Motion Planning in Complex Environments using Closed-loop Prediction

Yoshiaki Kuwata[*], Justin Teo[†], Sertac Karaman[‡], Gaston Fiore[§], Emilio Frazzoli[¶], and Jonathan P. How[‖]

This paper describes the motion planning and control subsystems of Team MIT's entry in the 2007 DARPA Grand Challenge. The novelty is in the use of closed-loop prediction in the framework of Rapidly-exploring Random Tree (RRT). Unlike the standard RRT, an input to the controller is sampled, followed by the forward simulation using the vehicle model and the controller to compute the predicted trajectory. This enables the planner to generate smooth trajectories much more efficiently, while the randomization allows the planner to explore cluttered environment. The controller consists of a Proportional-Integral speed controller and a nonlinear pure-pursuit steering controller, which are used both in execution and in the simulation-based prediction. The main advantages of the forward simulation are that it can easily incorporate any nonlinear control law and nonlinear vehicle dynamics, and the resulting trajectory is dynamically feasible. By using a stabilizing controller, it can handle vehicles with unstable dynamics. Several results obtained using MIT's race vehicle demonstrate these features of the approach.

I. Introduction

The 2007 DARPA Grand Challenge (DGC) was the third in a series of competitions organized by the Defense Advanced Research Projects Agency (DARPA) to accelerate research and development of autonomous vehicles for military applications. The goal was to drive a 60 mile mission in an urban environment within six hours, and Team MIT is one of the six teams that completed the race. Driving autonomously in an urban environment requires a much more sophisticated planning system than that for desert driving. For example, the planner must be able to handle cluttered environments, such as winding roads, partially blocked roads, and parking zones. Traffic vehicles make the environment change over time in an unpredictable way. Finally, traffic and rules of the road impose constraints on the vehicle's trajectories that depend not only on the instantaneous state of the vehicles, but also on their history.

Because the perceived world view changes as the vehicle traverses the environment, the plan must be made online. Although numerous algorithms have been studied in the past for robot motion planning problems,[1,2] real-time motion planning for vehicles with complex dynamics and constraints has still been a challenge. For example, grid-based searches such as A*, D*, or E*[3,4] can efficiently search for a global plan, but it is difficult to account for the nonlinear dynamics of the vehicle or tight maneuvers. Various techniques have been proposed and implemented for desert driving in DGC 2005, including manipulation of the center line of the corridor,[5] randomized approach,[6] and optimization-based planner.[7]

This paper presents a simulation-based closed-loop approach, in which the planner generates trajectories using a model of the vehicle and a low-level controller.[8] The controller typically runs at a rate that is much higher than that of the planner, stabilizing the system and dealing with small signals. By planning on this *stable* closed-loop system consisting of the controller and the vehicle (in contrast to a potentially unstable open-loop system), the planner can focus

[*]Y. Kuwata, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, kuwata@alum.mit.edu
[†]J. Teo, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, csteo@mit.edu
[‡]S. Karaman, Dept. of Mechanical Engineering, MIT, Cambridge, MA 02139, USA, sertac@mit.edu
[§]G. Fiore, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, gafiore@mit.edu
[¶]E. Frazzoli, Associate Professor, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, frazzoli@mit.edu
[‖]J. How, Professor, Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA, jhow@mit.edu

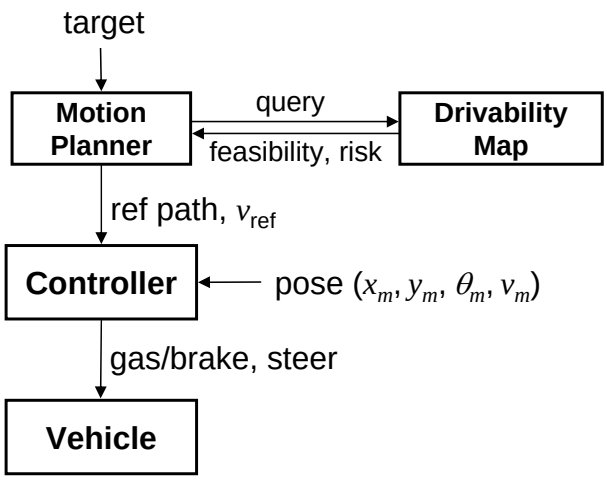


Figure 1. Planning and control system architecture.

on maneuvers of a longer time scale and efficiently generate trajectories, which is critical in real-time motion planning. In our architecture, the low-level controller follows a simple reference command while stabilizing the vehicle, and the path planner generates an input to the closed-loop system. By running forward simulation, the output of the closed-loop system is obtained, which is checked against external constraints such as obstacle avoidance for feasibility. The main advantage of the forward simulation is that the resulting trajectory is dynamically feasible by construction. Furthermore, it can easily incorporate nonlinear dynamics, nonlinear controllers, or input saturation. This closed-loop simulation is embedded in the Rapidly-exploring Random Trees (RRT)[9] framework, which can generate smooth trajectories much more efficiently than the standard RRT.

The paper starts with an overview of the system architecture. Section III describes the structure of the low-level controller. Section IV presents the planner that performs closed-loop propagation. Finally, Section V shows results of the planning approach applied to the race vehicle of Team MIT, which competed in DARPA Urban Challenge 2007.

II. Overview of the Approach

A. Problem Statement

The vehicle has nonlinear dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, where $\boldsymbol{x}$ is the states and $\boldsymbol{u}$ is the control inputs of the vehicle. Several constraints are imposed on the vehicle such as control saturation, actuator lag, and speed bounds. These are represented by $\boldsymbol{x} \in X$ and $\boldsymbol{u} \in U$. A set of environmental constraints on the states, such as staying in lane boundaries and avoiding obstacles and movers, are represented by $\boldsymbol{x}(t) \in X_{\text{free}}(t)$. This paper considers a motion planning problem that, given the current states $\boldsymbol{x}(t_0) \in X \cap X_{\text{free}}(t_0)$, generates a trajectory $\boldsymbol{x}(t),\ t \in [t_0, t_f]$ and associated control input history $\boldsymbol{u}(t),\ t \in [t_0, t_f]$ for a vehicle visiting a target region $X_{\text{goal}} \subseteq X$ while satisfying constraints $\boldsymbol{x}(t) \in X \cap X_{\text{free}}(t)$, $u(t) \in U$ for all $t \in [t_0, t_f]$. An implicit assumption is that there exists a $t_f \in (t_0, \infty)$ such that $X_{\text{goal}} \subseteq X \cap X_{\text{free}}(t_f)$, and the sets $X \cap X_{\text{free}}(t)$ and U are non-empty for $t \in [t_0, t_f]$.

The operating environment is dynamic and uncertain, so the plan must be generated online. Furthermore, to quickly react to the sudden change in the situational awareness, the planning interval must be as short as 0.1 second. This paper used a car as the vehicle, but the approach discussed is easily applied to vehicles with any type of dynamics.

B. Execution Modules

Figure 1 shows the system architecture of the closed-loop planning approach. The planner receives a short-term target X_{goal} from a high-level route planner. The planner runs at 10 Hz, and every 0.1 second, the path and the associated

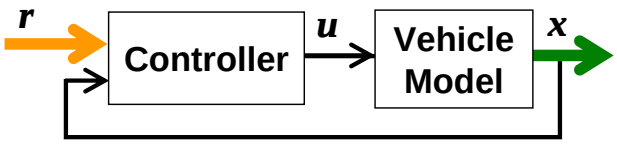


Figure 2. Closed-loop prediction. The red arrow shows the input to the controller. The green arrow shows the output of the closed-loop simulation.

speed command are sent to the controller. The controller computes the command $\boldsymbol{u}$ in terms of gas/brake and steer to track the planned path, and send them to the vehicle hardware at 25 Hz. The perceived environment is stored in the form of a drivability map, which the planner uses to evaluate the feasibility of the path. The update rate of the drivability map is the same as the running rate of the planner. The vehicle states $\boldsymbol{x}$ is coming into the system at around 100 Hz.

C. Closed-loop Prediction

The basic approach is that the planner generates a reference input $\boldsymbol{r}$ to the *stable* closed-loop system, which consists of the controller and the vehicle. By running forward simulation using the controller and the vehicle model, the output $\boldsymbol{x}$ of the closed-loop prediction is obtained, as shown in Figure 2. The feasibility of this output is checked against environmental constraints such as obstacle avoidance. The planner's role here is to generate a "large" signal in the form of the controller's input, and it is the controller's task to track the commanded path in a "small" signal sense.

Compared to the standard approach that samples the input $\boldsymbol{u}$ to the vehicle,[2,8] this closed-loop approach has several advantages. Firstly, by wrapping the vehicle with a stabilizing controller, this approach works with vehicles exhibiting unstable dynamics such as cars and helicopters. Secondly, the use of a stabilizing controller reduces the prediction mismatch typically caused by modeling errors of the vehicle. Thirdly, the forward simulation can easily incorporate nonlinear vehicle models and/or controllers, and the resulting trajectory is dynamically feasible by construction. Finally, a single input to the closed-loop system can create a long trajectory (on the order of a few seconds) while the stabilizing controller smoothly changes the input to the vehicle. This significantly improves the efficiency of randomized approaches such as RRT, because it is difficult to generate a good sequence of vehicle inputs if the input to the vehicle is drawn randomly, particularly for open-loop unstable systems.

III. Controller

The controller generates gas/brake and steer commands $\boldsymbol{u}$ that are used to move the vehicle along some desired trajectory[a]. It comprises two core modules: the pure-pursuit steer controller and the Proportional-Integral (PI) speed controller. These two core modules are used for both trajectory prediction in the planner and final execution by the controller.

For simplicity, a piece-wise linear reference path is used as a controller input $\boldsymbol{r}$, which is generated by connecting random samples. Tightly tracking such continuous but non-smooth reference trajectory would require relatively high bandwidth controllers. However, when viewed at the system level, the closed-loop simulation approach does not have a strict requirement for good tracking with respect to the reference trajectory. It is more important for the actual trajectory to match the prediction well. Hence, low bandwidth controllers can be used that may not tightly track the reference trajectory but are less sensitive to delays, noise, and modeling errors. Note, however, that because the performance is evaluated based on the resulting trajectory $\boldsymbol{x}$, if the resultant trajectory $\boldsymbol{x}$ deviates significantly from the reference trajectory $\boldsymbol{r}$, it would be difficult to generate a controller input $\boldsymbol{r}$ that achieves good overall objectives. Thus, there is a trade-off between the ease of controller design/implementation and ease of reference generation.

[a]The time independent path will be simply called "path" while the time parameterized path will be called "trajectory".

A. Steering Controller

The steering controller is based on the so-called pure-pursuit controller. Pure-pursuit control has been widely used in ground robots[10] and more recently in unmanned air vehicles[11] for path-following. It is adopted due to its simplicity, as an intuitive control law that has a clear geometric meaning.

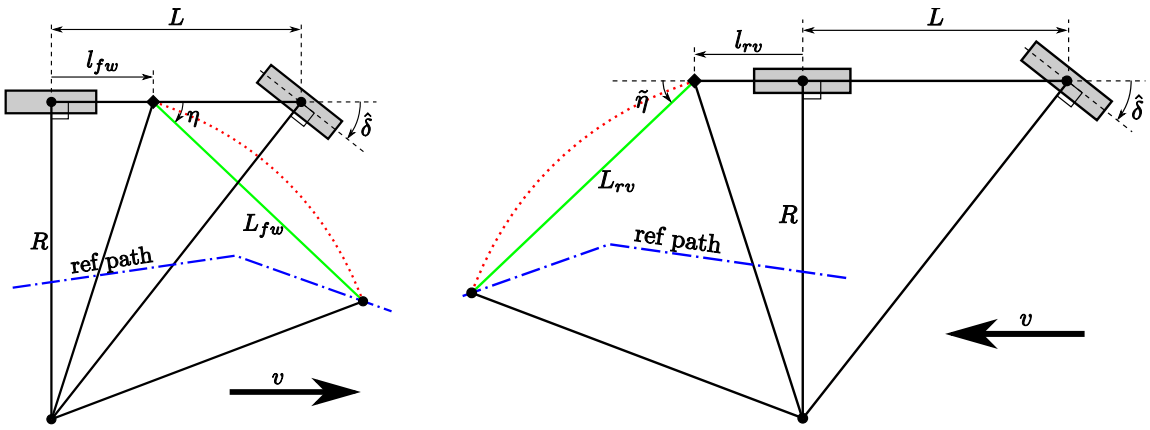
For the application of interest, namely the DARPA Urban Challenge, the operating speed is restricted to under 30 mph. By appropriately restricting the class of allowable maneuvers at the system level, the steer control problem can be treated as purely kinematic. For control design purposes only, dynamic effects such as sideslip are ignored. Treating the problem as purely kinematic simplifies it significantly and made pure-pursuit directly applicable. Note that the vehicle model used in the prediction includes sideslip.

1. Control Law

The kinematic bicycle model is described by

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\delta \end{aligned} \tag{1}$$

where, x and y refer to the rear axle position, θ is the vehicle heading with respect to the x-axis (positive counter-clockwise), v is the forward speed, δ is the steer angle (positive counter-clockwise), and L is the vehicle wheelbase. This model describes a slip-free nonholonomic vehicle moving in the plane at a speed of v. With the slip-free assumption, Figure 3(a) shows the definition of the variables that define the pure-pursuit control law when driving forwards, and Figure 3(b) shows the variables when driving in reverse. Here, R is the radius of curvature, and "ref path" is the



(a) Forward Drive

(b) Reverse Drive

Figure 3. Definition of pure-pursuit variables. Two shaded rectangles represent the rear tire and the steerable front tire in the bicycle model. The steering angle in this figure, $\hat{\delta}$, corresponds to negative δ in Eq. (1).

piecewise linear reference path given by the planner. In Figure 3(a), l_{fw} is the distance of the forward *anchor point* from the rear axle, L_{fw} is the forward drive look-ahead distance, and η is the heading of the look-ahead point (constrained to lie on the reference path) from the forward anchor point with respect to the vehicle heading. Similarly, in Figure 3(b), l_{rv} is the rearward distance of the reverse *anchor point* from the rear axle, L_{rv} is the reverse drive look-ahead distance, and $\tilde{\eta}$ is the heading of the look-ahead point from the reverse anchor point with respect to the vehicle heading offset by π rad. All angles and lengths shown in Figure 3 are positive by definition.

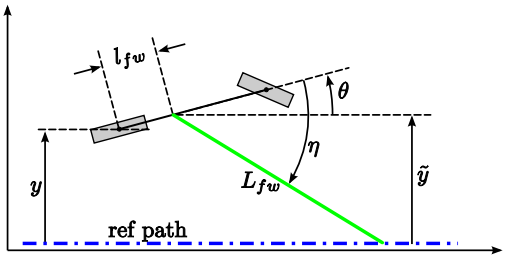


Figure 4. Straight reference path along x-axis

It can be shown that the instantaneous steer angle required to put the corresponding anchor points on a collision course with the corresponding look-ahead points are given by

$$\delta = -\tan^{-1}\left(\frac{L\sin\eta}{\frac{L_{\text{fw}}}{2} + l_{\text{fw}}\cos\eta}\right) \qquad \text{: forward drive}$$

$$\delta = -\tan^{-1}\left(\frac{L\sin\tilde{\eta}}{\frac{L_{\text{rv}}}{2} + l_{\text{rv}}\cos\tilde{\eta}}\right) \qquad \text{: reverse drive}$$

This is the modified pure-pursuit control law adopted. Note that by setting $l_{\text{fw}} = 0$ and $l_{\text{rv}} = 0$, the anchor points coincide with the rear axle, recovering the conventional pure-pursuit controller.[10] The subsection below shows that having $l_{\text{fw}} > 0$ and $l_{\text{rv}} > 0$ will offer a larger relative stability compared to having $l_{\text{fw}} = 0$ and $l_{\text{rv}} = 0$.

2. *Linear Stability Analysis*

The above steer control law did not account for the finite response times of the actuators. It has been previously shown that finite actuation imposes a stability constraint on the system.[12] This subsection uses the Routh-Hurwitz criterion and generalizes part of the results in Ref. [12] for the modified pure-pursuit control law.

The steering actuator for our system has a constant maximum slew rate, $|\dot{\delta}| \leq \dot{\delta}_{\max}$. This constraint can be approximated by modeling the actuator dynamics as a first order system

$$\dot{\delta} = \frac{1}{\tau}\left(-\delta + \delta_c\right) \tag{2}$$

where δ_c is the steer command, and τ is the actuator time constant. τ is chosen so that the steer angle reaches 90% of the commanded steer in the time that it would have taken to deflect from zero steer to full right/left steer. For forward drive, the steer control law is then written as

$$\delta_c = -\tan^{-1}\left(\frac{L\sin\eta}{\frac{L_{\text{fw}}}{2} + l_{\text{fw}}\cos\eta}\right). \tag{3}$$

Consider the case where the reference path is a straight line coincident with the x-axis, as shown in Figure 4. From the geometry, the following relation holds

$$\eta = \theta + \sin^{-1}\left(\frac{y + l_{\text{fw}}\sin\theta}{L_{\text{fw}}}\right). \tag{4}$$

For $v > 0$, the equilibrium of the kinematic model Eq. (1) augmented with the approximate actuator dynamics Eq. (2) is characterized by $\dot{y} = 0$, $\dot{\theta} = 0$, and $\dot{\delta} = 0$. From Eqs. (1)–(4), this implies $\theta = 0$, $\delta = 0$, $\delta_c = 0$, $\eta = 0$, and $y = 0$ at equilibrium. Define the state $z = [y \quad \theta \quad \delta]^{\text{T}}$. Linearizing the closed-loop system Eqs. (1)–(4) for z about

this equilibrium, we have the closed-loop linear dynamics as

$$\dot{z} = Az, \quad \text{where} \quad A = \begin{bmatrix} 0 & v & 0 \\ 0 & 0 & \frac{v}{L} \\ -\frac{L}{\tau L_{\text{fw}}\left(\frac{L_{\text{fw}}}{2}+l_{\text{fw}}\right)} & -\frac{L\left(L_{\text{fw}}+l_{\text{fw}}\right)}{\tau L_{\text{fw}}\left(\frac{L_{\text{fw}}}{2}+l_{\text{fw}}\right)} & -\frac{1}{\tau} \end{bmatrix}. \tag{5}$$

The characteristic equation of the above matrix, i.e., $\det(sI - A) = 0$, can be written as

$$s^3 + \frac{1}{\tau}s^2 + \frac{v\left(L_{\text{fw}} + l_{\text{fw}}\right)}{\tau L_{\text{fw}}\left(\frac{L_{\text{fw}}}{2} + l_{\text{fw}}\right)}s + \frac{v^2}{\tau L_{\text{fw}}\left(\frac{L_{\text{fw}}}{2} + l_{\text{fw}}\right)} = 0.$$

Applying the Routh-Hurwitz criterion, the system Eq. (5) will be stable if

$$\frac{1}{\tau} > 0, \tag{6}$$

$$\frac{v\left(L_{\text{fw}} + l_{\text{fw}} - v\tau\right)}{\tau L_{\text{fw}}\left(\frac{L_{\text{fw}}}{2} + l_{\text{fw}}\right)} > 0, \tag{7}$$

$$\frac{v^2}{\tau L_{\text{fw}}\left(\frac{L_{\text{fw}}}{2} + l_{\text{fw}}\right)} > 0. \tag{8}$$

Eqs. (6) and (8) are automatically satisfied since $\tau > 0$, $L_{\text{fw}} > 0$, and $l_{\text{fw}} > 0$. Because $v > 0$ for forward drive, Eq. (7) gives the following stability criterion

$$L_{\text{fw}} > v\tau - l_{\text{fw}}. \tag{9}$$

This shows that the look-ahead distance L_{fw} must increase with v to maintain stability. Also, for the same $L_{\text{fw}}(v)$, having $l_{\text{fw}} > 0$ will generally offer a larger relative stability compared to having $l_{\text{fw}} = 0$ [b]. The meaning of Eq. (9) in the small signal case is that the look-ahead point must always lie ahead of the rear axle by a scalar multiple of the speed. The more sluggish the actuator, the further the look-ahead point must be located. Note that by repeating the above procedure for reverse drive, an analogous criterion

$$L_{\text{rv}} > -v\tau - l_{\text{rv}}$$

is obtained, where $v < 0$ for reverse drive.

3. Choice of Look-Ahead Distance

Because of the random and non-smooth nature of the reference paths, the system frequently has to operate in regions that violate the small signal assumption. Furthermore, for reasons that will be highlighted later, the look-ahead distance is scheduled with the commanded speed v_{cmd} instead of the vehicle speed. As such, the look-ahead distance profile has to be chosen with some conservatism compared to Eq. (9). On the other hand, having an unnecessarily large look-ahead distance will reduce the ability to track the reference path. The rate limit of the vehicle was measured to be $\dot{\delta}_{\max} = 0.406$ rad/s, giving $\tau = 0.717$ sec. Through extensive simulations and field tests, the following numbers were selected

$$L_{\text{fw}}(v_{\text{cmd}}) = L_{\text{rv}}(v_{\text{cmd}}) = \begin{cases} 3 & \text{if } v_{\text{cmd}} < 1.34 \text{ m/s}, \\ 2.24\, v_{\text{cmd}} & \text{if } 1.34 \text{ m/s} \le v_{\text{cmd}} < 5.36 \text{ m/s}, \\ 12 & \text{otherwise.} \end{cases}$$

Note that $L_{\text{fw}}(v_{\text{cmd}})$ and $L_{\text{fw}}(v_{\text{cmd}})$ has units in meters. This is depicted in Figure 5 with the linear stability bound Eq. (9), across the speed range of interest. From here on, both L_{fw} and L_{rv} are referred to as L_1 in this paper.

[b] It can be shown that the maximum relative stability is bounded above by $\frac{1}{3\tau}$, independent of l_{fw} and l_{rv}. The relative stability cannot be improved indefinitely by simply increasing l_{fw} and l_{rv}.

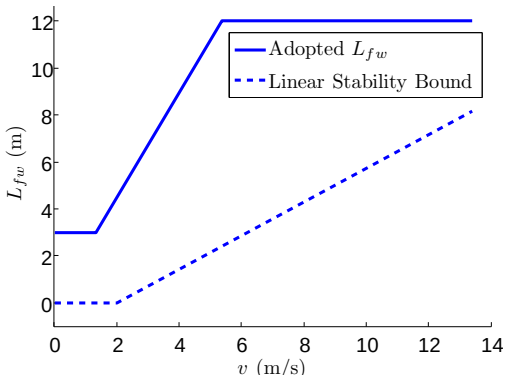


Figure 5. Look-ahead Distance as a function of the speed command

B. Speed Controller

The objective of having simple controllers led to the consideration of Proportional-Integral-Derivative (PID) type controllers for speed tracking. However, our assessment is that the PID controller offers no significant advantage over the PI controller because the vehicle has some inherent speed damping and the acceleration signal required for PID control is noisy. Hence, the PI controller of the following form is adopted

$$u = K_p \left(v_{\text{cmd}} - v\right) + K_i \int_0^t \left(v_{\text{cmd}} - v\right) d\tau \tag{10}$$

where u is the nondimensional speed control signal, K_p and K_i are the proportional and integral gains respectively, and v_{cmd} is the commanded speed.

1. *Vehicle Speed Model*

For simplicity, the same controller is used for acceleration (applying gas) and deceleration (applying brake). The following Linear Parameter Varying speed model is used for control design purposes.

$$\begin{aligned} \frac{V(s)}{U(s)} &= \frac{K_n(v)}{\tau_v s + 1} \\ K_n(v) &= 0.1013v^2 + 0.5788v + 49.1208 \end{aligned} \tag{11}$$

where $V(s)$ and $U(s)$ are the Laplace transforms of $v(t)$ and $u(t)$ respectively, τ_v $(= 12$ s$)$ is the speed dynamics time constant, and $K_n(v)$ is the speed dependent gain of the speed dynamics. The coefficients in Eq. (11) were found using system identification techniques.

2. *Parameter Space Approach*

For the design of the PI controller, the parameter space approach[13] is used. The basic idea is to translate design specifications into desirable regions in the controller parameter space, which in our case, is the space of the pair (K_p, K_i). The region in the parameter space that satisfies all design specifications defines a set of controllers achieving those specifications. Instead of giving exact controller parameters, this method provides a visual understanding of the effect of parameter changes to a set of criteria. Such methods have been used successfully to design controllers for road and air vehicles.[13] It is then a simple matter to choose one point in that region for implementation.

First, define a *D-Stable* region that corresponds to desired closed-loop pole locations in the complex plane, as shown in Figure 6. This D-Stable region is then mapped to the parameter space. Since the D-Stable region lies completely in the open left half complex plane, the corresponding region in the parameter space defines stabilizing

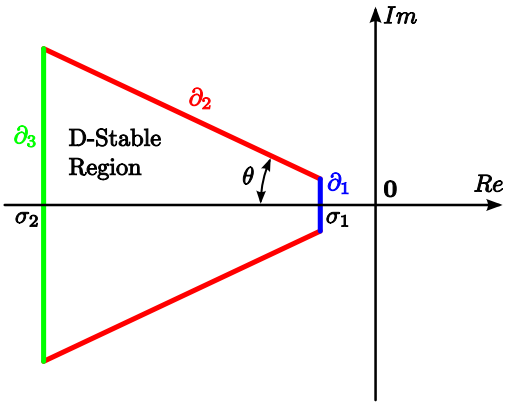


Figure 6. D-Stable region in the complex plane

controllers. The boundary ∂_1 ensures a minimum relative stability, ∂_2 ensures minimum damping, while ∂_3 ensures that the closed-loop bandwidth is not unrealistically large.

To perform this mapping, we note that the characteristic polynomial $p(s)$ of the closed-loop system, defined by Eq. (11) and the transfer function of Eq. (10), is given by

$$p(s) = \tau_v s^2 + (1 + K_n K_p)\, s + K_n K_i.$$

By making the substitution $s = -\sigma_1 + j\omega$ for ∂_1, solving for K_p and K_i from $p(s) = 0$, and sweeping ω over $(0, \infty)$, the *complex root boundary* is obtained as

$$\partial_{1c} = \left\{ (K_p, K_i) \mid K_p = \frac{2\sigma_1\tau_v - 1}{K_n},\, K_i > \frac{\tau_v \sigma_1^2}{K_n} \right\}.$$

By making the substitution $s = -\sigma_1$ for ∂_1, the *real root boundary* is obtained as

$$\partial_{1r} = \left\{ (K_p, K_i) \mid K_n K_i - (1 + K_n K_p)\, \sigma_1 + \tau_v \sigma_1^2 = 0 \right\}.$$

A simple check will show that the region in the parameter space defined by

$$R_p\left(\partial_1\right) = \left\{ (K_p, K_i) \mid K_p \geq \frac{2\sigma_1\tau_v - 1}{K_n},\, K_i \geq \frac{(1 + K_n K_p)\, \sigma_1 - \tau_v \sigma_1^2}{K_n} \right\}$$

is the desired region whose points correspond to closed-loop system poles lying on the left of ∂_1 in the complex plane.

Similarly, with appropriate substitutions, we can map ∂_2 and ∂_3 into the parameter space to obtain $R_p\left(\partial_2\right)$ and $R_p\left(\partial_3\right)$ as

$$R_p\left(\partial_2\right) = \left\{ (K_p, K_i) \mid K_p \geq -\frac{1}{K_n},\, \frac{\left(1 + K_n K_p\right)^2}{4\tau_v K_n \cos^2\theta} \geq K_i \geq 0 \right\}$$

$$R_p\left(\partial_3\right) = \left\{ (K_p, K_i) \mid K_p \leq \frac{2\sigma_2\tau_v - 1}{K_n},\, K_i \geq \frac{(1 + K_n K_p)\, \sigma_2 - \tau_v \sigma_2^2}{K_n} \right\}.$$

The region in the parameter space that maps from the D-Stable region is then given by $R_p\left(\partial_1\right) \cap R_p\left(\partial_2\right) \cap R_p\left(\partial_3\right)$.

Next, we consider the system delayed by T seconds

$$\frac{V(s)}{U(s)} = \frac{K_n(v)}{\tau_v s + 1} e^{-Ts}, \tag{12}$$

and map the constant phase margin boundary for this system to the parameter space. It can be shown[14] that for a desired phase margin of m_ϕ, and given T, the corresponding boundary is

$$\partial_{m_\phi,T} = \{(K_p, K_i) \mid K_p = f(m_\phi, T, \omega), K_i = g(m_\phi, T, \omega), \omega \in (0, \infty)\},$$

where

$$f(m_\phi, T, \omega) = \frac{\tau_v \omega \sin(m_\phi + T\omega) - \cos(m_\phi + T\omega)}{K_n}$$
$$g(m_\phi, T, \omega) = \frac{\tau_v \omega^2 \cos(m_\phi + T\omega) + \omega \sin(m_\phi + T\omega)}{K_n}.$$

The parameter space approach does not directly address the system's disturbance rejection, tracking performance, or noise attenuation properties. On the other hand, well established H_∞ robust control methods handles these directly by minimizing the weighted sensitivity and complementary sensitivity functions.[15] The relation between these two approaches have been established in the literature,[16] allowing ideas in H_∞ methods to be applied in the parameter space setting. The method in Ref. [16] is used to enforce the robust performance constraint. Accordingly, we find the region in the parameter space satisfying

$$\| |W_S S| + |W_T T| \| < 1. \tag{13}$$

In Eq. (13), S and T are the sensitivity and complementary sensitivity functions respectively, and W_S and W_T are the corresponding frequency dependent weights. The guidelines for choosing W_S and W_T[16] are: i) good tracking performance to reference inputs of frequencies up to ω_S requires $|S(j\omega)| \ll 1$ for $0 \leq \omega \leq \omega_S$; ii) rejection of sensor noise of frequencies in $[\omega_T, \infty)$ requires $|T(j\omega)| \ll 1$ for $\omega_T \leq \omega$; iii) stability against multiplicative unstructured uncertainty, modeled by $W_\Delta(s)\Delta(s)$ with $\|\Delta(s)\| \leq 1$, requires $\|W_\Delta(s)T(s)\| < 1$. Note that iii) is a consequence of the *robust stability theorem*[17] which states that the feedback control system with multiplicative unstructured uncertainty $W_\Delta(s)\Delta(s)$ is stable for all Δ such that $\|\Delta(s)\| \leq 1$ if and only if $\|W_\Delta(s)T(s)\| < 1$. Since $S(s) + T(s) = 1$, the weights must be chosen such that each function is minimized in the appropriate frequency range of interest, to achieve a small sensitivity S in the low-frequency regime and small complementary sensitivity T in the high-frequency regime. The method to find the boundaries in the parameter space satisfying Eq. (13) is detailed in Ref. [16], and is omitted for brevity.

3. Design Specifications and Feasible Region

From field tests, it was found that the vehicle cannot accelerate more than 3 m/s^2. Requiring the acceleration to a 5 mph step command to be at least 0.5 m/s^2, and not more than 3 m/s^2, produces $\sigma_1 = 0.224$ and $\sigma_2 = 1.34$. For a maximum overshoot of not more than 3%, we have $\theta = 41.9$ deg, as defined in Figure 6. From field tests, the system exhibits a communications delay of between 10 and 20 ms, reaching up to 150 ms for sporadic short periods of time. Hence, T in Eq. (12) is set to be 0.15 sec. The minimum desired phase margin is set at $m_\phi = 60$ deg. The robust performance criterion is specified by the weighting functions W_S and W_T which are chosen to be

$$W_S(s) = \frac{s + h_S\omega_S}{h_S s + h_S\omega_S l_S}$$
$$W_T(s) = \frac{h_T s + h_T\omega_T l_T}{s + h_T\omega_T}.$$

With $l_S = 0.5$, $h_S = 2$, $\omega_S = 1$, the tracking error is ensured to be less than half the reference for frequencies less than 1 rad/s, allowing it to be up to twice the reference for higher frequencies. Note that this choice of W_S does not correspond to a high tracking performance specification because our main objective is to have the resultant trajectory track the prediction well, rather than the reference, as previously discussed. Moreover, the specification of tracking performance is limited by the constraint to have a non-empty feasible region in the parameter space, dictated in part by fundamental limitations in engine performance. With $l_T = 0.1$, $h_T = 2$, and $\omega_T = 100$, robust stability against

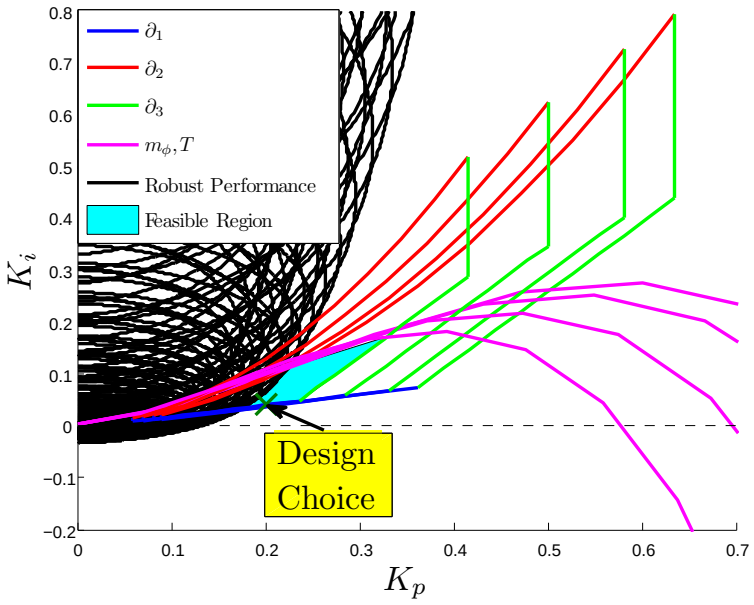


Figure 7. Feasible Region in Parameter Space

system gain variations up to 200% in high frequencies, and good sensor noise attenuation above 100 rad/s are ensured.

With these specifications, the feasible region in the parameter space for $v = 0, 10, 20, 30$ mph is shown in Figure 7, together with our design choice of $K_p = 0.2$, $K_i = 0.04$. The final design point lies within the feasible region near the real root boundaries of ∂_1. Considering all the design specifications, it is clear that our choice is a low bandwidth controller with good noise attenuation and small steady-state error properties.

IV. Planner

A. RRT Planner

As a planner, the standard RRT is extended to incorporate the closed-loop prediction discussed in Section II-C. Each node in the closed-loop RRT stores the states of the vehicle $\boldsymbol{x}$ and the corresponding controller input $\boldsymbol{r}$. The algorithm flow is shown in Algorithm 1 and is similar to the standard RRT, where randomly generated samples are added to the tree based on obstacle feasibility. The key difference is that the sample generated on line 5 of Algorithm 1 is an $(x,\ y)$ point but is not taken in the state space or the configuration space. Instead, it is in the controller input space.

As discussed in Section III-A, the input to the steering controller is a series of straight lines called a reference path. The planner generates this reference path by connecting a sample to the controller input at a node in the tree. Figure 8 shows a sample and potential reference paths.

For each sample, a node is selected from the tree using some heuristics on line 6. Then, a straight line from the controller input of the selected node to the sample is drawn. Using this straight line as an input, the forward state propagation on line 8 starts from the vehicle states at the selected node. The propagation stops when the controller finishes executing the input. The propagated trajectory is then checked with obstacle avoidance constraints on line 9, and if it is feasible, the sample is added to the tree. Otherwise, a different node in the tree is selected, and the process is repeated. Note that each node in the tree maintains two basic information, one for the input to the controller, and the other for the predicted trajectory of the vehicle, which is the output of the forward simulation. Figure 8 shows the tree of controller inputs with orange straight lines, and that of the predicted trajectories with curvy green lines.

Once the sample is either added to the tree or discarded, the next sample is taken, and the process repeats until the

Algorithm 1 RRT algorithm

```
1: repeat
2:    Measure the current vehicle states and environment.
3:    Propagate the states by the computation time limit.
4:    repeat
5:       Generate a sample for the input to the controller
6:       Sort the nodes in the tree using heuristics
7:       for each sorted node do
8:          Form the controller input by drawing a line from the node to the sample, then propagate
9:          if the propagated portion is collision free then
10:            Add the sample to the tree. break
11:         end if
12:      end for
13:   until the time limit is reached
14:   Choose the best path and repropagate from the current states
15:   if the repropagated trajectory is infeasible then
16:      Remove the infeasible portion from the tree and go to line 14
17:   end if
18:   Send the best path to the controller
19: until the vehicle reaches the target
```

time limit is reached. At the end of each planning iteration on line 14, the best path is selected, and the corresponding controller input, which is a sequence of $(x,\ y)$ points, is sent to the execution controller.

B. Accounting for the Prediction Error

When the controller does not accurately track the reference path due to modeling errors or disturbances, the planner could change the reference path so that the vehicle achieves the original desired path. However, it introduces an additional feedback loop, potentially destabilizing the overall system. When both the planner and the controller try to correct for the same error, they could be overcompensating or negating the effects of the other.

In our approach, the planner generates a "large" signal in the form of the reference path, but it does not do any adjustment. It is controller's responsibility to track the path in a "small" signal sense. Thus, the propagation on line 8 starts from the predicted vehicle states at the node. One challenge here is that the collision is checked with the predicted trajectory, which can be as long as several seconds. In order to ensure that the actual vehicle is free from collisions, it is critical to keep the prediction error small. This section presents several techniques to reduce prediction errors.

1. Use of v_{cmd} for L_1 scheduling

Eq. (9) shows that the look-ahead distance L_1 has to increase with vehicle speed to maintain stability. However, scheduling the steering gain L_1 based on the speed v means that any prediction error in the speed directly translates into a discrepancy in the steer command between prediction and execution, introducing a lateral prediction error. This is problematic because achieving a small speed prediction error is very challenging especially during the transient in the low-speed regime, where the engine dynamics and gear shifting of the automatic transmission exhibits complicated nonlinearities. Another disadvantage of scaling L_1 using v is that the noise in the speed estimate introduces jitters in the steering command.

The solution to this problem is to use the speed command v_{cmd} to schedule the L_1 distance. The speed command is designed by the planner during the propagation, as discussed later in Section D, so the planner and the controller have the same v_{cmd} with no ambiguity. Then, the same L_1 steering gain is used in the prediction and the execution, making the steering prediction decoupled from the speed prediction.

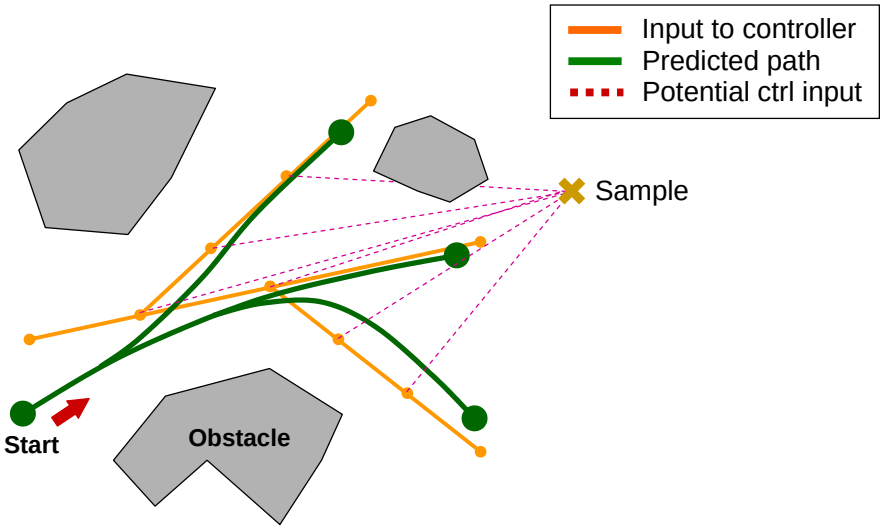


Figure 8. The extension of RRT, which maintains a pair of trees.

2. *Space-dependent speed command*

When the speed command is defined as a function of time, the prediction error in the speed could affect the steering performance, even if the look-ahead distance L_1 is scaled based on v_{cmd}. Suppose the vehicle accelerates from rest with a ramp command. If the actual vehicle accelerates more slowly than predicted, the time-based v_{cmd} would increase faster than the prediction with respect to the travel distance. This means that L_1 increases more in the execution before the vehicle moves much, and given a vehicle location, the controller places a look-ahead point farther down on the reference path compared to prediction.

To overcome this issue, the speed command is tied to the position of the vehicle with respect to the predicted path. Then, the steering command depends only on where the vehicle is, and is insensitive to the prediction error of the speed.

3. *Repropagation*

Due to the inherent modeling error or the disturbance, the state prediction always has non-zero errors. One approach to account for the prediction error is to discard the tree and rebuild it from scratch, because the current vehicle states are no longer on the predicted path stored in the tree. However, this would frequently discard the tree and is very inefficient. Because the tree is constructed after a series of computation, such as random samples landing on a good location, nonlinear state propagation, and feasibility check with the drivability map, the tree should be retained as much as possible especially in real-time applications.

The proposed approach is to reuse the controller input stored in the tree and repropagate from the latest states, as shown in Figure 9. The main advantage is that it retains the information stored in the tree no matter how large/small the prediction error is. Using a stable low-level controller, the difference between the original prediction and repropagation will converge to zero in the limit.

When applying the repropagation to the RRT framework, one can repropagate over the entire tree from the latest states. Although keeping the controller input, this approach is essentially discarding all the state trajectories that are previously computed, and can be computationally expensive. A more efficient approach is to repropagate from the latest states only along the best sequence of nodes at the end of computation time limit, as shown in line 14 of Algorithm 1. If the repropagated trajectory is collision free, the corresponding controller input is sent to the controller. Otherwise, the infeasible part of the tree is deleted from the tree, and the next best path is selected. This approach

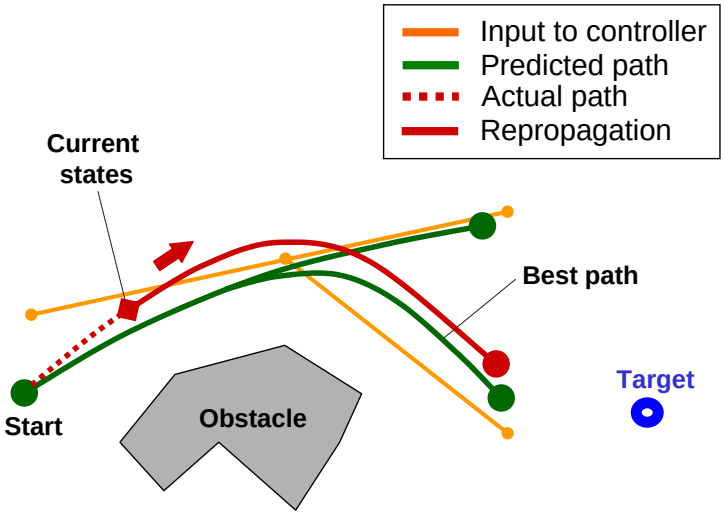


Figure 9. Repropagation from the current states.

requires very few re-evaluation while ensuring the feasibility of the plan that is sent to the controller even if the vehicle is not accurately following the original prediction.

C. Safety

The approach has several safety mechanisms.

1. Stopping Nodes

To ensure that the car can always come to a safe stop, all the leaf nodes have zero speed. This means that when the car finish executing the plan, it stops in X_{free}. To achieve this, the termination criteria for each forward simulation is that the vehicle comes to a complete stop, i.e.,

$$v = 0$$
$$v_{\text{cmd}} \leq 0.$$

The design of v_{cmd} is detailed later in Section D.

2. Maximum Lateral Acceleration

During the forward simulation, if the lateral acceleration exceeds a prespecified limit, the propagation is stopped and returns infeasibility. Then, the same path shape is tried with a lower speed limit, and if the lateral acceleration still exceeds the limit, the sample is discarded.

There are two thresholds - one used for growing the tree, and the other used during the repropagation. To minimize the chance of exceeding the limit during the repropagation, a tighter threshold is used for the tree growth compared to the repropagation.

3. Controller Override

While the planner has safety mechanisms in place, the controller also provide an independent layer of safety checks and override the motion plan when the safety conditions are triggered. The controller will override the motion plan when: i) predicted lateral acceleration exceeds rollover safety limit, ii) measured lateral acceleration exceeds rollover

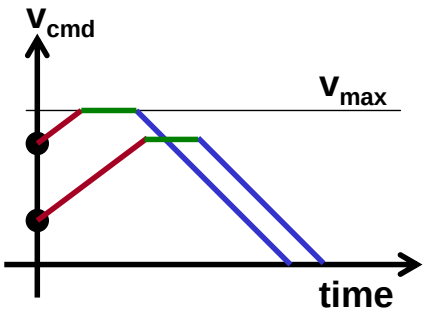


Figure 10. Speed command profile. The red line shows the initial ramp, the green line shows the coasting, and the blue line shows the ramp down. Note that in order to ensure enough coasting time for a smooth behavior, the speed command does not necessarily reach the speed limit $v_{\max}$.

safety limit, iii) state estimate not updated for up to some time limit, iv) measured roll angle exceeds safety limit, v) measured pitch angle exceeds safety limit, or vi) engine RPM exceeds safety limit. If the state estimate is not updated, this represents a critical system error and the controller will command maximum brake and center steer. For all other conditions, the controller will attempt to follow the reference path while applying moderate braking until the vehicle stops. The reason for moderate braking is so that it doesn't exacerbate the situation.

4. Emergency Braking

When the environment is dynamic and the constraints change significantly, all the trajectories stored in the tree could become infeasible. When no feasible trajectory is found while the vehicle is moving, the planner issues an emergency braking command. One simple approach is to remember the last steering command and slow down the vehicle with that steering command. However, this could easily drive the vehicle off the road when emergency braking is issued at the transition between curvy and straight road segments.

In our approach, the emergency plan consists of the last feasible steering profile over the previous plan and the emergency braking speed profile. To construct this plan, the predicted path is first obtained from the last feasible plan, which is stored by the planner. This gives the steering profile with respect to the vehicle location, and the emergency brake profile is then applied. The reference path is calculated by solving for η in Eq. (3), given δ_{cmd}, v_{cmd}, and hence $L_1(v_{\text{cmd}})$. This makes the vehicle follow the last feasible path while slowing down at the maximum deceleration.

D. Speed Design

Given a sample and a node to connect the sample to, a reference path is defined. The speed command profile is then designed so that the car comes to a stop at the end, ensuring the safety of the car. This "end" is defined as the location where the anchor point is minimum L_1 away from the end of the reference path.

To simplify the speed design, the speed command in every propagation has three segments: initial ramp up/down, coasting, and ramp down, as shown in Figure 10. A constant acceleration a_{accel} is used for the ramp up, and a constant deceleration a_{decel} is used for the ramp down.

The first step is to compute an estimate of the travel distance D. When the reference path is defined by an ordered list of 2D points, represented by $\boldsymbol{p}_{\text{ref}_i} \in \mathbb{R}^2$, $i = 0, \ldots, n$, and the look-ahead point $\boldsymbol{p}_{L_1} \in \mathbb{R}^2$ is on the ith segment, this estimate D is given by

$$D = L_1 + \|\boldsymbol{p}_{L_1} - \boldsymbol{p}_{\text{ref}_i}\| + \sum_{j=i}^{n-1} \|\boldsymbol{p}_{\text{ref}_j} - \boldsymbol{p}_{\text{ref}_{j+1}}\| - L_{1\min}$$

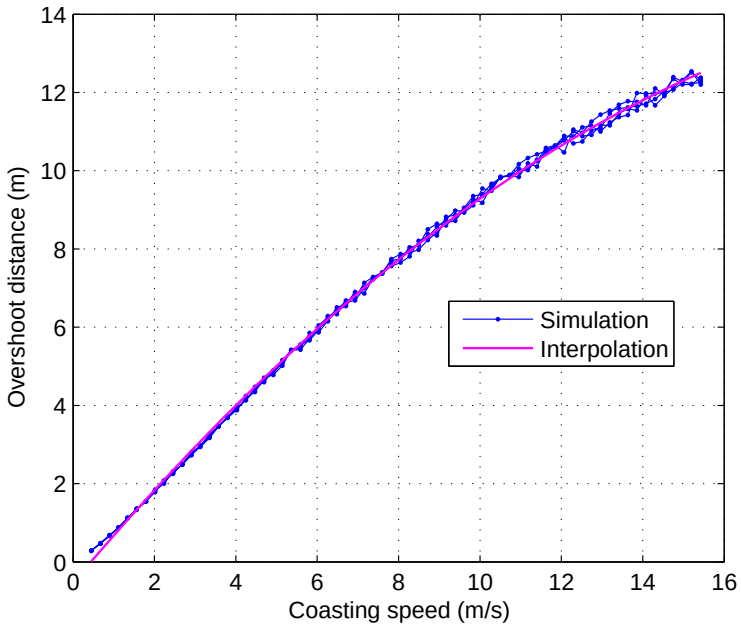


Figure 11. Overshooting distance as a function of speed.

where the first term is the distance from the anchor point to the look-ahead point on the reference path, the second and the third terms combined represent the distance from the look-ahead point to the end of the reference path, and the last term is the distance from the anchor point to the end of the reference path when the car stops exactly with L_1 point at the end of the reference path.

The second step is to compute the coasting speed v_{coast}. The coasting speed is a function of the initial speed v_0, the speed limit v_{max}, distance to go D, and the ramp acceleration $a_{\text{accel}}, a_{\text{decel}}$. To achieve a smooth behavior, the coasting time has a minimum duration t_{min}. If coasting at the maximum speed v_{max} ensures enough coasting time, i.e.,

$$\frac{{v_{\text{max}}}^2 - {v_0}^2}{2a_0} + v_{\text{max}} t_{\text{min}} + f(v_{\text{max}}) < D$$

where the function $f(v)$ gives a braking distance from a speed v, and a_0 is the initial ramp acceleration or deceleration, then, v_{max} is used as the coasting speed. Otherwise, the coasting speed is set such that there is t_{min} second of coasting, i.e.,

$$\frac{{v_{\text{coast}}}^2 - {v_0}^2}{2a_{\text{accel}}} + v_{\text{coast}} t_{\text{min}} + f(v_{\text{coast}}) = D.$$

The low-bandwidth speed controller does not precisely follow the ramp command. The tracking error during the ramp down leads to an overshoot in the stopping location. In order to account for this tracking error, the speed designer computes offline the overshooting distance as a function of the coasting speed. Figure 11 shows the overshooting distance as a function of the coasting speed. The plot is obtained by running three simulations for each speed with different coasting lengths. The variance comes from the quantization error with the discrete time simulation that runs at 25 Hz. Because it fits with a quadratic curve, the function $f(v)$ has three parameters that characterize the quadratic curve.

$$f(v) = \frac{v^2}{2a_{\text{decel}}} + \alpha_2 v^2 + \alpha_1 v + \alpha_0$$

V. Results

A. Vehicle Model

The following nonlinear bicycle model is used in the prediction.

$$\dot{x} = v\cos\theta \tag{14a}$$
$$\dot{y} = v\sin\theta \tag{14b}$$
$$\dot{\theta} = \frac{v}{L}\tan\delta \cdot G_{\text{ss}} \tag{14c}$$
$$\dot{\delta} = \frac{1}{T_d}(\delta_c - \delta) \tag{14d}$$
$$\dot{v} = a \tag{14e}$$
$$\dot{a} = \frac{1}{T_a}(a_c - a) \tag{14f}$$
$$a_{\min} \leq a \leq a_{\max} \tag{14g}$$
$$\|\delta\| \leq \delta_{\max} \tag{14h}$$
$$\|\dot{\delta}\| \leq \dot{\delta}_{\max} \tag{14i}$$

The position x and y are defined at the center of the rear axle, v and a represents the speed and acceleration also at the rear axle, θ is the car heading, δ is the steering angle, and L is the wheelbase. The inputs to this system are the steering angle command δ_c and the longitudinal acceleration command a_c. These inputs go through a first-order lag with time constants T_d and T_a for the steering and the acceleration respectively. The maximum steering angle is given by $\delta_{\max}$. Because of the actuator, the steering rate is also limited, and the maximum slew rate is given by $\dot{\delta}_{\max}$. The vehicle has a maximum deceleration $a_{\min}$ and maximum acceleration $a_{\max}$.

The term G_{ss} captures the effect of the side slip

$$G_{\text{ss}} = \frac{1}{1 + (v/v_{\text{CH}})^2}$$

and is a static gain of the yaw rate $\dot{\theta}$, i.e. the resulting yaw rate when the derivative of the side-slip angle and the derivative of yaw rate are both zero.[13] The parameter v_{CH} is called characteristic velocity[13,18] and can be determined experimentally. This side slip model has several advantages: it does not increase the model order, so it does not slow down the propagation: the model behaves the same as the kinematic model in the low speeds; and it requires only one tunable parameter. Moreover, this model does not differ much from the nonlinear single track model[13] for the urban driving conditions where extreme maneuvers are avoided and up to 35 mph speeds are considered.

B. Vehicle Hardware

Figure 12 shows the race vehicle developed by Team MIT for DARPA Urban Challenge. This robotic car is named Talos and all the results shown in this paper are obtained by running Talos. Although Talos has a blade cluster consisting of 40 cores (2.33 GHz Intel Xenon processor), the planner uses two cores and the controller uses one core. From the initial testing, the following values for the vehicle parameters were obtained.

$$L = 2.885 \text{ [m]} \qquad v_{\text{CH}} = 20.0 \text{ [m/s]}$$
$$\delta_{\text{max}} = 0.5435 \text{ [rad]} \qquad T_a = 0.3 \text{ [sec]}$$
$$\dot{\delta}_{\text{max}} = 0.3294 \text{ [rad/s]} \qquad a_{\min} = -6.0 \text{ [m/s}^2\text{]}$$
$$T_d = 0.05 \text{ [sec]} \qquad a_{\max} = 1.8 \text{ [m/s}^2\text{]}$$

Figure 12. Talos: Team MIT's race vehicle at the DARPA Urban Challenge

The parameters used to design the speed command are

$$a_{\text{accel}} = 1.0 \text{ [m/s}^2\text{]}$$
$$a_{\text{decel}} = 2.5 \text{ [m/s}^2\text{]}.$$

With the nominal deceleration of 2.5 m/s^2, the overshooting coefficients obtained from these parameters are as follow.

$$\alpha_0 = -0.5347, \quad \alpha_1 = 1.2344, \quad \alpha_2 = -0.0252$$

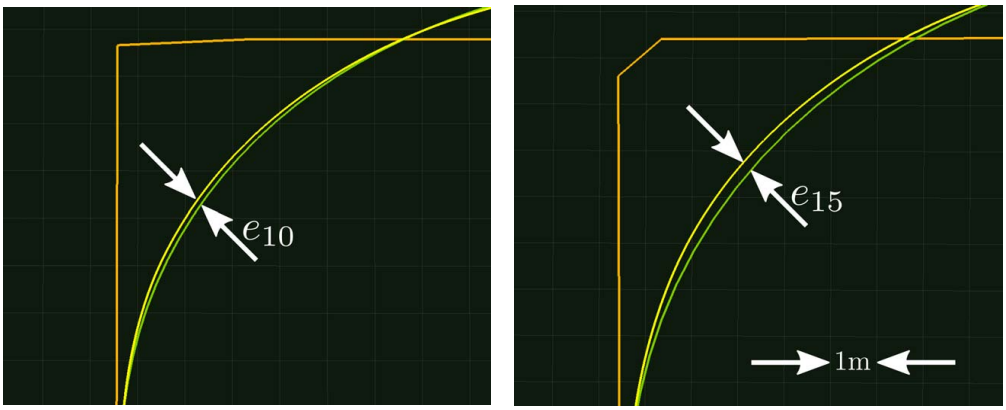
C. Prediction Results

Figure 13 shows the comparison of two trajectories, one obtained by the prediction and the other obtained by running Talos. The orange line shows the input to the controller, the green line shows the predicted path, and the yellow line shows the actual path that Talos took. The maximum error was $e_{10} = 0.114$ m for the 10 mph run, and $e_{15} = 0.240$ m for the 15 mph run. Note that the input to the vehicle changes at 25 Hz, but the input to the closed-loop system has only a few straight line segments. This demonstrates the efficiency of the RRT with closed-loop prediction.

Figure 14 shows the effect of the side slip. In this experiment, the speed was set to 30 mph. The left figure is without accounting for the side slip (i.e., $G_{\text{ss}} = 1$), and the right figure is with the side slip model. Even though the tuned set of parameters gives good prediction results in the low speed, without accounting for the side slip, the lateral prediction error can be as large as 1 m. Our simple side slip model decreased the lateral error by more than half, as seen with the white arrow in Figure 14.

D. DARPA Urban Challenge Race

This section shows several results obtained during the semi-final event that took place from October 26th to 31st and the final event that took place on November 3rd. The total traveling distance during the final race was 57.58 miles, and Talos completed all missions in 5 hours 35 minutes, which was under the 6 hour time limit. This section shows the algorithmic features demonstrated during the race.



(a) Right turn at 10 mph (b) Right turn at 15 mph

Figure 13. Comparison of the predicted and the actual path at 10 and 15 mph

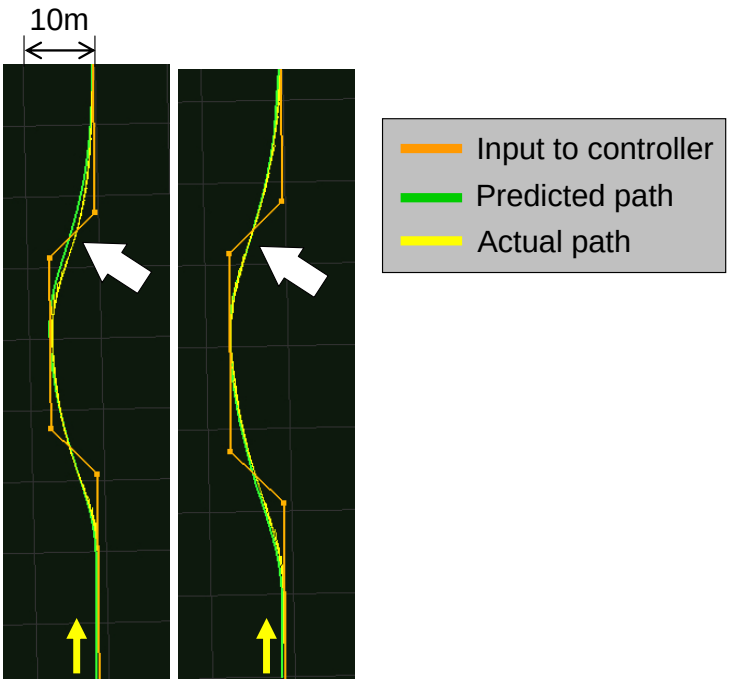


Figure 14. Comparison of the predicted and the actual path at 30 mph. Left: prediction without a side slip model. Right: prediction with the side slip model..

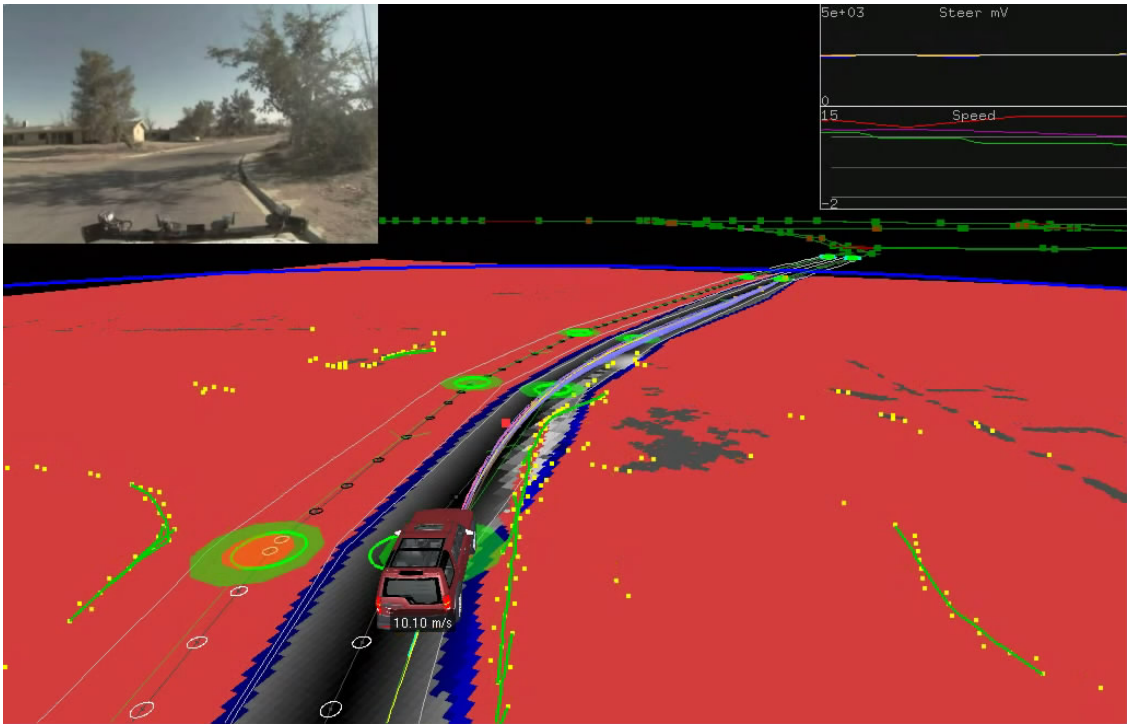


Figure 15. High speed curvy section

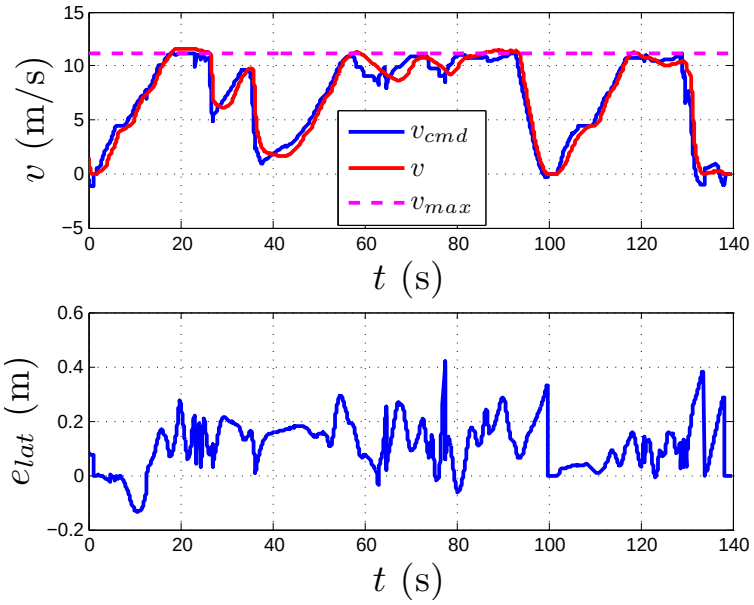


Figure 16. Prediction error during this segment

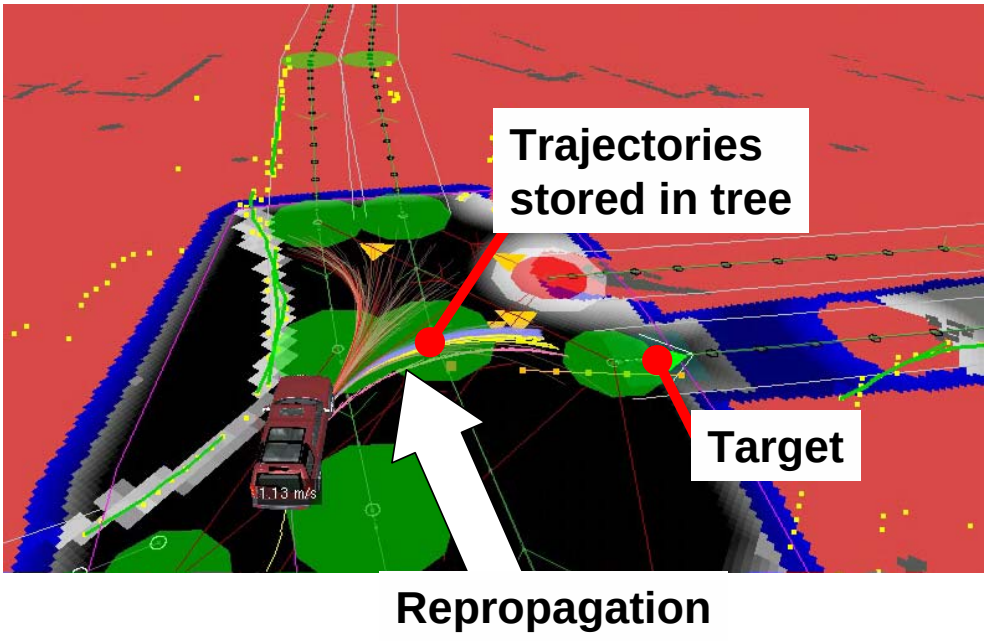


Figure 17. Repropagation after the evasive maneuver. The pink line with the white arrow shows the trajectory repropagated from the latest states using the controller input stored in the tree.

1. High speed behavior on a curvy road

Figure 15 shows a snapshot of the environment and the plan during the race. The red region represents the non-drivable part of the road, the green lines on the right side of the vehicle show the detected curb cuts, the green circles are the GPS points given by DARPA, and the purple lines represent the propagated trajectories in the tree. Note that even though the controller inputs are randomly generated to build the tree, the resulting trajectories naturally follow the curvy road. This section is about 0.5 mile long, and the speed limit specified by DARPA was 25 mph. Talos reached the maximum speed 25 mph on the straight segment of this road.

Figure 16 shows the speed profile and the lateral prediction error for this segment. The prediction versus execution error has the mean, maximum, and standard deviation of 0.11 m, 0.42 m, and 0.093 m respectively. Note from the plot that the prediction error has a constant offset of about 11 cm, making the maximum error much larger than the standard deviation. This is due to the fact that a) the steering wheel was not perfectly centered; and b) the pure-pursuit algorithm does not have any integral action to remove the steady state error.

Note that when the prediction error happens to become large, the planner/controller does not explicitly minimize it. This is because the vehicle keeps executing the same plan as long as the repropagated trajectory is feasible. In such a case, the prediction error could grow momentarily. For example, during a turn with a maximum steering angle, a small difference between the predicted initial heading and the actual heading can lead to a relatively large error as the vehicle turns. Even with a large mismatch, however, the repropagation always ensures the safety of the future path from vehicle's actual states.

2. Repropagation

Most of the time during the race, the trajectory repropagated from the latest states and the trajectory stored in the tree are close. However, there are some instances when the repropagation resulted in a different trajectory. Figure 17 shows the snapshot right after Talos took an evasive maneuver to avoid a moving car by steering left and applying strong brake. Talos slowed down more than the prediction, so the trajectory repropagated from the latest states (shown with an arrow in the figure) is a little off from the trajectory stored in the tree. However, as long as the repropagation

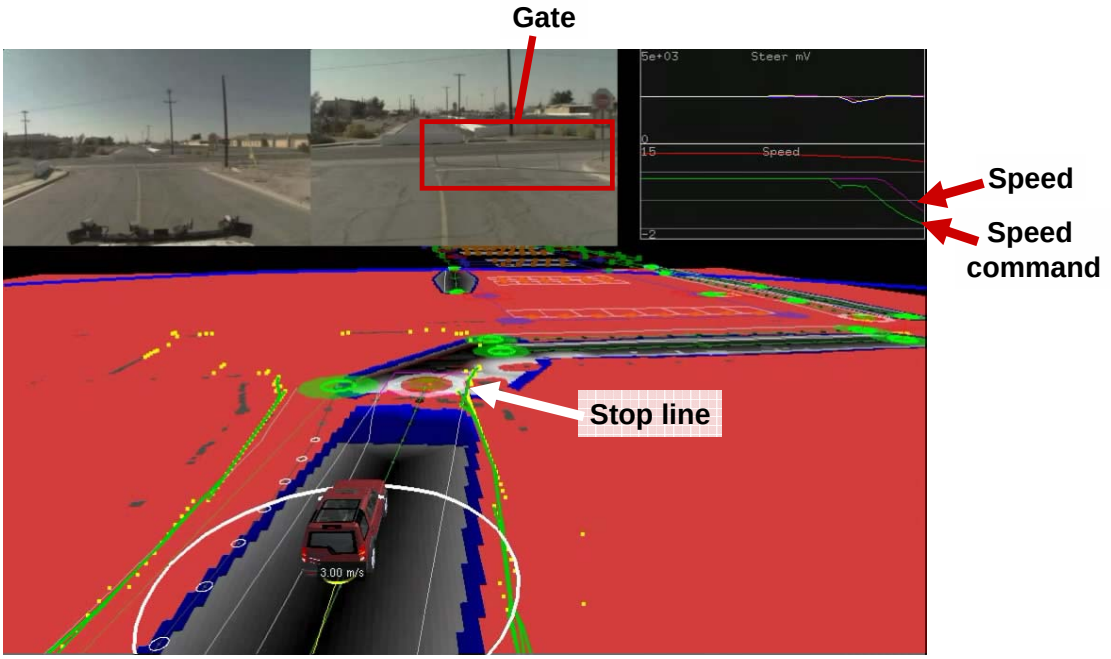


Figure 18. Emergency braking at a gate.

if feasible, Talos keeps executing the same controller input. This approach is much more efficient than discarding the tree when the prediction error exceeds an artificial limit and rebuilding the tree from scratch.

3. Hard brake with the gate

Figure 18 shows an emergency braking behavior seen during the race. The wind unexpectedly closed the gate near the stop line. This gate is made of relatively thin metal pipes and was detected by the perception system after the car started decelerating to stop at a stop line. When obstacles or vehicles block the road, Talos is designed to stop 10 m behind them to allow for some room for passing them later. The newly detected gate required Talos to stop about 10 m short of the stop line, requiring much stronger deceleration. The planner started commanding an emergency braking, which was 4.0 m/s^2 deceleration. The green line in the upper left shows the speed command, and the purple line above the green line shows the speed profile. The low-bandwidth controller has some tracking error, as expected, but the car stopped safely with enough clearance to the gate.

VI. Conclusion

This paper presented a motion planning algorithm that plans over the closed-loop dynamics. The approach is an extension of RRT that randomly generates an input to the controller rather than to the vehicle. By running forward simulation, dynamically feasible paths are generated, to be checked with environmental constraints such as obstacle avoidance. The paper also gave detailed description of the planner-controller interface and techniques to reduce prediction errors. Several results on the race vehicle of the DARPA Urban Challenge 2007 show various safe behaviors that are generated online.

Acknowledgment

Research was sponsored by Defense Advanced Research Projects Agency, Program: Urban Challenge, DARPA Order No. W369/00, Program Code: DIRO. Issued by DARPA/CMO under Contract No. HR0011-06-C-0149, with J. Leonard, S. Teller, J. How at MIT and D. Barrett at Olin College as the PI's. The authors are grateful to Karl Iagnemma for his expert advice, Stefan Campbell for his initial design and implementation of the controller, and Steven Peters for his technical support during the development of Team MIT's vehicle.

References

[1]Latombe, J. C., *Robot Motion Planning*, Kluwer Academic, 1991.

[2]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., 2006, Available at http://planning.cs.uiuc.edu/.

[3]Philippsen, R., Kolski, S., Macek, K., and Siegwart, R., "Path Planning, Replanning, and Execution for Autonomous Driving in Urban and Offroad Environments," *In Proceedings of the Workshop on Planning, Perception and Navigation for Intelligent Vehicles, ICRA*, Rome, Italy, 2007.

[4]Urmson, C., Ragusa, C., Ray, D., Anhalt, J., Bartz, D., Galatali, T., Gutierrez, A., Johnston, J., Harbaugh, S., ldquoYurdquo Kato, H., Messner, W., Miller, N., Peterson, K., Smith, B., Snider, J., Spiker, S., Ziglar, J., ldquoRedrdquo Whittaker, W., Clark, M., Koon, P., Mosher, A., and Struble, J., "A robust approach to high-speed navigation for unrehearsed desert terrain," *Journal of Field Robotics*, Vol. 23, No. 8, 2006, pp. 467–508.

[5]Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P., "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, Vol. 23, No. 9, 2006, pp. 661–692.

[6]Mason, R., Radford, J., Kumar, D., Walters, R., Fulkerson, B., Jones, E., Caldwell, D., Meltzer, J., Alon, Y., Shashua, A., Hattori, H., Frazzoli, E., and Soatto, S., "The Golem Group/University of California at Los Angeles autonomous ground vehicle in the DARPA grand challenge," *Journal of Field Robotics*, Vol. 23, No. 8, 2006, pp. 527–553.

[7]Cremean, L. B., Foote, T. B., Gillula, J. H., Hines, G. H., Kogan, D., Kriechbaum, K. L., Lamb, J. C., Leibs, J., Lindzey, L., Rasmussen, C. E., Stewart, A. D., Burdick, J. W., and Murray, R. M., "Alice: An information-rich autonomous vehicle for high-speed desert navigation," *Journal of Field Robotics*, Vol. 23, No. 9, 2001, pp. 777–810.

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

[9]LaValle, S. M., "Rapidly-exploring random trees: A new tool for path planning," Tech. rep., Computer Science Department, Iowa State University, 1998, TR 98-11.

[10]Amidi, O. and Thorpe, C., "Integrated Mobile Robot Control," *Proceedings of SPIE*, edited by W. H. Chun and W. J. Wolfe, Vol. 1388, SPIE, Boston, MA, Mar 1991, pp. 504–523.

[11]Park, S., Deyst, J., and How, J. P., "Performance and Lyapunov Stability of a Nonlinear Path-Following Guidance Method," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 6, November 2007, pp. 1718–1728.

[12]Ollero, A. and Heredia, G., "Stability Analysis of Mobile Robot Path Tracking," *International Conference on Intelligent Robots and Systems*, Vol. 3, Piscataway, NJ, 1995, pp. 461–466.

[13]Ackermann, J., *Robust Control: The Parameter Space Approach*, Communications and Control Engineering, Springer, 2nd ed., 2002.

[14]Krajewski, W., Lepschy, A., and Viaro, U., "Designing PI Controllers for Robust Stability and Performance," *IEEE Transactions on Control Systems Technology*, Vol. 12, No. 6, November 2004, pp. 973–983.

[15]Zhou, K. and Doyle, J., *Essentials of Robust Control*, Prentice Hall, 1998.

[16]Güvenç, L. and Ackermann, J., "Links between the Parameter Space and Frequency Domain Methods of Robust Control," *International Journal of Robust and Nonlinear Control*, Vol. 11, No. 15, December 2001, pp. 1435–1453.

[17]Doyle, J. C., Francis, B. A., and Tannenbaum, A. R., *Feedback Control Theory*, Macmillan, 1992.

[18]Gillespie, T., *Fundamentals of Vehicle Dynamics*, Society of Automotive Engineers, 1992.