametric Approach to Modeling Mobility Patterns," 2010.

[13]Joseph, J., Doshi-Velez, F., Huang, A. S., and Roy, N., "A Bayesian Nonparametric Approach to Modeling Motion Patterns," Tech. rep., 2011.

[14]Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., and How, J. P., "Real-time Motion Planning with Applications to Autonomous Urban Driving," *IEEE Transactions on Control Systems Technology*, Vol. 17, No. 5, September 2009, pp. 1105–1118.

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

[16]Joseph, J., Doshi-Velez, F., and Roy, N., "A Bayesian Nonparametric Approach to Modeling Mobility Patterns," *AAAI*, 2010.

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

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

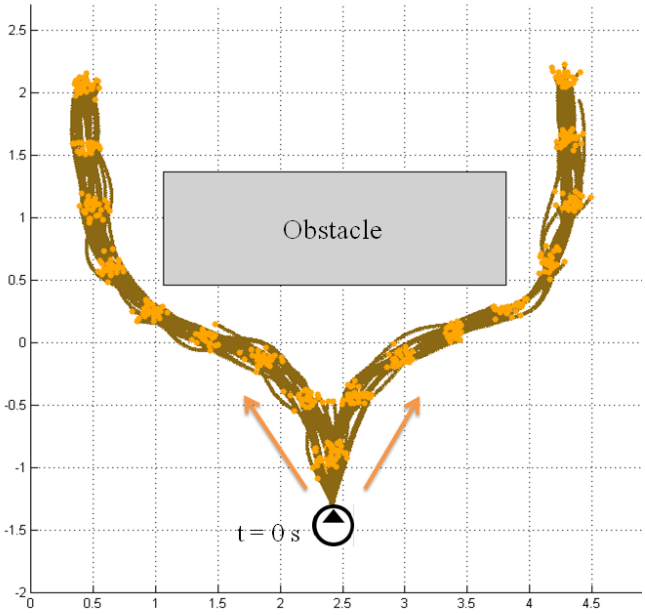
[19]Fletcher, L., Teller, S., Olson, E., Moore, D., Kuwata, Y., How, J., Leonard, J., Miller, I., Campbell, M., Huttenlocher, D., Nathan, A., and Kline, F.-R., "The MIT - Cornell Collision and Why it Happened," *Journal of Field Robotics*, Vol. 25, No. 10, October 2008, pp. 775 – 807.

[20]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, 2006.

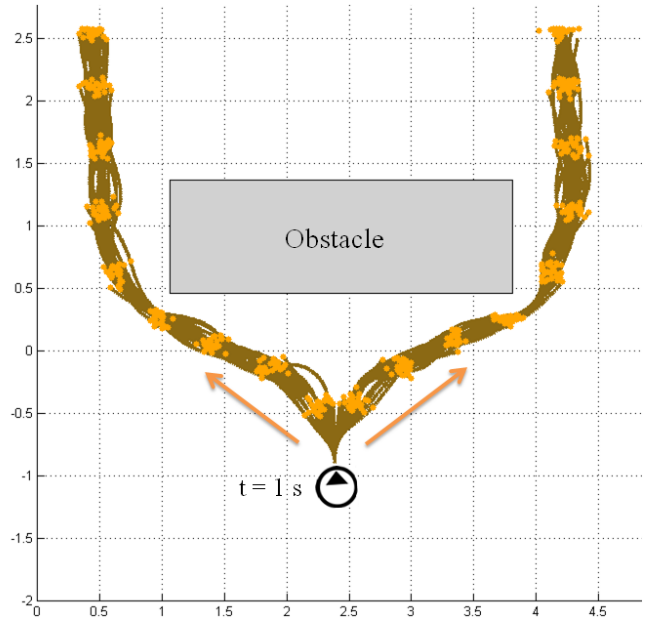
[21]Frazzoli, E., Dahleh, M. A., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, January-February 2002, pp. 116–129.

[22]iRobot, "iRobot: Education & Research Robots," Online `http://store.irobot.com/shop/index.jsp?categoryId=3311368`.

[23]Yepes, J., Hwang, I., and Rotea, M., "New algorithms for aircraft intent inference and trajectory prediction," *Journal of guidance, control, and dynamics*, Vol. 30, No. 2, 2007, pp. 370–382.

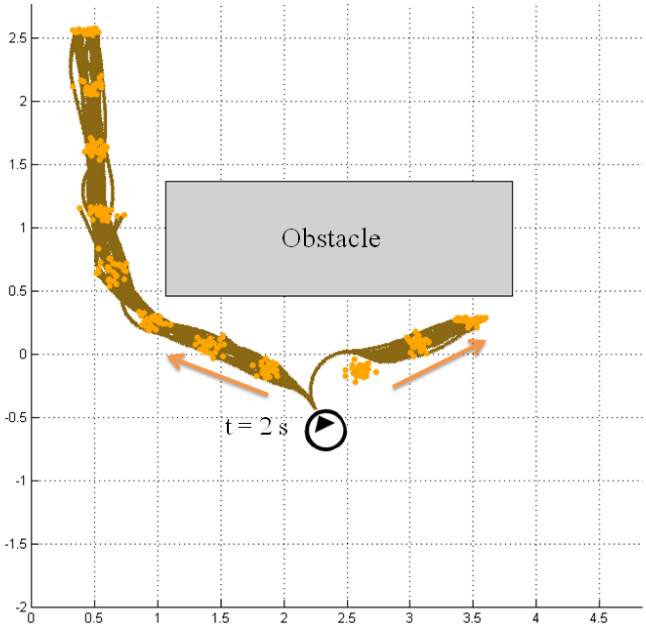


(a) t = 0 s



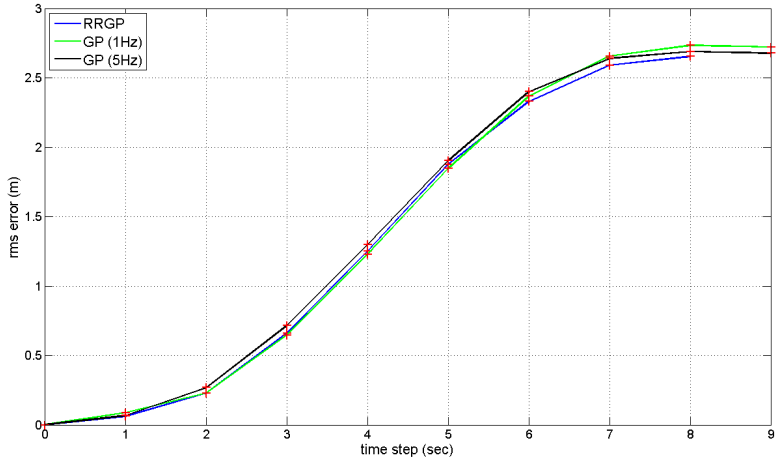
(b) t = 1 s

Figure 5: Snapshots of the dual-tree output of the RR-GP algorithm on a test trajectory following the left GP motion pattern

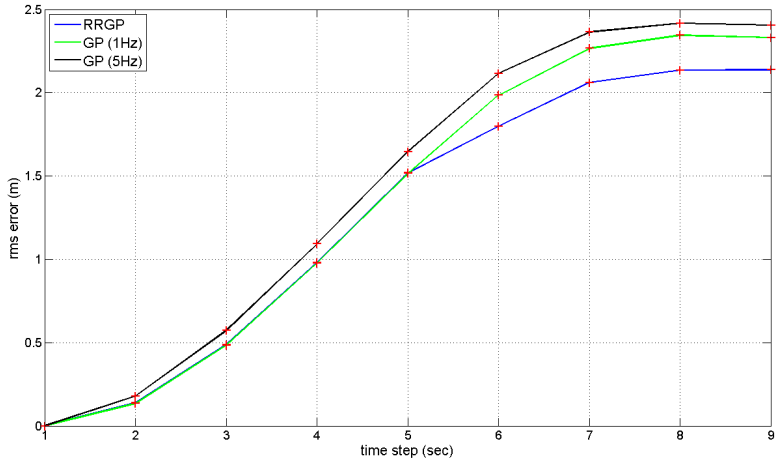


(c) t = 2 s

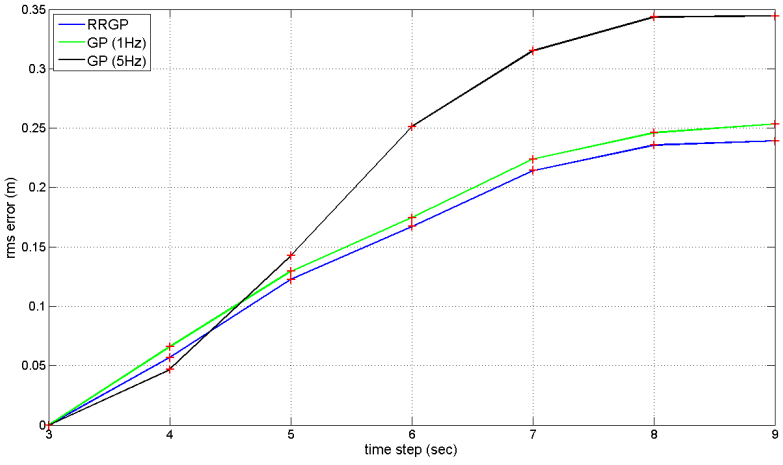
Figure 5: Snapshots of the dual-tree output of the RR-GP algorithm on a test trajectory following the left GP motion pattern



(a) t = 0 s



(b) t = 1 s



(c) t = 3 s

Figure 6: Average position prediction errors (over 42 trajectories) for Sparse-GP (1Hz), Dense-GP (5Hz), and RR-GP algorithms at different times of the example.

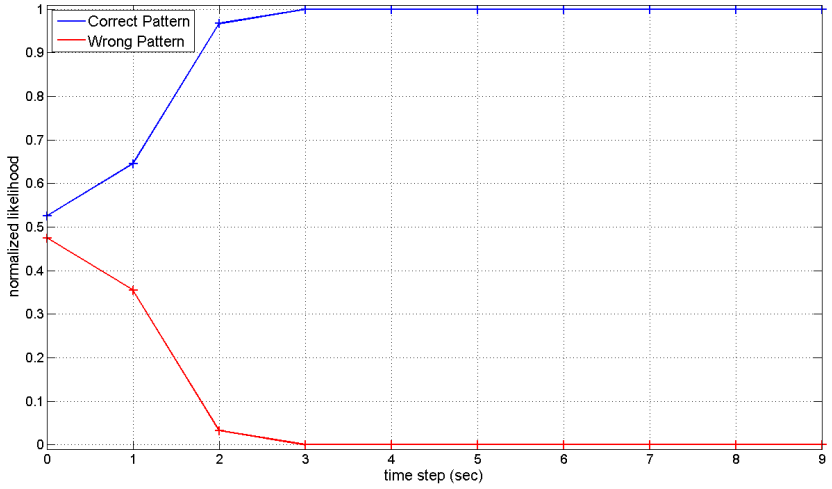


Figure 7: Average normalized likelihoods (over 42 trajectories) for Sparse-GP (1Hz) and RR-GP algorithms as function of time. For example, the values at $t = 2$ s represent the likelihoods of the correct vs. wrong motion patterns after the target vehicle has actually moved for two seconds on its trajectory.