Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment

Tim Mercy, Ruben Van Parys, and Goele Pipeleers

***Abstract*— Autonomous vehicles require a collision-free motion trajectory at every time instant. This brief presents an optimization-based method to calculate such trajectories for autonomous vehicles operating in an uncertain environment with moving obstacles. The proposed approach applies to linear system models, as well as to a particular class of nonlinear models, including industrially relevant vehicles, such as autonomous guided vehicles with front wheel, differential wheel, and rear-wheel steering. The method computes smooth motion trajectories, satisfying the vehicle's kinematics, by using a spline parameterization. Furthermore, it exploits spline properties to keep the resulting nonlinear optimization problem small scale and to guarantee constraint satisfaction, without the need for time gridding. The resulting problem is solved sufficiently fast for online motion planning, dealing with uncertainties and changes in the environment. This brief demonstrates the potential of the method with extensive numerical simulations. In addition, it presents an experimental validation in which a *KUKA youBot*, steered as a holonomic or differential drive vehicle, drives through an environment with moving obstacles. To facilitate the further development and the numerical and experimental validation of the presented method, it is embodied in a user-friendly open-source software toolbox.**

***Index Terms*— Autonomous vehicles, motion control, optimal control, splines, trajectory optimization.**

I. Introduction

AUTONOMOUS GUIDED VEHICLES (AGVs) are becoming more and more prevalent in modern industry. They are, for instance, essential for material transport through workshops and eCommerce warehouses, and are often used for container transport through large ports. In all applications, the AGVs require collision-free and safe motion trajectories at every time instant. To date, the computation of such trajectories is simplified by strongly conditioning the environments in which the AGVs operate: they are forced to move over dedicated tracks, are separated from humans, and their velocities are limited. However, these restrictions also reduce the performance, and therefore the potential of AGVs, to a great extent. As a consequence, there is a growing interest in more flexible motion planning techniques, allowing free motion of AGVs through warehouses and ports. Furthermore, this would unlock the possibility of using AGVs in less conditioned environments, which occur, for instance, in fruit picking, lawn mowing, crop harvesting, or personal assistance applications. Flexible navigation of AGVs through unconditioned, uncertain, and dynamic environments involves challenges in vehicle localization, safety, and, in particular, in trajectory generation.

Computing a trajectory that brings the vehicle from its current position to its destination as fast or as energy efficient as possible naturally translates into solving an optimal control problem (OCP). This OCP takes into account the vehicle's motion model and its limits. In addition, collision with all obstacles in the environment must be avoided, which introduces nonconvex constraints in the OCP. In many applications, the obstacle avoidance constraints are also subject to uncertainty, as the motion trajectories of moving obstacles are generally not fully known *a priori*, and the new obstacles may show up and the present obstacles may disappear from the environment. This uncertainty requires solving the OCP online, in order to always account for the latest information.

In motion planning, a distinction is generally made between coupled and decoupled approaches [1]. A coupled motion planning approach solves the resulting OCP in one step and returns both the optimal motion trajectory and the corresponding controls to steer the vehicle over the trajectory. However, because of the complexity of the problem, most coupled approaches consider a simplified environment (see [2]) and are often too slow to be used online. Therefore, most existing methods apply a decoupled approach, splitting the problem in a path planning and a path following phase. Various techniques have been developed to solve the path planning phase, including graph-based methods, such as A* [3] and D*-lite [4], random sampling-based methods, such as rapidly exploring random trees (RRT) [5] and probabilistic road maps (PRM) [6], and optimization-based methods, such as Trajopt [7] and ITOMP [8]. The path following phase often uses model predictive control (MPC) [9] to steer the system along the computed path, as MPC can explicitly account for the system limitations. While decoupled approaches generally lead to easier problems than coupled ones, they return suboptimal results. Furthermore, the computed paths are often infeasible, because the vehicle's kinematics and its limitations are not taken into account in the path planning phase. Recent developments [10], [11] each describes a path planning approach, which does allow accounting for the vehicle

Manuscript received December 23, 2016; revised June 8, 2017; accepted August 7, 2017. Date of publication August 30, 2017; date of current version October 9, 2018. Manuscript received in final form August 10, 2017. This work was supported in part by KU Leuven through the Centre of Excellence: Optimization in Engineering under Grant BOF PFV/10/002, in part by the Belgian Program on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office, in part by the Research Foundation-Flanders (FWO-Flanders) under Project G0C4515N, in part by the KU Leuven Research: B-spline-based certificates of positivity with applications in engineering under Project C14/15/067, and in part by the IWT ICON Project Sitcontrol of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) and Flanders Make. The work of R. Van Parys was supported by a Ph.D. fellowship from the Research Foundation-Flanders. Recommended by Associate Editor G. Notarstefano. *(Corresponding author: Tim Mercy.)*

The authors are with Flanders Make and the MECO Research Group, Department of Mechanical Engineering, Division PMA, KU Leuven, BE-3001 Leuven, Belgium (e-mail: tim.mercy@kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2017.2739706

kinematics. However, to this end, both methods require an ad hoc smoothing functionality to connect waypoints into a feasible path.

This brief presents a novel motion planning method that uses a coupled approach to compute time-optimal trajectories for AGVs moving through uncertain and dynamic environments, taking into account the kinematics of the vehicle. To translate the resulting OCP into a small-scale nonlinear optimization problem that is applicable for online motion planning, the method uses a spline parameterization of the trajectory. In contrast to the classical time gridding approach, the method exploits the properties of B-splines to efficiently enforce constraints to hold at all times. The approach applies to linear system models, as well as to a particular class of nonlinear models, including industrially relevant vehicles, such as AGVs with front wheel, differential wheel, and rear-wheel steering. The potential of the method is experimentally validated on a setup in which a *KUKA youBot* [12] drives through an environment with moving obstacles. To facilitate modeling, simulating, and exporting motion planning problems to practical setups, the method is embodied in OMG tools, a user-friendly open-source Python toolbox [13]. A simplified version of the motion planning method was presented in [14]. This formulation only considered obstacles with a fixed orientation and a holonomic vehicle with circular symmetry (of which the orientation can be neglected). This brief extends these preliminary results to AGVs with a nonlinear model. Furthermore, it includes vehicles and obstacles with changing orientation, tests a new solver, and provides an experimental validation of the online application of the method.

Section II describes the general OCP and its online implementation. In addition, it provides a short mathematical background on splines. Section III explains how to translate the general OCP to a spline-based motion planning problem, focusing on nonholonomic vehicles and collision avoidance constraints. Afterward, Section IV validates the presented method by showing numerical simulations together with experimental results. Finally, Section V concludes this brief.

II. Problem Formulation and Mathematical Background

As explained earlier, the proposed motion planning method uses a coupled approach to compute optimized spline motion trajectories. The considered OCP is described in Section II-A, while Section II-B provides a background on B-splines and their properties.

A. Online Motion Planning

The proposed method solves an OCP to find the trajectory $q(t)$ that optimally moves the vehicle through an environment containing moving obstacles. Together with the trajectory, the problem also returns the vehicle input $u(t)$, which steers the vehicle along the optimal trajectory. To obtain a trajectory that the vehicle is physically able to follow, it is necessary to take into account its kinematics. While the method applies to various vehicle models, the remainder of this brief focuses on two examples: a holonomic vehicle with circular symmetry, for the ease of explanation; and a bicycle model, to illustrate the potential of the presented method and because of its industrial relevance. For a holonomic vehicle with circular symmetry, the state is purely given by its position trajectory

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}. \tag{1}$$

When the vehicle is steered by the x- and y-velocities, the kinematics simply express that the inputs are the derivatives of the state: $u(t) = \dot{q}(t)$. On the other hand, the kinematics of a bicycle model are described as follows [15]:

$$\begin{aligned} \dot{x}(t) &= V(t) \cdot \cos\theta(t) \\ \dot{y}(t) &= V(t) \cdot \sin\theta(t) \\ \dot{\theta}(t) &= \frac{V(t)}{L} \cdot \tan\delta(t) \end{aligned} \tag{2}$$

in which $\theta(t)$ is the orientation of the vehicle, $\delta(t)$ is its steering angle, and L is the length of the wheel base. The model imposes that the velocity $V(t)$ is always tangent to the vehicle path. The bicycle model is nonholonomic, since it has two inputs: $u(t) = [V(t), \delta(t)]^T$, while it has three degrees of freedom [16], determining the state: $q(t) = [x(t), y(t), \theta(t)]^T$. Note that a differential drive vehicle is very similar, with the main difference that $\theta(t)$ is controlled directly by the difference in wheel velocities.

The remainder of this section further elaborates the motion planning problem for the bicycle model. To ensure that the trajectory connects the vehicle's current state q_0 to the desired state q_T, the problem contains initial and final conditions on the state. Since in most cases it is desired to start from, and end in complete standstill, also constraints on the initial and final inputs are necessary. These boundary conditions are expressed as

$$\begin{aligned} q(0) &= q_0, \quad q(T) = q_T \\ u(0) &= u_0, \quad u(T) = u_T. \end{aligned} \tag{3}$$

In this equation, T represents the total motion time that is required to reach the destination.

Furthermore, the vehicle is subject to input constraints, limiting the velocity, and steering angle. In addition, the dynamics of the vehicle are implicitly accounted for by kinodynamic constraints: the input trajectories are guaranteed to be sufficiently smooth by limiting the vehicle's acceleration and the rate of change of steering angle. Combining the input and kinodynamic constraints gives

$$\begin{aligned} V_{\min} \leq V(t) \leq V_{\max} \\ \dot{V}_{\min} \leq \dot{V}(t) \leq \dot{V}_{\max} \\ \delta_{\min} \leq \delta(t) \leq \delta_{\max} \\ \dot{\delta}_{\min} \leq \dot{\delta}(t) \leq \dot{\delta}_{\max} \\ \forall t \in [0, T]. \end{aligned} \tag{4}$$

In an environment that contains (moving) obstacles, collision avoidance constraints are essential. To account for obstacle movement, the constraints include a linear prediction model for every obstacle k. This model uses the current obstacle

position p_k and velocity v_k to predict the obstacle position p_k^{pred} as a function of time

$$p_k^{\text{pred}}(t) = p_k + t \cdot v_k. \tag{5}$$

The collision avoidance constraints themselves express that the shape of the vehicle must not overlap with the shape of any of the N obstacles

$$\text{dist}(\text{veh}(q(t)), \text{obs}_k\ (p_k^{\text{pred}}(t))) \geq \epsilon \quad \forall t \in [0, T] \tag{6}$$

where $k = 1 \ldots N$ and ϵ is a safety factor to ensure that the vehicle preserves a safe distance from the obstacles.

Finally, the goal is to obtain time-optimal trajectories. To allow optimizing the motion time T, the problem is reformulated using a dimensionless time $\tau = (t/T)$. Combining all the above-mentioned elements leads to the following OCP:

$$\begin{aligned}
&\min_{q(\cdot),u(\cdot),T} \quad T \\
&\text{s.t.}\ \ q(0) = q_0, \quad q(1) = q_T \\
&\qquad u(0) = u_0, \quad u(1) = u_T \\
&\qquad V_{\min} \leq V(\tau) \leq V_{\max} \\
&\qquad \dot{V}_{\min} \cdot T \leq \dot{V}(\tau) \leq \dot{V}_{\max} \cdot T \\
&\qquad \delta_{\min} \leq \delta(\tau) \leq \delta_{\max} \\
&\qquad \dot{\delta}_{\min} \cdot T \leq \dot{\delta}(\tau) \leq \dot{\delta}_{\max} \cdot T \\
&\qquad \text{constraints (2)} \\
&\qquad \text{dist}(\text{veh}(q(\tau)), \text{obs}_k\ (p_k^{\text{pred}}(\tau))) \geq \epsilon \\
&\qquad k = 1 \ldots N, \quad \forall \tau \in [0,\ 1].
\end{aligned} \tag{7}$$

This problem is infinite dimensional, both in terms of variables and in terms of constraints, since the motion trajectories live in an infinite function space, and the state and input constraints need to hold over the entire time horizon. In general, (7) is a complex optimization problem that is hard to solve. The collision avoidance constraints are nonlinear and nonconvex. As an additional complication, there is uncertainty in the environment: the new obstacles may appear and the present obstacles may change their movement direction or velocity. Therefore, the OCP generally needs to be solved online, with a receding horizon, allowing to always include the latest information into the problem. Section III describes how to transform the problem into an efficiently solvable finite dimensional problem. This transformation relies on the properties of B-splines, which are explained next.

B. Background on Splines

Splines are piecewise polynomial functions and can be expressed as a linear combination of B-spline basis functions B_i [17]

$$s(\tau) = \sum_{i=1}^{n} c_i \cdot B_i(\tau). \tag{8}$$

The dimension n is determined by the spline degree, the number of polynomial pieces the spline is composed of, and the continuity which they are glued together with. The last two properties are fully determined by the spline knots [17]. Fig. 1 shows this definition graphically.

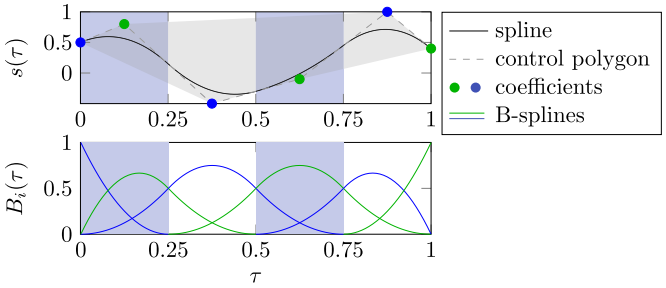


Fig. 1. Graphical illustration of a spline as a linear combination of B-spline basis functions.

The main reason for adopting the B-spline basis is the so-called convex hull property: because the basis functions are positive and sum up to one, a spline is always contained in the convex hull of its control polygon [17]. This polygon corresponds to the piecewise linear interpolation of the spline coefficients, as shown in Fig. 1. Therefore, constraints on a spline function can be enforced by imposing them on the B-spline coefficients

$$a \leq c_i \leq b, \quad \forall i \in \{1 \ldots n\} \Rightarrow a \leq s(\tau) \leq b, \ \forall \tau \in [0, 1]. \tag{9}$$

Replacing the infinite set of constraints on the right-hand side of (9) by the finite set on the left-hand side is hereafter called a B-spline relaxation.

Although B-spline relaxations provide an alternative to time gridding in order to guarantee constraint satisfaction at all times, they do introduce conservatism. This conservatism stems from the distance between the control polygon and the spline itself (see Fig. 1), since the relaxation comes down to replacing constraints on the spline by constraints on the control polygon. To reduce conservatism, the spline can be represented in a higher dimensional B-spline basis that includes the original one [18]. This basis is obtained by increasing the degree, or by inserting extra knots. Both approaches are shown in Fig. 2. Fig. 2 shows that each of the two techniques reduces the distance between the control polygon and the spline itself, and therefore reduces the introduced conservatism. Knot insertion is generally the preferred technique, since, in this way, the conservatism reduces quadratically with the number of constraints, while this decrease is only linear when using degree elevation [19]. In addition, knot insertion allows refining the control polygon only locally, whereas degree elevation always affects the entire control polygon. Since using a higher dimensional basis translates into more constraints, it is necessary to make a tradeoff between conservatism and computational complexity (number of constraints).

III. Spline-Based Motion Planning

In order to transform the OCP, describing the motion planning problem, into a small-scale nonlinear optimization problem, the method proposed in [18] is adopted. The key aspects of this method are: 1) a B-spline parameterization is used for the motion trajectory $q(\cdot)$ and 2) the properties of B-splines are exploited to enforce constraints over the entire time horizon through a small finite, yet conservative, set of constraints. Section III-A shows how to apply a spline

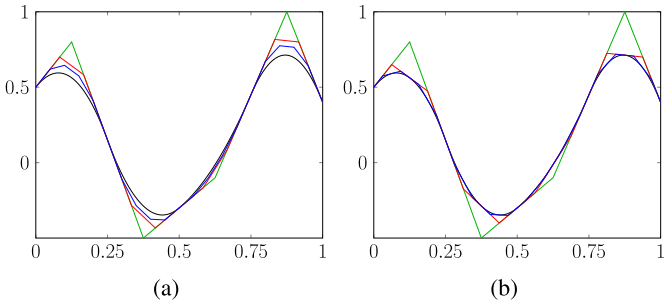


Fig. 2. Effect of (a) degree elevation and (b) knot insertion on the conservatism introduced by a spline parameterization.

parameterization to a problem with a holonomic vehicle. Afterward, Section III-B explains how to include a vehicle with a bicycle model. Finally, Section III-C describes how to add the collision avoidance constraints.

A. Spline Parameterization for Holonomic Vehicles

For a motion planning problem considering a holonomic vehicle with a circular shape, the usage of the spline parameterization and relaxations to transform the corresponding OCP into a small-scale nonlinear optimization problem is straightforward. The vehicle state $q(\tau)$, given by (1), is parameterized as a sufficiently smooth spline

$$q(\tau) = \sum_{i=1}^{n} c_i^q \cdot B_i^q(\tau) \tag{10}$$

with $c_i^q \in \mathbb{R}^{n_q}$, where n_q denotes the dimension of q. Constraints on the vehicle velocity and acceleration then amount to bounds on spline derivatives. Because the derivatives of splines are also splines, the B-spline relaxations can readily be applied to these constraints. Applying them, for instance, to the velocity constraints yields

$$V_{\min} \cdot T \leq c_i^{\dot{q}} \leq V_{\max} \cdot T \tag{11}$$

where the $c_i^{\dot{q}}$ values denote the B-spline coefficients of $\dot{q}(\tau)$, which is given by

$$\dot{q}(\tau) = \sum_{i=1}^{n-1} c_i^{\dot{q}}\left(c_i^q\right) \cdot B_i^{\dot{q}}(\tau). \tag{12}$$

The coefficients $c_i^{\dot{q}}$ can be verified to depend linearly on c_i^q [17]. Therefore, after applying B-spline relaxations, the problem variables become c_i^q, together with the motion time T (for more details, see [14]).

After combining the velocity constraints with initial and final conditions on the vehicle position and velocity, the optimization problem becomes

$$\begin{aligned} \min_{c_i^q, T} \ & T \\ \text{s.t.} \ & c_1^q = q_0, \quad c_n^q = q_T \\ & c_1^{\dot{q}} = \dot{q}_0, \quad c_{n-1}^{\dot{q}} = \dot{q}_T \\ & V_{\min} \cdot T \leq c_i^{\dot{q}} \leq V_{\max} \cdot T \\ & i = 1 \ldots n-1. \end{aligned} \tag{13}$$

B. Spline Parameterization for Nonholonomic Vehicles

The aforementioned approach does not straightforwardly extend to the bicycle model due to the trigonometric functions appearing in the state equations (2). B-spline relaxations can only be applied on splines, meaning that all constraints must be written as derivatives, antiderivatives, or polynomials of splines. Therefore, a nonlinear change of variables is adopted to transform all constraints into bounds on spline functions of τ. This change of variables is based on a spline parameterization of $r(\tau)$ and $\tilde{v}(\tau)$, which are defined as follows:

$$\begin{aligned} r(\tau) &= \tan\frac{\theta(\tau)}{2} \\ \tilde{v}(\tau) &= \frac{V(\tau)}{1+r(\tau)^2}. \end{aligned} \tag{14}$$

With this definition, $r(\tau)$ directly determines the vehicle orientation $\theta(\tau)$, while the corresponding trajectories for $x(\tau)$ and $y(\tau)$ are given by

$$\begin{aligned} x(\tau) &= x_0 + T \cdot \int_0^1 \tilde{v}(\tau) \cdot (1 - r(\tau)^2) d\tau \\ y(\tau) &= y_0 + T \cdot \int_0^1 \tilde{v}(\tau) \cdot (2 \cdot r(\tau)) d\tau \end{aligned} \tag{15}$$

in which (x_0, y_0) is the initial vehicle position. This expression is obtained after applying the tangent half-angle substitution [20]

$$\sin\theta(\tau) = \frac{2 \cdot r(\tau)}{1+r(\tau)^2}, \quad \cos\theta(\tau) = \frac{1-r(\tau)^2}{1+r(\tau)^2} \tag{16}$$

to the state equations (2). The longitudinal velocity $V(\tau)$ is given by

$$V(\tau) = \tilde{v}(\tau) \cdot (1 + r(\tau)^2) \tag{17}$$

and is a spline function of τ as well. Hence, bounds on $V(\tau)$ and its derivatives can directly be addressed by the spline relaxations. To include constraints on the steering angle, the following relation:

$$\dot{\theta}(\tau) = \frac{2 \cdot \dot{r}(\tau)}{1+r(\tau)^2} \tag{18}$$

is used in the kinematic bicycle model, to obtain the following expression for the steering angle:

$$\tan(\delta) = \frac{2 \cdot \dot{r}(\tau) \cdot L}{\tilde{v}(\tau) \cdot (1+r(\tau)^2)^2}. \tag{19}$$

In order to impose an initial condition on the steering angle for a vehicle starting from standstill, it is required to use l'Hôpital's rule, since both the numerator and denominator of (19) are zero in this case.

Taking the time derivative of (19) leads to the following expression for the rate of change of steering angle:

$$\dot{\delta}(\tau) = \frac{p(\tau) - 2 \cdot L \cdot \dot{r}(\tau) \cdot (q(\tau) + o(\tau))}{w(\tau)} \tag{20}$$

with

$$\begin{aligned} p(\tau) &= 2 \cdot L \cdot \ddot{r}(\tau) \cdot \tilde{v}(\tau) \cdot (1+r^2(\tau))^2 \\ q(\tau) &= (4 \cdot r^3(\tau) \cdot \dot{r}(\tau) + 4 \cdot r(\tau) \cdot \dot{r}(\tau)) \cdot \tilde{v}(\tau) \\ o(\tau) &= \dot{\tilde{v}}(\tau) \cdot (1+r^2(\tau))^2 \\ w(\tau) &= \tilde{v}^2(\tau) \cdot (1+r^2(\tau))^4 \cdot T^2 + (2 \cdot L \cdot \dot{r}(\tau))^2. \end{aligned} \tag{21}$$

Therefore, using the proposed nonlinear change of variables (14) allows writing all constraints (4) as spline constraints, making it possible to apply the spline-based motion planning method to the bicycle model. The application to the differential drive or to an AGV with rear-wheel steering follows a completely similar approach, using the same substitution and the same state vector.

C. Collision Avoidance Constraints

To generate a safe motion trajectory, it is vital to carefully include collision avoidance constraints in the optimization problem. In general, these constraints express that the vehicle shape must not overlap with the shape of any obstacle k

$$\operatorname{dist}\big(\operatorname{veh}(q(\tau)),\quad \operatorname{obs}_k\ \big(p_k^{\text{pred}}(\tau)\big)\big) \geq \epsilon \qquad k = 1 \ldots N,\quad \forall \tau \in [0,\ 1]. \tag{22}$$

Here, ϵ is a safety factor to avoid trajectories passing very close to obstacles. These trajectories are highly undesired, since the slightest deviation from the predicted movement of the vehicle or the obstacle may cause a collision in this case.

To include constraints (22) in the optimization problem, the proposed approach uses the separating hyperplane theorem [21]. This theorem states that two nonintersecting convex sets can always be separated by a hyperplane.

For the sake of clarity, only one obstacle is considered, and the subscript k is omitted in the notation. The separating hyperplane for the kth obstacle is described by the direction $a(\tau)$ and offset $b(\tau)$, which are both parameterized as splines in order to allow the hyperplane to rotate and translate over time

$$a(\tau) = \sum_{i=1}^{n} c_i^a \cdot B_i^a(\tau), \quad b(\tau) = \sum_{i=1}^{n} c_i^b \cdot B_i^b(\tau) \tag{23}$$

in which c_i^a and c_i^b are additional optimization variables. The constraints required to guarantee that the AGV stays on one side of the plane and the obstacle on the other depends on their particular shapes. In the following, the case of a rectangular AGV and a circular obstacle is further elaborated, clarifying how to handle polyhedral as well as circular shapes. Fig. 3 illustrates this approach graphically and shows the evolution of the separating hyperplane along the motion trajectory.

Let z_j^{veh} be the vertices of the AGV's rectangular shape in the local AGV frame; then, the trajectories $\hat{z}_j^{\text{veh}}(\tau)$ of the vertices in the global frame are given by

$$\hat{z}_j^{\text{veh}}(\tau) = [x(\tau),\ y(\tau)]^T + R(\tau) \cdot z_j^{\text{veh}} \tag{24}$$

in which $[x(\tau),\ y(\tau)]^T$ is the position of the vehicle and $R(\tau)$ is the rotation matrix around the vehicle center. When using the tangent half-angle substitution, as introduced earlier, $R(\tau)$ is given by

$$R(\tau) = \begin{bmatrix} \dfrac{1 - r(\tau)^2}{1 + r(\tau)^2} & -\dfrac{2 \cdot r(\tau)}{1 + r(\tau)^2} \\ \dfrac{2 \cdot r(\tau)}{1 + r(\tau)^2} & \dfrac{1 - r(\tau)^2}{1 + r(\tau)^2} \end{bmatrix}. \tag{25}$$

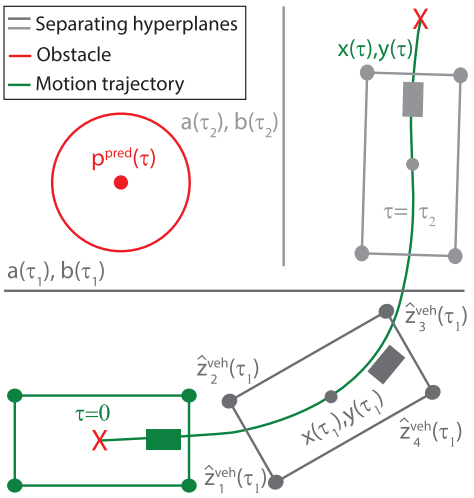


Fig. 3. Application of the separating hyperplane theorem.

Applying the separating hyperplane theorem to (22) gives [14]

$$\begin{aligned} a(\tau)^T \cdot p^{\text{pred}}(\tau) - b(\tau) &\geq d(\tau) - r^{\text{obs}}, \\ (1 + r^2(\tau)) \cdot \big(a(\tau)^T \cdot \hat{z}_j^{\text{veh}}(\tau) - b(\tau)\big) &\leq 0, \\ ||a(\tau)||_2 &\leq 1, \\ j = 1 \ldots 4, \quad \forall \tau \in [0, 1]. \end{aligned} \tag{26}$$

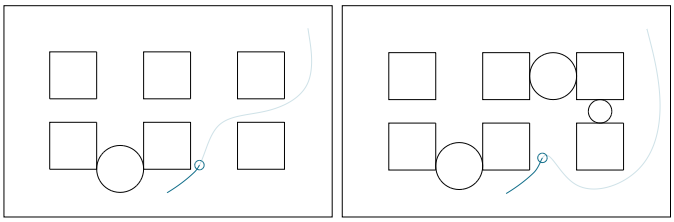
In these equations, $p^{\text{pred}}(\tau)$ gives the position of the circle center and r^{obs} represents its radius. Note that in the second line of (26), $(1+r^2(\tau))$ cancels out the denominators in $R(\tau)$. The safety factor ϵ leads to the addition of $d(\tau)$. This is also a spline, which expresses the distance between the vehicle and the corresponding obstacle as a function of time. Adding an extra term $||d(\tau) - \epsilon||_2$ to the objective penalizes $d(\tau)$ if it deviates from ϵ. Therefore, $d(\tau)$ will only become smaller than ϵ when there is no other option. Summarizing the procedure, the relaxed constraints (26) are added to the problem, introducing additional optimization variables c_i^a, c_i^b, and c_i^d.

IV. Numerical and Experimental Validation

This section validates the presented approach by means of numerical simulations (Section IV-A) and experimental validations (Section IV-B). In all validations, the spline-based motion planning problem is formulated using OMG tools: a user-friendly open-source toolbox [13]. This toolbox is written in Python and uses CasADi [22] as a symbolic framework to formulate the motion planning problem, transform it into a tractable optimization problem, and pass it to the solver. The default solver for all problems in this brief is Ipopt [23]. However, for some problems, the comparison is made with the active set solver blockSQP [24], which uses a feasible initial guess from Ipopt for its first iteration. Movies of all presented simulations and experiments can be found on the GitHub page of the toolbox [13], together with various additional examples and their corresponding code.

A. Numerical Simulations

To demonstrate the versatility of the presented approach, the current section shows four examples, using different



(a) (b)

Fig. 4. Holonomic vehicle moving through an obstructed warehouse. (a) $t = 3.3$ s. (b) $t = 3.5$ s.

(a) (b)

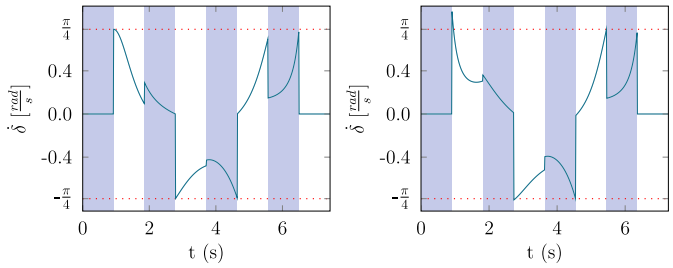
Fig. 5. Rear-wheel steered AGV parking between two obstacles. (a) $t = 0$ s. (b) $t = 5.9$ s.

vehicle models and considering various environments. All simulations were done on a notebook with Intel Core i5-4300M CPU at 2.60 GHz $\times$ 4 processor and 8 GB of memory.

The first example considers a holonomic vehicle moving through a warehouse, as shown in Fig. 4. To illustrate the merit of the receding horizon implementation of the motion planning approach, a scenario is considered in which after 3.5 s suddenly two new obstacles appear that block the initial trajectory. The receding horizon implementation always takes the latest information into account and, as shown in Fig. 4(b), it can deal with this drastic change in the environment: the method successfully computes a new trajectory to the goal position. The average solving time (t_{avg}) of this problem is 57 ms. The maximum solving time ($t_{\max}$) of 446 ms is reached in the iteration in which the new obstacles are detected.

In the second example, a rear-wheel steered AGV parks between two vehicles. A motion trajectory with eight internal spline knots is selected. In addition to the collision avoidance constraints, the problem formulation also contains the input and kinodynamic constraints (4). Fig. 5 shows two snapshots of the solution of the online motion planning approach for this problem. When using Ipopt, t_{avg} is 371 ms and $t_{\max}$ is 929 ms. When using blockSQP, t_{avg} is 107 ms and $t_{\max}$ is 286 ms. To demonstrate the merit of using B-spline relaxations to enforce constraints over the entire time horizon, the relaxation-based approach is compared with time gridding. In time gridding, which is currently a standard approach, constraints are only enforced on a finite set of time samples instead of over the entire time horizon. In order to obtain a fair comparison between the two approaches, the amount of gridpoints is chosen equal to the amount of spline coefficients (such that both approaches yield the same number of constraints). To get a good sample distribution, the so-called Greville abscissa [17] of the spline are used for the locations of the gridpoints. When using time gridding, this approach is applied to all constraints. In the comparison, both approaches consider the first OCP, at $t = 0$ s (see Fig. 5(a)).

Fig. 6 compares the trajectories of the rate of change of steering angle $\dot{\delta}$ for the two approaches and clearly shows their



(a) (b)

Fig. 6. Rate of change of steering angle for B-spline relaxations versus time gridding. (a) B-spline relaxations. (b) Time gridding.

main difference: time gridding does not guarantee constraints to hold in between the gridpoints, and may hence lead to constraint violations (0.949 rad/s versus $\dot{\delta}_{\max} \approx 0.785$ rad/s at 0.91 s). The B-spline relaxations, on the other hand, do provide this guarantee, but herein introduce some conservatism (0.7 rad/s versus $\dot{\delta}_{\max} \approx 0.785$ rad/s at 5.56 s). These effects are also represented in the optimal motion time obtained when using time gridding (7.27 s), compared with B-spline relaxations (7.43 s). While the obtained motion time for time gridding is lower, it is not physically achievable, due to the constraint violation. On the other hand, the optimal motion time obtained when using B-spline relaxations is conservative, yet physically achievable. In addition, the calculation time for the case with time gridding is about eight times higher than the one when using B-spline relaxations. As a consequence, this comparison clearly demonstrates the advantage of using B-spline relaxations.

To study the conservatism, which is introduced by using B-spline relaxations, constraint refinements are applied to the example of Fig. 5. Applying the presented motion planning approach results in a set of spline constraints. Afterward, before using B-spline relaxations, all spline constraints are refined by adding an additional knot in between every two distinct knots (a midpoint refinement). While this operation has no influence on the amount of variables, it does increase the number of constraints in order to reduce the gap between the control polygon and the spline, lowering the introduced conservatism. When using one and two midpoint refinements, the obtained motion time lowers from 7.43 to 7.39 and 7.38 s, respectively, proving that the conservatism is limited. Applying one and two midpoint refinements when using time gridding comes down to selecting two and four times as much gridpoints. In this case, the motion time rises from 7.27 to 7.36 and 7.37 s, respectively, since there are less constraint violations. However, using refinements increases the amount of constraints, complicating the resulting optimization problem, and leading to higher computation times.

In the third example, a differential drive vehicle moves past a central stationary obstacle with circular shape. The vehicle has a velocity limit of 0.7 m/s, and no acceleration limits. Table I gives an overview of the obtained motion time (T) as a function of the selected amount of internal spline knots (n_{knots}), together with the required average and maximum computation times (t_{avg} and $t_{\max}$). It becomes clear that for a reasonable amount of ten internal spline knots,

TABLE I
MOTION AND CALCULATION TIMES FOR A DIFFERENTIAL DRIVE VEHICLE MOVING PAST A CENTRAL OBSTACLE FOR SELECTED AMOUNTS OF INTERNAL KNOTS

n_{knots}	5	8	10	15	20	40
T[s]	6.81	6.52	6.42	6.30	6.24	6.16
t_{avg}[ms]	44	92	116	262	318	1202
t_{max}[ms]	73	504	229	1097	753	4771

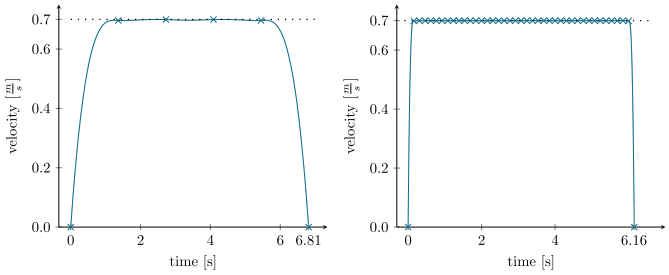


Fig. 7. Velocity profile of the differential drive vehicle. (a) 5 internal knots. (b) 40 internal knots.

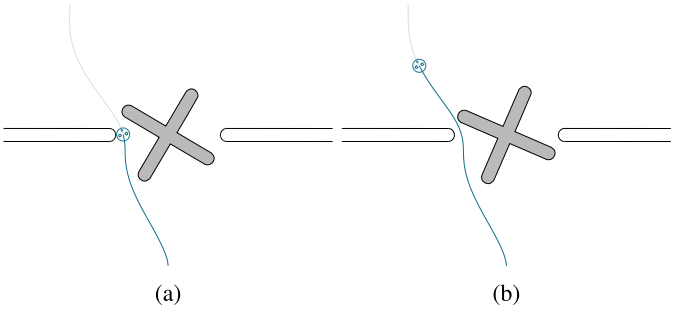


Fig. 8. Differential drive vehicle moving past a revolving door. (a) $t = 5.9$ s. (b) $t = 8.9$ s.

the computation time stays limited, while the obtained motion time is only 5.8% higher than the motion time for the shortest collision-free path for which the maximum velocity is always applicable (6.066 s). Since a time-optimal trajectory is computed, the vehicle wants to move as long as possible at its maximum velocity. Fig. 7 shows how the velocity trajectory converges to the expected bang-bang trajectory, when increasing the amount of internal spline knots from 5 to 40.

To illustrate the versatility of the presented motion planning approach, the fourth example considers a differential drive vehicle, which has to move through a revolving door. The rotation of the door is accounted for by including a prediction of its rotational velocity. Fig. 8 shows two snapshots of the online motion planning solution. When the safety factor is omitted, the time-optimal trajectory passes very closely to the left obstacle (see Fig. 8(a)). This example demonstrates the capability of the approach in handling complicated and nonconvex motion planning problems. When using Ipopt, t_{avg} is 134 ms, and apart from one very slow iteration of 3176 ms, t_{max} is 244 ms. When using blockSQP, t_{avg} is 22 ms and t_{max} is 222 ms.

The computation times for the examples with nonholonomic vehicles are substantially higher than the one considering a holonomic vehicle, due to the complexity introduced by the vehicle model. However, the currently investigated active set solver blockSQP shows promising computation times toward experiments with nonholonomic vehicles in rapidly changing environments.

B. Experimental Validations

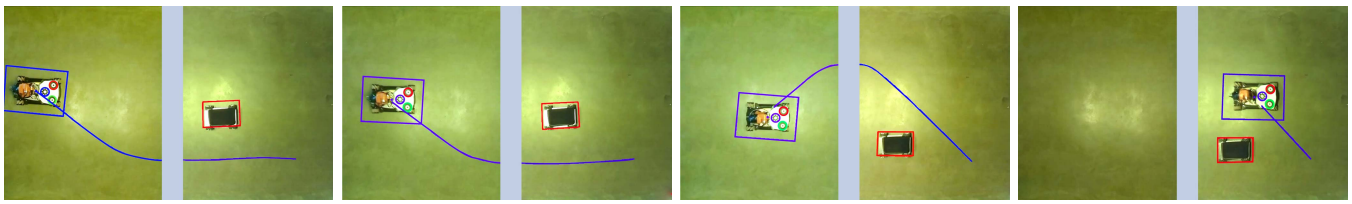
To experimentally validate the presented motion planning method, a *KUKA youBot* is used [12]. The *youBot* is a small holonomic platform, which can be programmed using *ROS* [25]. In addition to the *youBot*, the experimental setup contains two ceiling cameras to detect the *youBot*, as well as obstacles, in the test field. There is a small horizontal gap between both cameras, creating an uncovered zone in the middle of the field. The third component of the setup is an external desktop computer (Intel Core Xeon E5-1603 v3 CPU at 2.8 GHz $\times$ 4 processor and 16 GB of RAM) on which all computations are performed. The *youBot* is steered in velocity mode, meaning that it receives longitudinal, lateral, and rotational velocity set points from the external PC, which are translated on board to set points for the wheel velocity controllers. These set points are determined by the motion planning approach, which computes the fastest trajectory to move the *youBot* to its goal position, while satisfying its velocity limits (0.4 m/s) and acceleration limits (1 m/s^2). Note that, for a safe operation, it is essential to include a safety factor (see (22)) when calculating trajectories. For the presented experiments, a safety factor of 5 cm was adopted.

The motion planning problem is solved online, using a receding horizon implementation. For every update, the actual state of both the *youBot* and the obstacles is estimated first. To obtain an accurate estimation of the position and velocity of the *youBot*, a Kalman filter combines position and orientation estimates, extracted from the camera images, with odometry information of the *youBot* wheels. The obstacle positions are estimated by the second Kalman filter, using only the camera images. Based on the latest information, the online motion planning updates the *youBot* velocity trajectories in an asynchronous way: when a new trajectory is computed, it is immediately applied to the vehicle, replacing the trajectory from the previous iteration.

Fig. 9 shows the test results for an experiment in which the rectangular-shaped *youBot* crosses a room diagonally. After 1.5 s, the second vehicle (acting as an obstacle) starts to traverse the room in the vertical direction with a velocity of 0.3 m/s. Fig. 9 shows that the trajectory is adapted to avoid this moving obstacle, and a new trajectory to the endpoint is found. The average calculation time per iteration is 0.11 s, and the maximum time is 0.4 s.

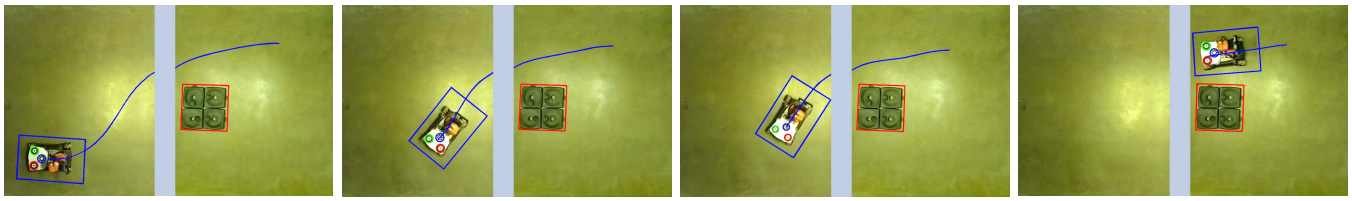
In the second experiment, the *youBot* is steered as a differential drive, in order to validate the presented method for nonholonomic vehicles. Fig. 10 shows how the orientation of the vehicle stays parallel to the trajectory, while it avoids a static obstacle in the center of the environment. The average calculation time per iteration is 0.21 s, and the maximum time is 0.98 s.

Both experiments show that the update rate of the motion planning is sufficiently fast to provide adequate position feedback for the *youBot*, and no additional position feedback controllers are needed.



(a) (b) (c) (d)

Fig. 9. Holonomic vehicle crossing a room with a moving obstacle. (a) $t = 0.5$ s. (b) $t = 1$ s. (c) $t = 1.7$ s. (d) $t = 4.5$ s.



(a) (b) (c) (d)

Fig. 10. Holonomic vehicle, steered as a differential drive, avoiding a static obstacle. (a) $t = 0$ s. (b) $t = 2.5$ s. (c) $t = 3$ s. (d) $t = 6.2$ s.

V. Conclusion

AGVs are becoming increasingly important in industry, leading to a growing interest in flexible motion planning methods. To tackle the corresponding challenges in trajectory generation, this brief presents a method to calculate motion trajectories for holonomic and nonholonomic vehicles in dynamic and uncertain environments. The method obtains an efficient problem formulation by using a spline parameterization of the motion trajectories, in combination with B-spline relaxations. To deal with uncertainties, the approach solves the problem with a receding horizon, to always account for the latest information. The output of the method consists of feasible motion trajectories, immediately taking into account the kinematics of the vehicle, in combination with the corresponding input trajectories. Experimental validation on a *KUKA youBot* shows that, due to the fast update rate, it is possible to generate collision-free motion trajectories for vehicles driving around in a dynamic environment. The whole process of modeling, simulating, and embedding spline-based motion planning problems is made easy by OMG-tools, a user-friendly open-source software toolbox.

References

[1] F. Debrouwere, "Optimal robot path following fast solution methods for practical non-convex applications," Ph.D. dissertation, Dept. Mech. Eng., KU Leuven, Leuven, Belgium, 2015.

[2] A. J. Häusler, A. Saccon, A. M. Pascoal, J. Hauser, and A. P. Aguiar, "Cooperative AUV motion planning using terrain information," in *Proc. MTS/IEEE OCEANS*, Bergen, Bergen, vol. 7. Jun. 2013, pp. 1–10.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[4] S. Koenig and M. Likhachev, "D* lite," in *Proc. 18th Nat. Conf. Artif. Intell.*, 2002, pp. 476–483.

[5] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[7] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," *Robot., Sci. Syst.*, vol. 9, no. 1, pp. 1–10, 2013.

[8] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proc. Int. Conf. Autom. Planning Scheduling*, May 2016, pp. 207–215.

[9] E. Camacho and C. B. Alba, *Model Predictive Control*. London, U.K.: Springer-Verlag, 2007.

[10] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, "A voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Eng. Pract.*, vol. 61, pp. 41–54, Apr. 2016.

[11] K. Yang, S. K. Gan, and S. Sukkarieh, "An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments," *J. Intell. Robot. Syst.*, vol. 57, nos. 1–4, pp. 101–122, 2010.

[12] R. Bischoff, U. Huggenberger, and E. Prassler, "KUKA youBot—A mobile manipulator for research and education," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 1–4.

[13] R. Van Parys and T. Mercy. (2016). *OMG-Tools*. [Online]. Available: https://github.com/meco-group/omg-tools

[14] T. Mercy, W. Van Loock, and G. Pipeleers, "Real-time motion planning in the presence of moving obstacles," in *Proc. Eur.Control Conf. (ECC)*, 2016, pp. 1586–1591.

[15] S. L. Valle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[16] M. J. Matarić, *The Robotics Primer*. Cambridge, MA, USA: MIT Press, 2007.

[17] C. de Boor, *A Practical Guide to Splines*. New York, NY, USA: Springer-Verlag, 1978.

[18] W. Van Loock, G. Pipeleers, and J. Swevers, "B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction," *Mech. Sci.*, vol. 6, no. 2, pp. 163–171, 2015.

[19] H. Prautzsch, B. Wolfgang, and M. Paluszny, *Bézier and B-Spline Techniques*. Berlin, Germany: Springer-Verlag, 2002.

[20] J. Stewart, *Calculus: Early Transcendentals*. Belmont, CA, USA: Brooks/Cole, 1991.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[22] J. Andersson, "A general-purpose software framework for dynamic optimization," Ph.D. dissertation, Dept. Elect. Eng., KU Leuven, Leuven, Belgium, 2013.

[23] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[24] D. Janka, C. Kirches, S. Sager, and A. Wächter, "An SR1/BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal Hessian matrix," *Math. Program. Comput.*, vol. 8, no. 4, pp. 435–459, 2016.

[25] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009, p. 5.