This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles

Brian Paden,[*,1] Michal Čáp,[*,1,2] Sze Zheng Yong,[1] Dmitry Yershov,[1] and Emilio Frazzoli[1]

***Abstract*—Self-driving vehicles are a maturing technology with the potential to reshape mobility by enhancing the safety, accessibility, efficiency, and convenience of automotive transportation. Safety-critical tasks that must be executed by a self-driving vehicle include planning of motions through a dynamic environment shared with other vehicles and pedestrians, and their robust executions via feedback control. The objective of this paper is to survey the current state of the art on planning and control algorithms with particular regard to the urban setting. A selection of proposed techniques is reviewed along with a discussion of their effectiveness. The surveyed approaches differ in the vehicle mobility model used, in assumptions on the structure of the environment, and in computational requirements. The side by side comparison presented in this survey helps to gain insight into the strengths and limitations of the reviewed approaches and assists with system level design choices.**

I. Introduction

The last three decades have seen steadily increasing research efforts, both in academia and in industry, towards developing driverless vehicle technology. These developments have been fueled by recent advances in sensing and computing technology together with the potential transformative impact on automotive transportation and the perceived societal benefit: In 2014 there were 32,675 traffic related fatalities, 2.3 million injuries, and 6.1 million reported collisions [1]. Of these, an estimated 94% are attributed to driver error with 31% involving legally intoxicated drivers, and 10% from distracted drivers [2]. Autonomous vehicles have the potential to dramatically reduce the contribution of driver error and negligence as the cause of vehicle collisions. They will also provide a means of personal mobility to people who are unable to drive due to physical or visual disability. Finally, for the 86% of the US work force that commutes by car, on average 25 minutes (one way) each day [3], autonomous vehicles would facilitate more productive use of the transit time, or simply reduce the measurable ill effects of driving stress [4].

Considering the potential impacts of this new technology, it is not surprising that self-driving cars have a long history. The idea has been around since as early as the 1920s, but it was not until the 1980s that driverless cars seemed like a real possibility. Pioneering work led by Ernst Dickmanns (e.g., [5]) in the 1980s paved the way for the development of autonomous vehicles. At that time a massive research effort, the PROMETHEUS project, was funded to develop an autonomous vehicle. A notable demonstration in 1994 resulting from the work was a 1,600 km drive by the VaMP driverless car, of which 95% was driven autonomously [6]. At a similar time, the CMU NAVLAB was making advances in the area and in 1995 demonstrated further progress with a 5,000 km drive across the US of which 98% was driven autonomously [7].

The next major milestone in driverless vehicle technology was the first DARPA Grand Challenge in 2004. The objective was for a driverless car to navigate a 150-mile off-road course as quickly as possible. This was a major challenge in comparison to previous demonstrations in that there was to be no human intervention during the race. Although prior works demonstrated nearly autonomous driving, eliminating human intervention at critical moments proved to be a major challenge. None of the 15 vehicles entered into the event completed the race. In 2005 a similar event was held; this time 5 of 23 teams reached the finish line [8]. Later, in 2007, the DARPA Urban Challenge was held, in which vehicles were required to drive autonomously in a simulated urban setting. Six teams finished the event demonstrating that fully autonomous urban driving is possible [9].

Numerous events and major autonomous vehicle system tests have been carried out since the DARPA challenges. Notable examples include the Intelligent Vehicle Future Challenges from 2009 to 2013 [10], Hyundai Autonomous Challenge in 2010 [11], the VisLab Intercontinental Autonomous Challenge in 2010 [12], the Public Road Urban Driverless-Car Test in 2013 [13], and the autonomous drive of the Bertha-Benz historic route [14]. Simultaneously, research has continued at an accelerated pace in both the academic setting as well as in industry. The Google self-driving car [15] and Tesla's Autopilot system [16] are two example commercial efforts that receive considerable media attention.

The extent to which a car is automated can vary from fully human operated to fully autonomous. The SAE J3016 standard [17] introduces a scale from 0 to 5 for grading vehicle automation. In this standard, the level 0 represents a vehicle where all driving tasks are the responsibility of a human driver. Level 1 includes basic driving assistance such as adaptive cruise control, anti-lock braking systems and electronic stability control [18]. Level 2 includes advanced assistance such as hazard-minimizing longitudinal/lateral control [19] or emergency braking [20], [21], often based upon set-based formal control theoretic methods to compute 'worst-case' sets of provably collision free (safe) states [22]–[24]. At level 3

[*] The first two authors contributed equally to this work.

[1] The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge MA, USA. email: bapaden@mit.edu, mcap@mit.edu, szyong@mit.edu, yershov@mit.edu, frazzoli@mit.edu

[2] Michal Čáp is also affiliated with Dept. of Computer Science, Faculty of Electrical Engineering, CTU in Prague, Czech Republic.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

the system monitors the environment and can drive with full autonomy under certain conditions, but the human operator is still required to take control if the driving task leaves the autonomous system's operational envelope. A vehicle with level 4 automation is capable of fully autonomous driving in certain conditions and will safely control the vehicle if the operator fails to take control upon request to intervene. Level 5 systems are fully autonomous in all driving modes.

The availability of on-board computation and wireless communication technology allows cars to exchange information with other cars and with the road infrastructure giving rise to a closely related area of research on connected intelligent vehicles [25]. This research aims to improve the safety and performance of road transport through information sharing and coordination between individual vehicles. For instance, connected vehicle technology has a potential to improve throughput at intersections [26] or prevent formation of traffic shock waves [27].

To limit the scope of this survey, we focus on aspects of decision making, motion planning, and control for self-driving cars, in particular, for systems falling into the automation level of 3 and above. For the same reason, the broad field of perception for autonomous driving is omitted and instead the reader is referred to a number of comprehensive surveys and major recent contributions on the subject [28]–[31].

The decision making in contemporary autonomous driving systems is typically hierarchically structured into route planning, behavioral decision making, local motion planning and feedback control. The partitioning of these levels are, however, rather blurred with different variations of this scheme occurring in the literature. This paper provides a survey of proposed methods to address these core problems of autonomous driving. Particular emphasis is placed on methods for local motion planning and control.

The remainder of the paper is structured as follows: In Section II, a high level overview of the hierarchy of decision making processes and some of the methods for their design are presented. Section III reviews models used to approximate the mobility of cars in urban settings for the purposes of motion planning and feedback control. Section IV surveys the rich literature on motion planning and discusses its applicability for self-driving cars. Similarly, Section V discusses the problems of path and trajectory stabilization and specific feedback control methods for driverless cars. Lastly, Section VI concludes with remarks on the state of the art and potential areas for future research.

II. Overview of the Decision-Making Hierarchy used in Driverless Cars

In this section we describe the decision making architecture of a typical self-driving car and comment on the responsibilities of each component. Driverless cars are essentially autonomous decision-making systems that process a stream of observations from on-board sensors such as radars, LIDARs, cameras, GPS/INS units, and odometry. These observations, together with prior knowledge about the road network, rules of the road, vehicle dynamics, and sensor models, are used to automatically select values for controlled variables governing the vehicle's motion. Intelligent vehicle research aims at automating as much of the driving task as possible. The commonly adopted approach to this problem is to partition and organize perception and decision-making tasks into a hierarchical structure. The prior information and collected observation data are used by the perception system to provide an estimate of the state of the vehicle and its surrounding environment; the estimates are then used by the decision-making system to control the vehicle so that the driving objectives are accomplished.

The decision making system of a typical self-driving car is hierarchically decomposed into four components (cf. Figure II.1): At the highest level a route is planned through the road network. This is followed by a behavioral layer, which decides on a local driving task that progresses the car towards the destination and abides by rules of the road. A motion planning module then selects a continuous path through the environment to accomplish a local navigational task. A control system then reactively corrects errors in the execution of the planned motion. In the remainder of the section we discuss the responsibilities of each of these components in more detail.

A. Route Planning

At the highest level, a vehicle's decision-making system must select a route through the road network from its current position to the requested destination. By representing the road network as a directed graph with edge weights corresponding to the cost of traversing a road segment, such a route can be formulated as the problem of finding a minimum-cost path on a road network graph. The graphs representing road networks can however contain millions of edges making classical shortest path algorithms such as Dijkstra [32] or A* [33] impractical. The problem of efficient route planning in transportation networks has attracted significant interest in the transportation science community leading to the invention of a family of algorithms that after a one-time pre-processing step return an optimal route on a continent-scale network in milliseconds [34], [35]. For a comprehensive survey and comparison of practical algorithms that can be used to efficiently plan routes for both human-driven and self-driving vehicles, see [36].

B. Behavioral Decision Making

After a route plan has been found, the autonomous vehicle must be able to navigate the selected route and interact with other traffic participants according to driving conventions and rules of the road. Given a sequence of road segments specifying the selected route, the behavioral layer is responsible for selecting an appropriate driving behavior at any point of time based on the perceived behavior of other traffic participants, road conditions, and signals from infrastructure. For example, when the vehicle is reaching the stop line before an intersection, the behavioral layer will command the vehicle to come to a stop, observe the behavior of other vehicles, bikes, and pedestrians at the intersection, and let the vehicle proceed once it is its turn to go.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

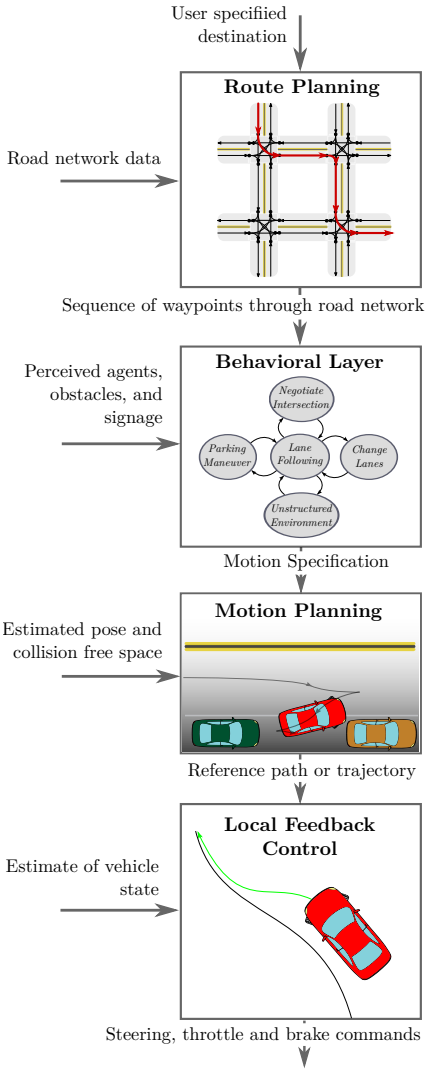


Figure II.1: Illustration of the hierarchy of decision-making processes. A destination is passed to a route planner that generates a route through the road network. A behavioral layer reasons about the environment and generates a motion specification to progress along the selected route. A motion planner then solves for a feasible motion accomplishing the specification. A feedback control adjusts actuation variables to correct errors in executing the reference path.

Driving manuals dictate qualitative actions for specific driving contexts. Since both driving contexts and the behaviors available in each context can be modeled as finite sets, a natural approach to automating this decision making is to model each behavior as a state in a finite state machine with transitions governed by the perceived driving context such as relative position with respect to the planned route and nearby vehicles. In fact, finite state machines coupled with different heuristics specific to considered driving scenarios were adopted as a mechanism for behavior control by most teams in the DARPA Urban Challenge [9].

Real-world driving, especially in an urban setting, is however characterized by uncertainty over the intentions of other traffic participants. The problem of intention prediction and estimation of future trajectories of other vehicles, bikes and pedestrians has also been studied. Among the proposed solution techniques are machine learning based techniques, e.g., Gaussian mixture models [37], Gaussian process regression [38] and the learning techniques reportedly used in Google's self-driving system for intention prediction [39], as well as model-based approaches for directly estimating intentions from sensor measurements [40], [41].

This uncertainty in the behavior of other traffic participants is commonly considered in the behavioral layer for decision making using probabilistic planning formalisms, such as Markov Decision Processes (MDPs) and generalizations. For example, [42] formulates the behavioral decision-making problem in MDP framework. Several works [43]–[46] model unobserved different driving scenarios and pedestrian intentions explicitly using a partially-observable Markov decision process (POMDP) framework and propose specific approximate solution strategies.

C. Motion Planning

When the behavioral layer decides on the driving behavior to be performed in the current context, which could be, e.g., cruise-in-lane, change-lane, or turn-right, the selected behavior has to be translated into a path or trajectory that can be tracked by the low-level feedback controller. The resulting path or trajectory must be dynamically feasible for the vehicle, comfortable for the passenger, and avoid collisions with obstacles detected by the on-board sensors. The task of finding such a path or trajectory is a responsibility of the motion planning system, which is discussed in greater detail in Section IV.

D. Vehicle Control

In order to execute the reference path or trajectory from the motion planning system a feedback controller is used to select appropriate actuator inputs to carry out the planned motion and correct tracking errors. The tracking errors generated during the execution of a planned motion are due in part to the inaccuracies of the vehicle model. Thus, a great deal of emphasis is placed on the robustness and stability of the closed loop system.

Many effective feedback controllers have been proposed for executing the reference motions provided by the motion planning system. A survey of related techniques are discussed in detail in Section V.

III. Modeling for Planning and Control

In this section we will survey the most commonly used models of mobility of car-like vehicles. Such models are widely used in control and motion planning algorithms to approximate a vehicle's behavior in response to control actions in relevant operating conditions. A high-fidelity model may accurately reflect the response of the vehicle, but the added

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

detail may complicate the planning and control problems. This presents a trade-off between the accuracy of the selected model and the difficulty of the decision problems. This section provides an overview of general modeling concepts and a survey of models used for motion planning and control.

Modeling begins with the notion of the vehicle *configuration*, representing its pose or position in the world. For example, configuration can be expressed as the planar coordinate of a point on the car together with the car's heading. This is a *coordinate system* for the configuration space of the car. This coordinate system describes planar rigid-body motions (represented by the Special Euclidean group in two dimensions, SE(2)) and is a commonly used configuration space [47]–[49]. Vehicle motion must then be planned and regulated to accomplish driving tasks and while respecting the constraints introduced by the selected model.

A. The Kinematic Single-Track Model

In the most basic model of practical use, the car consists of two wheels connected by a rigid link and is restricted to move in a plane [48]–[52]. It is assumed that the wheels do not slip at their contact point with the ground, but can rotate freely about their axes of rotation. The front wheel has an added degree of freedom where it is allowed to rotate about an axis normal to the plane of motion. This is to model steering. These two modeling features reflect the experience most passengers have where the car is unable to make lateral displacement without simultaneously moving forward. More formally, the limitation on maneuverability is referred to as a *nonholonomic* constraint [47], [53]. The nonholonomic constraint is expressed as a differential constraint on the motion of the car. This expression varies depending on the choice of coordinate system. Variations of this model have been referred to as the car-like robot, bicycle model, kinematic model, or single track model.

The following is a derivation of the differential constraint in several popular coordinate systems for the configuration. In reference to Figure III.1, the vectors p_r and p_f denote the location of the rear and front wheels in a stationary or inertial coordinate system with basis vectors $(\hat{e}_x, \hat{e}_y, \hat{e}_z)$. The heading θ is an angle describing the direction that the vehicle is facing. This is defined as the angle between vectors $\hat{e}_x$ and $p_f - p_r$.

Differential constraints will be derived for the coordinate systems consisting of the angle θ, together with the motion of one of the points p_r as in [54], and p_f as in [55].

The motion of the points p_r and p_f must be collinear with the wheel orientation to satisfy the no-slip assumption. Expressed as an equation, this constraint on the rear wheel is

$$\left(\dot{p}_r \cdot \hat{e}_y\right)\cos(\theta) - \left(\dot{p}_r \cdot \hat{e}_x\right)\sin(\theta) = 0, \tag{III.1}$$

and for the front wheel:

$$\left(\dot{p}_f \cdot \hat{e}_y\right)\cos(\theta+\delta) - \left(\dot{p}_f \cdot \hat{e}_x\right)\sin(\theta+\delta) = 0. \tag{III.2}$$

This expression is usually rewritten in terms of the component-wise motion of each point along the basis vectors. The motion of the rear wheel along the $\hat{e}_x$-direction is $x_r := p_r \cdot \hat{e}_x$. Similarly, for $\hat{e}_y$-direction, $y_r := p_r \cdot \hat{e}_y$. The forward speed

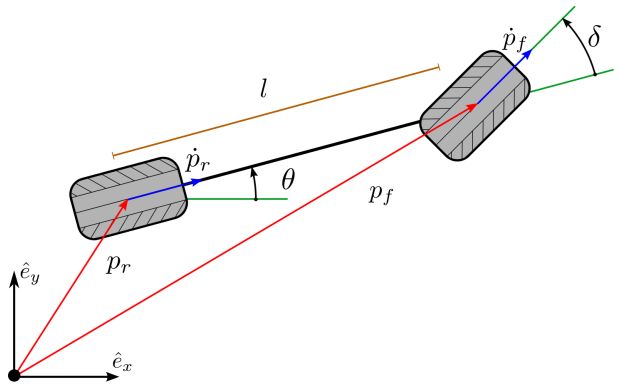


Figure III.1: Kinematics of the single track model. p_r and p_f are the ground contact points of the rear and front tire respectively. θ is the vehicle heading. Time derivatives of p_r and p_f are restricted by the nonholonomic constraint to the direction indicated by the blue arrows. δ is the steering angle of the front wheel.

is $v_r := \dot{p}_r \cdot (p_f - p_r)/\|(p_f - p_r)\|$, which is the magnitude of $\dot{p}_r$ with the correct sign to indicate forward or reverse driving. In terms of the scalar quantities x_r, y_r, and θ, the differential constraint is

$$\begin{aligned} \dot{x}_r &= v_r\cos(\theta), \\ \dot{y}_r &= v_r\sin(\theta), \\ \dot{\theta} &= \tfrac{v_r}{l}\tan(\delta). \end{aligned} \tag{III.3}$$

Alternatively, the differential constraint can be written in terms the motion of p_f,

$$\begin{aligned} \dot{x}_f &= v_f\cos(\theta+\delta), \\ \dot{y}_f &= v_f\sin(\theta+\delta), \\ \dot{\theta} &= \tfrac{v_f}{l}\sin(\delta), \end{aligned} \tag{III.4}$$

where the front wheel forward speed v_f is now used. The front wheel speed, v_f, is related to the rear wheel speed by

$$\frac{v_r}{v_f} = \cos(\delta). \tag{III.5}$$

The planning and control problems for this model involve selecting the steering angle δ within the mechanical limits of the vehicle $\delta \in [\delta_{min}, \delta_{max}]$, and forward speed v_r within an acceptable range, $v_r \in [v_{min}, v_{max}]$.

A simplification that is sometimes utilized, e.g. [56], is to select the heading rate ω instead of steering angle δ. These quantities are related by

$$\delta = \arctan\left(\frac{l\omega}{v_r}\right), \tag{III.6}$$

simplifying the heading dynamics to

$$\dot{\theta} = \omega, \quad \omega \in \left[\frac{v_r}{l}\tan\left(\delta_{min}\right), \frac{v_r}{l}\tan\left(\delta_{max}\right)\right]. \tag{III.7}$$

In this situation, the model is sometimes referred to as the unicycle model since it can be derived by considering the motion of a single wheel.

An important variation of this model is the case when v_r is fixed. This is sometimes referred to as the Dubins car, after Lester Dubins who derived the minimum time motion between to points with prescribed tangents [57]. Another

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

notable variation is the Reeds-Shepp car for which minimum length paths are known when v_r takes a single forward and reverse speed [58]. These two models have proven to be of some importance to motion planning and will be discussed further in Section IV.

The kinematic models are suitable for planning paths at low speeds (e.g. parking maneuvers and urban driving) where inertial effects are small in comparison to the limitations on mobility imposed by the no-slip assumption. A major drawback of this model is that it permits instantaneous steering angle changes which can be problematic if the motion planning module generates solutions with such instantaneous changes.

Continuity of the steering angle can be imposed by augmenting (III.4), where the steering angle integrates a commanded rate as in [49]. Equation (III.4) becomes

$$\begin{aligned} \dot{x}_f &= v_f \cos(\theta + \delta), \\ \dot{y}_f &= v_f \sin(\theta + \delta), \\ \dot{\theta} &= \tfrac{v_f}{l} \sin(\delta), \\ \dot{\delta} &= v_\delta. \end{aligned} \tag{III.8}$$

In addition to the limit on the steering angle, the steering rate can now be limited: $v_\delta \in \left[\dot{\delta}_{min}, \dot{\delta}_{max}\right]$. The same problem can arise with the car's speed v_r and can be resolved in the same way. The drawback to this technique is the increased dimension of the model which can complicate motion planning and control problems.

While the kinematic bicycle model and simple variations are very useful for motion planning and control, models considering wheel slip [59], inertia [18], [60]–[62], and chassis dynamics [60] can better utilize the vehicle's capabilities for executing agile maneuvers. These effects become significant when planning motions with high acceleration and jerk.

IV. Motion Planning

The motion planning layer is responsible for computing a safe, comfortable, and dynamically feasible trajectory from the vehicle's current configuration to the goal configuration provided by the behavioral layer of the decision making hierarchy. Depending on context, the goal configuration may differ. For example, the goal location may be the center point of the current lane a number of meters ahead in the direction of travel, the center of the stop line at the next intersection, or the next desired parking spot. The motion planning component accepts information about static and dynamic obstacles around the vehicle and generates a collision-free trajectory that satisfies dynamic and kinematic constraints on the motion of the vehicle. Oftentimes, the motion planner also minimizes a given objective function. In addition to travel time, the objective function may penalize hazardous motions or motions that cause passenger discomfort. In a typical setup, the output of the motion planner is then passed to the local feedback control layer. In turn, feedback controllers generate an input signal to regulate the vehicle to follow this given motion plan.

A motion plan for the vehicle can take the form of a path or a trajectory. Within the path planning framework, the solution path is represented as a function $\sigma(\alpha) : [0, 1] \to \mathcal{X}$, where $\mathcal{X}$ is the configuration space of the vehicle. Note that such a solution does not prescribe how this path should be followed and one can either choose a velocity profile for the path or delegate this task to lower layers of the decision hierarchy. Within the trajectory planning framework, the control execution time is explicitly considered. This consideration allows for direct modeling of vehicle dynamics and dynamic obstacles. In this case, the solution trajectory is represented as a time-parametrized function $\pi(t) : [0, T] \to \mathcal{X}$, where T is the planning horizon. Unlike a path, the trajectory prescribes how the configuration of the vehicle evolves over time.

In the following two sections, we provide a formal problem definition of the path planning and trajectory planning problems and review the main complexity and algorithmic results for both formulations.

A. Path Planning

The path planning problem is to find a path $\sigma(\alpha) : [0, 1] \to \mathcal{X}$ in the configuration space $\mathcal{X}$ of the vehicle (or more generally, a robot) that starts at the initial configuration and reaches the goal region while satisfying given global and local constraints. Depending on whether the quality of the solution path is considered, the terms *feasible* and *optimal* are used to describe this path. Feasible path planning refers to the problem of determining a path that satisfies some given problem constraints without focusing on the quality of the solution; whereas optimal path planning refers to the problem of finding a path that optimizes some quality criterion subject to given constraints.

The optimal path planning problem can be formally stated as follows. Let $\mathcal{X}$ be the configuration space of the vehicle and let $\Sigma(\mathcal{X})$ denote the set of all continuous functions $[0, 1] \to \mathcal{X}$. The initial configuration of the vehicle is $\mathbf{x}_{\text{init}} \in \mathcal{X}$. The path is required to end in a goal region $\mathcal{X}_{\text{goal}} \subseteq \mathcal{X}$. The set of all allowed configurations of the vehicle is called the free configuration space and denoted $\mathcal{X}_{\text{free}}$. Typically, the free configurations are those that do not result in collision with obstacles, but the free-configuration set can also represent other holonomic constraints on the path. The differential constraints on the path are represented by a predicate $D(\mathbf{x}, \mathbf{x}', \mathbf{x}'', \ldots)$ and can be used to enforce some degree of smoothness of the path for the vehicle, such as the bound on the path curvature and/or the rate of curvature. For example, in the case of $\mathcal{X} \subseteq \mathbb{R}^2$, the differential constraint may enforce the maximum curvature κ of the path using Frenet-Serret formula as follows:

$$D(\mathbf{x}, \mathbf{x}', \mathbf{x}'', \ldots) \Leftrightarrow \frac{\|\mathbf{x}' \times \mathbf{x}''\|}{\|\mathbf{x}'\|^3} \leq \kappa.$$

Further, let $J(\sigma) : \Sigma(\mathcal{X}) \to \mathbb{R}$ be the cost functional. Then, the optimal version of the path planning problem can be generally stated as follows.

Problem IV.1 (Optimal path planning)**.** Given a 5-tuple $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, \mathcal{X}_{\text{goal}}, D, J)$ find $\sigma^* =$

$$\begin{aligned} \underset{\sigma \in \Sigma(\mathcal{X})}{\arg\min} \quad & J(\sigma) \\ \text{subj. to} \quad & \sigma(0) = \mathbf{x}_{\text{init}} \text{ and } \sigma(1) \in \mathcal{X}_{\text{goal}} \\ & \sigma(\alpha) \in \mathcal{X}_{\text{free}} && \forall \alpha \in [0, 1] \\ & D(\sigma(\alpha), \sigma'(\alpha), \sigma''(\alpha), \ldots) && \forall \alpha \in [0, 1]. \end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

The problem of feasible and optimal path planning has been studied extensively in the past few decades. The complexity of this problem is well understood, and many practical algorithms have been developed.

The problem of finding an optimal path subject to holonomic and differential constraints as formulated in Problem IV.1 is known to be PSPACE-hard [63]. This means that it is at least as hard as solving any NP-complete problem and thus, assuming P $\neq$ NP, there is no efficient (polynomial-time) algorithm that is able to solve all instances of the problem. Research attention has since been directed toward studying approximate methods, or approaches to subsets of the general motion planning problem.

In particular, a shortest path for a holonomic vehicle in a 2-D environment with polygonal obstacles can be obtained using visibility graph approach in $O(n^2)$ [64]. Also, a shortest paths for a car-like vehicle in the absence of obstacles can be constructed analytically: Dubins [57] has shown that the shortest path having curvature bounded by κ between given two points p_1, p_2 and with prescribed tangents θ_1, θ_2 is a curve consisting of at most three segments, each one being either a circular arc segment or a straight line. Later, Reeds and Shepp [58] extended the method for a car that can move both forwards and backwards.

Since for most problems of interest in autonomous driving, exact algorithms with practical computational complexity are unavailable [65], one has to resort to more general, numerical solution methods. These methods generally do not find an exact solution, but attempt to find a satisfactory solution or a sequence of feasible solutions that converge to the optimal solution. The utility and performance of these approaches are typically quantified by the class of problems for which they are applicable as well as their guarantees for converging to an optimal solution. The numerical methods for path planning can be broadly divided in three main categories:

Variational methods represent the path as a function parametrized by a finite-dimensional vector and the optimal path is sought by optimizing over the vector parameter using non-linear continuous optimization techniques. These methods are attractive for their rapid convergence to *locally* optimal solutions; however, they typically lack the ability to find globally optimal solutions unless an appropriate initial guess in provided. For a detailed discussion on variational methods, see Section IV-C.

Graph-search methods discretize the configuration space of the vehicle as a graph, where the vertices represent a finite collection of vehicle configurations and the edges represent transitions between vertices. The desired path is found by performing a search for a minimum-cost path in such a graph. Graph search methods are not prone to getting stuck in local minima, however, they are limited to optimize only over a finite set of paths, namely those that can be constructed from the atomic motion primitives in the graph. For a detailed discussion about graph search methods, see Section IV-D.

Incremental search methods sample the configuration space and incrementally build a reachability graph (oftentimes a tree) that maintains a discrete set of reachable configurations and feasible transitions between them. Once the graph is large enough so that at least one node is in the goal region, the desired path is obtained by tracing the edges that lead to that node from the start configuration. In contrast to more basic graph search methods, sampling-based methods incrementally increase the size of the graph until a satisfactory solution is found within the graph. For a detailed discussion about incremental search methods, see Section IV-E.

Clearly, it is possible to exploit the advantages of each of these methods by combining them. For example, one can use a coarse graph search to obtain an initial guess for the variational method as reported in [66] and [67]. A comparison of key properties of select path planning methods is given in Table I. In the remainder of this section, we will discuss the path planning algorithms and their properties in detail.

B. Trajectory Planning

The motion planning problems in dynamic environments or with dynamic constraints may be more suitably formulated in the trajectory planning framework, in which the solution of the problem is a trajectory, i.e. a time-parametrized function $\pi(t) : [0, T] \to \mathcal{X}$ prescribing the evolution of the configuration of the vehicle in time.

Let $\Pi(\mathcal{X}, T)$ denote the set of all continuous functions $[0, T] \to \mathcal{X}$ and $x_{\text{init}} \in \mathcal{X}$. be the initial configuration of the vehicle. The goal region is $X_{\text{goal}} \subseteq \mathcal{X}$. The set of all allowed configurations at time $t \in [0, T]$ is denoted as $\mathcal{X}_{\text{free}}(t)$ and used to encode holonomic constraints such as the requirement on the path to avoid collisions with static and, possibly, dynamic obstacles. The differential constraints on the trajectory are represented by a predicate $D(\mathbf{x}, \mathbf{x}', \mathbf{x}'', \ldots)$ and can be used to enforce dynamic constraints on the trajectory. Further, let $J(\pi) : \Pi(\mathcal{X}, T) \to \mathbb{R}$ be the cost functional. Under these assumptions, the optimal version of the trajectory planning problem can be very generally stated as:

Problem IV.2 (Optimal trajectory planning)**.** Given a 6-tuple $(\mathcal{X}_{\text{free}}, \mathbf{x}_{\text{init}}, X_{\text{goal}}, D, J, T)$ find $\pi^* =$

$$\begin{aligned} &\underset{\pi \in \Pi(\mathcal{X}, T)}{\arg\min} && J(\pi) && \\ &\text{subj. to} && \pi(0) = \mathbf{x}_{\text{init}} \text{ and } \pi(T) \in X_{\text{goal}} && \\ &&& \pi(t) \in \mathcal{X}_{\text{free}} && \forall t \in [0, T] \\ &&& D(\pi(t), \pi'(t), \pi''(t), \ldots) && \forall t \in [0, T]. \end{aligned}$$

Since trajectory planning in a dynamic environment is a generalization of path planning in static environments, the problem remains PSPACE-hard. Moreover, trajectory planning in dynamic environments has been shown to be harder than path planning in the sense that some variants of the problem that are tractable in static environments become intractable when an analogous problem is considered in a dynamic environment [78].

Tractable exact algorithms are not available for non-trivial trajectory planning problems occurring in autonomous driving, making the numerical methods a popular choice for the task. Trajectory planning problems can be numerically solved using some variational methods directly in the time domain or by converting the trajectory planning problem to path planning in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

	Model assumptions	Completeness	Optimality	Time Complexity	Anytime
Geometric Methods					
Visibility graph [33]	2-D polyg. conf. space, no diff. constraints	Yes	Yes [a]	$O(n^2)$ [64] [b]	No
Cyl. algebr. decomp. [68]	No diff. constraints	Yes	No	Exp. in dimension. [69]	No
Variational Methods					
Variational methods (Sec IV-C)	Lipschitz-continuous Jacobian	No	Locally optimal	$O(1/\epsilon)$ [70] [k,l]	Yes
Graph-search Methods					
Road lane graph + Dijkstra (Sec IV-D1)	Arbitrary	No [c]	No [d]	$O(n + m \log m)$ [71] [e,f]	No
Lattice/tree of motion prim. + Dijkstra (Sec IV-E)	Arbitrary	No [c]	No [d]	$O(n + m \log m)$ [71] [e,f]	No
PRM [72] [g] + Dijkstra	Exact steering procedure available	Probabilistically complete [73] [*]	Asymptotically optimal [*] [73]	$O(n^2)$ [73] [h,f,*]	No
PRM* [73], [74] + Dijkstra	Exact steering procedure available	Probabilistically complete [73]–[75] [*,†]	Asymptotically optimal [73]–[75] [*,†]	$O(n \log n)$ [73], [75] [h,f,*,†]	No
RRG [73] + Dijkstra	Exact steering procedure available	Probabilistically complete [73] [*]	Asymptotically optimal [73] [*]	$O(n \log n)$ [73] [h,f,*]	Yes
Incremental Search					
RRT [76]	Arbitrary	Probabilistically complete [76] [i,*]	Suboptimal [73] [*]	$O(n \log n)$ [73] [h,f,*]	Yes
RRT* [73]	Exact steering procedure available	Probabilistically complete [73], [75] [*,†]	Asymptotically optimal [73], [75] [*,†]	$O(n \log n)$ [73], [75] [h,f,*,†]	Yes
SST* [77]	Lipschitz-continuous dynamics	Probabilistically complete [77] [†]	Asymptotically optimal [77] [†]	N/A [j]	Yes

Table I: Comparison of path planning methods. **Legend:** *a*: for the shortest path problem; *b*: n is the number of points defining obstacles; *c*: complete only w.r.t. the set of paths induced by the given graph; *d*: optimal only w.r.t. the set of paths induced by the given graph; *e*: n and m are the number of edges and vertices in the graph respectively; *f*: assuming $O(1)$ collision checking; *g*: batch version with fixed-radius connection strategy; *h*: n is the number of samples/algorithm iterations; *i*: for certain variants; *j*: not explicitly analyzed; *k*: ϵ is the required distance from the optimal cost; *l*: faster rates possible with additional assumptions; $*$: shown for systems without differential constraints; †: shown for some class of nonholonomic systems.

a configuration space with an added time-dimension [79]. A solution to such a path planning problem is then found using a path planning algorithm that can handle differential constraints and converted back to the trajectory form.

C. Variational Methods

We will first address the trajectory planning problem in the framework of non-linear continuous optimization. In this context, the problem is often referred to as trajectory optimization. Within this subsection we will adopt the trajectory planning formulation with the understanding that doing so does not affect generality since path planning can be formulated as trajectory optimization over the unit time interval. To leverage existing nonlinear optimization methods, it is necessary to project the infinite-dimensional function space of trajectories to a finite-dimensional vector space. In addition, most nonlinear programming techniques require the trajectory optimization problem, as formulated in Problem IV.2, to be converted into the following form

$$\begin{aligned} \underset{\pi \in \Pi(X,T)}{\arg\min} \quad & J(\pi) \\ \text{subj. to} \quad & \pi(0) = \mathbf{x}_{\text{init}} \text{ and } \pi(T) \in X_{\text{goal}} \\ & f(\pi(t), \pi'(t), \ldots) = 0 \qquad \forall t \in [0,T] \\ & g(\pi(t), \pi'(t), \ldots) \leq 0 \qquad \forall t \in [0,T], \end{aligned}$$

where the holonomic and differential constraints are represented as a system of equality and inequality constraints.

In some applications the constrained optimization problem is relaxed to an unconstrained one using penalty or barrier functions. In both cases, the constraints are replaced by an augmented cost functional. With the penalty method, the cost functional takes the form

$$\tilde{J}(\pi) = J(\pi) + \frac{1}{\varepsilon} \int_0^T \Big[\|f(\pi, \pi', \ldots)\|^2 + \| \max(0, g(\pi, \pi', \ldots))\|^2 \Big] \, \mathrm{d}t \; .$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

Similarly, barrier functions can be used in place of inequality constraints. The augmented cost functional in this case takes the form

$$\tilde{J}(\pi) = J(\pi) + \varepsilon \int_0^T h(\pi(t))\mathrm{d}t \,,$$

where the barrier function satisfies $g(\pi) < 0 \Rightarrow h(\pi) < \infty$, $g(\pi) \geq 0 \Rightarrow h(\pi) = \infty$, and $\lim_{g(\pi)\to 0}\{h(\pi)\} = \infty$. The intuition behind both of the augmented cost functionals is that, by making ε small, minima in cost will be close to minima of the original cost functional. An advantage of barrier functions is that local minima remain feasible, but must be initialized with a feasible solution to have finite augmented cost. Penalty methods on the other hand can be initialized with any trajectory and optimized to a local minima. However, local minima may violate the problem constraints. A variational formulation using barrier functions is proposed in [62] where a change of coordinates is used to convert the constraint that the vehicle remain on the road into a linear constraint. A logarithmic barrier is used with a Newton-like method in a similar fashion to interior point methods. The approach effectively computes minimum time trajectories for a detailed vehicle model over a segment of roadway.

Next, two subclasses of variational methods are discussed: Direct and indirect methods.

Direct Methods: A general principle behind direct variational methods is to restrict the approximate solution to a finite-dimensional subspace of $\Pi(\mathcal{X}, T)$. To this end, it is usually assumed that

$$\pi(t) \approx \tilde{\pi}(t) = \sum_{i=1}^{N} \pi_i \phi_i(t),$$

where π_i is a coefficient from $\mathbb{R}$, and $\phi_i(t)$ are *basis* functions of the chosen subspace. A number of numerical approximation schemes have proven useful for representing the trajectory optimization problem as a nonlinear program. We mention here the two most common schemes: Numerical integrators with collocation and pseudospectral methods.

1) Numerical Integrators with Collocation: With collocation, it is required that the approximate trajectory satisfies the constraints in a set of discrete points $\{t_j\}_{j=1}^{M}$. This requirement results in two systems of discrete constraints: A system of nonlinear equations which approximates the system dynamics

$$f(\tilde{\pi}(t_j), \tilde{\pi}'(t_j)) = 0 \quad \forall j = 1, \ldots, M$$

and a system of nonlinear inequalities which approximates the state constraints placed on the trajectory

$$g(\tilde{\pi}(t_j), \tilde{\pi}'(t_j)) \leq 0 \quad \forall j = 1, \ldots, M \,.$$

Numerical integration techniques are used to approximate the trajectory between the collocation points. For example, a piecewise linear basis

$$\phi_i(t) = \begin{cases} (t - t_{i-1})/(t_i - t_{i-1}) & \text{if } t \in [t_{i-1}, \ t_i] \\ (t_{i+1} - t)/(t_{i+1} - t_i) & \text{if } t \in [t_i, \ t_{i+1}] \\ 0 & \text{otherwise} \end{cases}$$

together with collocation gives rise to the Euler integration method. Higher order polynomials result in the Runge-Kutta family of integration methods. Formulating the nonlinear program with collocation and Euler's method or one of the Runge-Kutta methods is more straightforward than some other methods making it a popular choice. An experimental system which successfully uses Euler's method for numerical approximation of the trajectory is presented in [14].

In contrast to Euler's method, the Adams approximation, is investigated in [80] for optimizing the trajectory for a detailed vehicle model and is shown to provide improved numerical accuracy and convergence rates.

2) Pseudospectral Methods: Numerical integration techniques utilize a discretization of the time interval with an interpolating function between collocation points. Pseudospectral approximation schemes build on this technique by additionally representing the interpolating function with a basis. Typical basis functions interpolating between collocation points are finite subsets of the Legendre or Chebyshev polynomials. These methods typically have improved convergence rates over basic collocation methods, which is especially true when adaptive methods for selecting collocation points and basis functions are used as in [81].

Indirect Methods: Pontryagin's minimum principle [82], is a celebrated result from optimal control which provides optimality conditions of a solution to Problem IV.2. Indirect methods, as the name suggests, solve the problem by finding solutions satisfying these optimality conditions. These optimality conditions are described as an augmented system of ordinary differential equations(ODEs) governing the states and a set of co-states. However, this system of ODEs results in a two point boundary value problem and can be difficult to solve numerically. One technique is to vary the free initial conditions of the problem and integrate the system forward in search of the initial conditions which leads to the desired terminal states. This method is known as the *shooting* method, and a version of this approach has been applied to planning parking maneuvers in [83]. The advantage of indirect methods, as in the case of the shooting method, is the reduction in dimensionality of the optimization problem to the dimension of the state space.

The topic of variational approaches is very extensive and hence, the above is only a brief description of select approaches. See [84], [85] for dedicated surveys on this topic.

D. Graph Search Methods

Although useful in many contexts, the applicability of variational methods is limited by their convergence to only local minima. In this section, we will discuss the class of methods that attempts to mitigate the problem by performing global search in the discretized version of the path space. These so-called graph search methods discretize the configuration space $\mathcal{X}$ of the vehicle and represent it in the form of a graph and then search for a minimum cost path on such a graph.

In this approach, the configuration space is represented as a graph $G = (V, E)$, where $V \subset \mathcal{X}$ is a discrete set of selected configurations called vertices and $E = \{(o_i, d_i, \sigma_i)\}$ is the set of edges, where $o_i \in V$ represents the origin of the edge, d_i represents the destination of the edge and σ_i represents the path segments connecting o_i and d_i. It is assumed that the path

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

segment σ_i connects the two vertices: $\sigma_i(0) = o_i$ and $\sigma_i(1) = d_i$. Further, it is assumed that the initial configuration $\mathbf{x}_{\text{init}}$ is a vertex of the graph. The edges are constructed in such a way that the path segments associated with them lie completely in $\mathcal{X}_{\text{free}}$ and satisfy differential constraints. As a result, any path on the graph can be converted to a feasible path for the vehicle by concatenating the path segments associated with edges of the path through the graph.

There is a number of strategies for constructing a graph discretizing the free configuration space of a vehicle. In the following subsections, we discuss three common strategies: Hand-crafted lane graphs, graphs derived from geometric representations and graphs constructed by either control or configuration sampling.

1) Lane Graph: When the path planning problem involves driving on a structured road network, a sufficient graph discretization may consist of edges representing the path that the car should follow within each lane and paths that traverse intersections.

Road lane graphs are often partly algorithmically generated from higher-level street network maps and partly human edited. An example of such a graph is in Figure IV.1.

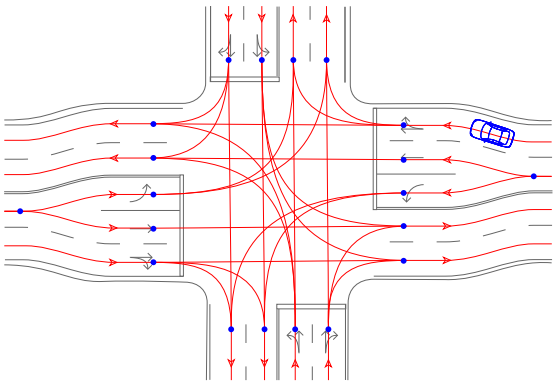


Figure IV.1: Hand-crafted graph representing desired driving paths under normal circumstances.

Although most of the time it is sufficient for the autonomous vehicle to follow the paths encoded in the road lane graph, occasionally it must be able to navigate around obstacles that were not considered when the road network graph was designed or in environments not covered by the graph. Consider for example a faulty vehicle blocking the lane that the vehicle plans to traverse – in such a situation a more general motion planning approach must be used to find a collision-free path around the detected obstacle.

The general path planning approaches can be broadly divided into two categories based on how they represent the obstacles in the environment. So-called geometric or combinatorial methods work with geometric representations of the obstacles, where in practice the obstacles are most commonly described using polygons or polyhedra. On the other hand, so-called sampling-based methods abstract away from how the obstacles are internally represented and only assumes access to a function that determines if any given path segment is in collision with any of the obstacles.

2) Geometric Methods: In this section, we will focus on path planning methods that work with geometric representations of obstacles. We will first concentrate on path planning without differential constraints because for this formulation, efficient exact path planning algorithms exist. Although not being able to enforce differential constraints is limiting for path planning for traditionally-steered cars because the constraint on minimum turn radius cannot be accounted for, these methods can be useful for obtaining the lower- and upper-bounds[1] on the length of a curvature-constrained path and for path planning for more exotic car constructions that can turn on the spot.

In path planning, the term roadmap is used to describe a graph discretization of $\mathcal{X}_{\text{free}}$ that describes well the connectivity of the free configuration space and has the property that any point in $\mathcal{X}_{\text{free}}$ is trivially reachable from some vertices of the roadmap. When the set $\mathcal{X}_{\text{free}}$ can be described geometrically using a linear or semi-algebraic model, different types of roadmaps for $\mathcal{X}_{\text{free}}$ can be algorithmically constructed and subsequently used to obtain complete path planning algorithms. Most notably, for $\mathcal{X}_{\text{free}} \subseteq \mathbb{R}^2$ and polygonal models of the configuration space, several efficient algorithms for constructing such roadmaps exists such as the vertical cell decomposition [86], generalized Voronoi diagrams [87], [88], and visibility graphs [33], [89]. For higher dimensional configuration spaces described by a general semi-algebraic model, the technique known as cylindrical algebraic decomposition can be used to construct a roadmap in the configuration space [47], [90] leading to complete algorithms for a very general class of path planning problems. The fastest of this class is an algorithm developed by Canny [69] that has (single) exponential time complexity in the dimension of the configuration space. The result is however mostly of a theoretical nature without any known implementation to date.

Due to its relevance to path planning for car-like vehicles, a number of results also exist for the problem of path planning with a constraint on maximum curvature. Backer and Kirkpatrick [91] provide an algorithm for constructing a path with bounded curvature that is polynomial in the number of features of the domain, the precision of the input and the number of segments on the simplest obstacle-free Dubins path connecting the specified configurations. Since the problem of finding a *shortest* path with bounded curvature amidst polygonal obstacles is NP-hard, it is not surprising that no exact polynomial solution algorithm is known. An approximation algorithm for finding shortest curvature-bounded path amidst polygonal obstacles has been first proposed by Jacobs and Canny [92] and later improved by Wang and Agarwal [93] with time complexity $O(\frac{n^2}{\epsilon^4} \log n)$, where n is the number of vertices of the obstacles and ϵ is the approximation factor.

3) Sampling-based Methods: In autonomous driving, a geometric model of $\mathcal{X}_{\text{free}}$ is usually not directly available and it would be too costly to construct from raw sensoric data. Moreover, the requirements on the resulting path are often far more complicated than a simple maximum curvature

[1]Lower bound is the length of the path without curvature constraint, upper bound is the length of a path for a large robot that serves as an envelope within which the car can turn in any direction.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

constraint. This may explain the popularity of sampling-based techniques that do not enforce a specific representation of the free configuration set and dynamic constraints. Instead of reasoning over a geometric representation, the sampling based methods explore the reachability of the free configuration space using *steering* and *collision checking* routines:

The steering function `steer(`$\mathbf{x}, \mathbf{y}$`)` returns a path segment starting from configuration $\mathbf{x}$ going towards configuration $\mathbf{y}$ (but not necessarily reaching $\mathbf{y}$) ensuring the differential constraints are satisfied, i.e., the resulting motion is feasible for the vehicle model in consideration. The exact manner in which the steering function is implemented depends on the context in which it is used. Some typical choices encountered in the literature are:

1) Random steering: The function returns a path that results from applying a random control input through a forward model of the vehicle from state $\mathbf{x}$ for either a fixed or variable time step [94].
2) Heuristic steering: The function returns a path that results from applying control that is heuristically constructed to guide the system from $\mathbf{x}$ towards $\mathbf{y}$ [95]–[97]. This includes selecting the maneuver from a predesigned discrete set (library) of maneuvers.
3) Exact steering: The function returns a feasible path that guides the system from $\mathbf{x}$ to $\mathbf{y}$. Such a path corresponds to a solution of a 2-point boundary value problem. For some systems and cost functionals, such a path can be obtained analytically, e.g., a straight line for holonomic systems, a Dubins curve for forward-moving unicycle [57], or a Reeds-Shepp curve for bi-directional unicycle [58]. An analytic solution also exists for differentially flat systems [98], while for more complicated models, the exact steering can be obtained by solving the two-point boundary value problem.
4) Optimal exact steering: The function returns an optimal exact steering path with respect to the given cost functional. In fact, the straight line, the Dubins curve, and the Reeds-Shepp curve from the previous point are optimal solutions assuming that the cost functional is the arc-length of the path [57], [58].

The collision checking function `col-free(`σ`)` returns true if path segment σ lies entirely in $\mathcal{X}_{\text{free}}$ and it is used to ensure that the resulting path does not collide with any of the obstacles.

Having access to steering and collision checking functions, the major challenge becomes how to construct a discretization that approximates well the connectivity of $\mathcal{X}_{\text{free}}$ without having access to an explicit model of its geometry. We will now review sampling-based discretization strategies from literature.

A straightforward approach is to choose a set of motion primitives (fixed maneuvers) and generate the search graph by recursively applying them starting from the vehicle's initial configuration $\mathbf{x}_{\text{init}}$, e.g., using the method in Algorithm 1. For path planning without differential constraints, the motion primitives can be simply a set of straight lines with different directions and lengths. For a car-like vehicle, such motion primitive might by a set of arcs representing the path the car would follow with different values of steering. A variety of techniques can be used for generating motion primitives for driverless vehicles. A simple approach is to sample a number of control inputs and to simulate forwards in time using a vehicle model to obtain feasible motions. In the interest of having continuous curvature paths, clothoid segments are also sometimes used [99]. The motion primitives can be also obtained by recording the motion of a vehicle driven by an expert driver [100].

Observe that the recursive application of motion primitives may generate a tree graph in which in the worst-case no two edges lead to the same configuration. There are, however, sets of motion primitives, referred to as lattice-generating, that result in regular graphs resembling a lattice. See Figure IV.2a for an illustration. The advantage of lattice generating primitives is that the vertices of the search graph cover the configuration space uniformly, while trees in general may have a high density of vertices around the root vertex. Pivtoraiko et al. use the term "state lattice" to describe such graphs in [101] and point out that a set of lattice-generating motion primitives for a system in hand can be obtained by first generating regularly spaced configurations around origin and then connecting the origin to such configurations by a path that represents the solution to the two-point boundary value problem between the two configurations.

Algorithm 1: Recursive Roadmap Construction

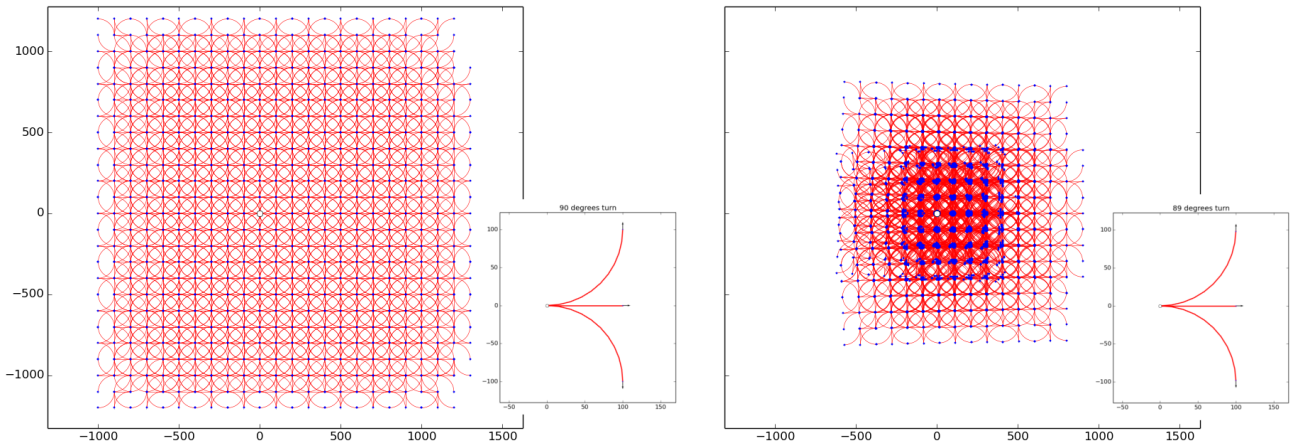
$V \leftarrow \{\mathbf{x}_{\text{init}}\}$; $E \leftarrow \emptyset$; $Q \leftarrow$ new queue($\mathbf{x}_{\text{init}}$);
while $Q \neq \emptyset$ **do**
  $\mathbf{x} \leftarrow$ pop element from Q;
  $M \leftarrow$ generate a set of path segments by applying motion primitives from configuration $\mathbf{x}$;
  for $\sigma \in M$ **do**
    if `col-free(`σ`)` **then**
      $E \leftarrow E \cup \{(\mathbf{x}, \sigma(1), \sigma)\}$;
      if $\sigma(1) \notin V$ **then**
        add $\sigma(1)$ to Q;
        $V \leftarrow V \cup \{\sigma(1)\}$;
return (V,E)

An effect that is similar to recursive application of lattice-generating motion primitives from the initial configuration can be achieved by generating a discrete set of samples covering the (free) configuration space and connecting them by feasible path segments obtained using an exact steering procedure.

Most sampling-based roadmap construction approaches follow the algorithmic scheme shown in Algorithm 2, but differ in the implementation of the `sample-points(`$\mathcal{X}, n$`)` and `neighbors(`$\mathbf{x}, V$`)` routines. The function `sample-points(`$\mathcal{X}, n$`)` represents the strategy for selecting n points from the configuration space $\mathcal{X}$, while the function `neighbors(`$\mathbf{x}, V$`)` represents the strategy for selecting a set of neighboring vertices $N \subseteq V$ for a vertex $\mathbf{x}$, which the algorithm will attempt to connect to $\mathbf{x}$ by a path segment using an exact steering function, $\texttt{steer}_{\text{exact}}(\mathbf{x}, \mathbf{y})$.

The two most common implementations of `sample-points(`$\mathcal{X}, n$`)` function are 1) return n points arranged in a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles



(a) Lattice graph (b) Non-lattice graph

Figure IV.2: Lattice and non-lattice graph, both with 5000 edges. (a) The graph resulting from recursive application of 90° left circular arc, 90° right circular arc, and a straight line. (b) The graph resulting from recursive application of 89° left circular arc, 89° right circular arc, and a straight line. The recursive application of those primitives does form a tree instead of a lattice with many branches looping in the neighborhood of the origin. As a consequence, the area covered by the right graph is smaller.

Algorithm 2: Sampling-based Roadmap Construction

$V \leftarrow \{\mathbf{x}_{\text{init}}\} \cup \texttt{sample-points}(\mathcal{X}, n)$; $E \leftarrow \emptyset$;
for $\mathbf{x} \in V$ **do**
  for $\mathbf{y} \in \texttt{neighbors}(\mathbf{x}, V)$ **do**
    $\sigma \leftarrow \texttt{steer}_{\text{exact}}(\mathbf{x}, \mathbf{y})$;
    if $\texttt{col-free}(\sigma)$ **then**
      $E \leftarrow E \cup \{(\mathbf{x}, \mathbf{y}, \sigma)\}$;
return (V,E)

regular grid and 2) return n randomly sampled points from $\mathcal{X}$. While random sampling has an advantage of being generally applicable and easy to implement, so-called Sukharev grids have been shown to minimize the radius of the largest empty ball with no sample point inside [102]. The two most commonly used strategies for implementing $\texttt{neighbors}(\mathbf{x}, V)$ function are to take 1) the set of k-nearest neighbors to $\mathbf{x}$ or 2) the set of points lying within the ball centered at $\mathbf{x}$ with radius r.

In particular, samples arranged deterministically in a d-dimensional grid with the neighborhood taken as 4 or 8 nearest neighbors in 2-D or the analogous pattern in higher dimensions represents a straightforward deterministic discretization of the free configuration space. This is in part because they arise naturally from widely used bitmap representations of free and occupied regions of robots' configuration space [103].

Kavraki et al. [104] advocate the use of random sampling within the framework of Probabilistic Roadmaps (PRM) in order to construct roadmaps in high-dimensional configuration spaces, because unlike grids, they can be naturally run in an anytime fashion. The batch version of PRM [72] follows the scheme in Algorithm 2 with random sampling and neighbors selected within a ball with fixed radius r. Due to the general formulation of PRMs, they have been used for path planning for a variety of systems, including systems with differential constraints. However, the theoretical analyses of the algorithm have primarily been focused on the performance of the algorithm for systems without differential constraints, i.e. when a straight line is used to connect two configurations. Under such an assumption, PRMs have been shown in [73] to be probabilistically complete and asymptotically optimal. That is, the probability that the resulting graph contains a valid solution (if it exists) converges to one with increasing size of the graph and the cost of the shortest path in the graph converges to the optimal cost. Karaman and Frazzoli [73] proposed an adaptation of batch PRM, called PRM*, that instead only connects neighboring vertices in a ball with a logarithmically shrinking radius with increasing number of samples to maintain both asymptotic optimality and computational efficiency.

In the same paper, the authors propose Rapidly-exploring Random Graphs (RRG*), which is an incremental discretization strategy that can be terminated at any time while maintaining the asymptotic optimality property. Recently, Fast Marching Tree (FMT*) [105] has been proposed as an asymptotically optimal alternative to PRM*. The algorithm combines discretization and search into one process by performing a lazy dynamic programming recursion over a set of sampled vertices that can be subsequently used to quickly determine the path from initial configuration to the goal region.

Recently, the theoretical analysis has been extended also to differentially constrained systems. Schmerling et al. [74] propose differential versions of PRM* and FMT* and prove asymptotic optimality of the algorithms for driftless control-affine dynamical systems, a class that includes models of non-slipping wheeled vehicles.

4) Graph Search Strategies: In the previous section, we have discussed techniques for the discretization of the free configuration space in the form of a graph. To obtain an actual optimal path in such a discretization, one must employ one of the graph search algorithms. In this section, we are going to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

review the graph search algorithms that are relevant for path planning.

The most widely recognized algorithm for finding shortest paths in a graph is probably the Dijkstra's algorithm [32]. The algorithm performs the best first search to build a tree representing shortest paths from a given source vertex to all other vertices in the graph. When only a path to a single vertex is required, a heuristic can be used to guide the search process. The most prominent heuristic search algorithm is A* developed by Hart, Nilsson and Raphael [106]. If the provided heuristic function is admissible (i.e., it never overestimates the cost-to-go), A* has been shown to be optimally efficient and is guaranteed to return an optimal solution. For many problems, a bounded suboptimal solution can be obtained with less computational effort using Weighted A* [107], which corresponds to simply multiplying the heuristic by a constant factor $\epsilon > 1$. It can be shown that the solution path returned by A* with such an inflated heuristics is guaranteed to be no worse than $(1 + \epsilon)$ times the cost of an optimal path.

Often, the shortest path from the vehicle's current configuration to the goal region is sought repeatedly every time the model of the world is updated using sensory data. Since each such update usually affects only a minor part of the graph, it might be wasteful to run the search every time completely from scratch. The family of real-time replanning search algorithms such as D* [108], Focussed D* [109] and D* Lite [110] has been designed to efficiently recompute the shortest path every time the underlying graph changes, while making use of the information from previous search efforts.

Anytime search algorithms attempt to provide a first suboptimal path quickly and continually improve the solution with more computational time. Anytime A* [111] uses a weighted heuristic to find the first solution and achieves the anytime behavior by continuing the search with the cost of the first path as an upper bound and the admissible heuristic as a lower bound, whereas Anytime Repairing A* (ARA*) [112] performs a series of searches with inflated heuristic with decreasing weight and reuses information from previous iterations. On the other hand, Anytime Dynamic A* (ADA*) [113] combines ideas behind D* Lite and ARA* to produce an anytime search algorithm for real-time replanning in dynamic environments.

E. Incremental Search Techniques

A disadvantage of the techniques that search over a fixed graph discretization is that they search only over the set of paths that can be constructed from primitives in the graph discretization. Therefore, these techniques may fail to return a feasible path or return a noticeably suboptimal one.

The incremental *feasible* motion planners strive to address this problem and provide a feasible path to any motion planning problem instance, if one exists, given enough computation time. Typically, these methods incrementally build increasingly finer discretization of the configuration space while concurrently attempting to determine if a path from initial configuration to the goal region exists in the discretization at each step. If the instance is "easy", the solution is provided quickly, but in general the computation time can be unbounded. Similarly, incremental *optimal* motion planning approaches on top of finding a feasible path fast attempt to provide a sequence of solutions of increasing quality that converges to an optimal path.

The term *probabilistically complete* is used in the literature to describe algorithms that find a solution, if one exists, with probability approaching one with increasing computation time. Note that probabilistically complete algorithm may not terminate if the solution does not exist. Similarly, the term *asymptotically optimal* is used for algorithms that converge to optimal solution with probability one.

A naïve strategy for obtaining completeness and optimality in the limit is to solve a sequence of path planning problems on a fixed discretization of the configuration space, each time with a higher resolution of the discretization. One disadvantage of this approach is that the path planning processes on individual resolution levels are independent without any information reuse. Moreover, it is not obvious how fast the resolution of the discretization should be increased before a new graph search is initiated, i.e., if it is more appropriate to add a single new configuration, double the number of configuration, or double the number of discrete values along each configuration space dimension. To overcome such issues, incremental motion planning methods interweave incremental discretization of configuration space with search for a path within one integrated process.

An important class of methods for incremental path planning is based on the idea of incrementally growing a tree rooted at the initial configuration of the vehicle outwards to explore the reachable configuration space. The "exploratory" behavior is achieved by iteratively selecting a random vertex from the tree and by expanding the selected vertex by applying the steering function from it. Once the tree grows large enough to reach the goal region, the resulting path is recovered by tracing the links from the vertex in the goal region backwards to the initial configuration. The general algorithmic scheme of an incremental tree-based algorithm is described in Algorithm 3.

Algorithm 3: Incremental Tree-based Algorithm

$V \leftarrow \{\mathbf{x}_{\text{init}}\} \cup \texttt{sample-points}(\mathcal{X}, n)$; $E \leftarrow \emptyset$;
while *not interrupted* **do**
  $\mathbf{x}_{\text{selected}} \leftarrow \texttt{select}(V)$;
  $\sigma \leftarrow \texttt{extend}(\mathbf{x}_{\text{selected}}, V)$;
  if $\texttt{col-free}(\sigma)$ **then**
    $\mathbf{x}_{\text{new}} = \sigma(1)$;
    $V \leftarrow V \cup \{\mathbf{x}_{\text{new}}\}$; $E \leftarrow E \cup \{(\mathbf{x}_{\text{selected}}, \mathbf{x}_{\text{new}}, \sigma)\}$;
return (V,E)

One of the first randomized tree-based incremental planners was the expansive spaces tree (EST) planner proposed by Hsu et al. [114]. The algorithm selects a vertex for expansion, $\mathbf{x}_{\text{selected}}$, randomly from V with a probability that is inversely proportional to the number of vertices in its neighborhood, which promotes growth towards unexplored regions. During expansion, the algorithm samples a new vertex $\mathbf{y}$ within a neighborhood of a fixed radius around $\mathbf{x}_{\text{selected}}$, and use the

same technique for biasing the sampling procedure to select a vertex from the region that is relatively less explored. Then it returns a straight line path between $\mathbf{x}_{\text{selected}}$ and $\mathbf{y}$. A generalization of the idea for planning with kinodynamic constraints in dynamic environments was introduced in [115], where the capabilities of the algorithm were demonstrated on different non-holonomic robotic systems and the authors use an idealized version of the algorithm to establish that the probability of failure to find a feasible path depends on the expansiveness property of the state space and decays exponentially with the number of samples.

Rapidly-exploring Random Trees (RRT) [94] have been proposed by La Valle as an efficient method for finding feasible trajectories for high-dimensional non-holonomic systems. The rapid exploration is achieved by taking a random sample $\mathbf{x}_{\text{rnd}}$ from the free configuration space and extending the tree in the direction of the random sample. In RRT, the vertex selection function `select(V)` returns the nearest neighbor to the random sample $\mathbf{x}_{\text{rnd}}$ according to the given distance metric between the two configurations. The extension function `extend()` then generates a path in the configuration space by applying a control for a fixed time step that minimizes the distance to $\mathbf{x}_{\text{rnd}}$. Under certain simplifying assumptions (random steering is used for extension), the RRT algorithm has been shown to be probabilistic complete [76]. We remark that the result on probabilistic completeness does not readily generalize to many practically implemented versions of RRT that often use heuristic steering. In fact, it has been recently shown in [116] that RRT using heuristic steering with fixed time step is not probabilistically complete.

Moreover, Karaman and Frazzoli [117] demonstrated that the RRT converges to a suboptimal solution with probability one and designed an asymptotically optimal adaptation of the RRT algorithm, called RRT*. The RRT* at every iteration considers a set of vertices that lie in the neighborhood of newly added vertex $\mathbf{x}_{\text{new}}$ and a) connects $\mathbf{x}_{\text{new}}$ to the vertex in the neighborhood that minimizes the cost of path from $\mathbf{x}_{\text{init}}$ to $\mathbf{x}_{\text{new}}$ and b) rewires any vertex in the neighborhood to $\mathbf{x}_{\text{new}}$ if that results in a lower cost path from $\mathbf{x}_{\text{init}}$ to that vertex. An important characteristic of the algorithm is that the neighborhood region is defined as the ball centered at $\mathbf{x}_{\text{new}}$ with radius being function of the size of the tree: $r = \gamma \sqrt[d]{(\log n)/n}$, where n is the number of vertices in the tree, d is the dimension of the configuration space, and γ is an instance-dependent constant. It is shown that for such a function, the expected number of vertices in the ball is logarithmic in the size of the tree, which is necessary to ensure that the algorithm almost surely converges to an optimal path while maintaining the same asymptotic complexity as the suboptimal RRT.

Sufficient conditions for asymptotic optimality of RRT* under differential constraints are stated in [73] and demonstrated to be satisfiable for Dubins vehicle and double integrator systems. In a later work, the authors further show in the context of small-time locally attainable systems that the algorithm can be adapted to maintain not only asymptotic optimality, but also computational efficiency [75]. Other related works focus on deriving distance and steering functions for non-holonomic systems by locally linearizing the system dynamics [118] or by deriving a closed-form solution for systems with linear dynamics [119]. On the other hand, RRT$^{\text{X}}$ is an algorithm that extends RRT* to allow for real-time incremental replanning when the obstacle region changes, e.g., in the face of new data from sensors [120].

New developments in the field of sampling-based algorithms include algorithms that achieve asymptotic optimality without having access to an exact steering procedure. In particular, Li at al. [77] recently proposed the Stable Sparse Tree (SST) method for asymptotically (near-)optimal path planning, which is based on building a tree of randomly sampled controls propagated through a forward model of the dynamics of the system such that the locally suboptimal branches are pruned out to ensure that the tree remains sparse.

F. Practical Deployments

Three categories of path planning methodologies have been discussed for self-driving vehicles: variational methods, graph-searched methods and incremental tree-based methods. The actual field-deployed algorithms on self-driving systems come from all the categories described above. For example, even among the first four successful participants of DARPA Urban Challenge, the approaches used for motion planning significantly differed. The winner of the challenge, CMU's Boss vehicle used variational techniques for local trajectory generation in structured environments and a lattice graph in 4-dimensional configuration space (consisting of position, orientation, and velocity) together with Anytime D* to find a collision-free paths in parking lots [121]. The runner-up vehicle developed by Stanford's team reportedly used a search strategy coined Hybrid A* that during search, lazily constructs a tree of motion primitives by recursively applying a finite set of maneuvers. The search is guided by a carefully designed heuristic and the sparsity of the tree is ensured by only keeping a single node within a given region of the configuration space [122]. Similarly, the vehicle arriving third developed by the VictorTango team from Virginia Tech constructs a graph discretization of possible maneuvers and searches the graph with the A* algorithm [123]. Finally, the vehicle developed by MIT used a variant of RRT algorithm called closed-loop RRT with biased sampling [124].

V. Vehicle Control

Solutions to Problem IV.1 or IV.2 are provided by the motion planning process. The role of the feedback controller is to stabilize to the reference path or trajectory in the presence of modeling error and other forms of uncertainty. Depending on the reference provided by the motion planner, the control objective may be *path stabilization* or *trajectory stabilization*. More formally, the path stabilization problem is stated as follows:

Problem V.1. (Path stabilization) Given a controlled differential equation $\dot{x} = f(x, u)$, reference path $x_{ref} : \mathbb{R} \to \mathbb{R}^n$, and velocity $v_{ref} : \mathbb{R} \to \mathbb{R}$, find a feedback law, $u(x)$, such that solutions to $\dot{x} = f(x, u(x))$ satisfy the following: $\forall \varepsilon > 0$ and $t_1 < t_2$, there exists a $\delta > 0$ and a differentiable $s : \mathbb{R} \to \mathbb{R}$ such that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

Controller		Model	Stability	Time Complexity	Comments/Assumptions
Pure Pursuit	(V-A1)	Kinematic	LES* to ref. path	$O(n)^*$	No path curvature
Rear wheel based feedback	(V-A2)	Kinematic	LES* to ref. path	$O(n)^*$	$C^2(\mathbb{R}^n)$ ref. paths
Front wheel based feedback	(V-A3)	Kinematic	LES* to ref. path	$O(n)^*$	$C^1(\mathbb{R}^n)$ ref. paths; Forward driving only
Feedback linearization	(V-B2)	Steering rate controlled kinematic	LES* to ref. traj.	$O(1)$	$C^1(\mathbb{R}^n)$ ref. traj.; Forward driving only
Control Lyapunov design	(V-B1)	Kinematic	LES* to ref. traj.	O(1)	Stable for constant path curvature and velocity
Linear MPC	(V-C)	$C^1(\mathbb{R}^n \times \mathbb{R}^m)$ model$^\sharp$	LES* to ref. or path	$O\left(\sqrt{N}\ln\left(\frac{N}{\varepsilon}\right)\right)^\dagger$	Stability depends on horizon length
Nonlinear MPC	(V-C)	$C^1(\mathbb{R}^n \times \mathbb{R}^m)$ model$^\sharp$	Not guaranteed	$O(\frac{1}{\varepsilon})^\ddagger$	Works well in practice

Table II: Overview of controllers discussed within this section. **Legend:** $\star$: local exponential stability (LES); $*$: assuming (V.1) is evaluated by a linear search over an n-point discretization of the path or trajectory; †: assuming the use of an interior-point method to solve (V.26) with a time horizon of n and solution accuracy of ε; ‡: based on asymptotic convergence rate to local minimum of (V.23) using steepest descent. Not guaranteed to return solution or find global minimum.; $\sharp$: vector field over the state space $\mathbb{R}^n$ defined by each input in $\mathbb{R}^m$ is a continuously differentiable function so that the gradient of the cost or linearization about the reference is defined.

1) $\left\|x(t_1) - x_{ref}(s(t_1))\right\| \leq \delta$
   $\Rightarrow \left\|x(t_2) - x_{ref}(s(t_2))\right\| \leq \varepsilon$
2) $\lim_{t\to\infty} \left\|x(t) - x_{ref}(s(t))\right\| = 0$
3) $\lim_{t\to\infty} \dot{s}(t) = v_{ref}(s(t))$.

Qualitatively, these conditions are that (1) a small initial tracking error will remain small, (2) the tracking error must converge to zero, and (3) progress along the reference path tends to a nominal rate.

Many of the proposed vehicle control laws, including several discussed in this section, use a feedback law of the form

$$u(x) = f\left(\underset{\gamma}{\operatorname{argmin}} \|x - x_{ref}(\gamma)\|\right), \tag{V.1}$$

where the feedback is a function of the nearest point on the reference path. An important issue with controls of this form is that the closed loop vector field $f(x, u(x))$ will not be continuous. If the path is self intersecting or not differentiable at some point, a discontinuity in which $f(x, u(x))$ is will lie directly on the path. This leads to unpredictable behavior if the executed trajectory encounters the discontinuity. This discontinuity is illustrated in Figure V.1. A backstepping control design which does not use a feedback law of the form (V.1) is presented in [125].

The trajectory stabilization problem is more straightforward, but these controllers are prone to performance limitations [126].

Problem V.2. (Trajectory stabilization) Given a controlled differential equation $\dot{x} = f(x, u)$ and a reference trajectory $x_{ref}(t)$, find $\pi(x)$ such that solutions to $\dot{x} = f(x, \pi(x))$ satisfy the following: $\forall \varepsilon > 0$ and $t_1 < t_2$, there exists a $\delta > 0$ such that

1) $\left\|x(t_1) - x_{ref}(t_1)\right\| \leq \delta \Rightarrow \left\|x(t_2) - x_{ref}(t_2)\right\| \leq \varepsilon$

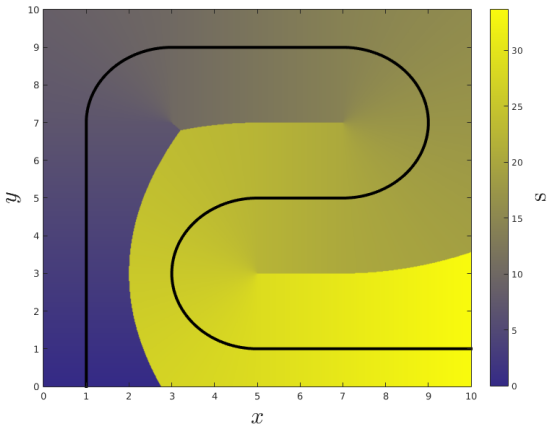


Figure V.1: Visualization of (V.5) for a sample reference path shown in black. The color indicates the value of s for each point in the plane and illustrates discontinuities in (V.5).

2) $\lim_{t\to\infty} \left\|x(t) - x_{ref}(t)\right\| = 0$

In many cases, analyzing the stability of trajectories can be reduced to determining the origin's stability in a time varying system. The basic form of Lyapunov's theorem is only applicable to time invariant systems. However, stability theory for time varying systems is also well established (e.g. [127, Theorem 4.9]).

Some useful qualifiers for various types of stability include:

- *Uniform asymptotic stability* for a time varying system which asserts that δ in condition 1 of the above problem is independent of t_1.
- *Exponential stability* asserts that the rate of convergence is bounded above by an exponential decay.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

A delicate issue that should be noted is that controller specifications are usually expressed in terms of the asymptotic tracking error as time tends to infinity. In practice, reference trajectories are finite so there should also be consideration for the transient response of the system.

The remainder of this section is devoted to a survey of select control designs which are applicable to driverless cars. An overview of these controllers is provided in Table II. Subsection V-A details a number of effective control strategies for path stabilization of the kinematic model, and subsection V-B2 discusses trajectory stabilization techniques. Predictive control strategies, discussed in subsection V-C, are effective for more complex vehicle models and can be applied to path and trajectory stabilization.

A. Path Stabilization for the Kinematic Model

1) Pure Pursuit: Among the earliest proposed path tracking strategies is pure pursuit. The first discussion appeared in [128], and was elaborated upon in [52], [129]. This strategy and its variations (e.g. [130], [131]) have proven to be an indispensable tool for vehicle control owing to its simple implementation and satisfactory performance. Numerous publications including two vehicles in the DARPA Grand Challenge [8] and three vehicles in the DARPA Urban challenge [9] reported using the pure pursuit controller.

The control law is based on fitting a semi-circle through the vehicle's current configuration to a point on the reference path ahead of the vehicle by a distance L called the lookahead distance. Figure V.2 illustrates the geometry. The circle is

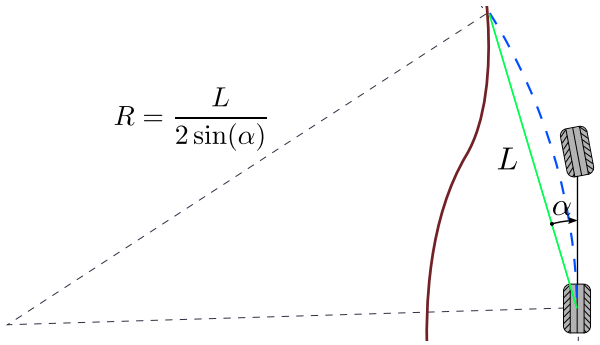


Figure V.2: Geometry of the pure pursuit controller. A circle (blue) is fit between rear wheel position and the reference path (brown) such that the chord length (green) is the look ahead distance L and the circle is tangent to the current heading direction.

defined as passing through the position of the car and the point on the path ahead of the car by one lookahead distance with the circle tangent to the car's heading. The curvature of the circle is given by

$$\kappa = \frac{2\sin(\alpha)}{L}. \tag{V.2}$$

For a vehicle speed v_r, the commanded heading rate is

$$\omega = \frac{2v_r\sin(\alpha)}{L}. \tag{V.3}$$

In the original publication of this controller [128], the angle α is computed directly from camera output data. However, α can be expressed in terms of the inertial coordinate system to define a state feedback control. Consider the configuration $(x_r, y_r, \theta)^T$ and the points on the path, $(x_{ref}(s), y_{ref}(s))$, such that $\|(x_{ref}(s), y_{ref}(s)) - (x_r, y_r)\| = L$. Since there is generally more than one such point on the reference, take the one with the greatest value of the parameter s to uniquely define a control. Then α is given by

$$\alpha = \arctan\left(\frac{y_{ref} - y_r}{x_{ref} - x_r}\right) - \theta. \tag{V.4}$$

Assuming that the path has no curvature and the vehicle speed is constant (potentially negative), the pure pursuit controller solves Problem V.1. For a fixed nonzero curvature, pure pursuit has a small steady state tracking error.

In the case where the vehicle's distance to the path is greater than L, the controller output is not defined. Another consideration is that changes in reference path curvature can lead to the car deviating from the reference trajectory. This may be acceptable for driving along a road, but can be problematic for tracking parking maneuvers. Lastly, the heading rate command ω becomes increasingly sensitive to the feedback angle α as the vehicle speed increases. A common fix for this issue is to scale L with the vehicle speed.

2) Rear wheel position based feedback: The next approach uses the rear wheel position as an output to stabilize a nominal rear wheel path [56]. The controller assigns

$$s(t) = \underset{\gamma}{\operatorname{argmin}} \|(x_r(t), y_r(t)) - (x_{ref}(\gamma), y_{ref}(\gamma))\|. \tag{V.5}$$

Detailed assumptions on the reference path and a finite domain containing the reference path where (V.5) is a continuous function are described in [56]. The unit tangent to the path at $s(t)$ is given by

$$\hat{t} = \frac{\left(\left.\frac{\partial x_{ref}}{\partial s}\right|_{s(t)}, \left.\frac{\partial y_{ref}}{\partial s}\right|_{s(t)}\right)}{\left\|\left(\frac{\partial x_{ref}(s(t))}{\partial s}, \frac{\partial y_{ref}(s(t))}{\partial s}\right)\right\|}, \tag{V.6}$$

and the tracking error vector is

$$d(t) := (x_r(t), y_r(t)) - \left(x_{ref}(s(t)), y_{ref}(s(t))\right). \tag{V.7}$$

These values are used to compute a transverse error coordinate from the path e which is a cross product between the two vectors

$$e = d_x\hat{t}_y - d_y\hat{t}_x \tag{V.8}$$

with the subscript denoting the component indices of the vector. The control uses the angle θ_e between the vehicle's heading vector and the tangent vector to the path.

$$\theta_e(t) = \theta - \arctan_2\left(\frac{\partial y_{ref}(s(t))}{\partial s}, \frac{\partial x_{ref}(s(t))}{\partial s}\right). \tag{V.9}$$

The geometry is illustrated in Figure V.3.

A change of coordinates to (s, e, θ_e) yields

$$\begin{aligned} \dot{s} &= \tfrac{v_r\cos(\theta_e)}{1-\kappa(s)e}, \\ \dot{e} &= v_r\sin(\theta_e), \\ \dot{\theta}_e &= \omega - \tfrac{v_r\kappa(s)\cos(\theta_e)}{1-\kappa(s)e}, \end{aligned} \tag{V.10}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

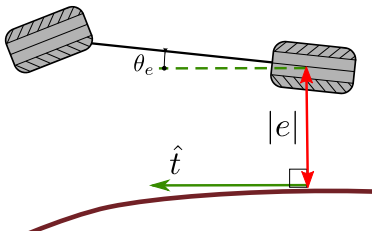


Figure V.3: Feedback variables for the rear wheel based feedback control. θ_e is the difference between the tangent at the nearest point on the path to the rear wheel and the car heading. The magnitude of the scalar value e is illustrated in red. As illustrated $e > 0$, and for the case where the car is to the left of the path, $e < 0$.

where $\kappa(s)$ denotes the curvature of the path at s. The following heading rate command provides local asymptotic convergence to twice continuously differentiable paths:

$$\omega = \frac{v_r \kappa(s) \cos(\theta_e)}{1 - \kappa(s)e} - g_1(e, \theta_e, t)\theta_e - k_2 v_r \frac{\sin(\theta_e)}{\theta_e} e, \quad \text{(V.11)}$$

with $g_1(e, \theta_e, t) > 0$, $k_2 > 0$, and $v_r \neq 0$ which is verified with the Lyapunov function $V(e, \theta_e) = e^2 + \theta_e^2/k_2$ in [56] using the coordinate system (V.10). The requirement that the path be twice differentiable comes from the appearance of the curvature in the feedback law. An advantage of this control law is that stability is unaffected by the sign of v_r making it suitable for reverse driving.

Setting $g_1(v_r, \theta_e, t) = k_\theta |v_r|$ for $k_e > 0$ leads to local exponential convergence with a rate independent of the vehicle speed so long as $v_r \neq 0$. The control law in this case is

$$\omega = \frac{v_r \kappa(s) \cos(\theta_e)}{1 - \kappa(s)e} - (k_\theta |v_r|)\,\theta_e - \left(k_e v_r \frac{\sin(\theta_e)}{\theta_e}\right) e. \quad \text{(V.12)}$$

3) Front wheel position based feedback: This approach was proposed and used in Stanford University's entry to the 2005 DARPA Grand Challenge [55], [8]. The approach is to take the front wheel position as the regulated variable. The control uses the variables $s(t)$, $e(t)$, and $\theta_e(t)$ as in the previous subsections, with the modification that $e(t)$ is computed with the front wheel position as opposed to the rear wheel position. Taking the time derivative of the transverse error reveals

$$\dot{e} = v_f \sin(\theta_e + \delta) \quad \text{(V.13)}$$

The error rate in (V.13) can be directly controlled by the steering angle for error rates with magnitude less than v_f. Solving for the steering angle such that $\dot{e} = -ke$ drives $e(t)$ to zero exponentially fast.

$$\begin{aligned} v_f \sin(\delta + \theta_e) &= -ke \\ \Rightarrow \qquad \delta &= \arcsin(-ke/v_f) - \theta_e. \end{aligned} \quad \text{(V.14)}$$

The term θ_e in this case is not interpreted as heading error since it will be nonzero even with perfect tracking. It is more appropriately interpreted as a combination of a feed-forward term of the nominal steering angle to trace out the reference path and a heading error term.

The drawback to this control law is that it is not defined when $|ke/v_f| > 1$. The exponential convergence over a finite domain can be relaxed to local exponential convergence with the feedback law

$$\delta = \arctan(-ke/v_f) - \theta_e, \quad \text{(V.15)}$$

which, to first order in e, is identical to the previous equation. This is illustrated in Figure V.4.

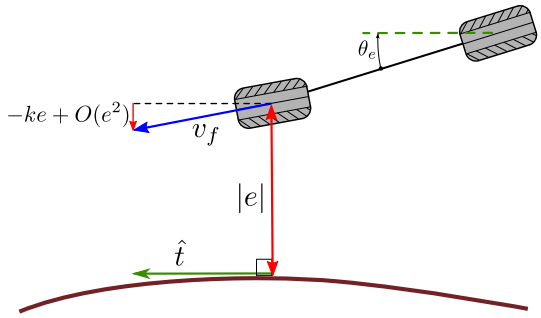


Figure V.4: Front wheel output based control. The control strategy is to point the front wheel towards the path so that the component of the front wheel's velocity normal to the path is proportional to the distance to the path. This is achieved locally and yields local exponential convergence.

Like the control law in (V.14) this controller locally exponentially stabilizes the car to paths with varying curvature with the condition that the path is continuously differentiable. The condition on the path arises from the definition of θ_e in the feedback policy. A drawback to this controller is that it is not stable in reverse making it unsuitable for parking.

Comparison of path tracking controllers for kinematic models: The advantages of controllers based on the kinematic model with the no-slip constraint on the wheels is that they have low computational requirements, are readily implemented, and have good performance at moderate speeds. Figure V.5 provides a qualitative comparison of the path stabilizing controllers of this sections based on, and simulated with, (III.3) for a lane change maneuver. In the simulation of the front wheel output based controller, the rear wheel reference path is replaced by the front wheel reference path satisfying

$$\begin{aligned} x_{ref}(s) &\mapsto x_{ref}(s) + l\cos(\theta), \\ y_{ref}(s) &\mapsto y_{ref}(s) + l\sin(\theta). \end{aligned} \quad \text{(V.16)}$$

The parameters of the simulation are summarized in Table III.

In reference to Figure V.5, the pure pursuit control tracks the reference path during periods with no curvature. In the region where the path has high curvature, the pure pursuit control causes the system to deviate from the reference path. In contrast, the latter two controllers converge to the path and track it through the high curvature regions.

In both controllers using (V.5) in the feedback policy, local exponential stability can only be proven if there is a neighborhood of the path where (V.5) is continuous. Intuitively, this means the path cannot cross over itself and must be differentiable.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

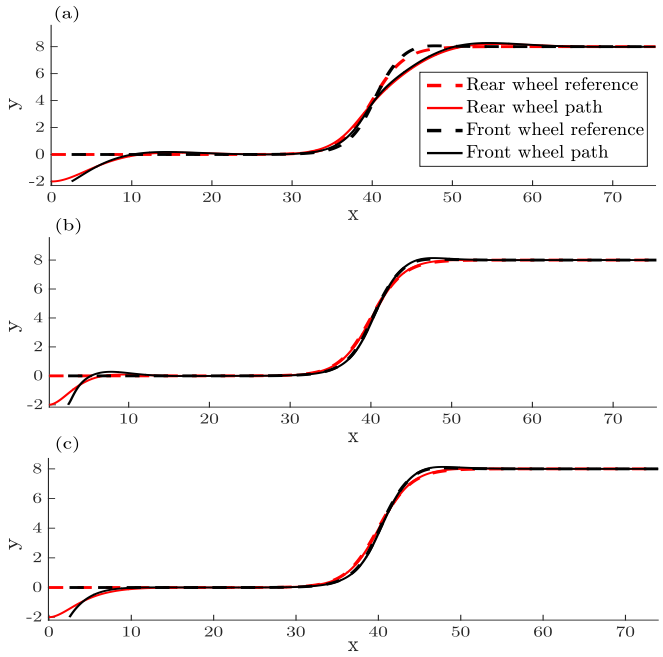


Figure V.5: Tracking performance comparison for the three path stabilizing control laws discussed in this section. (a) Pure pursuit deviates from reference when curvature is nonzero. (b) The rear wheel output based controller drives the rear wheel to the rear wheel reference path. Overshoot is a result of the second order response of the system. (c) The front wheel output based controller drives the front wheel to the reference path with a first order response and tracks the path through the maneuver.

Example Parameters			
Reference path	Wheel-base	Steering limit	Initial Configuration
$\left(x_{ref}(s), y_{ref}(s)\right) = (s, 4 \cdot \tanh\left(\frac{s-40}{4}\right)$	$L = 5$	$\lvert\delta\rvert \leq \pi/4$	$(x_r(0), y_r(0), \theta(0)) = (0, -2, 0)$

Controller Parameters		
Pure pursuit	Rear wheel feedback	Front wheel feedback
$L = 5$, $v_r = 1$	$k_e = 0.25$, $k_\theta = 0.75$, $v_r = 1$	$k = 0.5$, $v_r = 1$

Table III: Simulation and controller parameters used to generate Figure V.5.

B. Trajectory Tracking Control for the Kinematic Model

1) Control Lyapunov based design: A control design based on a control Lyapunov function is described in [132]. The approach is to define the tracking error in a coordinate frame fixed to the car. The configuration error can be expressed by a change of basis from the inertial coordinate frame using the reference trajectory, and velocity, $\left(x_{ref}, y_{ref}, \theta_{ref}, v_{ref}, \omega_{ref}\right)$,

$$\begin{pmatrix} x_e \\ y_e \\ \theta_e \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{ref} - x_r \\ y_{ref} - y_r \\ \theta_{ref} - \theta \end{pmatrix}.$$

The evolution of the configuration error is then

$$\begin{aligned} \dot{x}_e &= \omega y_e - v_r + v_{ref}\cos(\theta_e), \\ \dot{y}_e &= -\omega x_e + v_{ref}\sin(\theta_e), \\ \dot{\theta}_e &= \omega_{ref} - \omega. \end{aligned}$$

With the control assignment,

$$\begin{aligned} v_r &= v_{ref}\cos(\theta_e) + k_1 x_e, \\ \omega &= \omega_{ref} + v_{ref}\left(k_2 y_e + k_3 \sin(\theta_e)\right). \end{aligned}$$

the closed loop error dynamics become

$$\begin{aligned} \dot{x}_e &= (\omega_{ref} + v_{ref}\left(k_2 y_e + k_3 \sin(\theta_e)\right))y_e - k_1 x_e, \\ \dot{y}_e &= -\left(\omega_{ref} + v_{ref}\left(k_2 y_e + k_3 \sin(\theta_e)\right)\right) x_e + v_{ref}\sin(\theta_e), \\ \dot{\theta}_e &= \omega_{ref} - \omega. \end{aligned}$$

Stability is verified for $k_{1,2,3} > 0$, $\dot{\omega}_{ref} = 0$, and $\dot{v}_{ref} = 0$ by the Lyapunov function

$$V = \frac{1}{2}\left(x_e^2 + y_e^2\right) + \frac{(1 - \cos(\theta_e))}{k_2},$$

with negative semi-definite time derivative,

$$\dot{V} = -k_1 x_e^2 - \frac{v_{ref} k_3 \sin^2(\theta_e)}{k_2}.$$

A local analysis shows that the control law provides local exponential stability. However, for the system to be time invariant, ω_{ref} and v_{ref} are required to be constant.

A related controller is proposed in [133] which utilizes a backstepping design to achieve uniform local exponential stability for a finite domain with time varying references.

2) Output feedback linearization: For higher vehicle speeds, it is appropriate to constrain the steering angle to have continuous motion as in (III.8). With the added state, it becomes more difficult to design a controller from simple geometric considerations. A good option in this case is to output-linearize the system.

This is not easily accomplished using the front or rear wheel positions. An output which simplifies the feedback linearization is proposed in [134], where a point ahead of the vehicle by any distance $d \neq 0$, aligned with the steering angle is selected.

Let $x_p = x_f + d\cos(\theta + \delta)$ and $y_p = y_f + d\sin(\theta + \delta)$ be the output of the system. Taking the derivative of these outputs and substituting the dynamics of III.8 yields

$$\begin{pmatrix} \dot{x}_p \\ \dot{y}_p \end{pmatrix} = \underbrace{\begin{pmatrix} \cos(\theta+\delta) - \frac{d}{l}\sin(\theta+\delta)\sin(\delta) & -d\sin(\theta+\delta) \\ \sin(\theta+\delta) + \frac{d}{l}\cos(\theta+\delta)\sin(\delta) & d\cos(\theta+\delta) \end{pmatrix}}_{A(\theta,\delta)} \begin{pmatrix} v_f \\ v_\delta \end{pmatrix}. \tag{V.17}$$

Then, defining the right hand side of (V.17) as auxiliary control variables u_x and u_y yields

$$\begin{pmatrix} \dot{x}_p \\ \dot{y}_p \end{pmatrix} = \begin{pmatrix} u_x \\ u_y \end{pmatrix}, \tag{V.18}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

which makes control straightforward. From u_x and u_y, the original controls v_f and v_δ are recovered by using the inverse of the matrix in (V.17), provided below:

$$[A(\theta,\delta)]^{-1} = \begin{pmatrix} \cos(\theta+\delta) & \sin(\theta+\delta) \\ -\frac{1}{d}\sin(\theta+\delta) - \frac{1}{l}\cos(\theta+\delta)\sin(\delta) & \frac{1}{d}\cos(\theta+\delta) - \frac{1}{l}\sin(\theta+\delta)\sin(\delta) \end{pmatrix}. \tag{V.19}$$

From the input-output linear system, local trajectory stabilization can be accomplished with the controls

$$\begin{aligned} u_x &= \dot{x}_{p,ref} + k_x(x_{p,ref} - x_p), \\ u_y &= \dot{y}_{p,ref} + k_y(y_{p,ref} - y_p). \end{aligned} \tag{V.20}$$

To avoid confusion, note that in this case, the output position (x_p, y_p) and controlled speed v_f are not collocated as in the previously discussed controllers.

C. Predictive Control Approaches

The simple control laws discussed above are suitable for moderate driving conditions. However, a higher fidelity model may be required to plan and execute aggressive or emergency maneuvers. The added detail of more sophisticated models complicates the control design making it difficult to construct controllers from intuition and geometry of the configuration space.

Model predictive control [135] is a general control design methodology which can be very effective for this problem. Conceptually, the approach is to solve the motion planning problem over a short time horizon, take a short interval of the resulting open loop control, and apply it to the system. While executing, the motion planning problem is re-solved to find an appropriate control for the next time interval. Advances in computing hardware as well as mathematical programming algorithms have made predictive control feasible for real-time use in driverless vehicles. MPC is a major field of research on its own and this section is only intended to provide a brief description of the technique and to survey results on its application to driverless vehicle control.

Since model predictive control is a very general control technique, the model takes the form of a general continuous time control system with control, $u(t) \in \mathbb{R}^m$, and state, $x(t) \in \mathbb{R}^n$,

$$\dot{x} = f(x,u,t) \tag{V.21}$$

A feasible reference trajectory $x_{ref}(t)$, and for some motion planners $u_{ref}(t)$, are provided satisfying (V.21). The system is then discretized by an appropriate choice of numerical approximation so that (V.21) is given at discrete time instances by

$$x_{k+1} = F_k(x_k, u_k), \quad k \in \mathbb{N}, \tag{V.22}$$

To avoid over-complicating the following discussion, we assume the discretization is exact for the remainder of the section. The control law typically takes the form

$$\begin{aligned} u_k(x_{meas.}) = \underset{\substack{x_n \in \mathcal{X}_n, \\ u_n \in \mathcal{U}_n}}{\operatorname{argmin}} \Bigg\{ & h(x_N - x_{ref,N}, u_N - u_{ref,N}) \\ & + \sum_{n=k}^{k+N-1} g_n(x_n - x_{ref,n}, u_n - u_{ref,n}) \Bigg\} \end{aligned} \tag{V.23}$$

$$\begin{aligned} &\text{subject to} \\ &x_k = x_{meas.} \\ &x_{n+1} = F(x_n, u_n), \\ &n \in \{k, ..., k+N-1\}. \end{aligned}$$

The function g_n penalizes deviations from the reference trajectory and control at each time step, while the function h is a terminal penalty at the end of the time horizon. The set $\mathcal{X}_n$ is the set of allowable states which can restrict undesirable positions or velocities, e.g., excessive tire slip or obstacles. The set $\mathcal{U}_n$ encodes limits on the magnitude of the input signals. Important considerations are whether the solutions to the right hand side of (V.23) exist, and when they do, the stability and robustness of the closed loop system. These issues are investigated in the predictive control literature [136], [137].

To implement an MPC on a driverless car, (V.23) must be solved several times per second which is a major obstacle to its use. In the special case that h and g_n are quadratic, $\mathcal{U}_n$ and $\mathcal{X}_n$ are polyhedral, and F is linear, the problem becomes a quadratic program. Unlike a general nonlinear programming formulation, interior point algorithms are available for solving quadratic programs in polynomial time. To leverage this, the complex vehicle model is often linearized to obtain an approximate linear model. Linearization approaches typically differ in the reference about which the linearization is computed—current operating point [138]–[140], reference path [141] or more generally, about a reference trajectory, which results in the following approximate linear model:

$$\begin{aligned} x_{ref,k+1} + \xi_{k+1} = {} & F_k(x_{ref,k}, u_{ref,k}) \\ & + \underbrace{\nabla_x F_k(x_{ref,k}, u_{ref,k})}_{A_k}\xi_k + \underbrace{\nabla_u F_k(x_{ref,k}, u_{ref,k})}_{B_k}\eta_k \\ & + O(\|\xi_k\|^2) + O(\|\eta_k\|^2), \end{aligned} \tag{V.24}$$

where $\xi := x - x_{ref}$ and $\eta := u - u_{ref}$ are the deviations of the state and control from the reference trajectory. This first order expansion of the perturbation dynamics yields a linear time varying (LTV) system,

$$\xi_{k+1} = A_k\xi_k + B_k\eta_k. \tag{V.25}$$

Then using a quadratic objective, and expressing the polyhedral constraints algebraically, we obtain

$$\begin{aligned} u_k(x_{meas.}) = \underset{\xi_k, \eta_k}{\operatorname{argmin}} \Bigg\{ & \xi_N^T H \xi_N + \sum_{n=k}^{k+N-1} \xi_k^T Q_k \xi_k + \eta_k^T R_k \eta_k \Bigg\} \\ & \text{subject to} \\ & \xi_k = x_{meas.} - x_{ref} \\ & C_n\xi_n \le 0, \quad D_n\eta_n \le 0 \\ & \xi_{k+1} = A_k\xi_k + B_n\eta_n, \; n \in \{k, ..., k+N-1\}, \end{aligned} \tag{V.26}$$

where R_k and Q_k are positive semi-definite. If the states and inputs are unconstrained, i.e., $\mathcal{U}_n = \mathbb{R}^m$, $\mathcal{X}_n = \mathbb{R}^n$, a semi-closed form solution can be obtained by dynamic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIV.2016.2578706, IEEE Transactions on Intelligent Vehicles

programming requiring only the calculation of an N step matrix recursion [142]. Similar closed form recursive solutions have also been explored when vehicle models are represented by controlled auto-regressive integrated moving average (CARIMA) with no state and input constraints [143], [144].

A further variation in the model predictive control approach is to replace state constraints (e.g., obstacles) and input constraints by penalty functions in the performance functional. Such a predictive control approach based on a dynamic model with nonlinear tire behavior is presented in [145]. In addition to penalizing control effort and deviation from a reference path to a goal which does not consider obstacles, the performance functional penalizes input constraint violations and collisions with obstacles over the finite control horizon. In this sense it is similar to potential field based motion planning, but is demonstrated to have improved performance.

The following are some variations of the model predictive control framework that are found in the literature of car controllers:

1) Unconstrained MPC with Kinematic Models: The earliest predictive controller in [143] falls under this category, in which the model predictive control framework is applied without input or state constraints using a CARIMA model. The resulting semi-closed form solution has minimal computational requirements and has also been adopted in [144]. Moreover, the time-varying linear quadratic programming approach with no input or state constraints was considered in [144] using a linearized kinematic model.

2) Path Tracking Controllers: In [141], a predictive control is investigated using a center of mass based linear dynamic model (assuming constant velocity) for path tracking and an approximate steering model. The resulting integrated model is validated with a detailed automatic steering model and a 27 degree-of-freedom CarSim vehicle model.

3) Trajectory Tracking Controllers: A predictive controller using a dynamic model with nonlinear tire behavior was investigated in [138]. The full nonlinear predictive control strategy was carried out in simulation and shown to stabilize a simulated emergency maneuver in icy conditions with a control frequency of 20 Hz. However, with a control horizon of just two time steps the computation time was three times the sample time of the controller making experimental validation impossible. A linearization based approach was also investigated in [138]–[140] based on a single linearization about the state of the vehicle at the current time step. The reduced complexity of solving the quadratic program resulted in acceptable computation time, and successful experimental results are reported for driving in icy conditions at speeds up to 21m/s. The promising simulation and experimental results of this approach are improved upon in [140] by providing conditions for the uniform local-asymptotic stability of the time varying system.

D. Linear Parameter Varying Controllers

Many controller design techniques are available for linear systems making a linear model desirable. However, the broad range of operating points encountered under normal driving conditions make it difficult to rely on a model linearized about a single operating point. To illustrate this, consider the lateral error dynamics in (V.10). If the tracking error is assumed to remain small a linearization of the dynamics around the operating point $\theta_e = 0$ and $e = 0$ yields

$$\begin{pmatrix} \dot{e} \\ \dot{\theta}_e \end{pmatrix} = \begin{pmatrix} 0 & v_r \\ 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ \theta_e \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \omega + \begin{pmatrix} 0 \\ -v_r \kappa(s) \end{pmatrix}. \tag{V.27}$$

Introducing a new control variable incorporating a feedforward $u = \omega + v_r \kappa(s)$ simplifies the discussion. The dynamics are now

$$\begin{pmatrix} \dot{e} \\ \dot{\theta}_e \end{pmatrix} = \begin{pmatrix} 0 & v_r \\ 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ \theta_e \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u. \tag{V.28}$$

Observe that the model is indeed linear, but the forward speed v_r appears in the linear model. A simple proportional plus derivative control with gains k_p and k_d will stabilize the lateral dynamics but the poles of the closed loop system are given by $\left(-k_d \pm \sqrt{k_d^2 - 4k_p v_r}\right)/2$. At higher speeds the poles move into the complex plane leading to an oscillatory response. In contrast, a small k_p gain leads to a poor response at low speed. A very intuitive and widely used remedy to this challenge is gain scheduling. In this example, parameterizing k_p as a function of v_r fixes the poles to a single value for each speed. This technique falls into the category of control design for linear parameter varying (LPV) models [146]. Gain scheduling is a classical approach to this type of controller design. Tools from robust control and convex optimization are readily applied to address more complex models.

LPV control designs for lateral control are presented in [147]–[149]. LPV models are used in [150], [151] together with predictive control approaches for path and trajectory stabilization. At a lower level of automation, LPV control techniques have been proposed for integrated system control. In these designs, several subsystems are combined under a single controller to achieve improved handling performance. LPV control strategies for actuating active and semi-active suspension systems are developed in [152], [153], while integrated suspension and braking control systems are developed in [154], [155].

VI. Conclusions

The past three decades have seen increasingly rapid progress in driverless vehicle technology. In addition to the advances in computing and perception hardware, this rapid progress has been enabled by major theoretical progress in the computational aspects of mobile robot motion planning and feedback control theory. Research efforts have undoubtedly been spurred by the improved utilization and safety of road networks that driverless vehicles would provide.

This paper has provided a survey of the various aspects of driverless vehicle decision making problems with a focus on motion planning and feedback control. The aim of the discussion was to assist with gaining a system level understanding and to aid design decisions.

Driverless vehicles are complex systems which have been broken down into a hierarchy of decision making problems, where the solution of one problem is the input to the next. The breakdown into individual decision making problems has enabled the use of well developed solution techniques from a variety of research areas. However, one area which has received little attention is on tailoring and integrating these methods so that their interactions are semantically valid. One example of this is the reference produced by the motion planner must meet the assumptions used to prove the performance of the feedback controller so that the joint process can be relied on to function with provable performance. If the stability of the feedback controller relies on a level of smoothness of the reference path, and the motion planner outputs paths which are not sufficiently smooth, the performance of the system may be unsatisfactory. Another issue that arises with this breakdown into subproblems is that of the computational burden of the entire system. For example, a more efficient motion planning algorithm may only be compatible with a computationally intensive feedback controller such as model predictive control. Conversely, a simple control law may require less computation to execute, but is also less robust and requires using a more detailed model for motion planning. While it will be important to resolve these issues, they are not limiting factors preventing driverless vehicles from becoming a means of personal mobility.

REFERENCES

[1] "2014 crash data key findings." National Highway Traffic Safety Administration, Report No. DOT HS 812 219, 2014.

[2] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey." National Highway Traffic Safety Administration, Report No. DOT HS 812 115, 2014.

[3] B. McKenzie and M. Rapino, "Commuting in the United States: 2009, American Community Survey Reports," *U.S. Census Bureau*, 2011. ACS-15, 2011.

[4] D. A. Hennessy and D. L. Wiesenthal, "Traffic congestion, driver stress, and driver aggression," *Aggressive behavior*, pp. 409–423, 1999.

[5] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine vision and applications*, vol. 1, pp. 223–240, 1988.

[6] E. D. Dickmanns *et al.*, "Vehicles capable of dynamic vision," in *IJCAI*, pp. 1577–1592, 1997.

[7] "No Hands Across America Journal." http://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/Journal.html. Accessed: 2015-12-14.

[8] M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA Grand Challenge: The great robot race*, vol. 36. Springer Science & Business Media, 2007.

[9] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous vehicles in city traffic*, vol. 56. springer, 2009.

[10] J. Xin, C. Wang, Z. Zhang, and N. Zheng, "China future challenge: Beyond the intelligent vehicle," *IEEE Intell. Transp. Syst. Soc. Newslett*, vol. 16, pp. 8–10, 2014.

[11] P. Cerri, G. Soprani, P. Zani, J. Choi, J. Lee, D. Kim, K. Yi, and A. Broggi, "Computer vision at the Hyundai autonomous challenge," in *International Conference on Intelligent Transportation Systems*, pp. 777–783, IEEE, 2011.

[12] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta, "The vislab intercontinental autonomous challenge: an extensive test for a platoon of intelligent vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 10, pp. 147–164, 2012.

[13] A. Broogi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti, "Proud-public road urban driverless-car test," *Transactions on Intelligent Transportation Systems*, vol. 16, pp. 3508–3519, 2015.

[14] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making Bertha drive—an autonomous journey on a historic route," *Intelligent Transportation Systems Magazine*, vol. 6, pp. 8–20, 2014.

[15] "Google Self-Driving Car Project." https://www.google.com/selfdrivingcar/. Accessed: 2015-12-14.

[16] "Tesla Motors: Model S Press Kit." https://www.teslamotors.com/presskit/autopilot. Accessed: 2016-3-15.

[17] S. O.-R. A. V. S. Committee *et al.*, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," 2014.

[18] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[19] J. C. Gerdes and E. J. Rossetter, "A unified approach to driver assistance systems based on artificial potential fields," *Journal of Dynamic Systems, Measurement, and Control*, vol. 123, pp. 431–438, 2001.

[20] M. Brännström, E. Coelingh, and J. Sjöberg, "Model-based threat assessment for avoiding arbitrary vehicle collisions," *Transactions on Intelligent Transportation Systems*, vol. 11, pp. 658–669, 2010.

[21] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *Transactions on Intelligent Transportation Systems*, vol. 4, pp. 143–153, 2003.

[22] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1162–1175, 2013.

[23] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pp. 145–154, ACM, 2012.

[24] H. Kowshik, D. Caveney, and P. Kumar, "Provable systemwide safety in intelligent intersections," *Vehicular Technology, IEEE Transactions on*, vol. 60, pp. 804–818, 2011.

[25] A. Eskandarian, *Handbook of intelligent vehicles*. Springer London, UK, 2012.

[26] D. Miculescu and S. Karaman, "Polling-systems-based control of high-performance provably-safe autonomous intersections," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 1417–1423, IEEE, 2014.

[27] F. Zhou, X. Li, and J. Ma, "Parsimonious shooting heuristic for trajectory control of connected automated traffic part I: Theoretical analysis with generalized time geography," *arXiv preprint arXiv:1511.04810*, 2015.

[28] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *Transactions on Pattern Analysis & Machine Intelligence*, pp. 1239–1258, 2009.

[29] G. Ros, A. Sappa, D. Ponsa, and A. M. Lopez, "Visual SLAM for driverless cars: A brief survey," in *Intelligent Vehicles Symposium (IV) Workshops*, 2012.

[30] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 963–968, IEEE, 2011.

[31] P. Liu, A. Kurt, K. Redmill, and U. Ozguner, "Classification of highway lane change behavior to detect dangerous cut-in maneuvers," in *The Transportation Research Board (TRB) 95th Annual Meeting*, 2015.

[32] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, pp. 269–271, 1959.

[33] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," tech. rep., DTIC Document, 1969.

[34] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A search meets graph theory," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 156–165, Society for Industrial and Applied Mathematics, 2005.

[35] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter, "Exact routing in large road networks using contraction hierarchies," *Transportation Science*, vol. 46, pp. 388–404, 2012.

[36] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, "Route planning in transportation networks," *arXiv preprint arXiv:1504.05140*, 2015.

[37] F. Havlak and M. Campbell, "Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments," *Transactions on Robotics*, vol. 30, pp. 461–474, 2014.

[38] Q. Tran and J. Firl, "Modelling of traffic situations at urban intersections with probabilistic non-parametric regression," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 334–339, IEEE, 2013.

[39] A. C. Madrigal, "The trick that makes Google's self-driving cars work," *The Atlantic*, 2015. http://www.theatlantic.com/technology/archive/2014/05/all-the-world-a-track-the-trick-that-makes-googles-self-driving-cars-work/370871/.

[40] R. Verma and D. D. Vecchio, "Semiautonomous multivehicle safety," *Robotics & Automation Magazine, IEEE*, vol. 18, pp. 44–54, 2011.

[41] S. Z. Yong, M. Zhu, and E. Frazzoli, "Generalized innovation and inference algorithms for hidden mode switched linear stochastic systems with unknown inputs," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 3388–3394, IEEE, 2014.

[42] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic MDP-behavior planning for cars," in *14th International Conference on Intelligent Transportation Systems*, pp. 1537–1542, IEEE, 2011.

[43] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *16th International Conference on Intelligent Transportation Systems*, pp. 2063–2067, IEEE, 2013.

[44] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *17th International Conference on Intelligent Transportation Systems*, pp. 392–399, IEEE, 2014.

[45] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction," in *Proceedings of Robotics: Science & Systems Conference*, p. 2, 2015.

[46] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic Foundations of Robotics X*, pp. 475–491, Springer, 2013.

[47] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[48] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot motion planning and control*, pp. 171–253, Springer, 1998.

[49] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *Transactions on Automatic Control*, vol. 38, pp. 700–716, 1993.

[50] T. Fraichard and R. Mermond, "Path planning with uncertainty for car-like robots," in *International Conference on Robotics and Automation*, vol. 1, pp. 27–32, IEEE, 1998.

[51] M. Egerstedt, X. Hu, H. Rehbinder, and A. Stotsky, "Path planning and robust tracking for a car-like robot," in *Proceedings of the 5th symposium on intelligent robotic systems*, pp. 237–243, Citeseer, 1997.

[52] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," tech. rep., DTIC Document, 1992.

[53] H. Goldstein, *Classical mechanics*. Pearson Education India, 1965.

[54] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Transactions on Control Systems Technology*, vol. 17, pp. 1105–1118, 2009.

[55] M. D. Ventures, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of field Robotics*, vol. 23, pp. 661–692, 2006.

[56] C. Samson, "Path following and time-varying feedback stabilization of a wheeled mobile robot," in *2nd Int. Conf. on Automation, Robotics and Computer Vision*, 1992.

[57] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[58] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, pp. 367–393, 1990.

[59] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," tech. rep., SAE Technical Paper, 1987.

[60] E. Velenis and P. Tsiotras, "Minimum time vs maximum exit velocity path optimization during cornering," in *International symposium on industrial electronics*, pp. 355–360, 2005.

[61] S. C. Peters, E. Frazzoli, and K. Iagnemma, "Differential flatness of a front-steered vehicle with tire force control," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 298–304, IEEE, 2011.

[62] A. Rucco, G. Notarstefano, and J. Hauser, "Computing minimum lap-time trajectories for a single-track car with load transfer," in *Conference on Decision and Control*, pp. 6321–6326, IEEE, 2012.

[63] J. H. Reif, "Complexity of the mover's problem and generalizations," in *20th Annual Symposium on Foundations of Computer Science*, SFCS '79, (Washington, DC, USA), pp. 421–427, IEEE, 1979.

[64] M. H. Overmars and E. Welzl, "New methods for computing visibility graphs," in *Proceedings of the fourth annual symposium on Computational geometry*, pp. 164–171, ACM, 1988.

[65] S. Lazard, J. Reif, and H. Wang, "The complexity of the two dimensional curvatureconstrained shortest-path problem," in *Proceedings of the Third International Workshop on the Algorithmic Foundations of Robotics*, pp. 49–57, 1998.

[66] F. Lamiraux, E. Ferre, and E. Vallee, "Kinodynamic motion planning: Connecting exploration trees using trajectory optimization methods," in *International Conference on Robotics and Automation*, vol. 4, pp. 3987–3992, IEEE, 2004.

[67] F. Boyer and F. Lamiraux, "Trajectory deformation applied to kinodynamic motion planning for a realistic car model," in *International Conference on Robotics and Automation*, pp. 487–492, IEEE, 2006.

[68] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *28th Symposium on Foundations of Computer Science*, pp. 49–60, IEEE, 1987.

[69] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.

[70] D. P. Bertsekas, *Nonlinear programming*. Athena scientific, 1999.

[71] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM (JACM)*, vol. 34, pp. 596–615, 1987.

[72] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *Transactions on Robotics and Automation*, vol. 14, pp. 166–171, 1998.

[73] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.

[74] E. Schmerling, L. Janson, and M. Pavone, "Optimal sampling-based motion planning under differential constraints: the driftless case," *International Conference on Robotics and Automation*, 2015.

[75] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *International Conference on Robotics and Automation*, pp. 5041–5047, IEEE, 2013.

[76] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, pp. 378–400, 2001.

[77] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," in *Algorithmic Foundations of Robotics XI*, pp. 263–282, Springer, 2015.

[78] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," *Journal of the ACM*, vol. 41, pp. 764–790, 1994.

[79] T. Fraichard, "Trajectory planning in a dynamic workspace: a'state-time space'approach," *Advanced Robotics*, vol. 13, pp. 75–94, 1998.

[80] J. Kasac, J. Deur, B. Novakovic, I. Kolmanovsky, and F. Assadian, "A conjugate gradient-based BPTT-like optimal control algorithm with vehicle dynamics control application," *Transactions on Control Systems Technology*, vol. 19, pp. 1587–1595, 2011.

[81] C. L. Darby, W. W. Hager, and A. V. Rao, "An hp-adaptive pseudospectral method for solving optimal control problems," *Optimal Control Applications and Methods*, vol. 32, pp. 476–502, 2011.

[82] L. S. Pontryagin, *Mathematical theory of optimal processes*. CRC Press, 1987.

[83] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *International Conference on Robotics and Automation*, pp. 1168–1175, IEEE, 2014.

[84] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, pp. 193–207, 1998.

[85] E. Polak, "An historical survey of computational methods in optimal control," *SIAM Review*, vol. 15, pp. 553–584, 1973.

[86] B. Chazelle, "Approximation and decomposition of shapes," *Advances in Robotics*, vol. 1, pp. 145–185, 1987.

[87] C. Ó'Dúnlaing and C. K. Yap, "A "retraction" method for planning the motion of a disc," *Journal of Algorithms*, vol. 6, pp. 104–111, 1985.

[88] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized voronoi diagrams," *Transactions on Robotics and Automation*, vol. 5, pp. 143–150, 1989.

[89] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.

[90] J. T. Schwartz and M. Sharir, "On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds," *Advances in applied Mathematics*, vol. 4, pp. 298–351, 1983.

[91] J. Backer and D. Kirkpatrick, "Finding curvature-constrained paths that avoid polygonal obstacles," in *Proceedings of the twenty-third annual symposium on Computational geometry*, pp. 66–73, ACM, 2007.

[92] P. Jacobs and J. Canny, "Planning smooth paths for mobile robots," in *Nonholonomic Motion Planning*, pp. 271–342, Springer, 1993.

[93] H. Wang and P. K. Agarwal, "Approximation algorithms for curvature-constrained shortest paths.," in *SODA*, vol. 96, pp. 409–418, 1996.

[94] S. M. La Valle, "Rapidly-exploring random trees a new tool for path planning," tech. rep., Computer Science Dept., Iowa State University, 1998.

[95] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 2210–2215, IEEE, 2005.

[96] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Hybrid Systems: Computation and Control: 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004. Proceedings* (R. Alur and G. J. Pappas, eds.), pp. 142–156, Springer Berlin Heidelberg, 2004.

[97] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 5021–5028, IEEE, 2010.

[98] J. H. Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *American Control Conference*, pp. 188–193, IEEE, 2013.

[99] S. Fleury, P. Soueres, J.-P. Laumond, and R. Chatila, "Primitives for smoothing mobile robot trajectories," *Transactions on Robotics and Automation*, vol. 11, pp. 441–448, 1995.

[100] E. Velenis, P. Tsiotras, and J. Lu, "Aggressive maneuvers on loose surfaces: Data analysis and input parametrization," in *Mediterranean Conference on Control Automation*, pp. 1–6, 2007.

[101] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, pp. 308–333, 2009.

[102] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, pp. 673–692, 2004.

[103] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, *Real-time robot motion planning using rasterizing computer graphics hardware*, vol. 24. ACM, 1990.

[104] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

[105] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, 2015.

[106] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[107] I. Pohl, "First results on the effect of error in heuristic search," *Machine Intelligence*, pp. 219–236, 1970.

[108] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *International Conference Robotics and Automation*, pp. 3310–3317, IEEE, 1994.

[109] S. Anthony, "The focussed D* algorithm for real-time replanning.," in *IJCAI*, vol. 95, pp. 1652–1659, 1995.

[110] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *Transactions on Robotics*, vol. 21, pp. 354–363, 2005.

[111] E. A. Hansen and R. Zhou, "Anytime heuristic search.," *J. Artif. Intell. Res.(JAIR)*, vol. 28, pp. 267–297, 2007.

[112] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems*, p. None, 2003.

[113] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm.," in *ICAPS*, pp. 262–271, 2005.

[114] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *International Conference on Robotics and Automation*, vol. 3, pp. 2719–2726, IEEE, 1997.

[115] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, pp. 233–255, 2002.

[116] T. Kunz and M. Stilman, "Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete," in *Algorithmic Foundations of Robotics XI*, pp. 233–244, Springer, 2015.

[117] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Conference on Decision and Control*, pp. 7681–7687, IEEE, 2010.

[118] A. Perez, R. Platt Jr, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *International Conference on Robotics and Automation*, pp. 2537–2542, IEEE, 2012.

[119] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *International Conference on Robotics and Automation*, pp. 5054–5061, IEEE, 2013.

[120] M. Otte and E. Frazzoli, "RRT-X: Real-time motion planning/replanning for environments with unpredictable obstacles," in *International Workshop on the Algorithmic Foundations of Robotics*, 2014.

[121] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, pp. 425–466, 2008.

[122] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, p. 48105, 2008.

[123] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, *et al.*, "Odin: Team victorTango's entry in the DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, pp. 467–492, 2008.

[124] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, pp. 727–774, 2008.

[125] J. P. Hespanha *et al.*, "Trajectory-tracking and path-following of under-actuated autonomous vehicles with parametric modeling uncertainty," *Transactions on Automatic Control*, vol. 52, pp. 1362–1379, 2007.

[126] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotović, "Path-following for nonminimum phase systems removes performance limitations," *Transactions on Automatic Control*, vol. 50, pp. 234–239, 2005.

[127] H. K. Khalil, *Nonlinear systems*, vol. 3. Prentice hall New Jersey, 1996.

[128] R. Wallace, A. Stentz, C. E. Thorpe, H. Maravec, W. Whittaker, and T. Kanade, "First results in robot road-following.," in *IJCAI*, pp. 1089–1095, 1985.

[129] O. Amidi and C. E. Thorpe, "Integrated mobile robot control," in *Fibers' 91, Boston, MA*, pp. 504–523, International Society for Optics and Photonics, 1991.

[130] A. L. Rankin, C. D. Crane III, D. G. Armstrong II, A. D. Nease, and H. E. Brown, "Autonomous path-planning navigation system for site characterization," in *Aerospace/Defense Sensing and Controls*, pp. 176–186, International Society for Optics and Photonics, 1996.

[131] J. Wit, C. D. Crane, and D. Armstrong, "Autonomous ground vehicle path tracking," *Journal of Robotic Systems*, vol. 21, pp. 439–449, 2004.

[132] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *International Conference on Robotics and Automation*, pp. 384–389, IEEE, 1990.

[133] Z.-P. Jiang and H. Nijmeijer, "Tracking control of mobile robots: a case study in backstepping," *Automatica*, vol. 33, pp. 1393–1399, 1997.

[134] B. d'Andréa Novel, G. Campion, and G. Bastin, "Control of nonholonomic wheeled mobile robots by state feedback linearization," *The International journal of robotics research*, vol. 14, pp. 543–559, 1995.

[135] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice-a survey," *Automatica*, vol. 25, pp. 335–348, 1989.

[136] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[137] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

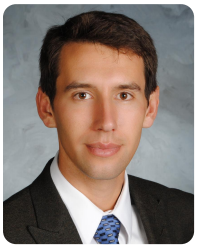
[138] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *Transactions on Control Systems Technology*, vol. 15, pp. 566–580, 2007.

[139] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *46th Conference on Decision and Control*, pp. 2980–2985, IEEE, 2007.

[140] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," *International journal of robust and nonlinear control*, vol. 18, pp. 862–875, 2008.

[141] E. Kim, J. Kim, and M. Sunwoo, "Model predictive control strategy for smooth path tracking of autonomous vehicles with steering actuator

dynamics," *International Journal of Automotive Technology*, vol. 15, pp. 1155–1164, 2014.
[142] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2012.
[143] A. Ollero and O. Amidi, "Predictive path tracking of mobile robots. application to the CMU Navlab," in *5th International Conference on Advanced Robotics*, vol. 91, pp. 1081–1086, 1991.
[144] G. V. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, and L. B. Becker, "A predictive controller for autonomous vehicle path tracking," *Transactions on Intelligent Transportation Systems*, vol. 10, pp. 92–102, 2009.
[145] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles," *Control Engineering Practice*, vol. 17, pp. 741–750, 2009.
[146] O. Sename, P. Gaspar, and J. Bokor, *Robust control and linear parameter varying approaches: application to vehicle dynamics*, vol. 437. Springer, 2013.
[147] P. Gáspár, Z. Szabó, and J. Bokor, "LPV design of fault-tolerant control for road vehicles," *International Journal of Applied Mathematics and Computer Science*, vol. 22, pp. 173–182, 2012.
[148] J. Huang and M. Tomizuka, "LTV controller design for vehicle lateral control under fault in rear sensors," *Transactions on Mechatronics*, vol. 10, pp. 1–7, 2005.
[149] S. Ç. Baslamisli, İ. E. Köse, and G. Anlaş, "Gain-scheduled integrated active steering and differential control for vehicle handling improvement," *Vehicle System Dynamics*, vol. 47, pp. 99–119, 2009.
[150] T. Besselmann and M. Morari, "Autonomous vehicle steering using explicit LPV-MPC," in *Control Conference (ECC), 2009 European*, pp. 2628–2633, IEEE, 2009.
[151] T. Besselmann, P. Rostalski, and M. Morari, "Hybrid parameter-varying model predictive control for lateral vehicle stabilization," in *European Control Conference*, pp. 1068–1075, IEEE, 2007.
[152] P. Gáspár, Z. Szabó, and J. Bokor, "The design of an integrated control system in heavy vehicles based on an LPV method," in *Conference on Decision and Control*, pp. 6722–6727, IEEE, 2005.
[153] C. Poussot-Vassal, O. Sename, L. Dugard, P. Gaspar, Z. Szabo, and J. Bokor, "A new semi-active suspension control strategy through LPV technique," *Control Engineering Practice*, vol. 16, pp. 1519–1534, 2008.
[154] M. Doumiati, O. Sename, L. Dugard, J.-J. Martinez-Molina, P. Gaspar, and Z. Szabo, "Integrated vehicle dynamics control via coordination of active front steering and rear braking," *European Journal of Control*, vol. 19, pp. 121–143, 2013.
[155] P. Gaspar, Z. Szabo, J. Bokor, C. Poussot-Vassal, O. Sename, and L. Dugard, "Toward global chassis control by integrating the brake and suspension systems," in *5th IFAC Symposium on Advances in Automotive Control, IFAC AAC 2007*, p. 6, IFAC, 2007.



Brian Paden received B.S. and M.S. degrees in Mechanical Engineering in 2011 and 2013 respectively from UC Santa Barbara. Concurrently from 2011 to 2013 he was an Engineer at LaunchPoint Technologies developing electronically controlled automotive valve-train systems. His research interests are in the areas of control theory and motion planning with a focus on applications for autonomous vehicles. His current affiliation is with the Laboratory for Information and Decision systems and he is a Doctoral Candidate in the department of Mechanical Engineering at MIT.

Michal Čáp received his Bc. degree in Information Technology from Brno University of Technology, the Czech Republic and MSc degree in Agent Technologies from Utrecht University, the Netherlands. He is currently pursuing PhD degree in Artificial Intelligence at CTU in Prague, Czech Republic. At the moment, Michal Cap holds the position of Fulbright-funded visiting researcher at Massachusetts Institute of Technology, USA. His research interests include motion planning, multi-robot trajectory coordination and autonomous transportation systems in general.

Sze Zheng Yong is currently a postdoctoral associate in the Laboratory for Information and Decision Systems at Massachusetts Institute of Technology. He obtained his Dipl.-Ing.(FH) degree in automotive engineering with a specialization in mechatronics and control systems from the Esslingen University of Applied Sciences, Germany in 2008 and his S.M. and Ph.D. degrees in mechanical engineering from MIT in 2010 and 2016, respectively. His research interests lie in the broad area of control and estimation of hidden mode hybrid systems, with applications to intention-aware autonomous systems and resilient cyber-physical systems.

Dmitry Yershov was born in Kharkov, Ukraine, in 1983. He received his B.S. and M.S. degree in applied mathematics from Kharkov National University in 2004 and 2005, respectively. In 2013, he graduated with a Ph.D. degree from the the Department of Computer Science at the University of Illinois at Urbana-Champaign. He joined the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology in years 2013–2015 where he was a postdoctoral associate. Currently, he is with nuTonomy inc. His research is focused primarily on the feedback planning problem in robotics, which extends to planning in information spaces and planning under uncertainties.

Emilio Frazzoli is a Professor of Aeronautics and Astronautics with the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. He received a Laurea degree in Aerospace Engineering from the University of Rome, "Sapienza" , Italy, in 1994, and a Ph. D. degree from the Department of Aeronautics and Astronautics of the Massachusetts Institute of Technology, in 2001. Before returning to MIT in 2006, he held faculty positions at the University of Illinois, Urbana-Champaign, and at the University of California, Los Angeles. He was the recipient of a NSF CAREER award in 2002, and of the IEEE George S. Axelby award in 2015. His current research interests focus primarily on autonomous vehicles, mobile robotics, and transportation systems.