Energy-Optimal Motion Planning for Multiple Robotic Vehicles with Collision Avoidance

Andreas J. Häusler, *Member, IEEE*, Alessandro Saccon, *Member, IEEE*, A. Pedro Aguiar, *Member, IEEE*, John Hauser, *Member, IEEE*, António M. Pascoal, *Member, IEEE*

***Abstract*—We propose a numerical algorithm for multiple vehicle motion planning that takes explicitly into account the vehicle dynamics, temporal and spatial specifications, and energy-related requirements. As a motivating example, we consider the case where a group of vehicles is tasked to reach a number of target points at the same time (*simultaneous arrival problem*) without colliding among themselves and with obstacles, subject to the requirement that the overall energy required for vehicle motion be minimized. With the theoretical setup adopted, the *vehicle dynamics are taken explicitly into account* at the planning level.**

The paper formulates the problem of multiple vehicle motion planning in a rigorous mathematical setting, describes the optimization algorithm used to solve it, and discusses key implementation details. The efficacy of the method is illustrated through numerical examples for the simultaneous arrival problem. The initial guess to start the optimization procedure is obtained from simple geometrical considerations, e.g., by joining the desired initial and final positions of the vehicles via straight lines. Even though the initial trajectories thus obtained may result in inter-vehicle and vehicle/obstacle collisions, we show that the optimization procedure that we employ in this paper will generate collision-free trajectories that also minimize the overall energy spent by each vehicle and meet the required temporal and spatial constraints.

The method developed applies to a very general class of vehicles; however, for clarity of exposition we adopt as an illustrative example the case of wheeled robots.

I. Introduction

MOTION PLANNING for autonomous vehicles in the presence of geometric, temporal, and energy-related constraints is a challenging problem that is at the core of many real applications. The rapid development of the field of robotics in the past decade, going from single robot tasks to missions that require coordination, cooperation, and communication among a number of networked vehicles makes the availability of versatile motion planners increasingly important. Ground [20], aerial [22], marine [10], and space robotics [12] alike illustrate the large spectrum of possible application domains that call for the use of reliable motion planners aimed at optimizing the performance that can be achieved in real scenarios, either in terms of reducing maneuvering time or minimizing energy consumption, or an appropriate combination thereof. The increasing sophistication of groups of autonomous vehicles that are tasked to carry out extremely challenging missions in cooperation, often in hazardous and complex non-structured environments, add to the complexity of the supporting mission planning systems, which must address explicitly inter-vehicle and vehicle-environment collision avoidance. For these reasons, cooperative motion planning continues to pose formidable challenges to system designers.

A. Problem Setting

It is envisioned that, in the not-too-distant future, groups of autonomous vehicles equipped with advanced sensor suites will roam the environment to acquire data at an unprecedented scale, detect and monitor episodic events, and inspect critical infrastructures on an almost permanent basis. Meeting these objectives under harsh operation conditions requires careful planning with a view to proper mission execution under the constraint of limited energy. The latter is a natural consequence of energy storage limitations, which become the overriding factor in the operation of robots that must operate in a fully autonomous mode over extended periods of time. Planning is also required to ensure, in the case of multiple vehicle operations, that the robots meet spatial configuration constraints.

It is against this backdrop of ideas that, in this paper, we address the following multiple vehicle motion planning problem: given a number of (possibly heterogeneous) vehicles tasked to arrive at desired target positions simultaneously and in the presence of known obstacles, compute trajectories that join the initial and target points, minimize overall vehicle energy expenditure, and meet the dynamical constraints of each vehicle while avoiding inter-vehicle and vehicle/obstacle collisions. Space limitations dictate that we focus on this simultaneous arrival problem. However, the core methodology developed for cooperative motion planning applies to a multitude of other mission scenarios.

Research supported in part by project Morph of the EU FP7 (grant agreement no. 288704), and by the FCT [UID/EEA/50009/2013] . The work of A. Häusler was supported with a Ph.D. scholarship of the Portuguese Science and Technology Foundation (FCT) under grant number SFRH/BD/68941/2010.

A. J. Häusler (a.haeusler@ieee.org) was formerly with the Laboratory of Robotics and Systems in Engineering and Science (LARSyS), IST, University of Lisbon, Portugal, and is now with the Marine Technology Group, MARUM (Center for Marine Environmental Sciences), University of Bremen, Germany. A. Saccon (a.saccon@tue.nl) is with the Dynamics and Control Section, Department of Mechanical Engineering, Eindhoven University of Technology, The Netherlands. A. P. Aguiar (pedro.aguiar@fe.up.pt) is with the Research Center for Systems and Technologies, Faculty of Engineering, University of Porto (FEUP), Portugal. J. Hauser (john.hauser@colorado.edu) is with the Department of Electrical, Computer, and Energy Engineering, University of Colorado at Boulder, CO, USA. A. M. Pascoal (antonio@isr.ist.utl.pt) is with the Laboratory of Robotics and Systems in Engineering and Science (LARSyS), IST, University of Lisbon, Portugal.

Draft paper submitted to IEEE Trans. Control Syst. Technol. on May 15, 2014. First revision submitted on December 13, 2014. Second revision submitted on June 2, 2015. Final version submitted on August 24, 2015.

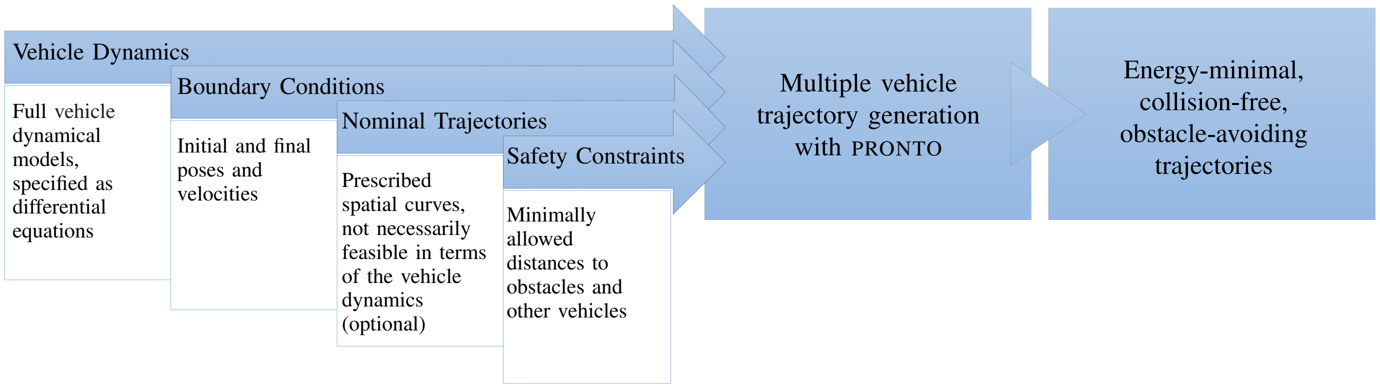

Vehicle Dynamics
Full vehicle dynamical models, specified as differential equations
Boundary Conditions
Initial and final poses and velocities
Nominal Trajectories
Prescribed spatial curves, not necessarily feasible in terms of the vehicle dynamics (optional)
Safety Constraints
Minimally allowed distances to obstacles and other vehicles
Multiple vehicle trajectory generation with PRONTO
Energy-minimal, collision-free, obstacle-avoiding trajectories

Figure 1. An overview of the trajectory planning framework introduced in this paper.

We tackle the problem of multiple vehicle motion planning by adopting an optimal control theoretical setting and including explicitly in the problem formulation the possibly complex dynamics of the vehicles involved. The planning tool that we develop affords mission designers an efficient methodology to understand how different cost functions and vehicle characteristics impact on the expected performance of a group of vehicles. In contrast with many strategies commonly reported in the literature for near real-time motion planning, we do not resort to approximations that often end up restricting the applicability of the algorithms proposed to a very limited set of possible mission scenarios. These would be, for instance, discrete (i.e., gridded) environments, discrete sets of motion primitives (i.e., prescribed "maneuver elements"), and/or discrete time, together with the underlying general assumption that no problems will occur after the transition from the planning space to the application space, such as the occurrence of obstacles that are too small for the discrete environment/motion/time grid size to have been captured in the planning phase. Our conviction is that once the nature of the problem of motion planning is understood in a general sense, choices affecting the efficiency of an algorithm (e.g. making the problem finite-dimensional instead of infinite-dimensional as in our proposed approach) can be made from an informed point of view, and the results of the planner presented here can then be used as "ground truth" against which to validate the results obtained later through a modified algorithm that reduces the computational load; e.g. by means of "educated heuristics" or by going from continuous to discrete time optimization. Our interest lies in understanding what kind of cost functions can be useful for motion planning of multiple vehicles in realistic scenarios. Making this approach applicable to on-line planning, possibly in a receding horizon fashion, is not the focus of this paper and is left for future work[1].

The methodology adopted builds on a numerical method for solving optimal control problems initiated by Hauser [15] and developed with coworkers, that is known as the PR*ojection Operator based Newton method for Trajectory Optimization* (PRONTO). It is a direct method that constructs a second order approximation of the optimization problem directly in continuous time (thus avoiding the transcription phase [3]), and employs a Newton method in the Banach manifold of system trajectories, with second order convergence.

[1] In [25], the authors claim that trajectory generation that returns a feasible result in "under a minute" can be considered as being an "on-line planner". Although being a rather vague definition, in this sense, depending on the complexity of the problem setting (number of vehicles and obstacles and time horizon), our method can be considered as on-line applicable.

B. State-of-the-art

The literature on vehicle motion planning is vast and defies a simple summary. The problem is well known and the methodologies for its solution proposed in the literature are diverse; see for example [26, 33, 9] and the references therein. Still, to the best of our knowledge, no motion planner is available that takes into account the full vehicle dynamics and addresses explicitly the issues of energy minimization and multiple vehicle collision unless some assumptions aimed at simplifying the task at hand are done. In many cases, the approaches are limited to planning *paths* without the associated timing laws that are an integral component of a *trajectory*—see e.g. [33, p. 161] for an illustration of the difference between these two terms. Common approaches that are used to make the motion planning problem easier to address are, e.g., discretization of the environment [35], simplified [27] or subclasses of possible vehicle models [13], and a limited set of possible motions [4] or path shapes [29]. In fact, one of the most common approaches is to include the limitations imposed by the vehicle dynamics at the planning level by adding them to the problem in the form of velocity and acceleration constraints that are imposed on a path planner, see, e.g., [22]. In this respect, the trajectory generation method we present in this paper also stands in contrast to our previous approach [19].

Many path and trajectory planning approaches limit themselves to a subset of possible path shapes, e.g. Dubins paths (i.e. lines and arcs) [29, 34]. This allows for the computation of possible maneuvers using a tree-based planner for finding a collision-free path [14, 21]. However, this comes at the expense of reduced flexibility of the solution, and possibly undesired first- and second-order discontinuities at the transition points of the path components, which put the vehicle actuators under more load than would be necessary. Differentiability issues can be avoided by planning polynomial paths of sufficiently high degree [25, 23, 38, 32]. A maneuver is then considered "feasible" if related kinematic and dynamic quantities such as velocity, acceleration, and curvature satisfy suitable bounds over the entire planning horizon to increase the confidence that the actual vehicle will be able to track the planned motion. Although with these latter planning approaches non-smoothness of the resulting trajectories ceases to be an issue, some conservatism is introduced in order to be able to express the vehicle limits in terms of simple kinematic or dynamic constraints. In contrast to the described approaches, our planner produces trajectories that satisfy nonlinear vehicle dynamics. In other words, the trajectories resulting from our planner are guaranteed to be feasible in terms of the vehicle dynamics and $\mathcal{C}^2$-smooth (or higher), for the simple reason that they are the solution of an optimization process that explicitly incorporates the nonlinear vehicle model, which the computed trajectories satisfy.

Alternatively, originally non-smooth paths are sometimes smoothed by fitting polynomials in a post-processing step [29], an approach that might be classified as separating path planning (finding a non-smooth path through the environment based on a given discretization) from trajectory optimization (smoothing the paths, possibly in an iterative manner [6]). Since this necessarily involves changing a path *after* a solution to the multiple vehicle planning problem has been found, the possibility exists that collisions may be re-introduced into a formerly collision-free set of paths. Planning in this sense is essentially a two-step process (i.e. planning and subsequent refinement) [7], which may result in the situation where the results of one step are influenced by the other step in such a manner that both alternate without finding an optimal solution. In contrast, our approach is a one-step solver that uses collision avoidance specifications directly in the minimization process, so that collisions have an immediate impact on the shape of the cost functional.

Post-processing of trajectories to verify their feasibility in terms of constraints is usually done on the basis of a discretization of the trajectories in space or time [7]. Such a discretization is used in the vast majority of methods available in the literature, and offers some of the advantages of direct methods, e.g. easy implementation and relative robustness to poor initialization. However, it also brings with it the problem of finding a suitable step size: the grid cannot be too small, because otherwise the finite-dimensional optimization problem would become intractable for a general-purpose solver, while, if the time grid is too coarse, it is possible that not all violations of the constraints (e.g. collisions) are detected. This problem is completely avoided in the motion planning approach that we propose in this paper, since it uses a variable step size solver. This is because the projection operator approach that is at the root of our methodology is not based on a transcription phase, which would be the standard for many direct methods in optimal control [8]. Using a variable step size solver ensures that no collision falls "in between two discrete points"; i.e., situations such as that in [7], where the authors need to do a sampling of the trajectories and to adapt the step size in certain regions according to the properties of the environment, do not occur. In other words, we propose to first find a solution and then approximate it (for in-memory storage), whereas most approaches first approximate the problem and then solve it on a discrete subspace.

C. Main Contributions

The main contributions of this paper are threefold:

1) Optimization Framework: We develop a numerical framework for the computation of minimum-energy, inter-vehicle and obstacle collision-free, *multiple-vehicle trajectories* that allows for the explicit inclusion of arbitrary full vehicle dynamics (instead of simple algebraic constraints) in the non-convex optimization problem (illustrated in Figure 1). By incorporating explicitly the vehicle dynamics, together with models of the vehicle's actuators and power source, it becomes possible to compute, for each candidate maneuver, the energy expenditure *at the energy source level*, and not simply the output mechanical energy imparted to the vehicle. It is not essential that the initial curves be feasible trajectories; if required, the projection operator may be used to transform them into feasible trajectories in terms of the vehicle dynamics before invoking the optimization algorithm. Furthermore, the initial curves do not have to satisfy the inter-vehicle and obstacle avoidance collision constraints.

2) Barrier Functional: We introduce a refinement of the barrier functional method introduced in [18] that shows improved performance of the algorithm in the presence of collisions among vehicles and with obstacles. This barrier functional is used as extension of the interior point optimization method to incorporate the collision avoidance constraints as part of the cost functional, thus achieving an approximate solution to the constrained optimization problem by means of a continuation approach.

3) Simulations: We show numerical examples to demonstrate the effectiveness of the method. In order to maintain the focus on the method itself, a simple second-order dynamical model of a non-holonomic, differential drive robot is developed, including battery energy consumption; obstacles are assumed to be enclosed in a circular hull. The demonstration scenarios include multiple vehicles performing a formation change, moving in a random obstacle field, and executing a survey mission with repeatedly intersecting trajectories.

D. Terminology

The basic terminology is defined next. In the remainder of this paper, we use the term "collision avoidance" to designate the avoidance of inter-vehicle collisions; the term "obstacle avoidance" will be used for situations where a vehicle avoids a stationary obstacle. With a slight abuse of semantics, we use formulations like "the projection operator" to mean both the mathematical operator as well as the full approach to solving an optimization problem; distinguishing both meanings will be clear from the particular context in which the terms are used. The notion "trajectory" is to be understood as referring to time-dependent state-input curves satisfying the vehicle dynamics.

This paper is organized as follows: Section II introduces the vehicle model adopted for illustration purposes, describes the cost functional involved in the motion planning problem, and formulates the latter rigorously. The projection operator approach adopted to solve the resulting optimization problem is briefly described in Section III, together with a summary of the barrier functional used to incorporate optimization constraints. Sections IV and V discuss the numerical implementation of the algorithm and show the results obtained from representative simulations. Conclusive remarks with a perspective on future research are given in Section VI. Two appendices give a detailed development of the vehicle model and its electrical system.

II. Problem Formulation

The methodology that we propose applies to arbitrarily complex vehicle dynamics. In this paper, however, in order to not clutter the presentation and distract the reader from the main contribution, we chose to build the presentation on the simple model of a unicycle-like ground robot.

A. Vehicle Dynamics

The vehicle model considered in this work is that of a two-wheeled mobile robot, depicted in Figure 2. The model, obtained from first principles and detailed in Appendix A, is given by

$$\begin{aligned}\dot{x} &= u\cos\psi\\ \dot{y} &= u\sin\psi\\ \dot{\psi} &= r\\ \bar{m}\dot{u} &= c_1 u + c_2(\tau_\mathrm{L}+\tau_\mathrm{R})\\ \bar{J}\dot{r} &= c_3 r + c_4(\tau_\mathrm{L}-\tau_\mathrm{R})\,,\end{aligned} \tag{1}$$

where $\bar{m}$ and $\bar{J}$ are auxiliary terms defining the equivalent mass and moment of inertia, respectively, and c_1, c_2, c_3, and c_4 are constants defined through the model's physical properties. The model states and inputs are listed in Table I. In what follows, $\{I\}$ and $\{B\}$ denote an inertial and body-axis reference frame, respectively.

Table I
List of state and input variables for the robot model at hand. The body frame is denoted as $\{B\}$, the intertial frame as $\{I\}$.

Var.	Description	Unit
x	x-coordinate of center of mass in $\{I\}$	[m]
y	y-coordinate of center of mass in $\{I\}$	[m]
ψ	yaw angle (in the x-y-plane)	[rad]
u	inertial velocity along the ${}^{\{B\}}x$ axis	[m]/[s]
r	angular speed about the ${}^{\{B\}}z$ axis	[rad]/[s]
τ_R	right motor driving torque	[Nm]
τ_L	left motor driving torque	[Nm]

Many trajectory optimization approaches are limited to differentially flat systems [36], since this allows for planning to occur in the output space [28], resulting in a fast algorithm. One must be careful, however, even when working with differentially flat systems, as the cost functions of interest may in fact be rather complicated (i.e., highly nonlinear) functions of the "flat coordinates", limiting the effectiveness of a simple performance analysis. The vehicle model (1) is also differentially flat; however, many systems of interest do not possess this property. We therefore avoid restricting our planner to this subclass of systems and avoid using properties of differentially flat systems, thus making our framework suitable for generic vehicle models.

In order to deal with more than one vehicle, we write the state and input of the i-th vehicle in compact notation as

$$\mathbf{x}^{[i]} = \begin{bmatrix}\mathbf{x}_1^{[i]} & \cdots & \mathbf{x}_5^{[i]}\end{bmatrix}^\mathsf{T} = \begin{bmatrix}x_i & y_i & \psi_i & u_i & r_i\end{bmatrix}^\mathsf{T} \tag{2}$$

$$\mathbf{u}^{[i]} = \begin{bmatrix}\mathbf{u}_1^{[i]} & \mathbf{u}_2^{[i]}\end{bmatrix}^\mathsf{T} = \begin{bmatrix}\tau_{\mathrm{L}i} & \tau_{\mathrm{R}i}\end{bmatrix}^\mathsf{T} \tag{3}$$

and rewrite (1) in standard notation as

$$\dot{\mathbf{x}}^{[i]}(t) = f\big(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t\big)\,. \tag{4}$$

B. Cost Functional

Our goal is to minimize the energy spent by each vehicle when moving from a given initial to a given final configuration. As detailed in Appendix B, the instantaneous power consumption for each robot can be computed as

$$\begin{aligned} l_\mathrm{pow}\big(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t)\big) &= R_\mathrm{a}\frac{\big(\mathbf{u}_1^{[i]}\big)^2 + \big(\mathbf{u}_2^{[i]}\big)^2}{K_\mathrm{t}^2}\\ &+ \frac{K_\mathrm{e}}{K_\mathrm{t}}\Big(\frac{\mathbf{u}_1^{[i]}}{\rho_\mathrm{w}}\big(\mathbf{x}_4^{[i]} + \rho_\mathrm{b}\mathbf{x}_5^{[i]}\big) + \frac{\mathbf{u}_2^{[i]}}{\rho_\mathrm{w}}\big(\mathbf{x}_4^{[i]} - \rho_\mathrm{b}\mathbf{x}_5^{[i]}\big)\Big) + P_\mathrm{p}\,, \end{aligned} \tag{5}$$

where K_e, K_t and R_a are suitable constants related to the electromechanical equations of the two DC motors mounted on each vehicle. The length of an axle (i.e., the distance of a wheel to the robot's center of mass) is denoted as ρ_b; the wheels' diameter is ρ_w. The term P_p is the (constant) hotel payload, i.e. the power required by all on-board devices that are *not* related to the execution of any maneuvering commands, and which require approximately constant input power, for example, the on-board computer and a sensor package.

C. Terminal Condition

A terminal condition is imposed on each vehicle state at the final time T adopted for the maneuver, i.e., its pose (position and orientation) and its forward and angular velocity. In other words, we desire that each robot reach its designated final pose at time T, satisfying

$$\mathbf{x}^{[i]}(T) = \mathbf{x}_\mathrm{f}^{[i]}\,, \tag{6}$$

where N_v is the total number of vehicles and $i \in \{1, \ldots, N_\mathrm{v}\}$. The constant $\mathbf{x}_\mathrm{f}^{[i]}$ denotes the terminal condition.

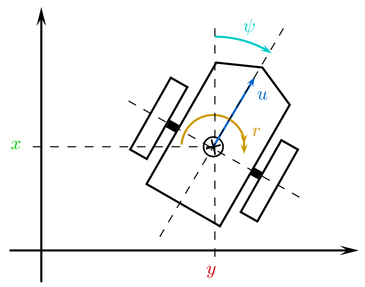


Figure 2. A schematic view of the robot used in this paper with all states of the dynamic model.

Instead of imposing the terminal condition as a hard constraint, we follow what is by now a classic approach (see e.g. [5]) and formulate it as a soft constraint by defining the terminal cost $m(\mathbf{x}^{[i]}(T))$ for each vehicle as

$$m(\mathbf{x}^{[i]}(T), \mathbf{x}_{\mathrm{f}}^{[i]}) = \frac{1}{2}||\mathbf{x}^{[i]}(T) - \mathbf{x}_{\mathrm{f}}^{[i]}||^2_{P_{\mathrm{f}}}\,, \tag{7}$$

where the positive definite weight matrix P_{f} is chosen to be sufficiently large so as to represent a good trade-off between integral and terminal cost.

Employing a terminal cost rather than a terminal constraint allows for a certain "soft boundary". If we had a terminal constraint and a system that would not allow for the exact satisfaction of that constraint, the problem itself would be infeasible, regardless of how close a feasible solution could exist to the terminal condition. When the exact satisfaction of the terminal condition is essential, the formulation of the terminal condition as a soft constraint can be further improved by using the approach described in [16]. This advanced method is, however, not discussed in this paper, as the results we obtain by just using the terminal cost (7) are already satisfying.

D. Constraints for Collision Avoidance

A further requirement is that the trajectories be generated in such a way that neither collisions between the vehicles nor among vehicles and obstacles may occur. The intuition behind our solution is shown in Figure 3. In the illustration for the inter-vehicle collision avoidance scenario (Figure 3a), both vehicles are shown at the same time $t = t_i$. At any time t, no vehicle should be closer to another than the sum of their security distances, $2r_{\mathrm{c}}$. However, crossings of trajectories *in space* are allowed: this happens, for example, when the left vehicle moves faster than the right one along the paths represented as dotted lines, so as to avoid intersection in time.

The requirement for the obstacle avoidance scenario in Figure 3b is that no robot may be closer to the boundary of an obstacle (defined by the obstacle's radius r_{o} around its known center coordinates) than r_{c}, the robot's security distance [2].

We define a set of trajectories as being inter-vehicle collision-free if at all instants of time all vehicles are separated from each other (i.e., pairwise) by at least a given security distance. This is equivalent to defining circles around each vehicle, centered at the origins of their body axes, with radii r_{c} that are half of the security distance; the disks are not allowed to overlap. To achieve collision-free trajectories, the motion planner must therefore ensure that the constraint

$$c_{\mathrm{col}}\big(\mathbf{x}^{[i]}(t), \mathbf{x}^{[j]}(t)\big) := \frac{\big(\mathbf{x}_1^{[i]}(t) - \mathbf{x}_1^{[j]}(t)\big)^2}{(2r_{\mathrm{c}})^2} + \frac{\big(\mathbf{x}_2^{[i]}(t) - \mathbf{x}_2^{[j]}(t)\big)^2}{(2r_{\mathrm{c}})^2} - 1 \geq 0 \tag{8}$$

be satisfied for all times $t \in (0, T)$ and all vehicles $i, j \in \{1, \dots, N_{\mathrm{v}}\}$ with $i \neq j$.

[2] To simplify the representation, only circular obstacles are considered at this stage, although it technically would not be a problem to represent other obstacle shapes as well by formulating the constraint differently, e.g. using intersections of half-planes. Also, we can cover any obstacle with a union of (circular) disks.

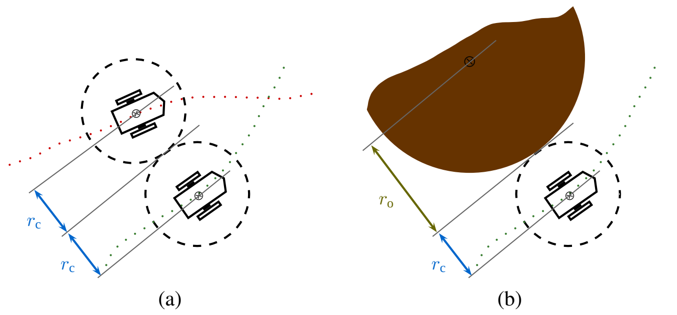


Figure 3. Illustration of the inter-vehicle collision avoidance and obstacle avoidance constraints. Figure 3a captures the definition of the (minimal) security distance between two robots; Figure 3b depicts the definition of the minimal security distance between a robot and an obstacle.

The obstacle avoidance constraint can be formulated in a similar manner. Let an obstacle be defined through the coordinates x_{o} and y_{o} of its center and its radius r_{o} (see Figure 3b). Then, each obstacle k can be represented by the 3×1 vector $\mathbf{o}^{[k]}$ with $\mathbf{o}_1^{[k]} = x_{\mathrm{o}_k}$, $\mathbf{o}_2^{[k]} = y_{\mathrm{o}_k}$, and $\mathbf{o}_3^{[k]} = r_{\mathrm{o}_k}$. Using this notation, trajectories that achieve obstacle avoidance satisfy the constraint

$$c_{\mathrm{obs}}\big(\mathbf{x}^{[i]}(t), \mathbf{o}^{[k]}\big) := \frac{\big(\mathbf{x}_1^{[i]}(t) - \mathbf{o}_1^{[k]}\big)^2}{\big(r_{\mathrm{c}} + \mathbf{o}_3^{[k]}\big)^2} + \frac{\big(\mathbf{x}_2^{[i]}(t) - \mathbf{o}_2^{[k]}\big)^2}{\big(r_{\mathrm{c}} + \mathbf{o}_3^{[k]}\big)^2} - 1 \geq 0\,, \tag{9}$$

where $k \in \{1, \dots, N_{\mathrm{o}}\}$ and N_{o} is the number of obstacles in the environment.

E. Cost Term for Desired Trajectories

In case the user may want to "bias" the optimization process towards trajectories that meet specified additional criteria, such as lying in preferred regions of the environment, desired curves[3] can be defined to guide the motion planning algorithm. The cost to be paid for a trajectory that does not match the specification of a desired trajectory is added as an additional term inside the cost integral. By doing so, we can penalize deviations of the planned trajectory from a desired curve without necessarily forcing it to exactly match the desired trajectory (which might not necessarily be a valid trajectory of the system or satisfy constraints). To this effect, we define positive definite weight matrices Q_{T} and R_{T} that, together with a desired system states $\mathbf{x}_{\mathrm{des}}^{[i]}$ and desired inputs $\mathbf{u}_{\mathrm{des}}^{[i]}$, specify the additional cost term

$$l_{\mathrm{trj}}\big(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t\big) = \frac{1}{2}\big|\big|\mathbf{x}^{[i]}(t) - \mathbf{x}_{\mathrm{des}}^{[i]}(t)\big|\big|^2_{Q_{\mathrm{T}}} + \frac{1}{2}\big|\big|\mathbf{u}^{[i]}(t) - \mathbf{u}_{\mathrm{des}}^{[i]}(t)\big|\big|^2_{R_{\mathrm{T}}}\,. \tag{10}$$

In case the optimizer is to be used solely as a trajectory generator, both weight matrices may be set to be zero.

[3]Since they do not necessarily satisfy the vehicle dynamics, these motions are referred to as "curves", not as "trajectories".

F. The Optimization Problem

The various components of the problem can now be put together to yield the optimization problem

$$\begin{aligned}\text{minimize}\quad & \int_0^T \sum_{i=1}^{N_\text{v}} \Big(l_\text{pow}\big(\mathbf{x}^{[i]}(\tau), \mathbf{u}^{[i]}(\tau)\big) \\ & \quad + l_\text{trj}\big(\mathbf{x}^{[i]}(\tau), \mathbf{u}^{[i]}(\tau), \tau\big)\Big) d\tau + \sum_{i=1}^{N_\text{v}} m(\mathbf{x}^{[i]}(T), \mathbf{x}_\text{f}^{[i]}) \\ \text{subject to}\quad & \dot{\mathbf{x}}^{[i]} = f(\mathbf{x}^{[i]}, \mathbf{u}^{[i]}, t) \\ & c_\text{col}\big(\mathbf{x}^{[i]}(t), \mathbf{x}^{[j]}(t)\big) \geq 0, \quad i \neq j \\ & c_\text{obs}\big(\mathbf{x}^{[i]}(t), \mathbf{o}^{[k]}\big) \geq 0\end{aligned} \tag{11}$$

with $i, j \in \{1, \ldots, N_\text{v}\}$ and $k \in \{1, \ldots, N_\text{o}\}$ and the boundary conditions $\mathbf{x}^{[i]}(0) = \mathbf{x}_0^{[i]}$.

III. Trajectory Optimization using the Projection Operator Approach

This section contains a description of the method adopted to deal with the optimization problem defined above. We make use of the notation in [15, 18]. Instead of solving (11) with the hard constraints as given above, we employ a barrier functional that allows for the satisfaction of interior point constraints as part of the cost functional itself. After discussing the approach adopted to solve the optimization problem, we give a formal introduction of the barrier method.

A. Projection Operator Approach to the Optimization of Trajectory Functionals

We review the projection operator based Newton method for the optimization of trajectory functionals, introduced by Hauser in [15]. The projection operator approach is suitable for addressing optimal control problems of the form

$$\begin{aligned}\text{minimize}\quad & \int_0^T l(x(\tau), u(\tau), \tau)\, d\tau + m(x(T)) \\ \text{subject to}\quad & \dot{x} = f(x, u, t), \quad x(0) = x_0\,.\end{aligned} \tag{12}$$

The key idea exploited is that a properly designed trajectory tracking controller defines a function space *operator* that maps a desired trajectory (a *curve*) to a system *trajectory* (that can be viewed as an element of a Banach manifold, henceforth referred to as the system's trajectory manifold). Composing the optimization objective (a *functional*) with the (trajectory tracking) *projection operator* converts the dynamically constrained optimal control problem into an essentially unconstrained optimization problem.

Let $\mathcal{I} = [0, T]$ or $[0, \infty)$ be an interval of interest and suppose that $\xi(t) = (\alpha(t), \mu(t))$, $t \in \mathcal{I}$, is a bounded curve (e.g., an approximate trajectory of f) and let $\eta(t) = (x(t), u(t))$, $t \in \mathcal{I}$, be the trajectory of f determined by the nonlinear feedback system

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t), t), \qquad x(0) = x_0, \\ u(t) &= \mu(t) + K(t)(\alpha(t) - x(t))\,.\end{aligned} \tag{13}$$

The mapping

$$\mathcal{P} : \xi = (\alpha(\cdot), \mu(\cdot)) \mapsto \eta = (x(\cdot), u(\cdot))$$

defines a nonlinear *projection operator*. Note again that we refer to a trajectory as including *both* time-dependent states and inputs. Roughly speaking, if $K(\cdot)$ stabilizes a trajectory ξ_0, then the trajectory $\eta = \mathcal{P}(\xi)$ will be well defined for all ξ near ξ_0 and the mapping $\mathcal{P}$ will be continuous (and C^r if f is). Properties of a projection operator defined for curves and trajectories on an infinite interval were deveoped in [17], where $\mathcal{P}$ is used to show that the set of exponentially stabilizable trajectories of f is in fact a Banach manifold. When the interval of interest is bounded, the same properties hold for $\mathcal{P}$ without any stability requirement (which is meaningless in that case), but $K(\cdot)$ can then be chosen as if to stabilize so that the modulus of continuity of $\mathcal{P}$ is small.

Let $K(\cdot)$ be bounded on $\mathcal{I} = [0, T]$ and use $\mathcal{T}$ to denote the manifold of trajectories of f with initial state x_0. Then $\mathcal{P}(\xi) \in \mathcal{T}$ for all $\xi \in \operatorname{dom}\mathcal{P}$ and $\xi \in \mathcal{T}$ if and only if $\xi = \mathcal{P}(\xi)$. Together, these imply that $\mathcal{P}(\xi) = \mathcal{P}(\mathcal{P}(\xi))$ for all $\xi \in \operatorname{dom}\mathcal{P}$ or $\mathcal{P} = \mathcal{P} \circ \mathcal{P}$ so that $\mathcal{P}$ is indeed a projection.

The curve $\xi = (\alpha(\cdot), \mu(\cdot))$ may be used to provide a *redundant* representation of the trajectory $\eta = \mathcal{P}(\xi) = (x(\cdot), u(\cdot))$. This robust representation of the trajectory $(x(\cdot), u(\cdot))$ is well suited for numerical computations since the approximation errors introduced by discretization in time and quantization in space are kept small by the stabilizing effect of the feedback. In contrast, if f describes an unstable system, one can find quite different trajectories for which the initial state and sampled control trajectories are the same to machine precision. Note that a suitable feedback gain $K(\cdot)$ may be constructed by, for example, solving a finite horizon linear regulator problem [1] about a trajectory $\eta = (x(\cdot), u(\cdot))$.

Defining the functional

$$h(\xi) = \int_0^T l(\alpha(\tau), \mu(\tau), \tau)\, d\tau + m(\alpha(T))$$

for curves $\xi = (\alpha(\cdot), \mu(\cdot))$, we see that the optimal control problem (12) can be written as $\min_{\xi \in \mathcal{T}} h(\xi)$. Defining

$$g(\xi) = h(\mathcal{P}(\xi))$$

for $\xi \in \mathcal{U}$ with $\mathcal{U}$ open and $\mathcal{P}(\mathcal{U}) \subset \mathcal{U} \subset \operatorname{dom}\mathcal{P}$, we see that the optimization problems

$$\min_{\xi \in \mathcal{T}} h(\xi) \quad \text{and} \quad \min_{\xi \in \mathcal{U}} g(\xi)$$

are equivalent in the following sense. If $\xi^* \in \mathcal{T} \cap \mathcal{U}$ is a constrained local minimum of h, then it is an unconstrained local minimum of g. If $\xi^+ \in \mathcal{U}$ is an unconstrained local minimum of g in $\mathcal{U}$, then $\xi^* = \mathcal{P}(\xi^+)$ is a constrained local minimum of $\mathcal{T}$. This observation is the basis for the development of a family of Newton and quasi-Newton descent methods for the optimization of h over $\mathcal{T}$.

The projection operator $\mathcal{P}$ provides a convenient parametrization of the trajectories in the neighborhood of a given trajectory. Indeed, the tangent space $T_\xi\mathcal{T}$ of bounded trajectories of the linearization of $\dot{x} = f(x, u, t)$ about $\xi \in \mathcal{T}$ can be used to parameterize *all* nearby trajectories [17]. That is, given $\xi \in \mathcal{T}$, there is an $\epsilon > 0$ such that, for each $\eta \in \mathcal{T}$ with $\|\eta - \xi\| < \epsilon$ there is a *unique* $\zeta \in T_\xi \mathcal{T}$ such that $\eta = \mathcal{P}(\xi + \zeta)$. Note also that $\zeta \mapsto D\mathcal{P}(\xi) \cdot \zeta$ is the bounded *linear* projection operator defined by linearizing (13) about ξ and that $\zeta \in T_\xi \mathcal{T}$ if and only if $\zeta = D\mathcal{P}(\xi) \cdot \zeta$.

We use the following Newton method for the optimization of trajectory functionals. The reader interested in understanding the optimization method at an intuitive level is invited to consult [31, Fig. 1], where a graphical illustration that captures the iterations involved in the algorithm can be found.

```
Algorithm (Projection operator Newton method)
  given initial trajectory $\xi_0 \in \mathcal{T}$
  for $i = 0, 1, 2, \ldots$
      design feedback $K(\cdot)$ defining $\mathcal{P}$ about $\xi_i$
      search direction
          $\zeta_i = \arg\min_{\zeta \in T_{\xi_i}\mathcal{T}} Dh(\xi_i) \cdot \zeta + \frac{1}{2} D^2 g(\xi_i) \cdot (\zeta, \zeta)$   (14)
      step size
          $\gamma_i = \arg\min_{\gamma \in (0,1]} h(\mathcal{P}(\xi_i + \gamma \zeta_i))$   (15)
      update
          $\xi_{i+1} = \mathcal{P}(\xi_i + \gamma_i \zeta_i)$   (16)
  end
```

This algorithm is similar to the Newton method for unconstrained optimization of a function $g(\cdot)$ in, e.g., a finite-dimensional space. As usual, the second order Taylor polynomial is used as a quadratic model function for determining a descent direction. The key differences are that (a) the search direction minimization (14) is performed over the tangent space to the trajectory manifold and (b) the update (16) *projects* each iterate onto the trajectory manifold, giving a descending sequence of system trajectories. The projection operator and the cost evaluation are easily computed by solving ODEs (initial value problems).

The Newton algorithm (with $\gamma_i \equiv 1$) provides (local) quadratic convergence to a local minimizer satisfying second order sufficiency conditions. The algorithm is easily generalized (or globalized) by replacing the Newton direction calculation (14) by a quasi-Newton search direction calculation

$$\zeta_i = \arg\min_{\zeta \in T_{\xi_i}\mathcal{T}} Dg(\xi_i) \cdot \zeta + \tfrac{1}{2}\, q(\xi_i) \cdot (\zeta, \zeta)\,, \tag{17}$$

where the functional $q(\xi_i)$ is a suitable positive definite approximation to $D^2 g(\xi_i)$. For instance, if $l(x, u, t)$ is positive definite (or strongly convex) in (x, u), one may use $q(\xi) \cdot (\zeta, \zeta) = D^2 h(\xi) \cdot (\zeta, \zeta)$. Note also that it is not necessary to do an accurate line search in (15). The standard *backtracking* line search (see, e.g., [5]) has been shown to work well.

B. A Barrier Functional for Constraints

Trajectory optimization with inequality constraints on the state and input can be approached with the help of a barrier functional [18] that generalizes the use of log barrier functions in finite dimensions [5]. Indeed, to approximate the solution of the constrained optimal control problem

$$\begin{aligned} \min\ & \int_0^T l(x(\tau), u(\tau), \tau) d\tau + m(x(T)) \\ \text{s.t.}\ & \dot{x}(t) = f(x(t), u(t), t), \quad x(0) = x_0 \\ & c_j(x(t), u(t), t) \le 0, \qquad t \in [0, T],\ j \in \{1, ..., k\} \end{aligned} \tag{18}$$

with the constraints c_j, we could try and solve

$$\begin{aligned} \min\ & \int_0^T l\big(x(\tau), u(\tau), \tau\big) \\ & \quad - \epsilon \sum_j \log\big(-c_j(x(t), u(t), t)\big) d\tau + m(x(T)) \\ \text{s.t.}\ & \dot{x}(t) = f\big(x(t), u(t), t\big), \quad x(0) = x_0 \end{aligned} \tag{19}$$

for $\epsilon > 0$. The log barrier terms in the cost prevent the constraint functions from crossing zero, allowing them to go to zero as ϵ approaches zero. From this point of view, the approach seems to be effective.

A key difficulty in the formulation in (19) is the fact that the use of the log barrier function implies that infeasibility will not be tolerated at all. That is, it is not possible to evaluate the cost in (19) unless ξ is a feasible curve. Furthermore, even if ξ is feasible, we can not be sure that $\mathcal{P}(\xi)$ will be.

To resolve this issue, we define, for $0 < \delta \le 1$, the approximate log barrier function

$$\beta_\delta(z) = \begin{cases} -\log z & z > \delta \\ \frac{1}{2}\left[\left(\frac{z - 2\delta}{\delta}\right)^2 - 1\right] - \log \delta & z \le \delta\,. \end{cases} \tag{20}$$

The $\mathcal{C}^2$ function $\beta_\delta(\cdot)$ retains many of the important properties of the log barrier function $z \mapsto -\log z$ while expanding the domain of finite values from $(0, \infty)$ to $(-\infty, \infty)$. Both functions are (strictly) convex and strictly decreasing on their domains. Thus, if $c : \mathbb{R} \to \mathbb{R}$ is a strictly convex proper function, the function $z \mapsto \beta_\delta(-c(z))$ is also a strictly convex proper function on $\mathbb{R}$. It follows that, for convex $f(x)$ and $c_j(x)$,

$$\min_{x \in C} f(x) + \epsilon \sum_j \beta_\delta(-c_j(x)) \tag{21}$$

is a convex problem that has the *same* solution ($x_\epsilon^\star$) as the one that would be obtained using the standard log barrier function, provided that $\delta > 0$ is small enough that $\delta < -c_j(x_\epsilon^\star)$ for all j.

Returning to infinite dimensions, we use the constraint functions in (18) to define the approximate barrier functional

$$b_\delta(\xi) = \int_0^T \sum_j \beta_\delta(-c_j(\alpha(t), \mu(t), t)) d\tau \tag{22}$$

for a given curve $\xi = (\alpha(\cdot), \mu(\cdot))$.

The unconstrained optimal control problem

$$\min_{\xi \in \mathcal{T}} h(\xi) + \epsilon b_\delta(\xi) \tag{23}$$

is an approximation of (18) in much the same way that (19) is. The key difference is that the functional $h(\cdot) + \epsilon b_\delta(\cdot)$ can be evaluated on any curve ξ in $\tilde{X}$ while the objective in (19) may only be evaluated on feasible curves. As in the finite dimensional case, a trajectory $\xi_\epsilon^\star$ that is a locally optimal solution of (19) is also a locally optimal solution of (23) provided $\delta > 0$ is sufficiently small. In particular, since $\xi_\epsilon^\star$ is continuous,

$$\delta < \min_{t\in[0,T]} \min_j -c_j(x_\epsilon^\star(t), u_\epsilon^\star(t), t) \tag{24}$$

is sufficient for δ.

The projection operator based Newton method may be used to optimize the functional

$$g_{\epsilon,\delta}(\xi) = h(\mathcal{P}(\xi)) + \epsilon b_\delta(\mathcal{P}(\xi)) \tag{25}$$

as part of a continuation method (i.e., following a central path) to seek an approximate solution to (18). The strategy is to start with a reasonably large ϵ and δ, for instance, $\epsilon = \delta = 1$. Then, for the current ϵ and δ, the problem

$$\min g_{\epsilon,\delta}(\xi) \tag{26}$$

is solved using the Newton method starting from the current trajectory. If necessary or desired, the value of δ is reduced to, for instance, ensure that (19) has been solved at the current level of ϵ. When satisfied, the trajectory is updated to the current optimal. Next, ϵ and δ are decreased using, for instance, $\epsilon \leftarrow \epsilon/10$ and $\delta \leftarrow \delta/10$. Then go back to the minimization step and continue. The effect that δ has in this extension of the logarithmic barrier functional is demonstrated in Figure 4a.

For efficiency, the line search in the Newton method can be modified to help in the search for feasible trajectories and to aid in maintaining feasibility once it has been achieved. This can be done reasonably well since the projection operator ensures that $\mathcal{P}(\xi + \gamma\zeta)$ will be close to $\xi + \gamma\zeta$ provided $\gamma\zeta$ is not too large. Thus, when ξ is a feasible trajectory, we expect that the trajectory $\xi^+ = \mathcal{P}(\xi + \gamma\zeta)$ will be feasible provided that the *curve* $\xi + \gamma\zeta$ is feasible and $\gamma\zeta$ is sufficiently small.

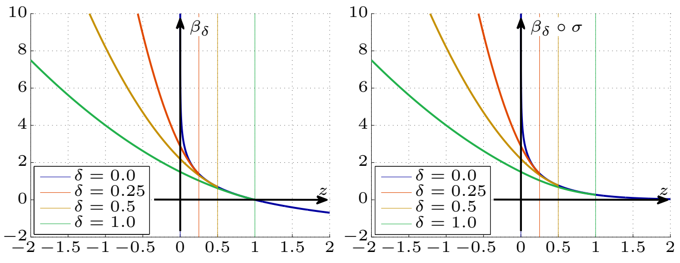


Figure 4. A comparison of $\beta_\delta(z)$ with $\beta_\delta(\sigma(z))$ for different values of δ with their $\mathcal{C}^2$-smooth continuation defined for $z <= \delta$. In Figure 4a we use the convention $\beta_0(z) := -\log(z)$ (i.e. for $\delta = 0$).

IV. Numerical Implementation

In what follows, we present a numerical method to solve the optimization problem defined above, using the previously described barrier function approach for the hard constraints. We begin by describing the procedure of converting the aforementioned optimization approach into a numerical algorithm. After formulating the constraints as barrier terms, we introduce an extension of the barrier functional formulation, so that it will not reward greater distances among the vehicles or between vehicles and obstacles, as would be the case for the original definition that was discussed above. The section concludes with some consideration on the implementation of the algorithm on a laptop computer.

In contrast to the vast majority of approaches present in the literature, we solve this optimization problem as a continuous-time problem on an infinite-dimensional function space. It is important to bear in mind that, to solve the optimization problem in this manner, we do not need to discretize it (neither in time, nor in the vehicles' state space or their input and output spaces). It is merely for representing and storing the results that discretization is required.

A. Optimization Constraints

Following the barrier functional formulation discussed in Section III-B, the constraint functions (8) and (9) will be incorporated as barrier terms in the cost integral. In contrast to the standard log-barrier method, this method allows expressing inequality constraints as a valid function on the full search space (i.e., extending the domain from $(0, \infty)$ to (∞, ∞)). Moreover, it is continuous and twice differentiable, thus meeting the prerequisites of the projection operator approach that we employ here to solve our optimization problem. By adjusting the tuning parameters, this barrier function can be shaped such that collisions of a robot with another one or with an obstacle have a strong impact on the integral cost, whereas, due to an improvement we made on the original barrier functional (to be discussed at the end of this section), the function's influence is effectively zero for the time intervals during which no collision occurs.

Making use of the barrier functional, the collision avoidance term of the cost functional is defined as

$$\begin{aligned} l_{\text{col}}(\mathbf{x}(t)) = &\sum_{\substack{i,j\in\{1,\ldots,N_\text{v}\}\\ i\neq j}} \epsilon\beta_\delta\Big(c_{\text{col}}\big(\mathbf{x}^{[i]}(\tau), \mathbf{x}^{[j]}(\tau)\big)\Big) \\ &+ \sum_{\substack{i\in\{1,\ldots,N_\text{v}\}\\ k\in\{1,\ldots,N_\text{o}\}}} \epsilon\beta_\delta\Big(c_{\text{obs}}\big(\mathbf{x}^{[i]}(\tau), \mathbf{o}^{[k]}\big)\Big). \end{aligned} \tag{27}$$

As mentioned above, the barrier functional's stiffness depends on the choice of the two tuning parameters ϵ and δ, which, after the initial run of the Newton descent method has converged, may be adapted and the algorithm re-started with the previous solution as the initial curve, thereby gradually "hardening" the barrier. This allows accommodating initial trajectories that may not be feasible in terms of the constraints expressed by the barrier functional; an initial "soft" barrier will naturally drive the trajectories out of the infeasibility region of the optimization space. Later adjustments of the barrier functional's parameters make the barrier more stiff, which prevents the trajectories from re-entering the infeasible regions. It is important to keep in mind that setting the parameters to a stiff barrier right from the beginning might not result in a feasible solution at all since the descent of the cost functional might be too steep for the Newton descent.

Thus, we can define the total optimization problem as

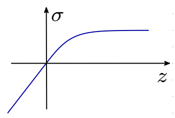
$$\begin{aligned} \min \quad & \int_0^T \sum_{\substack{i,j\in\{1,\dots,N_\text{v}\}\\ k\in\{1,\dots,N_\text{o}\}}} \Big(l_\text{pow}\big(\mathbf{x}^{[i]}(\tau),\mathbf{u}^{[i]}(\tau)\big) \\ & \qquad + l_\text{trj}\big(\mathbf{x}^{[i]}(\tau),\mathbf{u}^{[i]}(\tau),\tau\big) \\ & \qquad + l_\text{col}\big(\mathbf{x}^{[i]}(\tau),\mathbf{x}^{[j]}(\tau)\big)\Big) d\tau \\ & \qquad\qquad + m(\mathbf{x}(T)) \\ \text{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u},t), \quad \mathbf{x}(0) = \mathbf{x}_0 \,, \end{aligned} \tag{28}$$

where $\mathbf{x}_0$ is the vector of initial boundary conditions $\mathbf{x}_0^{[i]}$.

B. An Extension of the Barrier Functional

As shown in Figure 4a, $\beta_\delta(z)$ assumes (unbounded) negative values for $z > 1$. As a consequence, the negative part may dominate in the cost integral; in what concerns our application, this effectively puts a reward on a vehicle to stay away from another vehicle (or obstacle) as far as possible. In our area of application, i.e. trajectory generation and optimization for multiple vehicles, this formulation of the barrier functional (Section III-B), i.e. being unbounded from below, is disadvantageous: for example, with long distance trajectories, where potential collisions with other vehicles or obstacles become rare events over the total time interval, the integral over the cost functional is dominated by the huge negative area of the corresponding β_δ barrier term. This leads to the fact that the relative importance of saving energy diminishes, allowing for large-distance circumnavigation of the conflict areas, together with the associated large system inputs and high energy expenditure—instead of maintaining e.g. a low-value speed profile and staying closer to the boundaries of the collision zone. In addition, the desired positive peak in the cost functional at the time of potential collision can, over the course of the optimization, be lowered to negative values by staying as far away from the collision zone as other existing optimization constraints (e.g. vehicle dynamics) would allow. To overcome this disadvantage, we extended the β_δ barrier functional by forming a composition with what we sometimes call the "hockey stick": the $\mathcal{C}^2$-smooth function

$$\sigma(z) = \begin{cases} \tanh(z) & \text{if } z \geq 0 \\ z & \text{otherwise}\,, \end{cases} \tag{29}$$



so that every occurence of $\beta_\delta(z)$ is being replaced by $\beta_\delta(\sigma(z))$. By doing so, the value of the barrier goes to zero quickly, but never becomes negative. The effect of this extension is shown in Figure 4, where we compare it with the previous approach: $\beta_\delta(\sigma(z))$ does asymptotically approach 0 for $z \to \infty$. In contrast to what is shown in Figure 4a, this has the effect of only requiring the vehicles to avoid collisions taking into account the given security distance *without* driving them further away from other (and from obstacles) than necessary (Figure 4b).

In the remainder of this paper, we let $\beta_\delta(z)$ denote the extended barrier functional, i.e.,

$$\beta_\delta(z) := \bar{\beta}_\delta(\sigma(z))\,, \tag{30}$$

where $\bar{\beta}_\delta$ denotes the (original) barrier functional from Section III-B.

C. Implementation Considerations

The projection operator approach to the optimization of trajectory functionals can be implemented in Matlab as an iteration over sequential calls to a number of Simulink models, that act as wrapper functions for the actual algorithm code. For the purpose of execution time reduction, the latter was implemented in C. Using calls to those "models" makes it easy to obtain a numerical solution of the differential equations involved by leaving the numerical integration The ODE solver "ode45" was used in all simulations we show in this paper, with an absolute tolerance of 10^{-10} and a relative tolerance of 10^{-6}. to the ODE solver of choice.

The algorithm itself was defined as a M-file script taking care of the iterations, computing the Newton steps, adjusting the step size, and invoking the S-function files with new inputs. A number of S-functions were defined for different parts of the projection operator. Namely, to compute: (1) the regulator gain to drive the system to its desired terminal state, (2) the gain$K(\cdot)$ for the projection operator itself and simultaneously executing the projected trajectory and obtaining the cost, and (3) the first and second order derivatives of the cost functional required by the Armijo rule [2].

Solving the minimization problem (11) in the particular way we propose here immediately leads to numerous benefits: we get $\mathcal{C}^2$-smooth trajectories, which are highly desirable for reducing motor wear. In addition to this, the formulation in (11) automatically ensures *simultaneous arrival*[4], which is an absolute necessity for cooperative missions, especially if the planned trajectory only constitutes one part of a larger mission plan. We also have the benefits of a system state that is explicitly incorporated in the optimization problem (see, eg. [30]). Finally, by making use of the projection operator method, we have at our disposal the possibilities of infinite-dimensional optimization, which means that our approach will not suffer from the problems of discretization inherent to other optimization methods reported in the literature. (Notice that discretization must necessarily occur at a certain stage when we solve optimization problems with a digital computer, but in our case this is dependent on the settings of the ODE solver employed, which can be, and in our case is, a variable-step-size solver.)

The simulation experiments conducted for this paper (see Section V) are all based on the same set of constants specifying the robot parameters (see Table II). The extended version of the β_δ barrier functional introduced above was used in all runs of the optimizer. A required security distance of 2.0 meters between vehicles and 1.0 meters between a vehicle and an obstacle was specified, effectively resulting in $r_\text{c} = 1.0$ m (c.f. Figure 3).

Every optimization run started with the weights $\epsilon = 32.0$ and $\delta = 1.0$ for the inter-vehicle collision avoidance β_δ barrier. The obstacle avoidance scenarios additionally used the values

[4]Since we numerically solve this optimization problem using an ODE solver, simultaneous arrival is achieved in an approximate fashion.

Table II
ROBOT BODY AND WHEEL CONSTANTS (TABLE IIIA) AND MOTOR ELECTRIC CONSTANTS (TABLE IIIB) OF THE GROUND ROBOT.

m_b	10.0 kg
ρ_b	0.25 m
J_b	0.2 kg/m^2
m_w	0.15 kg
ρ_w	0.1 m
J_w	0.00075 kg/m^2

(a)

b	0.05 Nms
K_t	0.046 Nm/A
K_e	0.046 Vs/rad
R_a	0.66 Ω
P_p	26.0 W

(b)

$\epsilon = 16$ and $\delta = 0.25$ as initial setting for the obstacle avoidance barrier. The energy optimization term (5) of the cost function was weighted with a factor of 4.0 to ensure that it dominates the other components. The weight P_f on the terminal cost $m(x(T))$ was fixed as a diagonal matrix[5]

$$\begin{aligned} P_\mathrm{f} &= \mathrm{diag}\big([P_{\mathrm{f}_1}, \ldots, P_{\mathrm{f}_{N_\mathrm{v}}}]\big) \\ \text{with } P_{\mathrm{f}_i} &= \mathrm{diag}\big([100, 100, 50\pi/180, 1000, 1]\big) \,. \end{aligned} \tag{31}$$

In addition to this, the terminal cost (7) was weighted with a scaling factor of 3.0.

The projection sub-part of the algorithm was also used for obtaining initial curves that are feasible in terms of the robots' dynamics. By projecting a given, desired path together with a velocity profile (possibly non-smooth) onto the set of trajectories of the system, we obtain a feasible vehicle trajectory.

V. SIMULATION RESULTS

The trajectory generation and optimization problem method was tested with various scenarios. We first discuss a simple trajectory generation problem, and subsequently will move on to more complex scenarios showing the necessity of outer loop iterations that gradually adjust the barrier functional to make it stiffer. Finally, we will show results with problem configurations to obtain two different feasible paths through a random field of obstacles by merely changing the trade-off between different parts of the cost function.

A. Trajectory Generation for Multiple Vehicles

An example of a "pure trajectory generation" scenario is shown in Figure 5: by only specifying only the boundary conditions (i.e. initial and final poses as well as forward and rotational velocities) of the problem, the method generated an energy-optimal solution to the multiple vehicle planning problem.

Besides allowing for formation changes within a fleet of autonomous vehicles, this understanding of trajectory generation also has an application to the solution of the "Go-To-Formation" behavior that is required e.g. for autonomous underwater vehicles (AUVs) before a mission starts: since the vehicles are constantly subject to the influence of waves and currents, they cannot be deployed in formation. This

[5]Alternatively, the weight matrix P_f could also be obtained as the solution of the LQR problem given by the linearization of the augmented state system about the final time T.

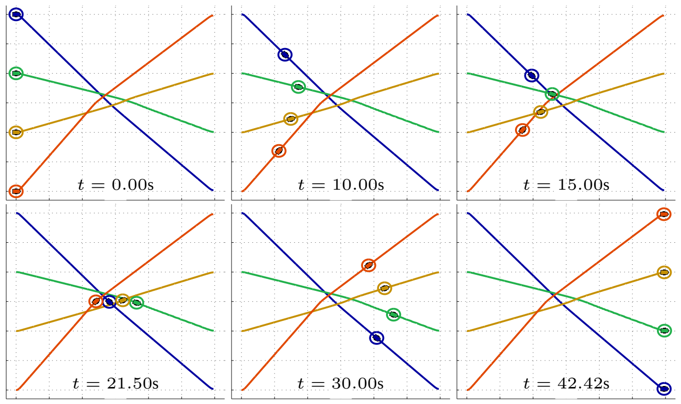


Figure 5. A formation change scenario. The circles around the robot pictograms allow us to visualize the security distance that must be maintained by the collision avoidance constraint: the circles are allowed to touch, but not to intersect.

makes it necessary to start multiple vehicle missions on-the-fly, thereby requiring the vehicles to reach the mission plan's first formation simultaneously and with equal forward velocities from positions scattered over the initial field of deployment from a supply vessel.

Managing a formation change by only specifying the boundary conditions has, in comparison to using desired trajectories for the same goal, the advantage that the planner has more flexibility in shaping the trajectories. This is especially useful in transition phases between different robot formations since collisions may only occur during these transitions. Having to move along desired trajectories would pose an unnecessary restriction to the problem. In other words, a formation change should be achieved not by telling the robots in a fleet *how* to change their formation, but only *that* they have to do so.

The computed solution is also optimal in the sense that deviations from the straight line connections between the initial and final positions are minimal; the vehicles avoid each other by exactly their security distance, which can be visually verified by considering Figure 5. The trajectories do intersect, in space but not at the same instants of time; the deconfliction constraints (illustrated by the circles around the vehicles that are shown at the times of closest approximation) are always satisfied. In a more colloquial manner, one could say that *the trajectories intersect in space, but are separated in time*, or, in other words, they intersect in space, but not in time.

It is also interesting to see the evolution of the optimal trajectories while iterating over the configuration parameters of the β_δ barrier functional. Figure 6 shows this for the initial curve, the final solution, and some intermediate steps for the formation change scenario. The frames illustrate the importance of beginning the descent with a "soft" value for ϵ and δ, so that the trajectories can move away from the collision points without being stuck (in the sense that a very stiff setting of the parameters would make it much harder to find a descent direction). Later on, while gradually increasing the stiffness of the barrier functional, and reducing its relative weight at the same time, the by now collision-free trajectories can move closer to the constraint boundary.

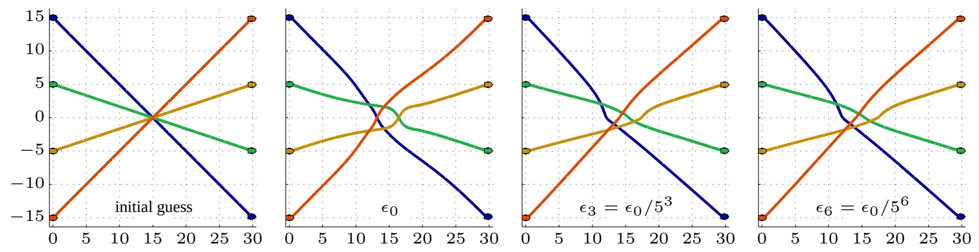


Figure 6. Development of the local optima obtained over various iterations on the configuration parameters of the β_δ barrier functional. Some intermediate steps are omitted, since they cannot be easily distinguished from each other at the resolution of a printed paper.

B. Optimization of Repeatedly Crossing Trajectories

A more sophisticated scenario was used to verify the capabilities of our method when optimizing a collection of desired trajectories. The desired mission is a "braid maneuver", where the robots repeatedly switch their relative positions (Figure 7). This is a maneuver that provides great ground coverage for exploration purposes, similar to the well-known "lawnmower maneuver" for autonomous surface craft in marine applications, but is most likely more economic since it avoids the fast transients that are inherent to the "lawnmower" when going from straight line to curved paths and back. A possible application for the "braid maneuver" would be e.g. observation missions for multiple vehicles that are aimed to maximizing ground coverage, or multiple vehicle SLAM.

This solution to optimizing the "braid maneuver" problem that is shown in Figure 7 was achieved by providing a high weight in Q_T for the states $\mathbf{x}_1^{[i]}$, $\mathbf{x}_2^{[i]}$ and $\mathbf{x}_4^{[i]}$ of each vehicle i. The other states and all inputs were weighted with 0. The initial curve was a straight line connection between the initial and final positions, with corresponding values for initial and final heading. Optimizing the trajectories with a non-zero weight on the energy term of (28) also serves the purpose of regularizing the problem.

This scenario additionally shows how important a proper parameter setting is for obtaining the desired solution. A relatively "soft" barrier defined through the initial setting of ϵ and δ as specified above, in conjunction with big weight matrices Q_T and R_T, is crucial to make the trajectories "jump" away from the initial curve (straight lines) during the first set of iterations. As a result, the vehicles pass the potential collision points that define the peaks of the barrier functional in the desired manner. In later (outer) iterations, when ϵ and δ are gradually reduced to make the barrier stiffer and force the solution into a collision-free area, the trajectories are more and more fixed to the respective side of the peak of the barrier functional. Thus, trajectories that are feasible in terms of the mission specifications are maintained even when the barrier terms' influence on the cost function diminishes with increasing stiffness of the barrier.

C. Obstacle Avoidance

As an example of how a differently weighted trade-off among the energy, trajectory tracking, and collision avoidance components of the cost function may yield very different solutions, consider the scenario shown in Figure 8. Here, we solve the problem of making two vehicles maneuver across a field of obstacles using two strategies. In the first strategy, the vehicles are given desired spatial trajectories that are simply straight lines from the initial to the final end-points (notice that the straight lines intersect the obstacles and are therefore not feasible). During the optimization process, the planner modifies the trajectories in order to avoid collisions while requiring minimum energy usage for doing so. In the second strategy, the planner is not given any desired (straight line) paths.

The obstacle field was randomly generated using a uniform distribution in $[0.0\mathrm{m}, 50.0\mathrm{m}]$ for the obstacle positions and in $[1.0\mathrm{m}, 3.0\mathrm{m}]$ for their radii. The result of an optimization run is shown in Figure 8a: the resultant trajectories stay as close as possible to the desired trajectories. The comparatively "sharp" turns are connected to a reduction of the corresponding velocities and feasible in terms of the vehicle dynamics. A zoomed view on a segment of the trajectories is shown in Figure 8b: the resultant trajectories maintain the feasibility criteria defined by the collision and obstacle avoidance constraints, visualized here by the circles around the red-rimmed robot pictograms. These circles mark the security margin around each robot and are plotted only at those instants of time where the β_δ functional has a peak related to a local minimum of the inter-vehicle distance.

It is interesting to compare this result with the one obtained from pure trajectory generation, (Figure 8c) using what previously were the desired trajectories as an initial curve in the iterative optimization process. Otherwise, i.e., in terms of obstacle data and boundary conditions, the scenario is identical to the previous one. The trajectories obtained are smoother, but at the expense of obtaining longer paths to follow. The final time T was the same in both cases. However, since the trajectories were not required to "stay close" to desired trajectories defined a priori, the algorithm was able to converge much faster than for the initial problem setting.

VI. Conclusion

We presented a numerical method for solving motion planning problems for multiple vehicles. The method takes explicitly into account the vehicle dynamics, and it generates and optimizes the corresponding trajectories so that the energy usage is minimized while satisfying temporal and spatial constraints, such as simultaneous arrival and the avoidance of collisions among the vehicles and with obstacles. We emphasize that, in our approach, energy usage is not to be

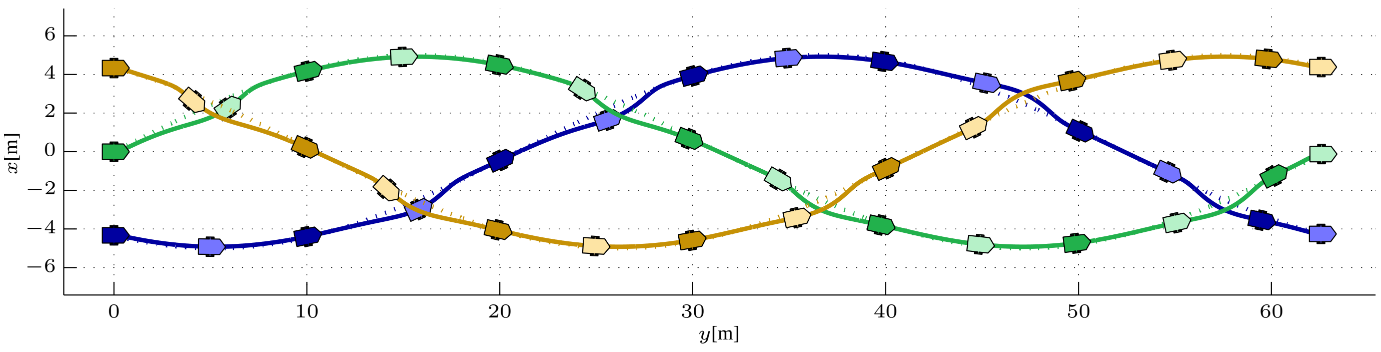


Figure 7. A "braid" maneuver for three vehicles. Desired trajectories are shown as dashed lines.

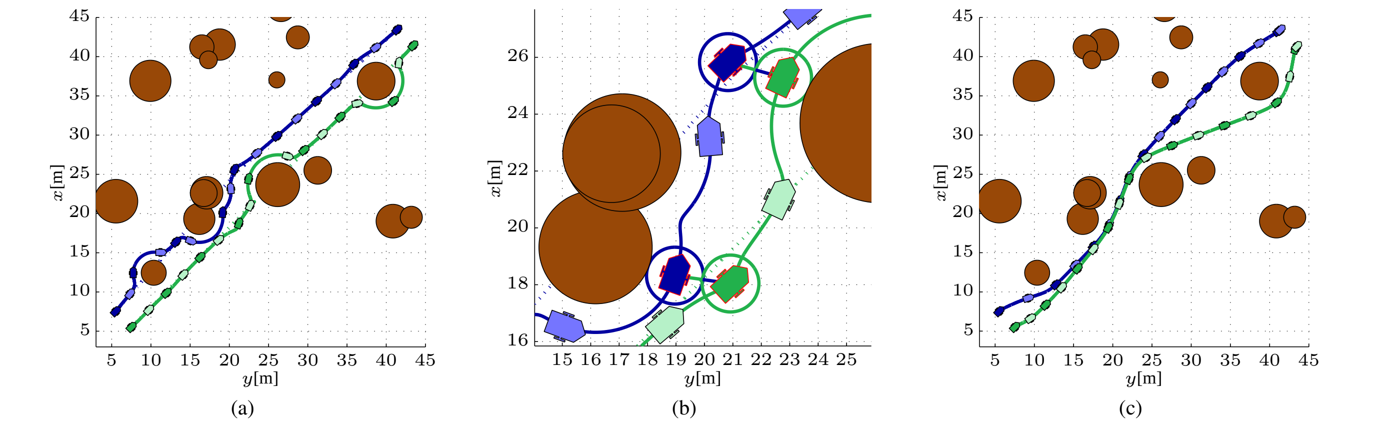


Figure 8. Two trajectories to maneuver across a random obstacle field. This scenario shows the versatility that can be achieved by combining a non-feasible "desired" trajectory with a cautious setting of the weights on the barrier functionals (Figure 8a). In Figure 8b, we show a zoomed portion of that solution, and Figure 8c gives an alternative solution that was obtained by *not* specifying a desired trajectory.

understood in the purely mechanical sense (i.e., the integral over the mechanical power obtained by multiplying each wheel's torque by the wheel's rotational velocity or the thrust force by the forward speed in the case of a two-wheeled robot or an underwater vehicle, respectively). Instead, the energy computation is done by including explicitly in the problem formulation models of the actuators and energy sources used to power the vehicles. In the case of a wheeled robot, this yields the energy drawn from an electric power source. The generated trajectories are feasible in terms of the vehicle dynamics, optimal in the sense of the energy consumed, and they ensure that, when executed by a trajectory tracking controller, the vehicles neither collide with each other nor with obstacles present in the environment. We remind the reader that the proposed approach requires the vehicle dynamics to be twice differentiable, and the analytical expressions need to be implemented as part of the problem setting, since the approach requires solving differential equations.

At its core, the algorithm makes use of a projection operator based Newton descent method for unconstrained trajectory optimization problems. Constraints imposed by inter-vehicle collision avoidance and avoidance of obstacles are converted into additive terms in the cost functional by the use of an existent barrier function method that we improved to capture the particularities of inter-vehicle and obstacle collision avoidance. It is important to remark that we achieve asymptotic convergence of the solution; in fact, the solutions converge to a local minimizer that fulfills the second order sufficiency condition (SSC). Since a solution obtained with the algorithm is second-order sufficient, we may claim that the overall problem presented and solved in this paper is well-posed.

In optimization, the initialization phase is an important part of the problem, and our planner is no exception. Coordination space planning [26, Section 7.2.2] might be a useful addition as a pre-planning heuristic to obtain a good initial guess. Of course, such an exploitation of the topological properties of the problem comes at the expense of biasing the solution toward a particular local minimum even before the core trajectory optimization starts. This effect might be reduced by using as a planner on the coordination space an algorithm that is weighted with the expected energy usage, e.g. A* with an appropriately designed goal function. However, since constraints as inter-vehicle collision avoidance immediately render the optimization problem non-convex anyway, such a "pre-planning" approach can be expected to help in finding a better local optimum.

In the future, we plan to extend our planning system by adding a waypoint finding method as a pre-planning step that incorporates topological information from the environment as well as a desired configuration of the vehicle formation. Because the robots' equations of motion are explicitly incorporated into the planner instead of merely implemented

Table III
LIST OF VARIABLES USED IN DERIVING THE GROUND ROBOT MODEL.

Var.	Description	Unit
b	motor viscous friction coefficient	[N][m][s]
ρ_b	wheel displacement on the axle (axle aligned with robot's y axis)	[m]
ρ_w	radius of one of the wheels	[m]
m_b	mass of the robot's body	[kg]
m_w	mass of one of the wheels	[kg]
$\dot{\theta}_\mathrm{R}$	right wheel rotational velocity	[rad]/[s]
$\dot{\theta}_\mathrm{L}$	left wheel rotational velocity	[rad]/[s]
u_R	right wheel longitudinal velocity	[m]/[s]
u_L	left wheel longitudinal velocity	[m]/[s]
F_R	force produced by right wheel	[N]
F_L	force produced by left wheel	[N]
J_b	inertia of the robot's body	[kg][m]^2
J_w	inertia of each wheel	[kg][m]^2
J_r	rotor inertia of one of the motors	[kg][m]^2
I_L	motor current of left motor	[A]
I_R	motor current of right motor	[A]
K_t	motor torque constant	[N][m]/[A]

as e.g. inequality constraints on velocities, accelerations and path curvature, it will be relatively easy to adapt the planner to a more complex vehicle model. We also plan to tune the algorithm and test it in the course of field trials with a fleet of autonomous marine robots. A further step aimed at extending the planner will be its decentralization (see e.g. [24, 37]) so that planning can be executed faster. This will also require, amongst other extensions, the implementation of a communication topology based on the robots' formation.

APPENDIX A
DERIVATION OF THE ROBOT DYNAMICS

We present a model of a two-wheeled ground robot with linear and rotational dynamics and ideal tires, i.e., no sideslip and ideal rolling of the wheels. The variables describing the robot's physical dimensions are detailed in Table III. The model can be obtained from first physics principles. We chose to present it in detail in order to stress the fact that in our approach the model includes not only the vehicle dynamics but also its actuators (electrical motors). This allows for the computation of the electrical energy that is drawn from the batteries that power the motors. This is in contrast to a number of approaches, where only the output mechanical energy is computed.

The kinematic equations of the vehicle are given by

$$\dot{x} = u\cos\psi \tag{A-1}$$
$$\dot{y} = u\sin\psi \tag{A-2}$$
$$\dot{\psi} = r\,. \tag{A-3}$$

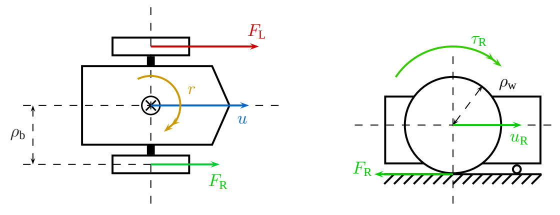
Its dynamical model can be written as a set of equations consisting of the linear (forward) and rotational dynamics of a body, subjected to the two forces F_L and F_R in Figure 9, together with the dynamics of the motors for the left and right wheel. Assuming that the moments of inertia of the latter are equal, this yields the equations

$$(m_\mathrm{b} + 2m_\mathrm{w})\dot{u} = F_\mathrm{L} + F_\mathrm{R} \tag{A-4}$$
$$J_\mathrm{b}\dot{r} = \rho_\mathrm{b}(F_\mathrm{L} - F_\mathrm{R}) \tag{A-5}$$
$$(J_\mathrm{r} + J_\mathrm{w})\ddot{\theta}_\mathrm{L} = K_\mathrm{t} I_\mathrm{L} - b\dot{\theta}_\mathrm{L} - \rho_\mathrm{w} F_\mathrm{L} \tag{A-6}$$
$$(J_\mathrm{r} + J_\mathrm{w})\ddot{\theta}_\mathrm{R} = K_\mathrm{t} I_\mathrm{R} - b\dot{\theta}_\mathrm{R} - \rho_\mathrm{w} F_\mathrm{R}\,, \tag{A-7}$$

where we consider the torques

$$K_\mathrm{t} I_\mathrm{L} = \tau_\mathrm{L} \tag{A-8}$$
$$K_\mathrm{t} I_\mathrm{R} = \tau_\mathrm{R} \tag{A-9}$$

as model inputs. The terms $\rho_\mathrm{w} F_\mathrm{L}$ and $\rho_\mathrm{w} F_\mathrm{R}$ that appear in (A-6) and (A-7) constitute the reaction force that is imparted on the ground by the vehicle in motion.



(a) Top view. (b) Side view (right).

Figure 9. Illustration of the robot's dynamics when undergoing a right turn. The figure shows the position of the wheels in body coordinates and the forces and induced torque acting on the vehicle's body.

The robot dynamic equations are accompanied by the following kinematic equations describing the velocities of the wheels at the contact points, computed as the sum of the body's forward velocity and the tangential velocity of the wheels:

$$u_\mathrm{L} = u + \rho_\mathrm{b} r \tag{A-10}$$
$$u_\mathrm{R} = u - \rho_\mathrm{b} r\,. \tag{A-11}$$

We further add a set of nonholonomic constraints (describing the velocities of the wheels at their contact points under the assumption that the only friction in the system comes from the motors, i.e., the wheels are rolling freely):

$$u_\mathrm{L} = \rho_\mathrm{w}\dot{\theta}_\mathrm{L} \tag{A-12}$$
$$u_\mathrm{R} = \rho_\mathrm{w}\dot{\theta}_\mathrm{R}\,. \tag{A-13}$$

With (A-10) and (A-12), together with (A-11) and (A-13), the wheels' rotational velocities can be described in terms of the body velocities as

$$\dot{\theta}_\mathrm{L} = \frac{1}{\rho_\mathrm{w}}(u + \rho_\mathrm{b} r) \tag{A-14}$$
$$\dot{\theta}_\mathrm{R} = \frac{1}{\rho_\mathrm{w}}(u - \rho_\mathrm{b} r)\,. \tag{A-15}$$

Using (A-14) and (A-15) and their time derivatives, we now derive a set of linear relations that later on will be needed for substitution in the vehicle model, namely

$$\dot{\theta}_\mathrm{L} + \dot{\theta}_\mathrm{R} = \frac{2}{\rho_\mathrm{w}}u \qquad \ddot{\theta}_\mathrm{L} + \ddot{\theta}_\mathrm{R} = \frac{2}{\rho_\mathrm{w}}\dot{u} \tag{A-16}$$
$$\dot{\theta}_\mathrm{L} - \dot{\theta}_\mathrm{R} = \frac{2\rho_\mathrm{b}}{\rho_\mathrm{w}}r \qquad \ddot{\theta}_\mathrm{L} - \ddot{\theta}_\mathrm{R} = \frac{2\rho_\mathrm{b}}{\rho_\mathrm{w}}\dot{r}\,. \tag{A-17}$$

Table IV
LIST OF VARIABLES USED IN DERIVING THE INTEGRAL COST TERM OF THE ROBOT'S ENERGY USAGE.

Variable	Description	Unit
V_L	left motor armature voltage	[V]
V_R	right motor armature voltage	[V]
K_e	motor back EMF constant	[V][s][rad]
L_a	armature inductance	[H]
R_a	armature resistance	[Ω]
P_p	hotel payload power consumption	[W]
l_pow	total power needed by the robot	[W]

Our aim is to find a final model with just two dynamic equations, i.e., one for the forward dynamics and another for the heading dynamics, that take the motor torques as inputs. We begin by deriving the forward dynamics by adding (A-7) to (A-6), resulting in

$$(J_\mathrm{r}+J_\mathrm{w})(\ddot\theta_\mathrm{L}+\ddot\theta_\mathrm{R}) = \tau_\mathrm{L}+\tau_\mathrm{R} - b(\dot\theta_\mathrm{L}+\dot\theta_\mathrm{R}) - \rho_\mathrm{w}(F_\mathrm{L}+F_\mathrm{R})\,. \tag{A-18}$$

Inserting (A-16) and (A-4) in the above equation yields

$$\frac{2}{\rho_\mathrm{w}}(J_\mathrm{r}+J_\mathrm{w})\dot u = \tau_\mathrm{L}+\tau_\mathrm{R} - \frac{2b}{\rho_\mathrm{w}}u - \rho_\mathrm{w}(m_\mathrm{b}+2m_\mathrm{w})\dot u \tag{A-19}$$

$$\Big(m_\mathrm{b}+2m_\mathrm{w}+\frac{2}{\rho_\mathrm{w}^2}(J_\mathrm{r}+J_\mathrm{w})\Big)\dot u = \frac{1}{\rho_\mathrm{w}}(\tau_\mathrm{L}+\tau_\mathrm{R}) - \frac{2b}{\rho_\mathrm{w}^2}u \tag{A-20}$$

and finally

$$\Big(m_\mathrm{b}+2m_\mathrm{w}+2\frac{1}{\rho_\mathrm{w}^2}(J_\mathrm{r}+J_\mathrm{w})\Big)\dot u = \frac{1}{\rho_\mathrm{w}}(\tau_\mathrm{L}+\tau_\mathrm{R}) - \frac{2b}{\rho_\mathrm{w}^2}u\,, \tag{A-21}$$

which is the equation for the robot's forward dynamics. The correctness of (A-21) can be verified by considering the units: both sides of the equation represent forces.

In a similar manner, we derive the heading dynamics by subtracting (A-7) from (A-6), which yields

$$(J_\mathrm{r}+J_\mathrm{w})(\ddot\theta_\mathrm{L}-\ddot\theta_\mathrm{R}) = \tau_\mathrm{L}-\tau_\mathrm{R} - b(\dot\theta_\mathrm{L}-\dot\theta_\mathrm{R}) - \rho_\mathrm{w}(F_\mathrm{L}-F_\mathrm{R})\,. \tag{A-22}$$

Using (A-17) and (A-5), this equation can be rewritten as

$$\frac{2\rho_\mathrm{b}}{\rho_\mathrm{w}}(J_\mathrm{r}+J_\mathrm{w})\dot r = \tau_\mathrm{L}-\tau_\mathrm{R} - \frac{2\rho_\mathrm{b}}{\rho_\mathrm{w}}br - \frac{\rho_\mathrm{w}}{\rho_\mathrm{b}}J_\mathrm{b}\dot r \tag{A-23}$$

leading to the rotational dynamics

$$\Big(J_\mathrm{b}+2\frac{\rho_\mathrm{b}^2}{\rho_\mathrm{w}^2}(J_\mathrm{r}+J_\mathrm{w})\Big)\dot r = \frac{\rho_\mathrm{b}}{\rho_\mathrm{w}}(\tau_\mathrm{L}-\tau_\mathrm{R}) - \frac{2\rho_\mathrm{b}^2}{\rho_\mathrm{w}^2}br\,. \tag{A-24}$$

The same analysis as before can be made for the robot's heading dynamics (A-24): both sides of the equation have to be torques. This is obviously true for the left-hand side, where relative inertia terms are multiplied with the rotational acceleration. The first term on the right-hand side constitutes the input torques with a dimensionless constant multiplier, and the second term represents the rotational damping.

Using (A-21), (A-24), and the vehicle kinematics (A-1) to (A-3), the final robot model becomes a nonholonomic mechanical model with damping terms and no dynamic coupling, consisting of five states, given by

$$\begin{aligned}\dot x &= u\cos\psi\\ \dot y &= u\sin\psi\\ \dot\psi &= r\\ \bar m\dot u &= -\frac{2b}{\rho_\mathrm{w}^2}u + \frac{1}{\rho_\mathrm{w}}(\tau_\mathrm{L}+\tau_\mathrm{R})\\ \bar J\dot r &= -\frac{2\rho_\mathrm{b}^2}{\rho_\mathrm{w}^2}br + \frac{\rho_\mathrm{b}}{\rho_\mathrm{w}}(\tau_\mathrm{L}-\tau_\mathrm{R})\,,\end{aligned} \tag{A-25}$$

where the motor torques τ_R and τ_L are the system inputs. The auxiliary variables are defined as

$$\begin{aligned}\bar m &= m_\mathrm{b}+2m_\mathrm{w}+2\frac{1}{\rho_\mathrm{w}^2}J_\mathrm{w}\\ \bar J &= J_\mathrm{b}+2\frac{\rho_\mathrm{b}^2}{\rho_\mathrm{w}^2}J_\mathrm{w}\,.\end{aligned} \tag{A-26}$$

For clarity of the exposition, the model can be written as

$$\begin{aligned}\dot x &= u\cos\psi\\ \dot y &= u\sin\psi\\ \dot\psi &= r\\ \bar m\dot u &= c_1u + c_2(\tau_\mathrm{L}+\tau_\mathrm{R})\\ \bar J\dot r &= c_3r + c_4(\tau_\mathrm{L}-\tau_\mathrm{R})\,,\end{aligned} \tag{A-27}$$

where c_1, c_2, c_3 and c_4 are parameters defined as

$$c_1 = -\frac{2b}{\rho_\mathrm{w}^2} \quad c_2 = \frac{1}{\rho_\mathrm{w}} \quad c_3 = -\frac{2\rho_\mathrm{b}^2}{\rho_\mathrm{w}^2}b \quad c_4 = \frac{\rho_\mathrm{b}}{\rho_\mathrm{w}}\,. \tag{A-28}$$

APPENDIX B
COMPUTING THE INSTANTANEOUS POWER REQUIREMENT

We aim to minimize the vehicle's energy consumption by solving a dynamically constrained optimization problem, where the constraints represent the vehicle dynamics. For this reason, we need to model the actual total power consumed by the robot, which is given by

$$l_\mathrm{pow} = I_\mathrm{L}V_\mathrm{L} + I_\mathrm{R}V_\mathrm{R} + P_\mathrm{p}\,. \tag{B-1}$$

In this equation, P_p represents the instantaneous power that is consumed by the hotel payload, e.g. on-board computers and sensors, which, for the simlicity of exposition, we assume to be constant. All variables needed in the following section are given in Table IV[6]. The robot's complete electrical system is depicted in Figure 10: it contains two DC motors and a resistance representing the constant consumption of the computer system.

The equations associated with the electric circuits (or armatures) of the two DC motors can be written as

$$L_\mathrm{a}\rho_\mathrm{b}\frac{d}{dt}I_\mathrm{L} + R_\mathrm{a}I_\mathrm{L} = V_\mathrm{L} - K_\mathrm{e}\dot\theta_\mathrm{L} \tag{B-2}$$

$$L_\mathrm{a}\rho_\mathrm{b}\frac{d}{dt}I_\mathrm{R} + R_\mathrm{a}I_\mathrm{R} = V_\mathrm{R} - K_\mathrm{e}\dot\theta_\mathrm{R}\,, \tag{B-3}$$

[6]We recall that, in consistent units (that is, K_t in [N][m]/[A] and K_e in [V][s]/[rad]), we have the numerical equivalence $K_\mathrm{t} = K_\mathrm{e}$—see e.g. [11, Figure 2.26].

where $_\mathrm{L}$ and $_\mathrm{R}$ refer to the right and left wheels, respectively. There is virtually no interference between the three parallel circuits, so the motors can be dealt with in a decoupled manner using separate equations.

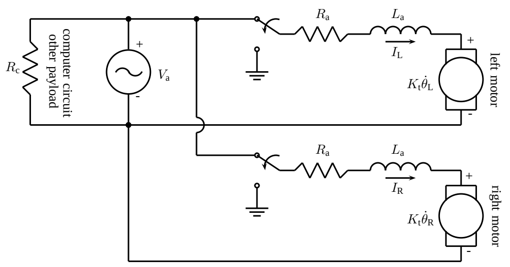


Figure 10. The robot's electrical circuit.

For our purpose it is sufficient to consider the motors' electrical equations at steady-state (obtained by neglecting the inductances in the armature circuits), so that we remain with static equations. Making the reasonable assumption that both the left and the right motor are equally fabricated, we can rewrite (B-2) and (B-3) as

$$\begin{aligned} V_\mathrm{L} &= R_\mathrm{a} I_\mathrm{L} + K_\mathrm{e}\dot{\theta}_\mathrm{L} \\ V_\mathrm{R} &= R_\mathrm{a} I_\mathrm{R} + K_\mathrm{e}\dot{\theta}_\mathrm{R}\,. \end{aligned} \tag{B-4}$$

Notice that (A-8) and (A-9) can be rewritten as

$$\begin{aligned} I_\mathrm{L} &= \tau_\mathrm{L}/K_\mathrm{t} \\ I_\mathrm{R} &= \tau_\mathrm{R}/K_\mathrm{t}\,. \end{aligned} \tag{B-5}$$

By substituting (B-4) and (B-5) into (B-1), and making use of the equations for for the rotational velocities (A-14) and (A-15), we obtain the instantaneous power

$$\begin{aligned} l_\mathrm{pow} = R_\mathrm{a}\frac{\tau_\mathrm{L}^2+\tau_\mathrm{R}^2}{K_\mathrm{t}^2} + \frac{K_\mathrm{e}}{K_\mathrm{t}}\Big(\frac{\tau_\mathrm{L}}{\rho_\mathrm{w}}(u+\rho_\mathrm{b} r) \\ + \frac{\tau_\mathrm{R}}{\rho_\mathrm{w}}(u-\rho_\mathrm{b} r)\Big) + P_\mathrm{p}\,, \end{aligned} \tag{B-6}$$

which forms the cost for the optimization problem. Notice that the first summand is a dissipative term related to the electrical circuit, while the second term is the *mechanical power* required to move the robot.

Ignoring the constant P_p, we can write (B-6) in matrix form as

$$l_\mathrm{pow} = \begin{bmatrix}\mathbf{x}(\tau)\\ \mathbf{u}(\tau)\end{bmatrix}^\mathsf{T} \begin{bmatrix} Q_\mathrm{E} & S_\mathrm{E} \\ S_\mathrm{E}^\mathsf{T} & R_\mathrm{E}\end{bmatrix} \begin{bmatrix}\mathbf{x}(\tau)\\ \mathbf{u}(\tau)\end{bmatrix}, \tag{B-7}$$

where the quadratic cost is determined by the matrices

$$\begin{aligned} Q_\mathrm{E} &= 0_{5\times 5} \qquad R_\mathrm{E} = \begin{bmatrix} \frac{R_\mathrm{a}}{K_\mathrm{t}^2} & 0 \\ 0 & \frac{R_\mathrm{a}}{K_\mathrm{t}^2}\end{bmatrix} \\ S_\mathrm{E} &= \begin{bmatrix} 0 & 0 & 0 & \frac{K_\mathrm{e}}{2K_\mathrm{t}\rho_\mathrm{w}} & \frac{\rho_\mathrm{b}K_\mathrm{e}}{2K_\mathrm{t}\rho_\mathrm{w}} \\ 0 & 0 & 0 & \frac{K_\mathrm{e}}{2K_\mathrm{t}\rho_\mathrm{w}} & -\frac{\rho_\mathrm{b}K_\mathrm{e}}{2K_\mathrm{t}\rho_\mathrm{w}} \end{bmatrix}^\mathsf{T}. \end{aligned} \tag{B-8}$$

It is easy to verify that the quadratic form (B-7) has seven eigenvalues, four of which are non-zero. In particular, the matrix has two positive eigenvalues corresponding to dissipating energy, and two negative ones corresponding to recovering energy from the motion. Note also that, since all constants in (B) are positive, the magnitude of the dissipating eigenvalues is larger than the magnitude of the energy storing eigenvalues, which confirms the intuition that dissipating energy is "easier" than storing it.

References

[1] B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, Mineola (NY), USA, 2007. ISBN 9780486457666.

[2] L. Armijo. Minimization of Functions Having Lipschitz Continuous First Partial Derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[3] J. T. Betts. *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*, volume 19 of *Advances in design and control*. SIAM, Philadelphia (PA), USA, 2nd edition, 2010. ISBN 978-0898716887.

[4] C. Bottasso, D. Leonello, and B. Savini. Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives. *IEEE Transactions on Control Systems Technology*, 16(6):1152–1168, 2008.

[5] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK and New York (NY), USA, 2004. ISBN 9780521833783.

[6] V. Delsart, T. Fraichard, and L. Martinez. Real-Time Trajectory Generation for Car-like Navigating Dynamic Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3401–3406, 2009.

[7] M. G. Earl and R. D'Andrea. Iterative MILP Methods for Vehicle-Control Problems. *IEEE Transactions on Robotics*, 21(6):1158–1167, 2005.

[8] A. Engelsone. *Direct Transcription Methods in Optimal Control: Theory and Practice*. PhD thesis, North Carolina State University, Raleigh (NC), USA, 2006.

[9] F. Fahimi. *Autonomous Robots: Modeling, Path Planning, and Control*. Springer US, Boston (MA), USA, 2009. ISBN 978-0-387-09537-0.

[10] E. Fiorelli, N. E. Leonard, P. Bhatta, D. A. Paley, R. Bachmayer, and D. M. Fratantoni. Multi-AUV Control and Adaptive Sampling in Monterey Bay. *IEEE Journal of Oceanic Engineering*, 31(4):935–948, 2006.

[11] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall, Upper Saddle River (NJ), USA, 4th edition, 2002. ISBN 0-13-032393-4.

[12] I. Garcia and J. P. How. Trajectory Optimization for Satellite Reconfiguration Maneuvers with Position and Attitude Constraints. In *Proceedings of the 2005 American Control Conference (ACC)*, volume 2, pages 889–894, 2005.

[13] Q. Gong, W. Kang, and I. M. Ross. A Pseudospectral Method for the Optimal Control of Constrained Feedback Linearizable Systems. In *Proceedings of the 2005 IEEE Conference on Control Applications (CCA)*, pages 1033–1038, 2005.

[14] M. Häselich, N. Handzhinyski, C. Winkens, and D. Paulus. Spline Templates for Fast Path Planning in

Unstructured Environments. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3545–3550, 2011.

[15] J. Hauser. A Projection Operator Approach to the Optimization of Trajectory Functionals. In *Proceedings of the 15th IFAC World Congress*, volume 15, pages 310–315, 2002.

[16] J. Hauser. On the Computation of Optimal State Transfers with Application to the Control of Quantum Spin Systems. In *Proceedings of the 2003 American Control Conference (ACC)*, pages 2169–2174, 2003.

[17] J. Hauser and D. G. Meyer. The Trajectory Manifold of a Nonlinear Control System. In *Proceedings of the 37th IEEE Conference on Decision and Control (CDC)*, volume 1, pages 1034–1039, 1998.

[18] J. Hauser and A. Saccon. A Barrier Function Method for the Optimization of Trajectory Functionals with Constraints. In *Proceedings of the 45th IEEE Conference on Decision and Control (CDC)*, pages 864–869, 2006.

[19] A. J. Häusler, R. Ghabcheloo, A. M. Pascoal, and A. P. Aguiar. Multiple Marine Vehicle Deconflicted Path Planning with Currents and Communication Constraints. In *Proceedings of the 7th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, volume 7, pages 491–496, 2010.

[20] T. M. Howard and A. Kelly. Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots. *The International Journal of Robotics Research*, 26(2):141–166, 2007.

[21] B. Johnson and R. Lind. Improving Tree-Based Trajectories through Order Reduction/Expansion and Surrogate Models. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2010.

[22] I. I. Kaminer, O. A. Yakimenko, V. Dobrokhodov, A. M. Pascoal, N. Novakimyan, C. Cao, A. Young, and V. Patel. Coordinated Path Following for Time-Critical Missions of Multiple UAVs via $\mathcal{L}_1$ Adaptive Output Feedback Controllers. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2007.

[23] T. Kröger and F. M. Wahl. Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events. *IEEE Transactions on Robotics*, 26(1):94–111, 2010.

[24] Y. Kuwata and J. How. Decentralized Cooperative Trajectory Optimization for UAVs with Coupling Constraints. In *Proceedings of the 45th IEEE Conference on Decision and Control (CDC)*, pages 6820–6825, 2006.

[25] C.-K. Lai, M. Lone, P. Thomas, J. Whidborne, and A. Cooke. On-Board Trajectory Generation for Collision Avoidance in Unmanned Aerial Vehicles. In *Proceedings of the IEEE Aerospace Conference*, pages 1–14, 2011.

[26] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, UK and New York (NY), USA, 2006. ISBN 978-0521862059.

[27] J.-W. Lee and H. J. Kim. Trajectory Generation for Rendezvous of Unmanned Aerial Vehicles with Kinematic Constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1056–1061, 2007.

[28] F.-L. Lian and R. M. Murray. Real-Time Trajectory Generation for the Cooperative Path Planning of Multi-Vehicle Systems. In *Proceedings of the 41st IEEE Conference on Decision and Control (CDC)*, pages 3766–3769, 2002.

[29] T. G. McGee and J. K. Hedrick. Path Planning and Control for Multiple Point Surveillance by an Unmanned Aircraft in Wind. In *Proceedings of the 2006 American Control Conference (ACC)*, pages 4261–4266, 2006.

[30] B. Mettler and Z. Kong. Receding Horizon Trajectory Optimization with a Finite-State Value Function Approximation. In *Proceedings of the 2008 American Control Conference (ACC)*, pages 3810–3816, 2008.

[31] A. Saccon, J. Hauser, and A. P. Aguiar. Optimal Control on Lie Groups: The Projection Operator Approach. *IEEE Transactions on Automatic Control*, 58(9):2230–2245, 2013.

[32] H.-S. Shin, C. Leboucher, and A. Tsourdos. Resource Allocation with Cooperative Path Planning fur Multiple UAVs. In *Proceedings of the UKACC International Conference on Control (CONTROL)*, pages 298–303, 2012.

[33] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning, and Control*. Springer, London, UK, 2009. ISBN 978-1-84628-642-1.

[34] A. Tsourdos, B. White, and M. Shanmugavel. *Path Planning Strategies for Cooperative Autonomous Air Vehicles*. Aerospace series (PEP). Wiley-Blackwell, Oxford, UK, 2010. ISBN 978-0-470-74129-0.

[35] J. van den Berg and M. Overmars. Roadmap-based motion planning in dynamic environments. *IEEE Transactions on Robotics*, 21(5):885–897, 2005.

[36] M. J. van Nieuwstadt and R. M. Murray. Real-Time Trajectory Generation for Differentially Flat Systems. *International Journal of Robust & Nonlinear Control*, 8 (11):995–1020, 1998.

[37] E. Wei, A. Ozdaglar, and A. Jadbabaie. A Distributed Newton Method for Network Utility Maximization—I: Algorithm. *IEEE Transactions on Automatic Control*, 58 (9):2162–2175, 2013.

[38] E. Xargay, V. N. Dobrokhodov, I. I. Kaminer, A. M. Pascoal, N. Hovakimyan, and C. Cao. Time-Critical Cooperative Control of Multiple Autonomous Vehicles: Robust Distributed Strategies for Path-Following Control and Time-Coordination over Dynamic Communications Networks. *IEEE Control Systems Magazine*, 32(5):49–73, 2012.