Particle Filtering and Reweighting Smoothing Motion Planning

Mason Lieb

University of Kansas – Department of Mechanical Engineering

M.S. Project Report

Abstract

This study describes an application of particle filtering and reweighting particle smoothing to solving a motion planning and decision-making problem for autonomous road vehicles. The method utilizes a set of driving requirements as a reference for the generated trajectory to track, where these driving requirements can be determined a priori, and no global waypoints are required. The motion planning problem is viewed as an estimation problem which is solved with the particle filtering and reweighting smoothing (PF-RS) motion planning method. The driving requirements and therefore the reference can vary as the selected driving mode changes, which characterizes several different maneuvers a vehicle might make in a standard highway driving scenario. The decision of which driving mode to select is part of the solution framework. Collision avoidance and road boundary constraints are built into the filtering and decision-making to enhance the safety of the generated trajectories. The trajectories are applied in a receding horizon manner to facilitate the dynamic environment. The specific scenario considered in this study is overtaking, with obstacle vehicles travelling at different speeds in each of the two lanes in the simulated highway. A GitHub repository with animated simulation results is included with this study. The results show the capability of the method to track reference lane centers and velocities while satisfying safety constraints with multiple driving mode changes throughout the simulation.

List of Symbols

p_{X_k}	Global x-coordinate of the ego vehicle
p_{Y_k}	Global y-coordinate of the ego vehicle
ψ_k	Heading (yaw) angle of the ego vehicle [rad]
v_{x_k}	Longitudinal velocity of the ego vehicle [m/s]
δ_k	Steering rate [rad/s]
β_k	Body slip angle (rad)
L	Vehicle wheelbase
l_r	Distance from vehicle center to vehicle rear axle
v_{nom}	Nominal velocity (speed limit) [m/s]
e_L	Lane centerline tracking error [m]
e_ψ	Heading angle tracking error [rad]
g_o	Obstacle barrier function for obstacle $o = 1, \dots, N_{obs}$
g_b	Road boundary barrier function
$\boldsymbol{r}$	Reference vector for measurement tracking
T	Planning horizon length [s]
Δt	Discretization time [s]
$\boldsymbol{x}_k$	State vector
$\boldsymbol{u}_k$	Input vector
$\overline{\boldsymbol{x}}_k$	Virtual system state vector with augmented controls
w	Lane width [m]
$\mathcal{M}$	Set of driving modes
$\boldsymbol{q}_k$	Input noise
$\boldsymbol{Q}$	Input noise covariance
$\overline{\boldsymbol{q}}_k$	Virtual system process noise
$\overline{\boldsymbol{Q}}$	Virtual system process noise covariance
$\boldsymbol{y}_k$	Measurement vector
$h(\cdot)$	Measurement function $h\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$
α, β	Softplus barrier function parameters for obstacle avoidance constraint
m, n	Softplus barrier function parameters for road boundary constraint
$\overline{\boldsymbol{x}}_k^i$	Particle i
N	Number of particles
w_k^i	Filtering weight for particle i
$w_{k\|k+1}^i$	Smoothing weight for particle i
$\delta(\cdot)$	Dirac delta function
a, b, c	Ellipse parameters
Δ	Edge-to-edge distance between ellipses
Δ_l	Edge-to-edge distance between ego vehicle and only the obstacle in same lane
$\gamma, \epsilon, \zeta, \lambda$	Weighting parameters for trajectory cost function
ν, η	Exponential parameters of Softplus barrier functions in trajectory cost function
ξ, ρ	Offset terms for barrier function terms of trajectory cost function

Introduction

There is ever-increasing interest in autonomous transportation systems in our modern society as seen in the popularity of adaptive driver assistance systems (ADAS) such as General Motors Super Cruise, Mercedes-Benz Drive Pilot, Ford BlueCruise, and Tesla Autopilot. To implement such systems, there are many different modules that must work together to receive instructions from a high-level navigation system, make decisions, perform motion planning, and execute controller inputs for low-level systems like steering, acceleration, or braking actuations. In addition to performing these actions, there are sensors that provide the necessary information to facilitate the decision making and motion planning processes. Figure 1 outlines a high-level overview of the system schematics for an autonomous vehicle [3].

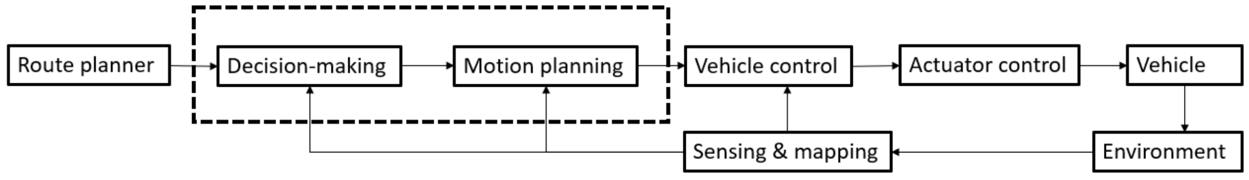

Fig. 1 Autonomous vehicle high-level system architecture.

The route planner sets the high-level goals of the autonomous driving system. An example of this sort of high-level route planning would be a mapping application directing a driver to take a certain exit from a highway or turn a certain direction at an intersection. The next lowest highest level is the decision-making system. This system determines things like which lane the vehicle should be in, which will be described by the *driving mode* in this study. The driving modes considered are to stay in the current lane, to change lanes to the left, and change lanes to the right. These modes capture most of the possible behaviors in a highway driving scenario apart from emergency braking, which can be handled by a separate emergency system that would also handle other evasive maneuvers.

Below the decision-making level is the motion planner, which determines a trajectory that the vehicle should take based on the decision-making output and the sensor information. The generated trajectory should be collision-free and dynamically feasible while adhering to all other safety or road constraints like minimum distance to obstacles and speed limit.

Vehicle control, actuator control, and final motion of the vehicle is handled by lower-level controllers, and the vehicle interacts with its dynamic environment which requires continuous sensing and mapping to determine the vehicles position relative to the road and other vehicles or static obstacles.

The sensing and mapping in a current ADAS implementation could include radar and lidar sensors, stereo camera systems, wheel speed and steering angle sensors, and GPS. With the information provided from the sensing and mapping, the vehicle should be able to determine where it is relative to the road it is driving on, identify obstacles and their relative location to the vehicle, as well as identifying the road constraints such as speed limit.

Motion planning for autonomous road vehicles has several different classes of solutions, including graph search-based planners, interpolating curve planners, numerical optimization, and sampling-based planners [1]. Example implementations of each class of planning methods can be found in [1], as well as descriptions of their advantages and disadvantages. For this study, there are inspirations drawn from the numerical optimization and sampling-based classes of motion planning solutions.

An example of an optimization-based solution method is model predictive control (MPC), where over some finite horizon, a trajectory is optimized over the inputs to the system. These inputs are then applied in a receding horizon manner, where only the first generated input is applied to the true system while the optimization repeats over the next horizon. MPC has been applied to robotic motion planning problems for legged robots [2] and autonomous vehicles [3]. A challenge facing the MPC method for autonomous vehicles is nonconvexity, which may cause the optimization to fall into local minima.

From a sampling-based perspective, one can sample configurations or inputs. Rapidly exploring random trees were originally proposed as a method of computing collision-free kinodynamic trajectories for high degree-of-freedom problems [4]. RRTs were applied to autonomous driving motion planning in [5] Since their introduction, they have been expanded into the probabilistically complete RRT* [6] where, as the number of samples approaches infinity, the probability that a feasible trajectory will be found if one exists approaches one. These methods sample the configuration space and will generate satisfactory paths, but for an autonomous vehicle, the steering inputs must then be calculated via inverse steering dynamics. By constructing a virtual system that augments the system inputs with the states, sampling the augmented state space also samples the control inputs. This avoids the calculation of inverse steering laws, and by propagating the samples through the system's dynamic model, the resulting trajectories are guaranteed to be dynamically feasible. This study does not seek to find an optimal trajectory, as in normal driving there are many trajectories that are suitable and satisfy the driving requirements and there is not a general requirement to minimize the amount of time a maneuver takes. Thus, the trajectories that are developed with this method are not necessarily optimal but are suitable and safe.

To accomplish this, the samples drawn are used as particles in the particle filtering framework. The generated trajectories from filtering are then smoothed via a reweighting particle smoother before the inputs are extracted and applied in a receding horizon manner. Applying particle filtering to perform model predictive control was applied in [7] [8]. The inspiration for this study is the particle filter motion planning and decision-making algorithm presented in [9] [10].

Report Format

First, the high-level motion planning problem is outlined with the driving scenario considered, the vehicle model, measurement model, and the driving modes for decision-making. Then, the particle filtering and smoothing solution methodology is described. Next, a detailed description for each measurement and the driving mode selection scheme is given. Finally, simulation results are displayed before final statements.

Problem Formulation

Driving Scenario

This study considers the problem of generating control inputs that drive a vehicle such that its trajectories satisfy a given reference that includes road requirements such as speed limit and safety constraints like road boundary satisfaction and obstacle avoidance. The driving scenario is on a straight two-lane highway, with dynamic obstacles. Under normal highway driving conditions, acceleration and steering inputs are relatively small, which means that the executed maneuvers are well within the vehicle's performance envelope. The motion planning module of an autonomous vehicle would feed control inputs to lower-level controllers for the vehicle actuators, which in a practical application would handle the errors between this kinematic model and the true vehicle dynamics.

It is assumed that the ego vehicle can measure its own position relative to the road boundaries and lane centerlines, as well as the positions and velocities of the obstacle vehicles. Other available information includes the speed limit and the ego vehicle's heading angle relative to the road.

Vehicle Model

For the reasons stated above, the kinematic bicycle (or single-track) model provides satisfactory accuracy. The equations of motion for this model are below in equation 1 [9].

$$\dot{x} = \begin{bmatrix} \dot{p}_X \\ \dot{p}_Y \\ \dot{\psi} \\ \dot{v}_x \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \frac{v_x \cos(\psi+\beta)}{\cos(\beta)} \\ \frac{v_x \sin(\psi+\beta)}{\cos(\beta)} \\ \frac{v_x \tan(\delta)}{L} \\ u_1 \\ u_2 \end{bmatrix} \tag{1}$$

$$\beta = \arctan\left(\frac{l_r \tan(\delta)}{L}\right)$$

Where β is the kinematic body slip angle, L is the wheelbase of the vehicle and l_r is the distance from the center of the vehicle to the rear axle. The control inputs are taken to be the steering rate and the acceleration, which guarantees smooth trajectories. The vehicle state variables are the vehicles x- and y-positions (p_X, p_Y), yaw (heading) angle (ψ), longitudinal velocity (v_x), and steering angle (δ). The state variables are illustrated below in figure 2. The inputs to the system are acceleration and steering rate, which guarantees smooth trajectories when the inputs are applied to the actual system.

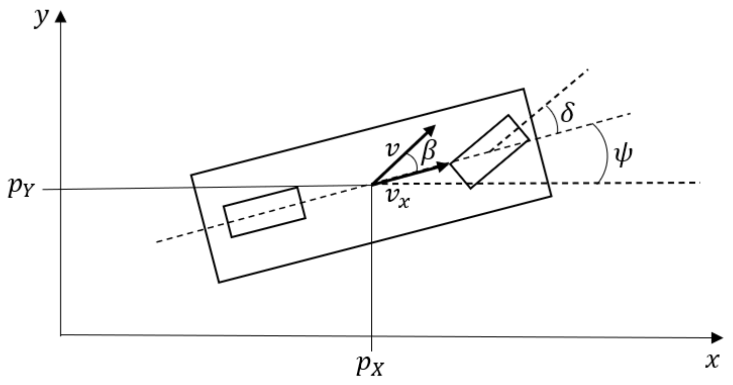

Figure 2. Illustration of state variables for kinematic bicycle model.

The model is discretized to be compatible with the sequential estimation procedure of the particle filter which is described in detail in the Problem Solution – *Preliminaries on Particle Filtering and Smoothing* section.

$$x_{k+1} = x_k + \dot{x}_k \Delta t$$

$$\begin{bmatrix} p_X \\ p_Y \\ \psi \\ v_x \\ \delta \end{bmatrix}_{k+1} = \begin{bmatrix} p_{X_k} + \dfrac{v_{x_k} cos(\psi_k + \beta_k)}{cos(\beta_k)} \Delta t \\ p_{Y_k} + \dfrac{v_{x_k} sin(\psi_k + \beta_k)}{cos(\beta_k)} \Delta t \\ \psi_k + \dfrac{v_{x_k} tan(\delta_k)}{L} \Delta t \\ v_{x_k} + u_{1_k} \Delta t \\ \delta_k + u_{2_k} \Delta t \end{bmatrix} = f(x_k, u_k) \tag{2}$$

Measurement Model

The reference that the vehicle should track needs to satisfy several requirements for generated trajectories to be safe and efficient. The first set of requirements can be thought of as the road requirements, which are determined by the road itself. These requirements include tracking the road's speed limit, denoted as v_{nom} or nominal velocity, track the centerline of the vehicle's target lane, do not leave the road, and maintain no deviation in heading angle from the road angle. The second set of requirements concerns the obstacles present in the scenario, namely the other vehicles driving on the road. The only requirement concerning the obstacle vehicles is that the ego vehicle maintain a minimum distance from them that is a function of the ego vehicle's velocity. This means that the faster the ego vehicle is travelling, the larger the collision avoidance zone will be. This is illustrated below in figure 3.

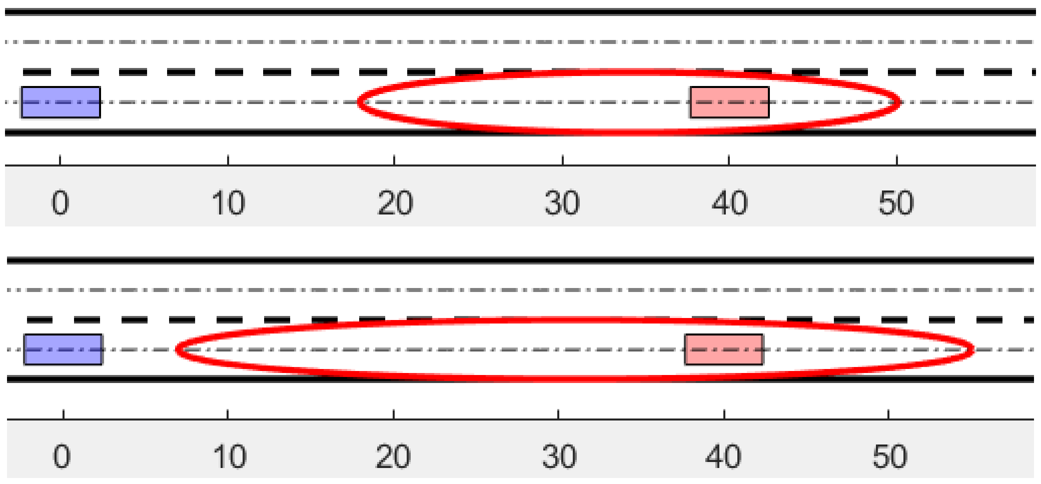

Figure 3. (Top) Obstacle region with ego vehicle speed of 20 m/s. (Bottom) Obstacle region with ego vehicle speed of 30 m/s

This reference is assumed fixed during each planning horizon, so only a single realization of the reference vector is required, denoted as r. The reference vector is described below in equation 3.

$$\boldsymbol{r} = [v_{nom} \quad e_L = 0 \quad g_o = 0 \quad g_b = 0 \quad e_\psi = 0]^T \tag{3}$$

The reference has four terms for the road conditions and $o = 1, \ldots, N_{obs}$ terms for obstacle avoidance.

Driving Modes

Distinct driving modes are required to determine the reference $\boldsymbol{r}$. The driving modes will tell the vehicle what velocity and lane centerline to track. Driving modes can be selected in different ways. One method is to consider a feasible set of driving modes $\mathcal{M}$ and generate a single trajectory according to each driving mode. Then, the "best" trajectory as determined by a cost function will be selected for execution in a receding horizon manner. Another method is by generating samples from a transition probability $m_j \sim p(m_j|Z)$ of switching to mode j conditioned on the currently available road and obstacle information Z.

The benefit of deterministically testing each viable driving mode is that each mode is guaranteed to be checked. This contrasts to the transition probability sampling, where it is possible that a feasible mode may not be checked during trajectory generation unless there is a large enough amount of samples which comes with significant computational cost. However, if the transition probabilities are designed well and appropriately conditioned on currently available information, then that method may lead to a more optimal trajectory, as the mode that is expected to produce high quality trajectories will get sampled more, and more possible trajectories corresponding to that mode will be generated.

The considered driving modes are lane-keeping, change lanes left, change lanes right, and stop. These are abbreviated and described in the set $\mathcal{M} = \{LK, CLL, CLR\}$. The nominal velocity for all driving modes is the speed limit. For lane-keeping, the target lane is the current lane. For the lane-changing modes, the target lane is one lane left or right of the current lane. In a practical sense, lane changing is shifting the reference lane by *w* meters, where *w* is the width of a lane. The other elements of the reference signal (obstacle avoidance, road boundary constraint, and heading angle constraint) are all left unchanged for all driving modes, as those elements always need to be satisfied. However, each driving mode can have varied covariance matrices for the input and measurement noise. The design of the individual covariance matrices used for the simulations in this study is described in the *Simulation Study* section.

With the system and measurement model described here, the problem being addressed is how to generate a viable sequence of control inputs $u_{k:k+T}$ over the planning horizon that drives the system towards the reference signal. In the next section, the specific solution to this problem used in this study is given.

Problem Solution

System Overview

Particle filtering is used to address the motion planning problem by estimating a trajectory over the planning horizon and applying the generated inputs from said trajectory in a receding horizon fashion. This method is like model predictive control (MPC), where an optimization over the inputs is performed over the planning horizon under a set of constraints, and then those inputs are applied in a receding horizon manner. Equivalence between the solution to a nonlinear model-predictive control problem via constrained optimization and via estimation is described further in [8]. The constraints in this problem are present in the design of the reference signal and the corresponding covariance matrix for the measurement noise. This implementation of constraints results in only soft constraints being present where there are only probabilistic guarantees of constraint satisfaction. These constraints are made harder via the implementation of barrier functions in the measurement function $h(\cdot)$.

The goal is to estimate a posterior distribution of states and inputs given the reference.

$$p(\boldsymbol{x}_{k:k+T}, \boldsymbol{u}_{k:k+T}|\boldsymbol{r}) \tag{4}$$

The reference $\boldsymbol{r}$ does not carry a subscript because it is constant over each planning horizon. The distribution in (4) describes a joint posterior over the states and inputs over a horizon of length T.

To convert the model into one that lends itself to the sequential estimation of the particle filter, the controls are modeled as random noise acting on the system. It is desired that the noise be additive, as this simplifies the modeling of the process. The system is rewritten as follows, starting from the previously described system dynamics.

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5}$$

The inputs are replaced with random process noise $\boldsymbol{w}_k$. Then the system is rewritten in terms of the augmented state vector $\overline{\boldsymbol{x}}_k$.

$$\boldsymbol{u}_k = \boldsymbol{q}_k \tag{6}$$

$$\overline{\boldsymbol{x}}_{k+1} = f(\overline{\boldsymbol{x}}_k) + \overline{\boldsymbol{q}}_k \tag{7}$$

$$\overline{\boldsymbol{x}}_k = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{bmatrix} = \begin{bmatrix} p_X \\ p_Y \\ \psi \\ v_x \\ \delta \\ \dot{v}_x \\ \dot{\delta} \end{bmatrix}_k , \overline{\boldsymbol{q}}_k = \begin{bmatrix} \boldsymbol{0}_{nx\times 1} \\ \boldsymbol{q}_k \end{bmatrix} \tag{8}$$

In this *virtual system*, the mean and covariance of the process noise q_k can be designed to constrain the generated inputs. The noise is modeled as white Gaussian, hence has zero mean and a $n_u \times n_u$ covariance matrix $\boldsymbol{Q}$, where n_u is the dimension of the control input space (in this case $n_u = 2$).

$$\boldsymbol{q}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}) \tag{9}$$

Thus, the virtual system also follows a Gaussian distribution with mean $\overline{\boldsymbol{x}}_k$ and covariance $\overline{\boldsymbol{Q}}$, where $\overline{\boldsymbol{Q}}$ is a block diagonal matrix constructed as follows.

$$\overline{\boldsymbol{Q}} = \begin{bmatrix} \sim\boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q} \end{bmatrix} \tag{10}$$

Here, the lower right submatrix is the tuned covariance for the inputs and the upper $n_x \times n_x$ submatrix is diagonal with each element set to be approximately zero. This is done to facilitate numerical stability for the reweighting smoother, discussed later in this section. The covariance for the inputs can vary according to the driving mode, however the process noise for the state variables of the original system (upper left submatrix) is maintained as constant.

Note that the values corresponding with the heading angle ψ and the steering angle δ are much smaller than the other values. This is because these values are in radians, so the magnitude of ψ and δ is much smaller than that of the other states.

The measurements $\boldsymbol{y}_k$ are designed to evaluate how well the current vehicle augmented state satisfies the driving requirements in the reference vector $\boldsymbol{r}$.

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{11}$$

The measurement noise $\boldsymbol{v}_k$ is taken to be white Gaussian with covariance $\boldsymbol{R}$, which is designed to put more emphasis on certain driving requirements for safety purposes.

$$\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}) \tag{12}$$

For example, the covariance term for velocity tracking could be set to be relatively large, which allows for a decent amount of deviation from the reference velocity. This is balanced by setting tight constraints on obstacle avoidance or lane centerline tracking. The measurement function $\boldsymbol{h}(\cdot)$ is described below.

$$\boldsymbol{h}(\boldsymbol{x}_k) = \begin{bmatrix} \boldsymbol{x}_k(4) \\ e_L \\ g_o(\boldsymbol{x}_k) \\ g_b(\boldsymbol{x}_k) \\ e_\psi \end{bmatrix} = \begin{bmatrix} longitudinal\ velocity \\ lateral\ lane\ center\ tracking\ error \\ obstacle\ barrier\ function \\ road\ boundary\ barrier\ function \\ heading\ angle\ tracking\ error \end{bmatrix} \tag{13}$$

The inputs are not considered in the measurement function, so the original state vector x_k is the input. The first measurement is direct state feedback of the vehicle's longitudinal velocity. The remaining measurements should nominally be zero and are described in further detail in the *Measurement Model* section. Before expanding on the measurements, the estimation methodology is described in detail.

Preliminaries on Particle Filtering and Smoothing

Forward Filtering

Particle filtering is an estimation method that can achieve arbitrarily good estimates and can handle nonlinear and potentially multimodal non-Gaussian signals. Further detailed derivation of the particle filter used in this study is provided in the appendix, but the main filtering equations for sequential importance sampling/resampling are covered here.

Particle filtering is a general method of finding expectations over the posterior distribution. In the implementation here, the reference is constant over the planning horizon, however that is not true in a general case.

$$\mathbb{E}[g(\boldsymbol{x})|\boldsymbol{y}_{1:T}] = \int g(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{y}_{1:T})d\boldsymbol{x} \tag{14}$$

Here, $g(\boldsymbol{x})$ is an arbitrary function of the random variable $\boldsymbol{x}$ which represents the latent state vector of the system. This integral generally does not have a closed-form solution, so Monte Carlo sampling-based methods are a common choice for solving this type of problem. For the integral in equation 14, if one could draw samples directly from the posterior distribution, then the evaluation of the integral could be approximated as follows.

$$\mathbb{E}[g(\boldsymbol{x})|\boldsymbol{y}_{1:T}] = \int g(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{y}_{1:T})d\boldsymbol{x} \approx \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}^i) \tag{15}$$

The convergence of this approximation is guaranteed by the central limit theorem [11]. Unfortunately, the true posterior distribution may be difficult to draw samples from, so importance sampling can be leveraged to get around this challenge. Importance sampling is a method of drawing samples from an importance distribution, then by applying weights to those samples determined by how well that sample matches the target distribution (the true posterior), the target distribution can be approximated.

$$\int g(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{y}_{1:T})d\boldsymbol{x} = \int \left[g(\boldsymbol{x})\frac{p(\boldsymbol{x}|\boldsymbol{y}_{1:T})}{\pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})}\right]\pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})d\boldsymbol{x}$$
$$\approx \frac{1}{N}\sum_{i=1}^{N}\frac{p(\boldsymbol{x}|\boldsymbol{y}_{1:T})}{\pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})}g(\boldsymbol{x}^i) = \sum_{i=1}^{N}\widetilde{w}^i g(\boldsymbol{x}^i) \qquad (16)$$

Where $\boldsymbol{x}^i \sim \pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})$. Here, the weights are defined as

$$\widetilde{w}^i = \frac{1}{N}\frac{p(\boldsymbol{x}^i|\boldsymbol{y}_{1:T})}{\pi(\boldsymbol{x}^i|\boldsymbol{y}_{1:T})} \qquad (17)$$

An example of importance sampling is shown below with a simple case of a standard normal distribution as the target (acting as the posterior in this problem) and a uniform distribution over the range of [-5,5] acting as the proposal.

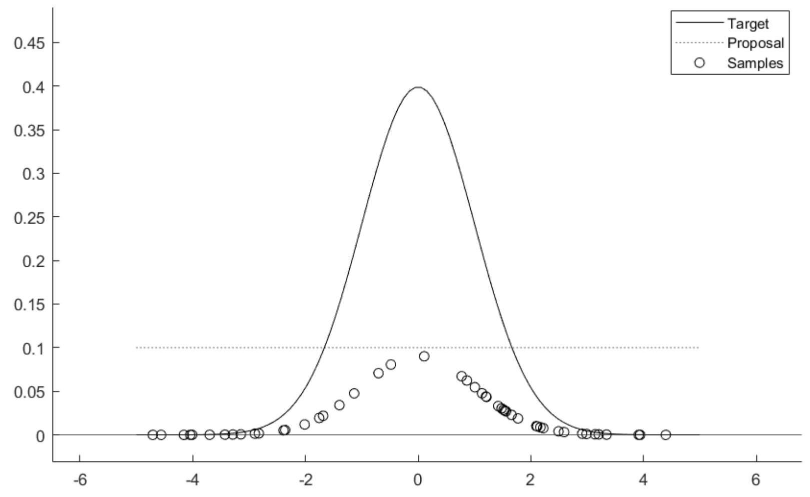

Figure 4. Importance Sampling Example. The circles represent the approximation formed by the weighted samples.

Problems can arise with this method due to difficulty in evaluating the full posterior at each sample $p(\boldsymbol{x}^i|\boldsymbol{y}_{1:T})$. Bayes' rule can be applied to the posterior distribution to provide a different method of arriving at an expression for normalized weights while avoiding this evaluation. First, Bayes' rule is applied to the posterior distribution for the samples.

$$p(\boldsymbol{x}^i|\boldsymbol{y}_{1:T}) = \frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x}^i)p(\boldsymbol{x}^i)}{p(\boldsymbol{y}_{1:T})} = \frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x}^i)p(\boldsymbol{x}^i)}{\int p(\boldsymbol{y}_{1:T}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}} \qquad (18)$$

The terms in the numerator can generally be evaluated, but another importance sampling approximation is required to compute the marginal likelihood term in the denominator.

$$\begin{aligned}
\mathbb{E}[g(\boldsymbol{x})|\boldsymbol{y}_{1:T}] &= \int g(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{y}_{1:T})d\boldsymbol{x} = \frac{\int g(\boldsymbol{x})p(\boldsymbol{y}_{1:T}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}}{\int p(\boldsymbol{y}_{1:T}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}} \\
&= \frac{\int \left[\frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x})p(\boldsymbol{x})}{\pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})}\right] g(x)\pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})d\boldsymbol{x}}{\int \left[\frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x})p(\boldsymbol{x})}{\pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})}\right] \pi(\boldsymbol{x}|\boldsymbol{y}_{1:T})d\boldsymbol{x}} \\
&\approx \frac{\frac{1}{N}\sum_{i=1}^{N} \frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x}^i)p(\boldsymbol{x}^i)}{\pi(\boldsymbol{x}^i|\boldsymbol{y}_{1:T})} g(\boldsymbol{x}^i)}{\frac{1}{N}\sum_{j=1}^{N} \frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x}^j)p(\boldsymbol{x}^j)}{\pi(\boldsymbol{x}^j|\boldsymbol{y}_{1:T})}} \qquad (19) \\
&= \sum_{i=1}^{N} \left[\frac{\frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x}^i)p(\boldsymbol{x}^i)}{\pi(\boldsymbol{x}^i|\boldsymbol{y}_{1:T})}}{\sum_{j=1}^{N} \frac{p(\boldsymbol{y}_{1:T}|\boldsymbol{x}^j)p(\boldsymbol{x}^j)}{\pi(\boldsymbol{x}^j|\boldsymbol{y}_{1:T})}}\right] g(\boldsymbol{x}^i) = \sum_{i=1}^{N} w^i g(\boldsymbol{x}^i)
\end{aligned}$$

With this process, the final expression for the sample weights is normalized and only requires evaluation of the measurement likelihood and the prior for each sample. The posterior distribution can then be approximated by the samples and their weights by equation 20 below.

$$p(\boldsymbol{x}|\boldsymbol{y}_{1:T}) \approx \sum_{i=1}^{N} w^i \delta(\boldsymbol{x} - \boldsymbol{x}^i) \qquad (20)$$

Here, $\delta(\cdot)$ is the Dirac delta function. Up to this point, the estimated density is the full posterior. If measurements are obtained at each time step, which is the case in this study, then the filtering density needs to be updated sequentially as new information becomes available. This requires extending the above method to sequential importance sampling.

Sequential importance sampling in general considers Markov chain systems that have dynamics of the following form.

$$\begin{aligned}
&\boldsymbol{x}_k \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \\
&\boldsymbol{y}_k \sim p(\boldsymbol{y}_k|\boldsymbol{x}_k)
\end{aligned} \qquad (21)$$

Instead of estimating the full posterior $p(\boldsymbol{x}|\boldsymbol{y}_{1:T})$ at each time step, the filtering density $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ can be estimated and used to recursively update the full posterior as follows.

$$\begin{aligned} p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k}) &\propto p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k},\boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k-1}) \\ &= p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1},\boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1}) \\ &= p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1}) \end{aligned} \tag{22}$$

With this recursive representation of the posterior distribution, a proposal distribution needs to be developed to develop a recursive weight update equation (24).

$$\begin{aligned} w_k^i &\propto \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)p(\boldsymbol{x}_{0:k-1}^i|\boldsymbol{y}_{1:k-1})}{\pi(\boldsymbol{x}_{0:k}^i|\boldsymbol{y}_{1:k})} \\ &= \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)}{\pi(\boldsymbol{x}_k^i|\boldsymbol{x}_{0:k-1}^i,\boldsymbol{y}_{1:k-1})} \times \frac{p(\boldsymbol{x}_{0:k-1}^i|\boldsymbol{y}_{1:k-1})}{\pi(\boldsymbol{x}_{0:k-1}^i|\boldsymbol{y}_{1:k-1})} \end{aligned} \tag{23}$$

$$w_k^i \propto \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)}{\pi(\boldsymbol{x}_k^i|\boldsymbol{x}_{0:k-1}^i,\boldsymbol{y}_{1:k-1})} \times w_{k-1}^i \tag{24}$$

Now that the weights can be updated sequentially once new information y_k arrives, the filtering distribution can also be updated sequentially and is represented as shown in equation 25 below.

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i) \tag{25}$$

For this study, a variation of the particle filter called the *bootstrap particle filter* is used, where the proposal distribution is chosen to be the system dynamics.

$$\pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1},\boldsymbol{y}_{1:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \tag{26}$$

This simplifies the weight update in equation 24 to be proportional to just the measurement likelihood.

$$w_k^i \propto w_{k-1}^i p(\boldsymbol{y}_k|\boldsymbol{x}_k^i) \tag{27}$$

Adapting this development of the bootstrap particle filter to the notation of the problem at hand, the measurements are taken to be the reference that the vehicle needs to track $\boldsymbol{r}$, and the states will be the augmented state and input vector from the virtual system.

$$w_k^i \propto w_{k-1}^i p(\boldsymbol{r}|\bar{\boldsymbol{x}}_k) \tag{28}$$

$$p(\bar{\boldsymbol{x}}_k|\boldsymbol{r}) \approx \sum_{i=1}^{N} w_k^i \delta(\bar{\boldsymbol{x}}_k - \bar{\boldsymbol{x}}_k^i) \tag{29}$$

One way to think about this implementation is that a state that satisfies the reference signal well will have a higher measurement likelihood and therefore a larger weight than a state that poorly satisfies the reference.

Resampling

As the state evolves over time, the estimation quality can degrade due to a phenomenon called particle degeneracy. This happens when only a few of the overall particles have appreciable weights, which makes the estimate of the posterior distribution poor. Resampling the particles is a method to combat the particle degeneracy problem by replacing bad or low-weight particles. There are many different methods of resampling [12], and in this study traditional multinomial resampling is used.

Multinomial resampling is illustrated with the figure below where a uniform distribution is sampled for an index i, and then the particle that is resampled is indexed as j. The cumulative distribution function that the resampling index relates to is constructed with the weighted set of particles. The particles with larger weights are more likely to get resampled, which will replace the particles with low weights.

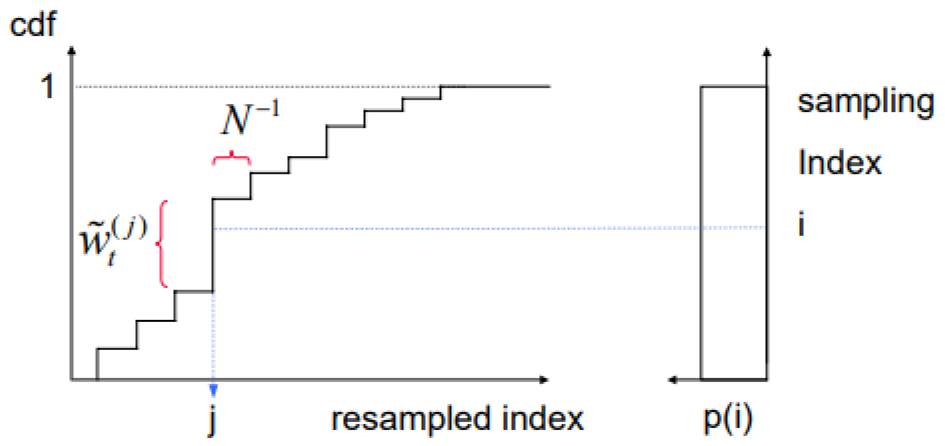

Figure 5. Multinomial resampling illustration

The resampling procedure will increase the computational effort required by the algorithm, so it is not performed at every step. Whether to do resampling or not is based on the number of effective particles, which is calculated with equation 30 below.

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2} \tag{30}$$

A threshold value of effective particles required N_{req} is set, and if the number calculated with (30) is less than this value then resampling is performed. The implementation of the bootstrap particle filter is described in the algorithm below.

```
Algorithm 1 – Particle Filtering

• Given the constant reference $\boldsymbol{r}$ for the planning horizon of length $T$
• Sample the initial set of particles from the prior distribution $x_0^i \sim p(x)$
• For $k = 1\ to\ T$
    o Predict obstacle regions, update road information $Z$
    o For $i = 1\ to\ N$
        ▪ Generate $\bar{x}_k^i$
        ▪ Update weight
    o End for
    o Normalize weights
    o Calculate $N_{eff}$ with (30)
    o If $N_{eff} < N_{req}$
        ▪ Resample particles with replacement
        ▪ Normalize weights
    o End if
• End for
• Pass generated sequence of weighted particles to reweighting smoother
```

Reweighting Particle Smoother

At the end of the planning horizon, when all measurements have been obtained, the quality of the estimation can be improved by smoothing due to the additional information available. The effects of this smoothing on the trajectory generation can be seen below in figure 6.

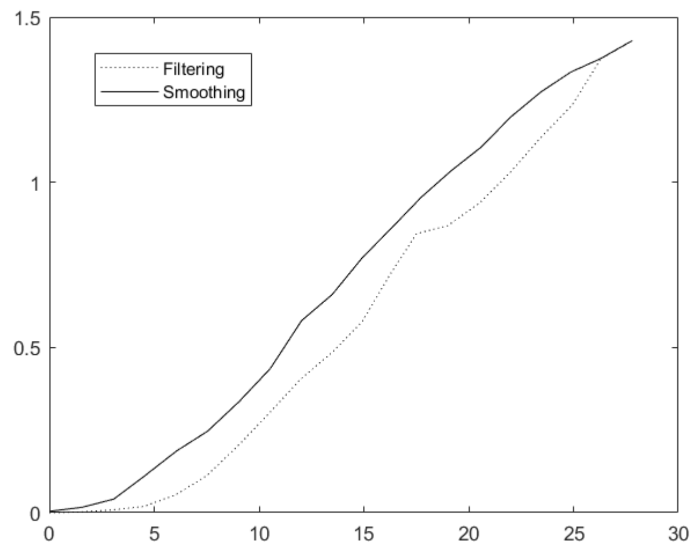

Figure 6. Filtering and smoothing trajectory lateral positions compared during a lane change maneuver. The trajectory is a single one-second planning horizon.

The goal of Bayesian smoothing is to compute the marginal posterior distribution of x_k given the measurements in a set window (horizon) where $T > k$.

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:T}) \tag{31}$$

The reweighting particle smoother computes new weights for the particles to get an approximation of this smoothing distribution. As with the forward filtering equations, the main results are shown here, and the derivations are included in the appendix.

Given the result of the forward filtering procedure, there are marginal distributions $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ and the complete measurement set $\boldsymbol{y}_{1:T}$ available. The smoothing weights are initialized as follows.

$$w_{T|T}^{i} = w_T^i \ \forall\ i = 1, \dots, N \tag{32}$$

The notation $w_{k|T}^{i}$ indicates the smoothing weight for particle i at time k given information up to time $T > k$. With the initialized smoothing weights, the smoothing weights can be updated recursively backwards through the horizon.

$$w_{k|T}^{i} = \sum_{j} w_{k+1|T}^{j} \frac{w_k^i p\left(\boldsymbol{x}_{k+1}^{i}|\boldsymbol{x}_k^i\right)}{\left[\sum_l w_k^l p\left(\boldsymbol{x}_{k+1}^{j}|\boldsymbol{x}_k^l\right)\right]} \tag{33}$$

With these weights, then the marginal distribution at each time step can be approximated as follows.

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:T}) \approx \sum_i w_{k|T}^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i) \tag{34}$$

Again, adapting these equations to the notation of the system in this study, the smoothing density is for the augmented state vector $\overline{\boldsymbol{x}}_k$ given the reference $\boldsymbol{r}$ with the newly generated smoothing weights.

$$p(\overline{\boldsymbol{x}}_k|\boldsymbol{r}) \approx \sum_{i=1}^{N} w_{k|T}^i \delta(\overline{\boldsymbol{x}}_k - \overline{\boldsymbol{x}}_k^i) \tag{35}$$

With these equations, the reweighting smoother procedure is defined in the algorithm below.

```
Algorithm 2 – Reweighting Particle Smoothing

•  Given the states and weights for each particle from the
   particle filtering procedure and the complete measurement
   set over the horizon
•  Initialize the smoothing weights with (32)
•  For $k = T - 1\ to\ 0$
      o  For $i = 1\ to\ N$
            ▪  Calculate smoothing weights using (33)
      o  End for
      o  Generate marginal smoothing distribution with (35)
•  End for
```

Once the smoothing distribution is generated, the smoothed vehicle state estimate can then be extracted and then combined into a trajectory over the planning horizon.

$$\begin{aligned} \overline{\boldsymbol{x}}_k &= \sum_{i=1}^{N} w_{k|T}^i \overline{\boldsymbol{x}}_k^i \\ \overline{\boldsymbol{x}}_{0:T} &= [\overline{\boldsymbol{x}}_0, \overline{\boldsymbol{x}}_1, \dots, \overline{\boldsymbol{x}}_{T-1}, \overline{\boldsymbol{x}}_T] \end{aligned} \tag{36}$$

To drive the actual vehicle along the generated smoothing trajectory, the inputs are extracted from the augmented states and applied in a receding horizon fashion where only the first input is used, and the rest are discarded and replanned. As the virtual system vector includes the inputs as the 6th and 7th elements, these inputs are obtained by taking the 6th and 7th elements from $\overline{\boldsymbol{x}}_0$ as found in (36) above.

With the estimation method fully developed, the specific measurements that will be used are described in detail in the following section.

Measurement Model

Lane Tracking Measurement

In the considered straight road scenario, the road axis can be aligned with the global x-direction which allows for simple implementation of quantifying the lane tracking error. As shown in figure 7 below, the lane centerlines can be described by a global y-coordinate, to which the vehicles global y-position p_Y can be directly compared.

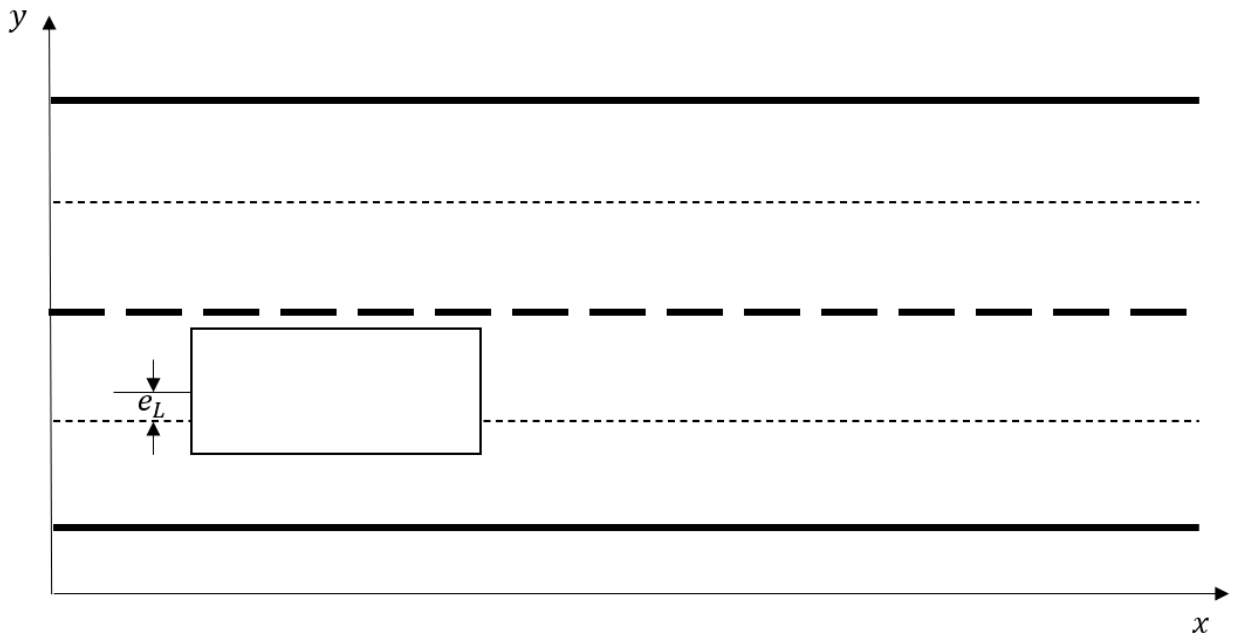

Figure 7. Lane centerline illustration for straight road case.

Obstacle Avoidance

The next set of measurements are for obstacle avoidance, where there are N_{obs} terms. To model the obstacles in a parametric manner that is computationally efficient, an ellipse is generated around the obstacle and the ego vehicle, each with known functional forms. The obstacle ellipse should be designed such that the obstacle itself is completely enclosed by the region. The region is extended within the obstacle lane to provide a safety buffer for passing maneuvers to incentivize the faster-moving ego vehicle to change lanes sooner when approaching the obstacle from behind.

The general polar form of an ellipse is given in equation 37.

$$r(\theta) = \frac{ab}{\sqrt{(bcos\theta)^2 + (asin\theta)^2}} \tag{37}$$

The parameters of the ellipse (a and b) are the major and minor radii, respectively. The length of the ellipse is parametrized by the ego vehicle velocity, so the obstacle region elongates as the velocity increases. To perform this parametrization, the foci-to-center distance c is modeled as a function of EV velocity in equation 38 and the major radius is then calculated as follows in equation 39.

$$c = 0.8v_x \tag{38}$$

$$a = \sqrt{b^2 + c^2} \tag{39}$$

The leading coefficient in the c function is designed to give the distance between the obstacle and the trailing edge of the ellipse a length of approximately v_x. The chosen value accomplishes this because for highway speeds, the ellipse is elongated to the point where $c \approx a$, and setting the coefficient as such results in an ellipse approximately $1.5v_x$ in overall length. The minor radius b is simply the lane half-width to make the ellipse occupy the entire width of the obstacle lane.

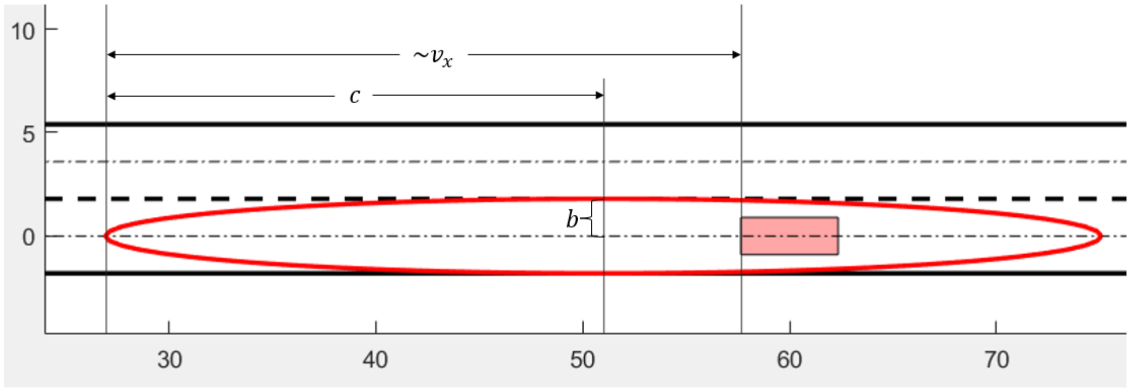

Figure 8. Elliptical obstacle region with parameters b and c shown.

As $a \approx c$, it is not shown separately in figure 8 above (c = 24, a = 24.0674). The adjustment of the ellipse center via equation 40. results in the distance from the obstacle to the trailing edge being approximately $v_x\ m$ long.

$$center = obs_x + \frac{v_x}{2} - c \tag{40}$$

Here, obs_x denotes the center of the obstacle in the global longitudinal direction, or in this case, the global x-coordinate of the obstacle center. Note that if $c = 0.75v_x$, this results in a distance between the obstacle center and the trailing edge of the ellipse that is exactly equal to the ego velocity. However, the distance that matters is to the rear of the vehicle. This motivates choosing the factor of 0.8, which generates an ellipse with the desired properties that is shown in figure 8 above.

An ellipse is also used for modeling the EV's collision region, where the major and minor radii are specified by the vehicle's dimensions. For the straight road scenario, the obstacle vehicles maintain a heading angle of zero, so the ellipse the describes their obstacle region does not need any rotation. However, the ego vehicle's heading angle will change, so the ellipse that bounds it must be rotated accordingly by the heading angle ψ. In the following equations, the subscript ego indicates the ego vehicle, and the subscript obs indicates the obstacle vehicle. The bounding ellipse for any state that describes the ego vehicle is described in equation 41 below. In the particle filter motion planning implementation, this means that each particle's state vector has an ellipse that is used in the measurement function that evaluates particle weights.

$$r(\theta) = \frac{a_{ego}b_{ego}}{\sqrt{\left(b_{ego}cos(\theta+\psi)\right)^2 + \left(\mathrm{a_{ego}}\sin(\theta+\psi)\right)^2}} \tag{41}$$

To calculate the distance between the ellipse edges the center-to-center distance and angle between the ellipse centers $\bar{\theta}$ is calculated.

$$d_{centers} = \sqrt{\left(y_{ego} - y_{obs}\right)^2 + \left(x_{ego} - x_{obs}\right)^2} \tag{42}$$

$$\bar{\theta} = \tan^{-1}\left(\frac{y_{ego} - y_{obs}}{x_{ego} - x_{obs}}\right) \tag{43}$$

Then, the *specific radius* for each ellipse is evaluated.

$$\bar{r}_{ego} = \frac{a_{ego}b_{ego}}{\sqrt{\left(b_{ego}cos(\bar{\theta}+\psi)\right)^2 + \left(\mathrm{a_{ego}}\sin(\bar{\theta}+\psi)\right)^2}} \tag{44}$$

$$\bar{r}_{obs} = \frac{a_{obs}b_{obs}}{\sqrt{\left(b_{obs}cos(\bar{\theta})\right)^2 + (\mathrm{a_{obs}}\sin(\bar{\theta}))^2}} \tag{45}$$

These specific radii are illustrated below in figure 9. Calculating the edge-to-edge distance between the ellipses is straightforward once the specific radii are calculated.

$$\Delta = d_{centers} - \bar{r}_{ego} - \bar{r}_{obs} \tag{46}$$

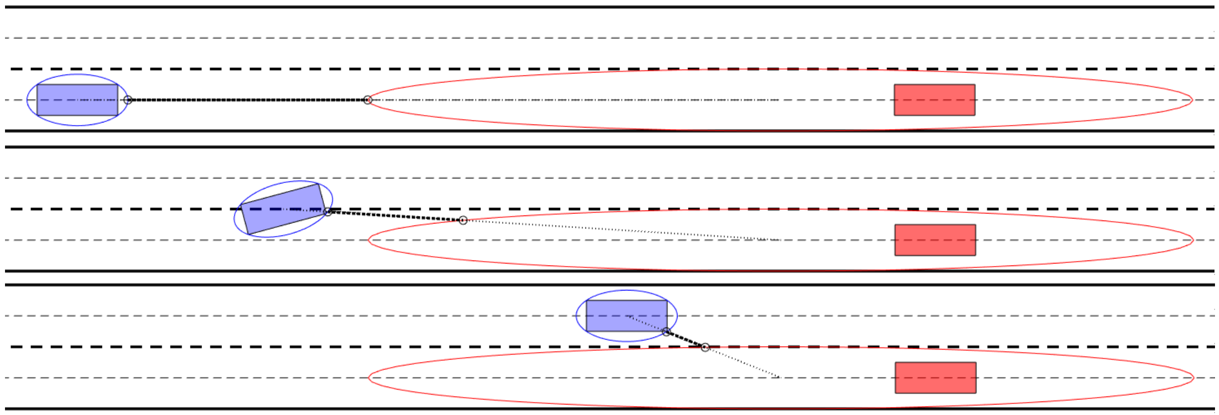

Figure 9. Passing maneuver with obstacle distance indicated with the solid line connecting the two circles lying on each ellipse. The dashed line between the circle and the ellipse center is the specific radius.

It can be seen in figure 9 that this method does not return the actual minimum distance between the two ellipse edges, especially when looking at the third subfigure. Because the obstacle region is conservative, the approximation made with this method of finding obstacle distance is acceptable.

Once Δ is calculated, there remains the evaluation of the measurement function term g_o. To enforce strict adherence to obstacle avoidance requirements, a barrier function is used in the measurement function that will give particles that violate the obstacle region very low weights. The Softplus barrier function is chosen.

$$g(\Delta) = \frac{1}{\alpha}\log\left(1 + e^{\beta\Delta}\right) \tag{47}$$

This function has two parameters α and β which, if chosen properly, cause the function output to be zero for positive arguments and grow rapidly for negative arguments. This can be thought of as an inequality constraint, where the requirement is for $g(x) \geq 0$. To get this behavior, β must be negative. An illustration of the Softplus function for different values of α and β is shown below in figure 10.

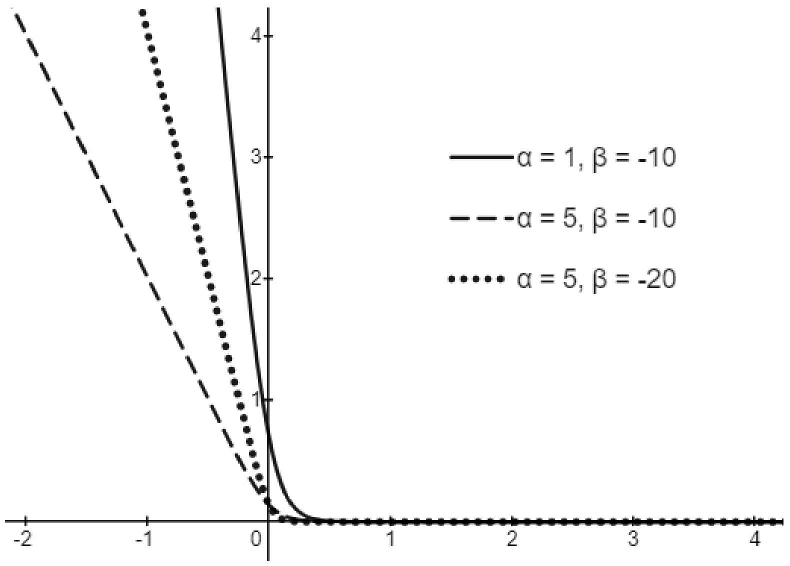

Figure 10. Softplus barrier function for 3 different parameter pairs

The selection of the parameter values depends on the design of the measurement covariance for obstacle avoidance. The specific parameter values used will be described in the *Simulation Study* section.

Road Boundary Satisfaction

For the driving requirement of staying within the road boundaries, another barrier function is used. The input to this barrier function can be calculated in a similar manner to the lane centerline deviation for the straight road case. The global y-coordinate of the particle state can be compared to the y-coordinate of the road boundaries, and if the EV's y-coordinate is outside of the road boundary, the measurement function will provide almost zero weight to that particle due to the strict parameter design of this driving requirement. This road boundary satisfaction measurement is a redundant safety mechanism, as the lane centerline and heading angle tracking should only generate trajectories that stay within the road boundary. The specific equation for this barrier function is below, and the plotted function is in figure 11.

$$\frac{1}{m} ln\left(1 + e^{n(|y|-w)}\right) \tag{48}$$

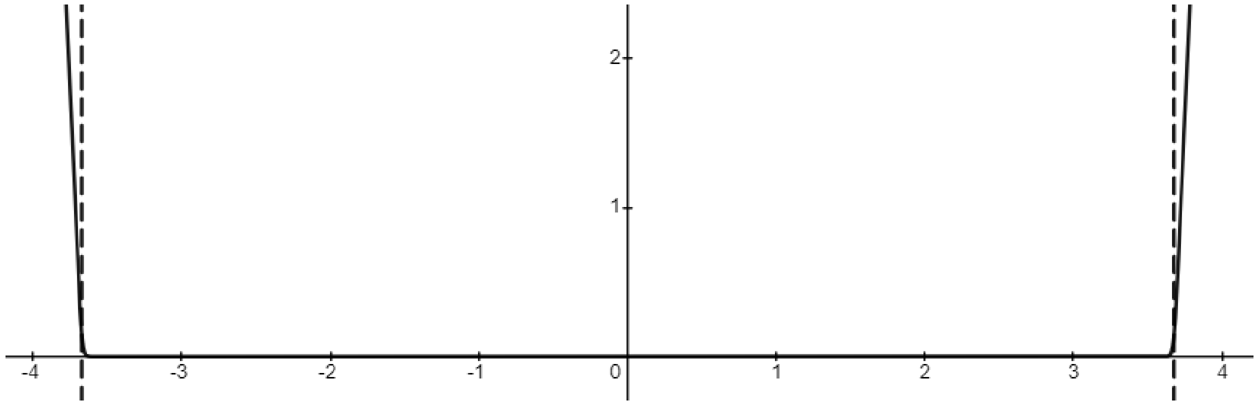

Figure 11. Barrier function illustration for road boundary satisfaction. The solid line is the barrier function. The dashed line is the road boundary.

Heading Angle Tracking

The requirement to keep the EV's heading angle close to the road angle is to combat overshooting the target lane during a lane change. This problem is illustrated below in figure 12.

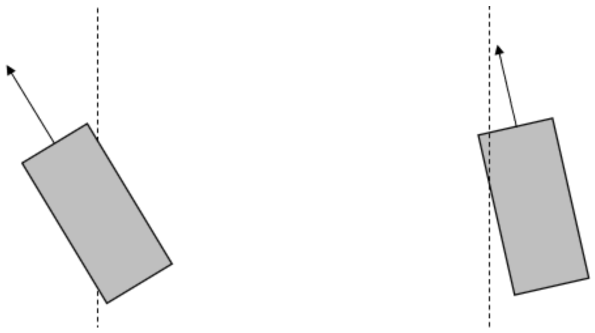

Figure 11. Heading angle tracking requirement illustration

On the left of this figure is a vehicle state that would have a high weight if this driving requirement was not considered. The vehicle is exactly on the lane centerline, but its velocity direction will cause it to overshoot the reference. By implementing a heading angle tracking term, the vehicle state on the right will have a higher weight. The heading angle tracking and lane centerline tracking are at odds with each other during a lane change, but if the parameters associated with these terms in the measurement function and measurement noise covariance are designed properly, then smooth lane-changing trajectories will be generated.

The heading angle error is measured by taking direct state feedback of the vehicle's heading angle and evaluating how well it satisfies the requirement of zero deviation from road angle.

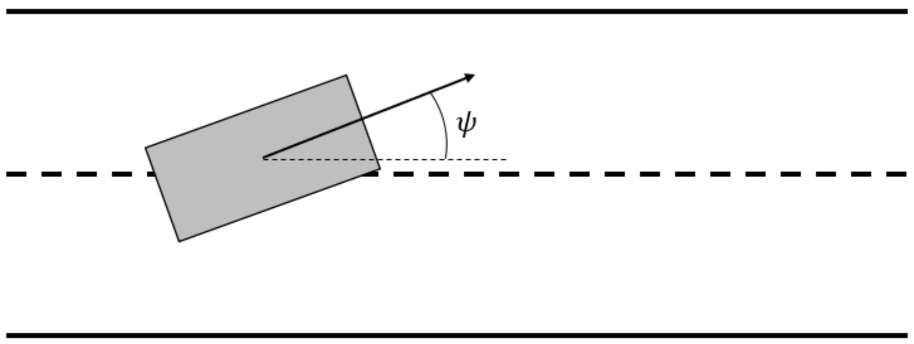

Figure 12. Vehicle heading angle illustration

Driving Mode Design and Selection

For each driving mode, the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ can be adjusted to increase trajectory quality. The adjustments can be explained intuitively by understanding what each driving mode is trying to accomplish.

The lane-keeping driving mode is the default mode that should be selected for most of the time. When lane-keeping in a straight road scenario, the vehicle does not need to make any turning maneuvers except for correcting for small errors, so the covariance for the steering rate can be decreased in the process noise. Following the same logic, the measurement covariance elements corresponding to the lane centerline and heading angle tracking can also be reduced. This tightens up the lateral range of the generated trajectories and the result will track straighter than it would with "looser" covariance values.

For lane-changing modes, the vehicle is trying to turn out of its current lane and track a new reference lane, so it is required to use larger steering rate inputs than lane-keeping. For the measurements, the covariance for the lane and heading angle tracking must be increased to allow for sufficient exploration of the input space to reach a region of high enough likelihood for the filtering weights to not collapse.

When sampling the driving mode for the next trajectory, the transition probabilities first need to be calculated. In this study, the probabilities are conditioned on the distance from the ego vehicle to the obstacle occupying the same lane. Probabilistically, this can be expressed as follows.

$$m_j \sim p\left(m_j | \Delta_l\right) \qquad (49)$$

This is done to incentivize the lane changing modes to be selected more frequently when the ego vehicle gets closer to the obstacle avoidance region. Intuitively, one can think about this as modeling the probability that a certain mode will produce a good trajectory given the distance to the leading obstacle.

To implement this, first the probability of sampling a lane change is calculated with the following equation. There are two parameters, P_{base} and P_{min} in this equation. P_{base} sets the max sampling probability for a lane change and is set at 0.9 for the results in this study. P_{min} sets a lower limit for the lane-keeping sampling probability and is set at 0.1. These can be tuned depending on how many modes are selected or based on how large the feasible mode set is. In this study, the feasible modes can be restricted to just two once the ego vehicle's current lane is identified, as the mode that would cause the vehicle to leave the road can be eliminated.

$$p(LK) = \begin{cases} \frac{-(P_{base} - P_{min})}{v_x^2}(\varDelta_l - v_x)^2 + P_{max}, & 0 < \varDelta_l < v_x \\ P_{max}, & \varDelta_l \geq v_x \end{cases} \tag{50}$$

Plotting this equation reveals the nature of how the lane-keeping sampling probability is determined as a function of distance to the obstacle in the same lane and the longitudinal velocity of the ego vehicle. The plot also demonstrates how changing the parameters effect the probability of sampling a lane change.

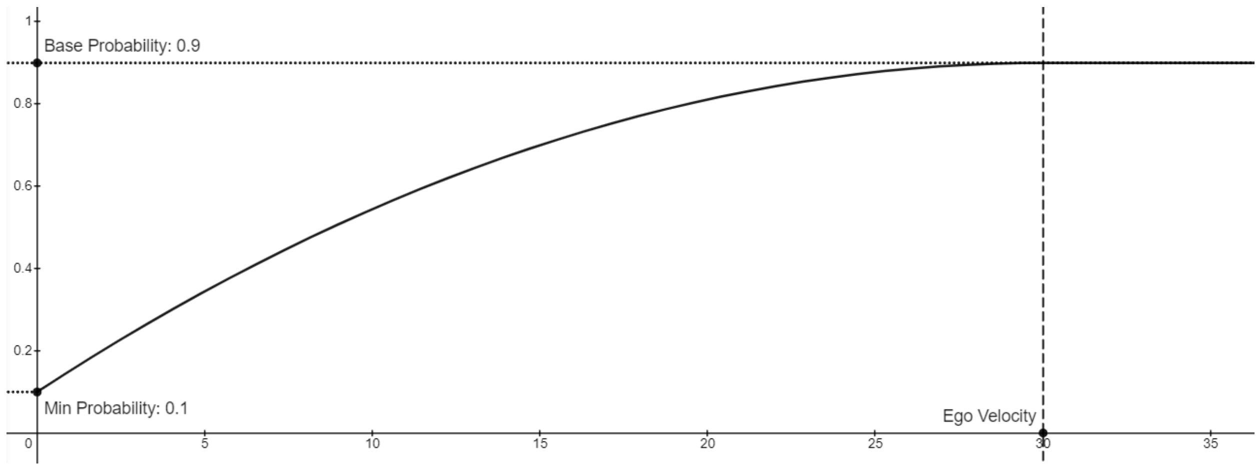

Figure 13. Plot of equation 50

In this example, the longitudinal velocity is taken to be 30 m/s. As the distance to the obstacle decreases below the distance the ego vehicle will travel in 1 second, the probability of sampling the lane-keeping mode decreases quadratically to P_{min}. At this point, the sampled trajectories likely include a lane-changing mode that is better than lane-keeping. As the feasible set of driving modes has been pared down to two, if lane-keeping is not sampled, then the other feasible mode will be selected. When the ego vehicle is far from any obstacles in its lane, then the algorithm does not expect a lane-changing mode to provide a better trajectory than lane-keeping.

The method of determining which driving mode to commit to is by implementing a cost function that evaluates the cumulative cost of each generated trajectory. The trajectories are generated according to the reference signal associated with the driving mode sampled. The cost function is formulated below in (51).

$$C = \sum_{horizon} \gamma \|e_v\|^2 + \epsilon \|e_l\|^2 + \zeta \ln\left(1 + e^{\nu(\Delta_{obs} - \xi)}\right) + \lambda \ln\left(1 + e^{\eta(|y| - w + \rho)}\right) \tag{51}$$

There are several weighting parameters that can be tuned to adjust the cost function's preference towards certain behaviors. For example, the lane boundary satisfaction and obstacle avoidance barrier functions are safety-critical, so their parameters should be designed such that any violation ensures that the trajectory will not be selected. Through the filtering and smoothing, the trajectories should already satisfy the constraints of the environment, but the cost function allows for a second pass over multiple trajectories to further improve the applied trajectory quality.

The cost function works in a similar manner to the measurement function and evaluates how well a trajectory tracks the nominal velocity, reference lane centerline, and satisfaction of obstacle and road boundary constraints. There is some bias implicit to this formulation and this bias explains why there is no heading angle tracking term in the cost function. This bias is because at the beginning of a lane-changing trajectory, the lane tracking error will be large (as the vehicle is in the wrong lane) but will decrease towards the end of the trajectory as the states moves towards the target lane. Because the cost function is cumulative over the entire trajectory, this error can lead to inflated cost even though the trajectory is good. The heading angle also must change during a lane-changing maneuver, so adding a heading angle penalty term in the cost function would further increase the bias against selecting a lane-changing trajectory.

As seen in (51), the cumulative velocity and lane tracking terms are simply a sum of squares. The obstacle avoidance and lane boundary terms are again a Softplus barrier function. The cost function can have different parameters than the measurement function and has offset terms applied in the Softplus function that are not in the measurement function. The cost function term for obstacle avoidance includes an offset ξ which disincentivizes trajectories that remain near the obstacle region, even if they don't violate the obstacle region. This is effectively inflating the size of the obstacle ellipse.

The lane boundary barrier function includes an offset ρ, which can be thought of as narrowing the available road width. This is illustrated below in figure 14 where the boundary barrier function is plotted with several values of ρ.

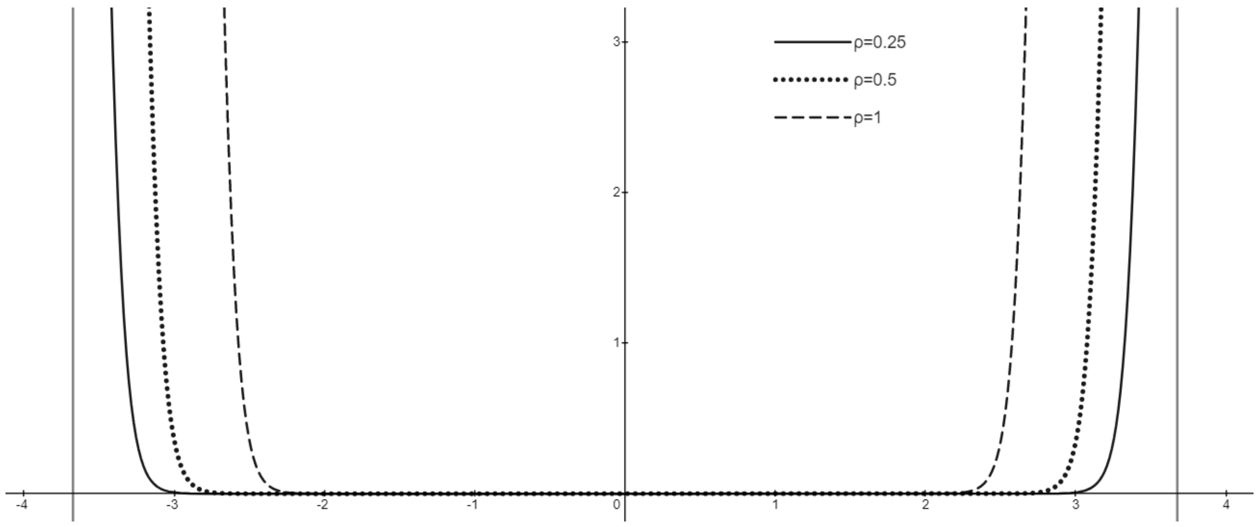

Figure 14. Road boundary cost function – Softplus function with varying offset parameters. The gray lines indicate the lane boundary. Other parameters are $\lambda = 5, \eta = 15$ for all plots.

These offsets combat the possibility of a lane-keeping trajectory to skirt around the edge of the obstacle ellipse and maintain a lower cost than lane-changing. This results in a somewhat indecisive planner that would settle into lane-changing later than expected. The implementation of these offset terms in the cost function remedies this.

The final algorithm is described below.

```
Algorithm 3 – PF-RS Motion Planning

Algorithm describes the process for a single planning phase

• Observe road information (obstacle positions and velocities,
  current ego vehicle lane)
• For each sampled driving mode
    o Identify reference lane, input noise covariance, and
      measurement noise covariance
    o Execute forward filtering with algorithm 1
    o Execute reweighting smoothing with algorithm 2
    o Evaluate the cumulative cost of the generated
      trajectory with (51)
• End for
• Choose best trajectory (minimum cost)
• Apply 1st set of inputs from best trajectory to the true system
```

Simulation Study

Single Planning Horizon Simulation

In this section, the simulation will be built up to illustrate what is happening in each step and how certain parameters effect the resulting trajectories. Because there is random sampling involved, the results shown here will all be generated with the MATLAB random number generation seed set to 'default'. This will ensure consistent results when the simulation is run multiple times.

First, the particles themselves are illustrated in increasing quantities. As the particle count increases, it becomes harder to distinguish them, but the exploration of the state space becomes much better and higher quality trajectories are the result.

In the lane-keeping driving mode, the vehicle is just trying to maintain its lane center tracking for its current lane, and for a straight road this results in a straight trajectory with nominally zero steering input. The vehicle is still trying to maintain the nominal velocity (speed limit) as well. For the lane-changing driving mode, the vehicle tracks the nominal velocity but must make steering corrections to track a different lane center than that of its currently occupied lane.

The figures on the following pages demonstrate how varying the particle count affects the generated trajectory. Individual particles are plotted along with the resulting weighted sum trajectory. Note that these trajectories are after using particle filtering and reweighting particle smoothing. The resampling steps are occurring whenever the particles collapse back into a smaller area.

All the lane-keeping simulated trajectories have the same process and measurement covariances. All the lane-changing trajectories have the same process and measurement covariances. The black **x** at the left end of each trajectory is the initial position of the ego vehicle.

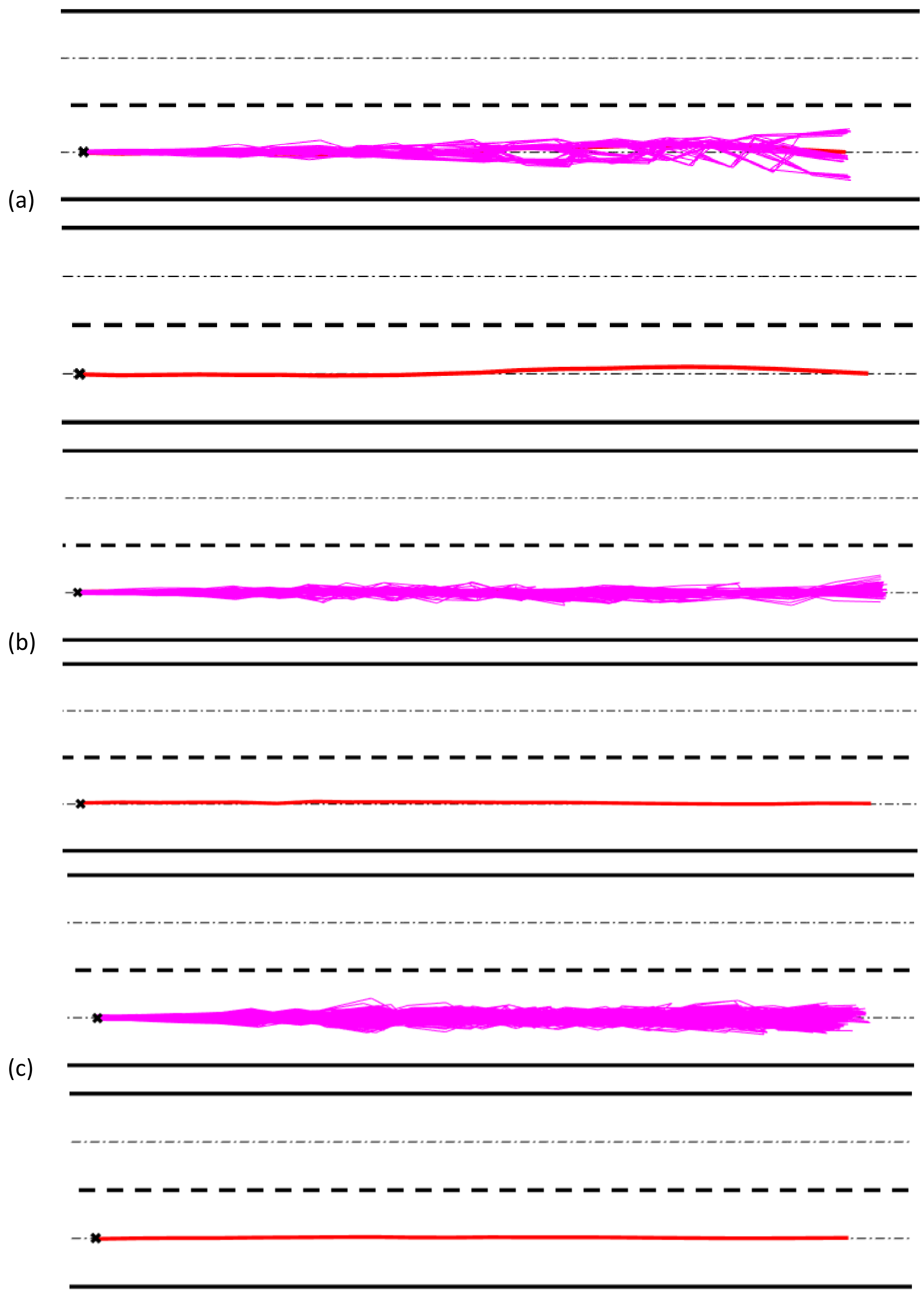

Figure 15. Individual particle evolution is plotted in the upper figures, with the resulting trajectory plotted below. (a) 50 particles, (b) 100 particles, and (c) 500 particles.

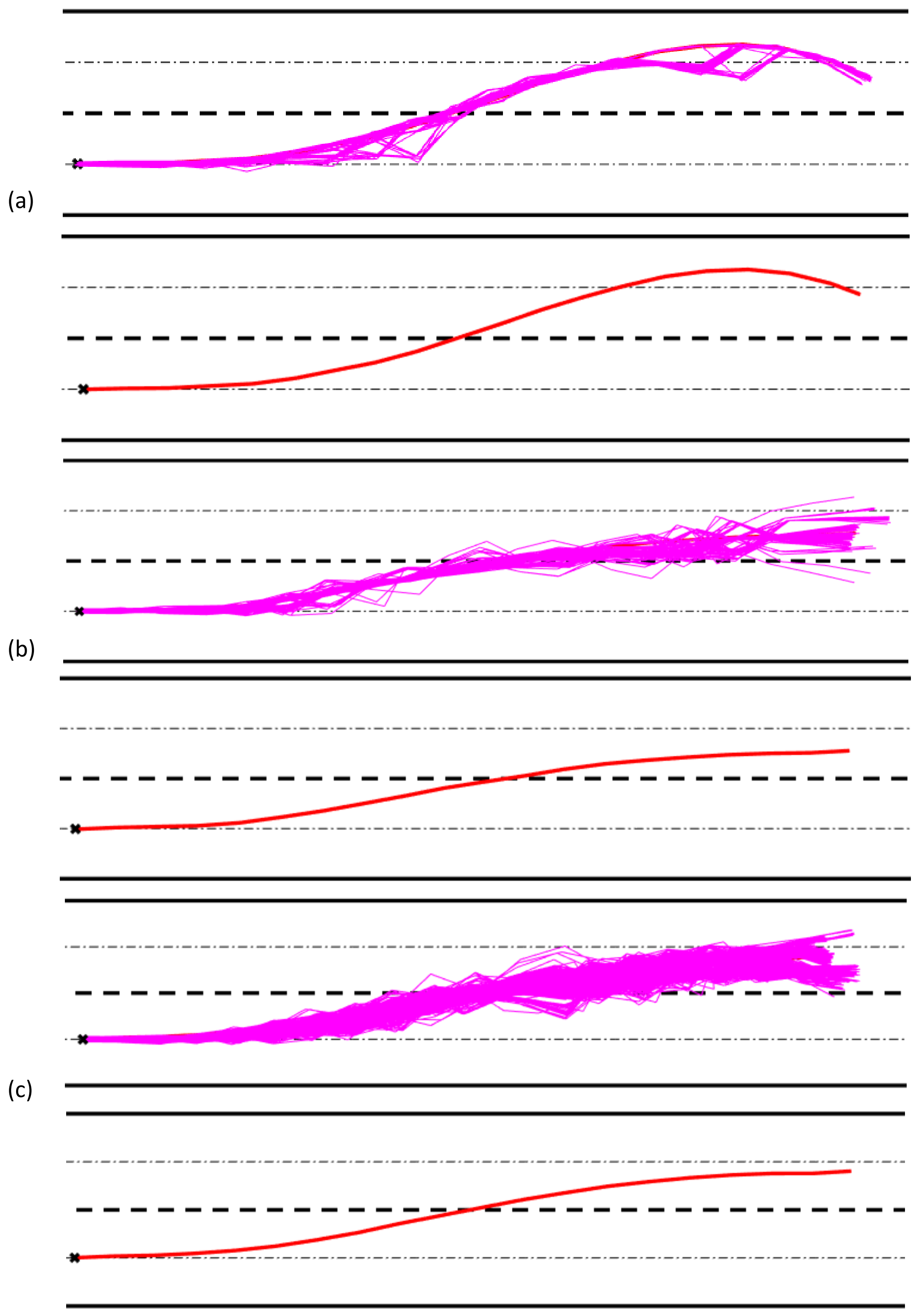

Figure 16. Individual particle evolution is plotted in the upper figures, with the resulting trajectory plotted below. (a) 50 particles, (b) 100 particles, and (c) 500 particles.

For all driving modes, the state variable process noise covariance (the ~ 0 submatrix in (10)) is kept constant. The covariance for the noise term that generates the inputs to the system is varied for each driving mode. First, the state process noise is given. The subscript indicates the matrix dimension.

$$\overline{\boldsymbol{Q}}_{\mathrm{n_x} \times n_x} = \begin{bmatrix} 1\times 10^{-4} & 0 & 0 & 0 & 0 \\ 0 & 1\times 10^{-4} & 0 & 0 & 0 \\ 0 & 0 & 1\times 10^{-7} & 0 & 0 \\ 0 & 0 & 0 & 1\times 10^{-4} & 0 \\ 0 & 0 & 0 & 0 & 1\times 10^{-7} \end{bmatrix} \tag{52}$$

This matrix is designed to give the system increased confidence in the steering angle and heading angle states, each of which can have detrimental effects on the trajectory tracking if the initial state estimate is deviated from the true state.

The table below provides the specific input and measurement covariance matrices used for each driving mode. The values reflect the goals of each mode. Compared to lane-keeping, the lane-changing modes have an increased steering rate noise covariance to allow for more steering exploration and the lane tracking and heading angle tracking covariances are increased accordingly in the measurement noise covariance.

Mode	Input Noise Covariance	Measurement Noise Covariance
Lane-Keeping	$\boldsymbol{Q}_{n_u \times n_u} = \begin{bmatrix} 1 & 0 \\ 0 & 0.005 \end{bmatrix}$	$\boldsymbol{R} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.025 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0002 \end{bmatrix}$
Lane-Changing	$\boldsymbol{Q}_{n_u \times n_u} = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}$	$\boldsymbol{R} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0007 \end{bmatrix}$

After constructing the virtual system process noise covariance, a multiplier can be applied to increase the covariance for large particle counts. This helps with numerical stability, as the region of high likelihood is larger, and more particles can be used to get a good estimate of the posterior. For 500 particles, this multiplier is set to $G = 10$.

$$\boldsymbol{Q}' = G \times \boldsymbol{Q},\ \boldsymbol{R}' = G \times \boldsymbol{R} \tag{53}$$

The driving scenario analyzed in the full-length simulations has two obstacles; one obstacle occupies each lane. The right lane obstacle is traveling at 15 m/s and the left lane obstacle is travelling at 17 m/s. The ego vehicle starts the simulation at 20 m/s and must speed up to the nominal velocity of 30 m/s. The obstacle positions and their elliptical avoidance regions are shown below. The figure displays the five generated trajectories in two different cases. The first case is when the lane-keeping mode sampling probability is 90% due to the distance to the red obstacle being less than the distance the vehicle will travel in 1 second. The second case is when the lane-keeping mode sampling probability is reduced due to the ego vehicle being closer to the obstacle.

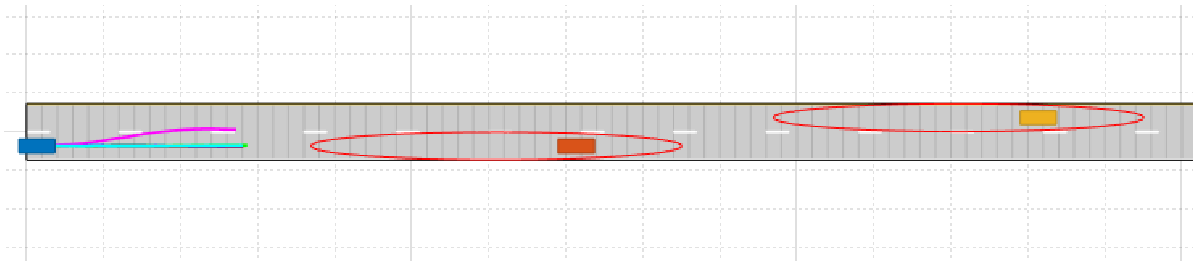

Figure 17. Scenario layout with 5 generated trajectories (4 LK, 1 CLL) with LK sampling probability of 90%.

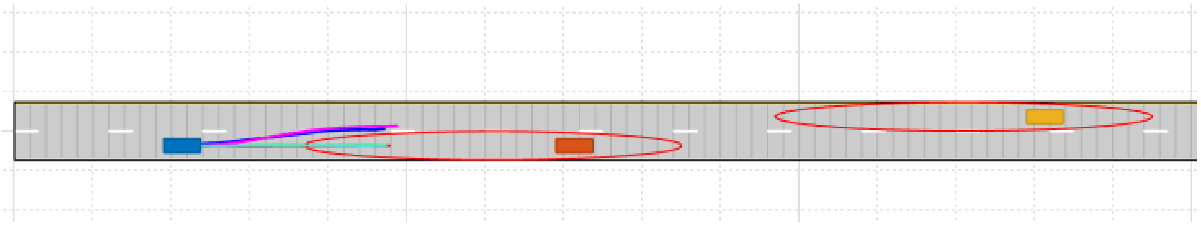

Figure 18. Scenario with ego vehicle adjusted to show impact of (50). of the 5 generated trajectories, 3 are LK and 2 are CLL. The LK sampling probability is 64.6%.

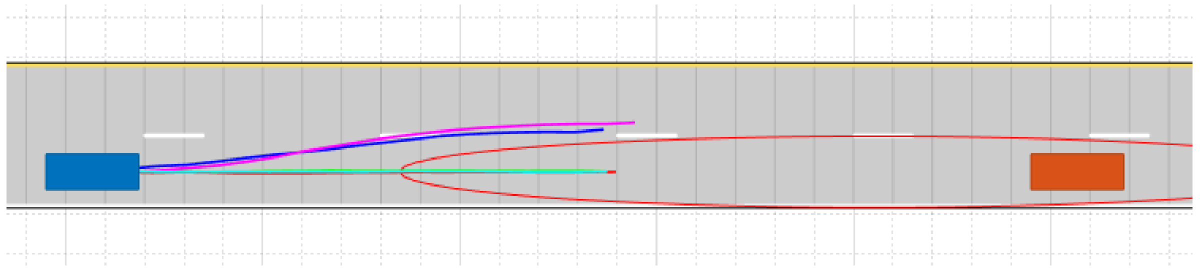

Figure 19. Zoomed in on trajectory generation for figure 18 bottom.

In figures 18 and 19 above, it may seem that the lane-keeping trajectories are violating the obstacle ellipse, however this is not the case. Recall that during the planning phase, the obstacle position is predicted based on its measured velocity. As the obstacle moves, the elliptical region will move with it, and the trajectories shown do not violate the region at the time when they coexist. Next, the results of full-length simulations are presented.

The first simulation was run with 250 particles and 5 driving modes sampled per planning phase. The plots below show the lane tracking and velocity tracking. On the lane-tracking plot, the target shown is shown as the dotted line. There are some instances of momentary unexpected or inconsistent mode choices that occur. These happen during the transient phases of the ego vehicles trajectory when deciding to start a lane change. Sometimes, just through random chance and the MATLAB random number generation method, only 1 driving mode will be sampled 5 times. This can cause lane-keeping to be selected during a lane-changing maneuver, which appears in these plots. Once the ego vehicle gets close enough to the leading obstacle for the lane-keeping sampling probability to drop sufficiently low, the chances of sampling all 5 as lane-keeping diminishes to 0.001% as per the binomial distribution with $n = 5, p = 0.1, x = 5$.

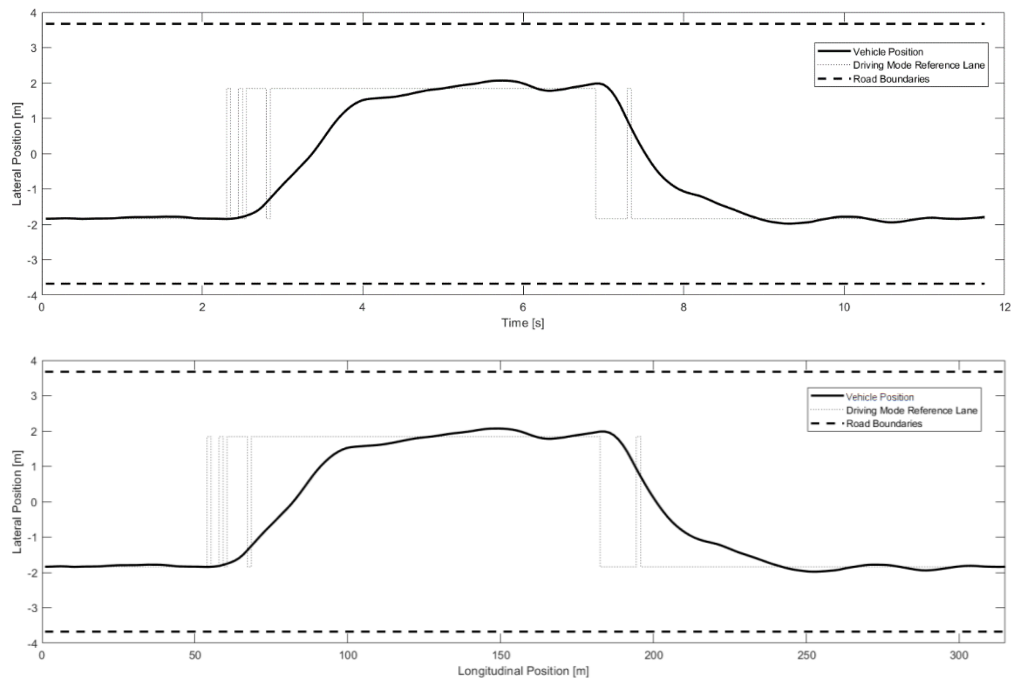

Figure 20. Lane centerlines of driving mode and vehicle position. 250 particles, 5 modes sampled, lane-keeping sampling probability parameters $P_{base} = 0.9, P_{min} = 0.1$

The inputs and the corresponding state variables are shown below in figures 21 and 22.

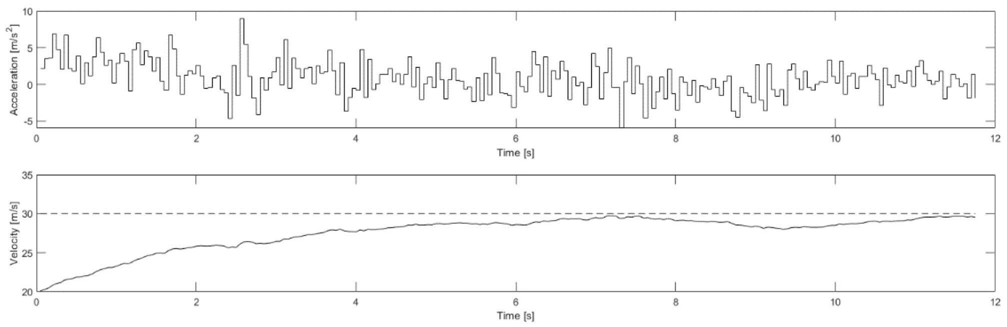

Figure 21. Acceleration inputs and corresponding velocity vs time.

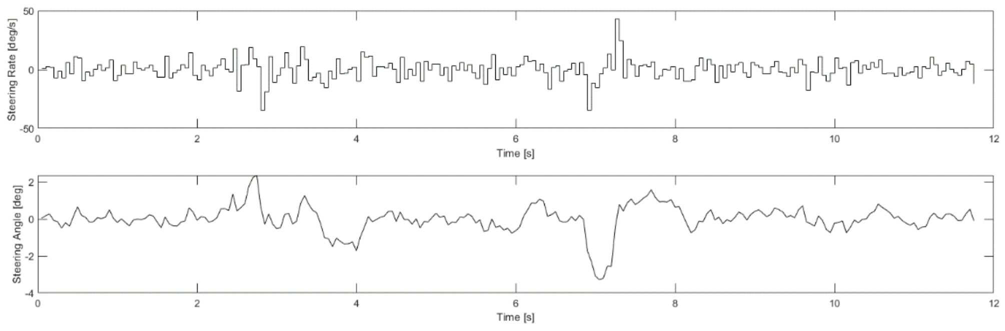

Figure 22. Steering rate inputs and corresponding angles vs time.

In these 250-particle simulations, the estimated trajectories are somewhat jagged, although the driving requirements are still satisfied. The next simulation has the same setup, except 500 particles are used. In these simulations, it is expected that the estimation performance will increase. Again, the results for lane tracking are shown first with the position and time axis.

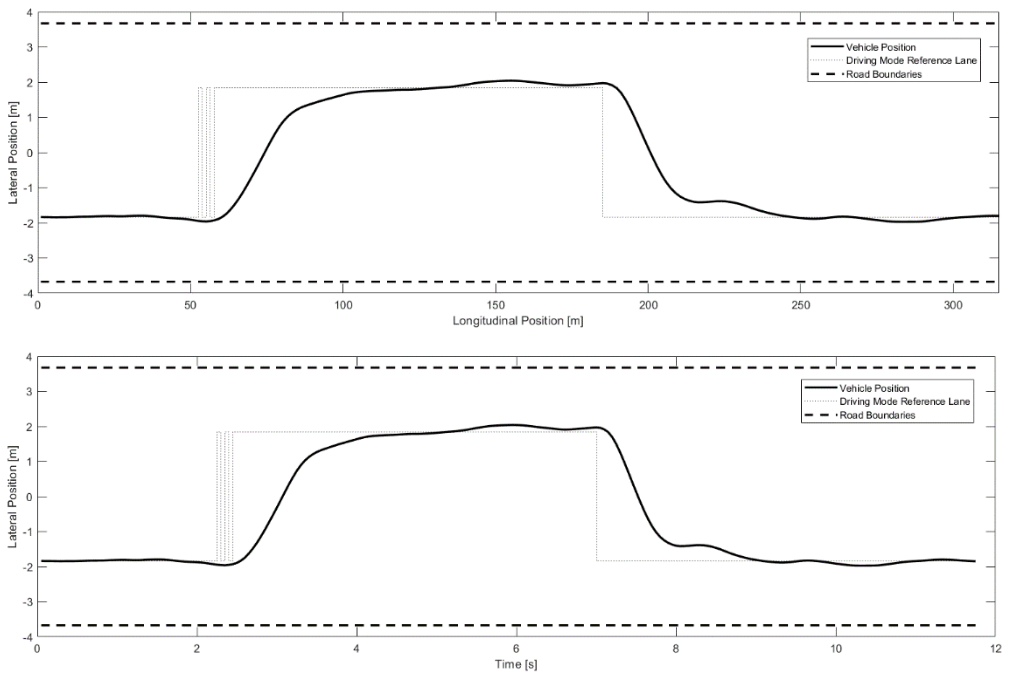

Figure 23. Lane centerlines of driving mode and vehicle position. 500 particles, 5 modes sampled, lane-keeping sampling probability parameters $P_{base} = 0.9, P_{min} = 0.1$

The inputs and the corresponding state variables are shown below in figures 24 and 25.

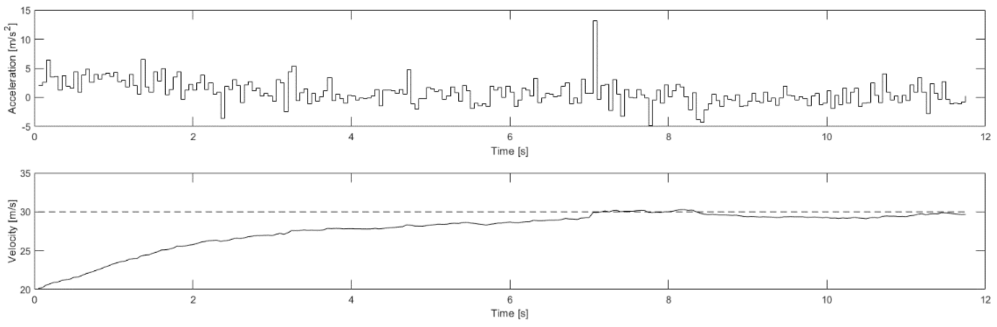

Figure 24. Acceleration inputs and corresponding velocity vs time.

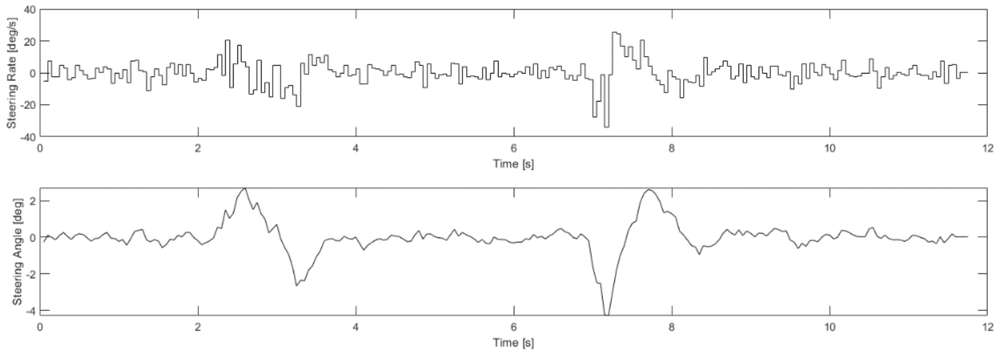

Figure 25. Steering rate inputs and corresponding angles vs time.

When these two results are compared, the improvement in performance by doubling the particle count can be seen. The vehicle's behavior is overall smoother, as seen in the lane tracking graph and the smaller changes in velocity (smaller accelerations).

The next set of results are all from a 500-particle simulation, with additional plots showing the inputs applied to the system. The same covariance matrices described earlier in this section are used here, as well as 5 modes sampled with 90% base lane-keeping sampling.

Animated simulation results are available in the GitHub repository for this project.

Conclusion

This study has developed a particle filter and reweighting particle smoother-based motion planning method for autonomous road vehicles in a dynamic highway driving environment. The generated trajectories are dynamically feasible but may be suboptimal. Barrier functions in the particle filter weight update equations and trajectory selection cost function are used to prevent collisions or road boundary violation. The generated trajectories are applied to the system in a receding horizon manner, which enables responsiveness to the dynamic environment.

The applicability of this method to more sophisticated scenarios as presented is limited, as the formulation of the obstacle regions and driving mode selection will need reworking for curved roads or more than two lanes. A potential improvement to the driving modes would be an implementation of a "following" mode for cases when there is no open lane to make a passing maneuver. This mode would set the nominal velocity of the ego vehicle to the velocity of the leading vehicle. The driving mode sampling could also be modified to be conditioned on the current mode as in [3].

Potential improvements to the trajectory generation method of forward particle filtering and reweighting particle smoothing can come from the choice of proposal distribution from which the particles are sampled at each time step. There are variations of the particle filter that seek to generate more optimal proposal distributions for nonlinear systems than the bootstrap particle filter, which may be worth exploration. This method could potentially be expanded to motion planning for multiple vehicles in a collaborative manner to enable platooning or cooperative merging.

References

[1] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135-1145, April 2016, doi: 10.1109/TITS.2015.2498841.

[2] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 577-584, doi: 10.1109/HUMANOIDS.2017.8246930.

[3] X. Qian, I. Navarro, A. de La Fortelle and F. Moutarde, "Motion planning for urban autonomous driving using Bézier curves and MPC," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 826-833, doi: 10.1109/ITSC.2016.7795651.

[4] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 1999, pp. 473-479 vol.1, doi: 10.1109/ROBOT.1999.770022.

[5] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli and J. P. How, "Motion planning for urban driving using RRT," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 1681-1686, doi: 10.1109/IROS.2008.4651075.

[6] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli and S. Teller, "Anytime Motion Planning using the RRT*," *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1478-1483, doi: 10.1109/ICRA.2011.5980479.

[7] D. Stahl and J. Hauth, “PF-MPC: Particle filter-model predictive control,” *Systems & Control Letters*, vol. 60, no. 8, pp. 632-643, 2011

[8] I. Askari, S. Zeng and H. Fang, "Nonlinear Model Predictive Control Based on Constraint-Aware Particle Filtering/Smoothing," *2021 American Control Conference (ACC)*, 2021, pp. 3532-3537, doi: 10.23919/ACC50511.2021.9482774.

[9] K. Berntorp, T. Hoang and S. Di Cairano, "Motion Planning of Autonomous Road Vehicles by Particle Filtering," in *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 197-210, June 2019, doi: 10.1109/TIV.2019.2904394.

[10] K. Berntorp, P. Inani, R. Quirynen and S. Di Cairano, "Motion Planning of Autonomous Road Vehicles by Particle Filtering: Implementation and Validation," *2019 American Control Conference (ACC)*, 2019, pp. 1382-1387, doi: 10.23919/ACC.2019.8815309.

[11] Särkkä, S. (2013). *Bayesian Filtering and Smoothing* (Institute of Mathematical Statistics Textbooks). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139344203

[12] C. Kuptametee and N. Aunsri, “A review of resampling techniques in particle filtering framework,” *Measurement*, vol. 193, p. 110836, 2022.

Appendix

Equation (22) Derivation

Markov properties of the model for developing the sequential importance sampling equations.

$$\boldsymbol{x}_k \sim p(\boldsymbol{x}_k | \boldsymbol{x}_{k-1})$$

$$\boldsymbol{y}_k \sim p(\boldsymbol{y}_k | \boldsymbol{x}_k)$$

Create recursive estimate for the full posterior $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k})$. By applying the definition of conditional probability, this can be rewritten as follows.

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k}) = \frac{p(y_k, y_{1:k-1}, x_{0:k})}{p(y_k, y_{1:k-1})} = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k-1})p(\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})p(\boldsymbol{y}_{1:k-1})}$$

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})}$$

The denominator can be removed by taking a proportionality.

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k}) \propto p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k-1})$$

Expanding the first term in this equation and applying Markov properties of the system.

$$p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1}) = \frac{p(\boldsymbol{y}_k, \boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1})}{p(\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1})}$$

$$= \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1})p(\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k-1})p(\boldsymbol{y}_{1:k-1})}$$

$$p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k-1})}$$

Plugging this back into the equation and applying the Markov property for the states and measurements.

$$p(\boldsymbol{y}_k|\boldsymbol{x}_{0:k}, \boldsymbol{y}_{1:k-1}) = p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{1:k-1})p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1})$$

$$= p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1})$$

Now, the result of (22) has been reached.

Equations (23)-(24) Derivation

These equations develop the recursive weight update for the sequential importance sampling algorithm. First, the recursive property of the proposal distribution $\pi\left(\boldsymbol{x}_{0:k}^{i}|\boldsymbol{y}_{1:k}\right)$ is demonstrated.

$$\pi\left(\boldsymbol{x}_{0:k}^{i}|\boldsymbol{y}_{1:k}\right)=\frac{\pi\left(\boldsymbol{x}_{0:k}^{i},\boldsymbol{y}_{1:k}\right)}{\pi\left(\boldsymbol{y}_{1:k}\right)}=\frac{\pi\left(\boldsymbol{x}_{k}^{i}|\boldsymbol{x}_{0:k-1}^{i},\boldsymbol{y}_{1:k}\right)\pi\left(\boldsymbol{x}_{0:k-1}^{i}|\boldsymbol{y}_{1:k}\right)\pi\left(\boldsymbol{y}_{1:k}\right)}{\pi\left(\boldsymbol{y}_{1:k}\right)}$$

$$\pi\left(\boldsymbol{x}_{0:k}^{i}|\boldsymbol{y}_{1:k}\right)=\pi\left(\boldsymbol{x}_{k}^{i}|\boldsymbol{x}_{0:k-1}^{i},\boldsymbol{y}_{1:k}\right)\pi\left(\boldsymbol{x}_{0:k-1}^{i}|\boldsymbol{y}_{1:k}\right)$$

We enforce the following property on the proposal distribution to facilitate the recursive weight update.

$$\pi\left(\boldsymbol{x}_{0:k-1}^{i}|\boldsymbol{y}_{1:k}\right)=\pi\left(\boldsymbol{x}_{0:k-1}^{i}|\boldsymbol{y}_{1:k-1}\right)$$

This property essentially says that particles at time k are independent of the future measurement $\boldsymbol{y}_{k+1}$. Applying this proposal distribution to the weight update and including the expanded proposal, arrive at (23)

$$w_{k}^{i}\propto\frac{p\left(\boldsymbol{y}_{k}|\boldsymbol{x}_{k}^{i}\right)p\left(\boldsymbol{x}_{k}^{i}|\boldsymbol{x}_{k-1}^{i}\right)p\left(\boldsymbol{x}_{0:k-1}^{i}|\boldsymbol{y}_{1:k-1}\right)}{\pi\left(\boldsymbol{x}_{0:k}^{i}|\boldsymbol{y}_{1:k}\right)}=\frac{p\left(\boldsymbol{y}_{k}|\boldsymbol{x}_{k}^{i}\right)p\left(\boldsymbol{x}_{k}^{i}|\boldsymbol{x}_{k-1}^{i}\right)}{\pi\left(\boldsymbol{x}_{k}^{i}|\boldsymbol{x}_{0:k-1}^{i},\boldsymbol{y}_{1:k-1}\right)}\times\frac{p\left(\boldsymbol{x}_{0:k-1}^{i}|\boldsymbol{y}_{1:k-1}\right)}{\pi\left(\boldsymbol{x}_{0:k-1}^{i}|\boldsymbol{y}_{1:k-1}\right)}$$

Observing that the weight for the previous estimate of the posterior at time $k-1$ is presented inside of (23), arrive at (24).

$$w_{k}^{i}\propto\frac{p\left(\boldsymbol{y}_{k}|\boldsymbol{x}_{k}^{i}\right)p\left(\boldsymbol{x}_{k}^{i}|\boldsymbol{x}_{k-1}^{i}\right)}{\pi\left(\boldsymbol{x}_{k}^{i}|\boldsymbol{x}_{0:k-1}^{i},\boldsymbol{y}_{1:k-1}\right)}\times w_{k-1}^{i}$$

Equation (33) Derivation

The derivation starts from the Bayesian fixed interval smoothing equations.

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k}) = \int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})d\boldsymbol{x}_k$$

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:T}) = p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})\int\left[\frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:T})}{p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k})}\right]d\boldsymbol{x}_{k+1}$$

From the forward filtering stage, the marginal filtering distributions are already calculated.

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:T}) \approx \sum_i w^i_{k+1|T}\delta(\boldsymbol{x}_{k+1} - \boldsymbol{x}^i_{k+1})$$

With this approximation available, the integral term can be approximated.

$$\int\left[\frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:T})}{p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k})}\right]d\boldsymbol{x}_{k+1} \approx \int\left[\frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)}{p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k})}\right]\sum_i\left[w^i_{k+1|T}\delta(\boldsymbol{x}_{k+1} - \boldsymbol{x}^i_{k+1})\right]d\boldsymbol{x}_{k+1}$$

$$= \sum_i w^i_{k+1|T}\frac{p(\boldsymbol{x}^i_{k+1}|\boldsymbol{x}_k)}{p(\boldsymbol{x}^i_{k+1}|\boldsymbol{y}_{1:k})}$$

This is by applying the following property of the Dirac delta function.

$$\int_{-\infty}^{\infty} f(x)\delta(x-a)dx = f(a)$$

The denominator term $p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k})$ is the predictive distribution also approximated using the filtering method.

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k}) \approx \sum_j w^j_k p(\boldsymbol{x}_{k+1}|\boldsymbol{x}^j_k)$$

With this approximation, arrive at the following approximation for the integral.

$$\int\left[\frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:T})}{p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k})}\right]d\boldsymbol{x}_{k+1} \approx \sum_i w^i_{k+1|T}\frac{p(\boldsymbol{x}^i_{k+1}|\boldsymbol{x}_k)}{\left[\sum_j w^j_k p(\boldsymbol{x}^i_{k+1}|\boldsymbol{x}^j_k)\right]}$$

Returning to the overall Bayesian optimal smoothing equation and substituting this approximation, arrive at the following.

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})\int\left[\frac{p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:T})}{p(\boldsymbol{x}_{k+1}|\boldsymbol{y}_{1:k})}\right]d\boldsymbol{x}_{k+1} \approx p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})\sum_i w^i_{k+1|T}\frac{p(\boldsymbol{x}^i_{k+1}|\boldsymbol{x}_k)}{\left[\sum_j w^j_k p(\boldsymbol{x}^i_{k+1}|\boldsymbol{x}^j_k)\right]}$$

Where $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ is again approximated as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \approx \sum_l w^l_k\delta(\boldsymbol{x}_k - \boldsymbol{x}^l_k)$$

Substituting this approximation, arrive at the following.

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})\sum_i w_{k+1|T}^i \frac{p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k)}{\left[\sum_j w_k^j p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^j)\right]} \approx \sum_l w_k^l \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^l)\sum_i w_{k+1|T}^i \frac{p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^l)}{\left[\sum_j w_k^j p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^j)\right]}$$

Finally, the weight update equation for the smoothing weights for particle $\boldsymbol{i}$ at time $\boldsymbol{k}$ is reached.

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:T}) \approx \sum_l w_k^l \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^l)\sum_i w_{k+1|T}^i \frac{p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^l)}{\left[\sum_j w_k^j p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^j)\right]}$$

Combining all the terms except the $\delta(\cdot)$ gives the simplified approximation to the smoothing distribution.

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:T}) \approx \sum_l w_{k|T}^l \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^l)$$

Where the weights are updated backwards using

$$w_{k|T}^l = \sum_i w_{k+1|T}^i \frac{w_k^l p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^l)}{\left[\sum_j w_k^j p(\boldsymbol{x}_{k+1}^i|\boldsymbol{x}_k^j)\right]}$$