An Optimization-Based Receding Horizon Trajectory Planning Algorithm

Kristoffer Bergman * Oskar Ljungqvist * Torkel Glad *
Daniel Axehill *

* Division of Automatic Control, Linköping University, Sweden
(e-mail: {kristoffer.bergman, oskar.ljungqvist, torkel.glad,
daniel.axehill}@liu.se).

Abstract: This paper presents an optimization-based receding horizon trajectory planning algorithm for dynamical systems operating in unstructured and cluttered environments. The proposed approach is a two-step procedure that uses a motion planning algorithm in a first step to efficiently find a feasible, but possibly suboptimal, nominal solution to the trajectory planning problem where in particular the combinatorial aspects of the problem are solved. The resulting nominal trajectory is then improved in a second optimization-based receding horizon planning step which performs local trajectory refinement over a sliding time window. In the second step, the nominal trajectory is used in a novel way to both represent a terminal manifold and obtain an upper bound on the cost-to-go online. This enables the possibility to provide theoretical guarantees in terms of recursive feasibility, objective function value, and convergence to the desired terminal state. The established theoretical guarantees and the performance of the proposed algorithm are verified in a set of challenging trajectory planning scenarios for a truck and trailer system.

Keywords: Trajectory & Path Planning, Optimal Control, Autonomous Vehicles

1. INTRODUCTION

In recent decades, an extensive amount of research has been conducted in the area of motion planning for autonomous vehicles (LaValle, 2006; Paden et al., 2016). However, the problem of computing locally optimal trajectories for dynamical systems in confined and unstructured environments is still considered as a difficult task. In this paper, the optimal motion planning problem is defined as the problem of finding a feasible and collision-free trajectory that brings the system from its initial state to a desired terminal state while a performance measure is minimized. The computed trajectory is then intended to be used as reference to a trajectory tracking or path following controller (Andersson et al., 2018b; Paden et al., 2016; Ljungqvist et al., 2019).

The optimal motion planning problem is in general hard to solve by directly applying optimal control techniques, since the problem in general is nonconvex due to obstacle-imposed constraints and nonlinear system dynamics. Therefore, approximate methods in terms of motion planning algorithms are commonly used (LaValle, 2006). One commonly used approach for dynamical systems is to apply sampling-based planners, which are either based on random or deterministic exploration of the vehicle's state space (LaValle, 2006). One approach based on random sampling is RRT* which is a popular motion planning algorithm for dynamical systems where an efficient steering function is available (Karaman and Frazzoli, 2013; Banzhaf et al., 2018). Unless an efficient steering function is available, the RRT* algorithm becomes computationally

inefficient as multiple optimal control problems (OCPs) have to be solved online at each tree expansion (Stoneman and Lampariello, 2014).

A popular deterministic sampling-based motion planner is the lattice-based motion planner, which uses a finite set of precomputed motion segments, or motion primitives, online to find an optimal solution to a discretized version of the motion planning problem (Pivtoraiko et al., 2009). A benefit with this method is that efficient graph-search algorithms can be used online such as A* (Hart et al., 1968), making it real-time applicable (Pivtoraiko et al., 2009; Ljungqvist et al., 2019). However, since the lattice-based planner uses a discretized search space, the computed solution can be noticeably suboptimal and a latter post-optimization step is often desirable to use (Dolgov et al., 2010; Andreasson et al., 2015). A related technique is proposed in our previous work in (Bergman et al., 2019b), where an optimization-based improvement step is added, aiming at locally improving the solution from a lattice-based planner without being limited to a discrete search space. Compared to previous work, a tight integration between the motion planner and the optimization step was introduced. This new approach was shown to have significant benefits over existing related methods in terms of solution quality and reliability. However, the introduced improvement step increases the motion planner's latency time and hence, the time before the trajectory can start being executed. To reduce the computation time of the improvement step, and thus enable a faster start of the execution phase, a receding horizon trajectory planning approach is proposed in this paper where the nominal trajectory from the motion planning algorithm is improved iteratively during the execution phase.

Optimization-based receding horizon planning (RHP) is commonly used in on-road applications, where the structure of the road environment is utilized to evaluate several candidates with different terminal states centered around the vehicle's lane. In (Werling et al., 2012), these candidates are efficiently computed using quintic polynomials. In unstructured environments, optimization-based RHP has mainly been applied on unmanned areal vehicles (UAVs) (Schouwenaars et al., 2004; Kuwata et al., 2005; Liu et al., 2017). The RHP approach is motivated in many applications due to limited sensing range, which makes it unnecessary to optimize the full horizon trajectory to the terminal state (Liu et al., 2017). Common for these methods are that outside the vehicle's planning range, a geometric planning algorithm is used to compute a simplified trajectory to the goal, e.g., a shortest distance trajectory that avoids known obstacles but disregards the system dynamics. The simplified trajectory is then used to estimate the cost-to-go, which enables a trade-off between short term and long term trajectory selection. This technique has been shown to work well for agile systems such as quadcopters. However, for systems that are less agile (such as truck and trailer systems), using, e.g., a geometric algorithm to estimate the cost-to-go can in worst case lead to infeasibility (Pivtoraiko et al., 2009; Bergman et al., 2019b).

The main contribution in this work is to use a nominal trajectory computed by a motion planning algorithm in a novel way to define a terminal manifold and an upper bound on the optimal cost-to-go. This use of a nominal trajectory makes it possible to avoid potential infeasibility caused by using a simplified cost-to-go estimate when solving the RHP problem. The result is utilized to provide theoretical guarantees on feasibility during the entire planning horizon, objective function value improvement and convergence to the terminal state. These theoretical results are used to define a practical RHP algorithm, whose performance is verified in a number of challenging motion planning problems for a truck and trailer system.

The remainder of the paper is organized as follows. The optimal motion planning problem is posed in Section 2. In Section 3, the RHP problem is defined and theoretical guarantees presented. These results are used in Section 4 to present an algorithm to iteratively improve the nominal trajectory using RHP. A simulation study for a truck and trailer system is presented in Section 5, followed by conclusions and future work in Section 6.

2. PROBLEM FORMULATION

In this paper, continuous-time nonlinear systems in the form
$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \tag{1}$$
are considered, where $\boldsymbol{x} \in \mathbf{R}^n$ and $\boldsymbol{u} \in \mathbf{R}^m$ denote the state and control signal of the system, respectively. These are subject to the following constraints:
$$\boldsymbol{x} \in \mathcal{X} \subseteq \mathbf{R}^n, \quad \boldsymbol{u} \in \mathcal{U} \subseteq \mathbf{R}^m. \tag{2}$$
Furthermore, the system should not collide with obstacles, where the obstacle region is defined as $\mathcal{X}_{\text{obst}} \subset \mathbf{R}^n$. Thus, in motion planning problems, the state space is constrained as:
$$\boldsymbol{x} \in \mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obst}}. \tag{3}$$
This constraint is in general non-convex since $\mathcal{X}_{\text{free}}$ is defined as the complement set of $\mathcal{X}_{\text{obst}}$.

The motion planning problem can now be defined as the problem of computing a feasible (i.e. satisfying (1)-(3)) state and control signal trajectory $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot))$ that moves the system from $\boldsymbol{x}_0 \in \mathcal{X}_{\text{free}}$ to a desired terminal state, $\boldsymbol{x}_f \in \mathcal{X}_{\text{free}}$, while a performance measure J_{tot} is minimized. This problem can be posed as a continuous-time OCP:

$$\begin{aligned} \underset{\boldsymbol{u}(\cdot),\, t_f}{\text{minimize}} \quad & J_{\text{tot}}(\boldsymbol{x}_0, \boldsymbol{u}(\cdot)) = \int_{t_0}^{t_f} \ell(\boldsymbol{x}(t), \boldsymbol{u}(t)) \mathrm{d}t \\ \text{subject to} \quad & \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{x}(t_f) = \boldsymbol{x}_f, \\ & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \\ & \boldsymbol{x}(t) \in \mathcal{X}_{\text{free}}, \ \boldsymbol{u}(t) \in \mathcal{U} \quad t \in [t_0, t_f]. \end{aligned} \tag{4}$$

Here, the decision variable t_f represents the time when the terminal state is reached. Furthermore, $\ell(\boldsymbol{x}, \boldsymbol{u})$ forms the cost function that is used to define the objective functional J_{tot}.

Assumption 1. $\ell : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^1$ is continuous, and $\ell(\boldsymbol{x}, \boldsymbol{u}) \geq \varepsilon > 0$ for all $(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X} \times \mathcal{U}$.

Remark 1. Assumption 1 provides an explicit penalty on the terminal time. Hence, $J_{\text{tot}} \to \infty$ as $t_f \to \infty$.

One commonly used cost function for motion planning and optimal control problems can be written in the form:
$$\ell(\boldsymbol{x}, \boldsymbol{u}) = 1 + ||\boldsymbol{x}||_Q^2 + ||\boldsymbol{u}||_R^2, \tag{5}$$
in which the weight matrices $Q \succeq 0$ and $R \succeq 0$ are used to determine the trade-off between time duration (captured by the first term in (5)) and other measures such as smoothness of a motion (Ljungqvist et al., 2019).

As discussed in Section 1, the problem in (4) is hard to solve by applying direct optimal control techniques due to the non-convex obstacle avoidance constraints and the nonlinear dynamics. Hence, a good initialization strategy is required to enable the possibility of computing efficient and reliable solutions (Bergman et al., 2019b). In this work, it is assumed that a motion planning algorithm (such as the ones described in Section 1) has provided a nominal trajectory that moves the system from $\boldsymbol{x}_0$ to $\boldsymbol{x}_f$ and is at least a feasible solution to (4). This trajectory is represented by $(\bar{\boldsymbol{x}}(\tau), \bar{\boldsymbol{u}}(\tau))$, $\tau \in [t_0, \ \bar{t}_f]$, where $\bar{\boldsymbol{x}}(\tau)$ satisfies:
$$\bar{\boldsymbol{x}}(\tau) = \boldsymbol{x}_0 + \int_{t_0}^{\tau} f(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{u}}(t)) \mathrm{d}t \tag{6}$$
This nominal trajectory $(\bar{\boldsymbol{x}}(\cdot), \bar{\boldsymbol{u}}(\cdot), \bar{t}_f)$ is used computationally to warm-start the second RHP step, but also theoretically to guarantee convergence to the terminal state. A detailed description of this procedure is given in the next section.

3. RECEDING HORIZON PLANNING

In this section, it will be shown how to use an optimization-based receding horizon planner to optimize a nominal trajectory already computed by a motion planning algorithm. The nominal trajectory is used in the RHP approach to represent a terminal manifold, which ensures the existence of a feasible trajectory to the terminal state beyond the current receding planning horizon.

3.1 Receding horizon planning formulation

The problem of optimizing the nominal trajectory is solved using an iterative receding horizon approach. At each RHP iteration k at time $t_k = t_0 + k\delta$, $\delta > 0$, $k \in \mathbf{Z}_0$, an OCP

is solved over a sliding time window $[t_k, t_k + T]$, where $T \in (\delta, T_{\max}]$ denotes its length in time. This optimization-based RHP problem is defined as:

$$
\begin{aligned}
\underset{\boldsymbol{u}_k(\cdot),\,\tau_k}{\text{minimize}} \quad & J(\boldsymbol{x}_{\text{cur}}, \boldsymbol{u}_k(\cdot), \tau_k) = \\
& \Psi_k(\tau_k) + \int_{t_k}^{t_k+T} \ell(\boldsymbol{x}_k(t), \boldsymbol{u}_k(t)) \mathrm{d}t
\end{aligned}
$$
$$
\begin{aligned}
\text{subject to} \quad & \boldsymbol{x}_k(t_k) = \boldsymbol{x}_{\text{cur}}, \quad \boldsymbol{x}_k(t_k + T) = \bar{\boldsymbol{x}}_{k-1}(\tau_k) \quad (7)\\
& \dot{\boldsymbol{x}}_k(t) = f(\boldsymbol{x}_k(t), \boldsymbol{u}_k(t)), \\
& \boldsymbol{x}_k(t) \in \mathcal{X}_{\text{free}}, \qquad t \in [t_k, t_k + T] \\
& \boldsymbol{u}_k(t) \in \mathcal{U}.
\end{aligned}
$$

Here, $\boldsymbol{x}_{\text{cur}} = \bar{\boldsymbol{x}}_{k-1}(t_k)$ is the predicted state of the system at time t_k, $\bar{\boldsymbol{x}}_{k-1}(\cdot)$ the previously optimized state trajectory at time t_k (with $\bar{\boldsymbol{x}}_{-1}(\cdot) = \bar{\boldsymbol{x}}(\cdot)$) and $\Psi_k(\tau_k)$ the cost-to-go function. Compared to (4), a subindex k has been added to the state and control signal to clarify that it is related to the k:th RHP iteration. Furthermore, an additional decision variable τ_k has been added. This variable can be seen as a timing parameter and is used in the terminal constraint to select at what time instance the state at the end of the horizon $\boldsymbol{x}_k(t_k + T)$ is connected to the previously optimized state trajectory $\bar{\boldsymbol{x}}_{k-1}(\cdot)$, which defines the terminal manifold. From this state on the terminal manifold, an open-loop control law is known that moves the system from $\bar{\boldsymbol{x}}_{k-1}(\tau_k), \tau_k \in [t_0, \bar{t}_f^{k-1}]$ to $\boldsymbol{x}_f$. Note that if the previous solution is already locally optimal, the optimal solution to (7) is given by $(\boldsymbol{u}_k^\star(\cdot), \tau_k^\star)$, where $\boldsymbol{u}_k^\star(t) = \bar{\boldsymbol{u}}_{k-1}(t), \; t \in [t_k, t_k + T]$ and $\tau_k^\star = t_k + T$. Otherwise, a time shift to connect to the previous solution might occur, which is defined as

$$
\Delta t_k = \tau_k^\star - (t_k + T). \tag{8}
$$

Hence, a new optimized solution $\bar{\boldsymbol{u}}_k(\cdot)$ is available in the end of each RHP iteration and is given by

$$
\bar{\boldsymbol{u}}_k(t) = \begin{cases} \bar{\boldsymbol{u}}_{k-1}(t), & t \in [t_0, t_k) \\ \boldsymbol{u}_k^\star(t) \in \mathcal{U}, & t \in [t_k, t_k + T) \\ \bar{\boldsymbol{u}}_{k-1}(t + \Delta t_k), & t \in [t_k + T, \bar{t}_f^{k-1} - \Delta t_k], \end{cases} \tag{9}
$$

where $\bar{\boldsymbol{u}}_{-1}(\cdot) = \bar{\boldsymbol{u}}(\cdot)$ which is the nominal control trajectory. Furthermore, the new terminal time is updated according to $\bar{t}_f^k = \bar{t}_f^{k-1} - \Delta t_k$ and the new optimized state trajectory $\bar{\boldsymbol{x}}_k(\cdot)$ is defined analogously as in (9).

In order to be able to select the optimal choice of τ_k, i.e., where to connect onto the terminal manifold given by $\bar{\boldsymbol{x}}_{k-1}(\cdot)$, a terminal cost $\Psi_k(\tau_k)$ is added that represents the cost to transfer the system from $\bar{\boldsymbol{x}}_{k-1}(\tau_k)$ to $\boldsymbol{x}_f$ using the previously optimized solution. This cost-to-go function is given by

$$
\Psi_k(\tau_k) = \int_{\tau_k}^{\bar{t}_f^{k-1}} \ell(\bar{\boldsymbol{x}}_{k-1}(t), \bar{\boldsymbol{u}}_{k-1}(t)) \mathrm{d}t, \tau_k \in [t_0, \bar{t}_f^{k-1}], \tag{10}
$$

which represents an admissible overestimate of the optimal cost-to-go, obtained from the previous solution.

3.2 Feasibility, optimality and convergence

It will now be shown that the RHP problem in (7) possesses the following properties: i) recursive feasibility, ii) the total objective function value will be non-increasing at every RHP iteration, and iii) convergence to the terminal state. The reasoning behind most of the results are inspired by stability analysis for nonlinear model predictive control (MPC) (Mayne et al., 2000).
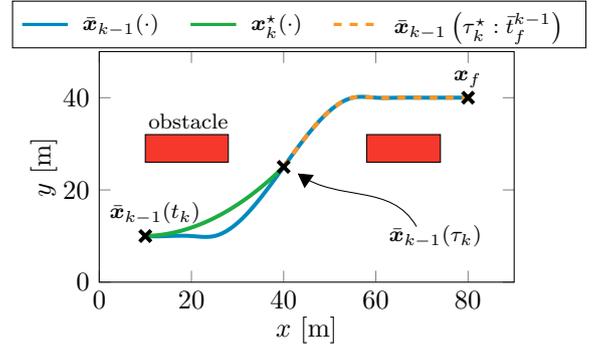

Fig. 1. An illustrative example of one RHP iteration. The problem in (7) is solved from $\bar{\boldsymbol{x}}_{k-1}(t_k)$, which results in an optimal state trajectory (green). The previous solution $\bar{\boldsymbol{x}}_{k-1}(\cdot)$ (blue) is used to provide guarantees that a feasible trajectory to the terminal state exist beyond the receding planning horizon (dashed).

Lemma 1. (**Recursive feasibility**).
Assume that the nominal trajectory $(\bar{\boldsymbol{x}}_{-1}(\cdot), \bar{\boldsymbol{u}}_{-1}(\cdot))$ is feasible in (4). Then, at all RHP iterations k satisfying $t_k + T \leq \bar{t}_f^{k-1}$, there exists a feasible solution to (4).

Proof. Assume that $\bar{\boldsymbol{u}}_{k-1}(\cdot)$ is feasible in (4) at RHP iteration $k-1$. Then, at any RHP iteration k, $\forall k : t_k + T \leq \bar{t}_f^{k-1}$, one choice of feasible decision variables in (7) is:

$$
\begin{aligned}
\tau_k^i &= t_k + T, \\
\boldsymbol{u}_k^i(t) &= \bar{\boldsymbol{u}}_{k-1}(t), \; t \in [t_k, t_k + T].
\end{aligned} \tag{11}
$$

After solving (7), an updated full horizon open-loop control law feasible in (4) at RHP iteration k is obtained from (9) as $\bar{\boldsymbol{u}}_k(\cdot)$. The desired result follows from induction by noting that at RHP iteration 0, $\bar{\boldsymbol{u}}_{-1}(\cdot)$ is feasible. $\square$

Theorem 1. (**Full horizon objective function value**).
Assume that the nominal trajectory $(\bar{\boldsymbol{x}}_{-1}(\cdot), \bar{\boldsymbol{u}}_{-1}(\cdot))$ is feasible in (4). Then, the result in the end of each RHP iteration k satisfying $t_k + T \leq \bar{t}_f^{k-1}$ is a full horizon open-loop control law $\bar{\boldsymbol{u}}_k(\cdot)$ that is feasible in (4) and satisfies $J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_k(\cdot)) \leq J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{k-1}(\cdot)) \leq \ldots \leq J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{-1}(\cdot))$.

Proof. From Lemma 1, it is known that $\bar{\boldsymbol{u}}_{k-1}(\cdot)$ is feasible in (4). Furthermore, the objective function value is $J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{k-1}(\cdot))$, which can be equivalently expanded as

$$
\begin{aligned}
J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{k-1}(\cdot)) = \; & \Psi_{ctc}(t_{k-1}) \\
& + J(\bar{\boldsymbol{x}}_{k-1}(t_{k-1}), \boldsymbol{u}_{k-1}^\star(\cdot), \tau_{k-1}^\star),
\end{aligned} \tag{12}
$$

where $\Psi_{ctc}(t)$ is the cost-to-come function, i.e., the accumulated cost up until t, with $\Psi_{ctc}(t_0) = 0$, while J and $(\boldsymbol{u}_{k-1}^\star(\cdot), \tau_{k-1}^\star)$ are the objective function and the solution to (7) at RHP iteration $k-1$, respectively. By using (7), (9), (10), (11) in (12), it follows that

$$
J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{k-1}(\cdot)) =
$$
$$
\underbrace{\Psi_{ctc}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \ell(\bar{\boldsymbol{x}}_{k-1}(t), \bar{\boldsymbol{u}}_{k-1}(t)) \mathrm{d}t}_{\Psi_{ctc}(t_k)}
$$
$$
+ \underbrace{\int_{t_k}^{t_k+T} \ell(\bar{\boldsymbol{x}}_{k-1}(t), \bar{\boldsymbol{u}}_{k-1}(t)) \mathrm{d}t + \Psi_k(t_k + T)}_{\text{Using (11) in (7)} : J(\bar{\boldsymbol{x}}_k(t_k), \boldsymbol{u}_k^i(\cdot), \tau_k^i)} = \tag{13}
$$
$$
\Psi_{ctc}(t_k) + J(\bar{\boldsymbol{x}}_k(t_k), \boldsymbol{u}_k^i(\cdot), \tau_k^i) \geq
$$
$$
\Psi_{ctc}(t_k) + J(\bar{\boldsymbol{x}}_k(t_k), \boldsymbol{u}_k^\star(\cdot), \tau_k^\star) = J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_k(\cdot)).
$$

Thus, using induction, it is possible to conclude that:

$$J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_k(\cdot)) \leq J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{k-1}(\cdot)) \leq \ldots \leq J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{-1}(\cdot)).$$

which holds $\forall k : t_k + T \leq \bar{t}_f^{k-1}$. When $t_k + T > \bar{t}_f^{k-1}$, an optimal solution within the current planning horizon already exists and no re-planning is required. $\square$

Remark 2. Note that Assumption 1 on the cost function $\ell(\boldsymbol{x}, \boldsymbol{u})$ is not required in Lemma 1 nor in Theorem 1.

Remark 3. When $t_k + T > \bar{t}_f^{k-1}$, one possibility is to perform re-planning by iteratively decreasing the planning horizon T. However, the optimal solution will stay the same during these last T/δ RHP iterations using arguments from principle of optimality.

Theorem 2. (**Finite number of RHP iterations**). Under Assumption 1, the maximum number of RHP iterations $k_{\max}$ is upper bounded by

$$k_{\max} \leq \frac{J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{-1}(\cdot))}{\varepsilon \delta}, \tag{14}$$

where δ is the time between two consecutive RHP iterations.

Proof. At RHP iteration k, Assumption 1 and (12) give

$$J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_k(\cdot)) \geq \Psi_{ctc}(t_k) =$$
$$\int_{t_0}^{t_0+\delta k} \underbrace{\ell(\bar{\boldsymbol{x}}_k(t), \bar{\boldsymbol{u}}_k(t))}_{\geq \varepsilon} \, \mathrm{d}t \geq \varepsilon \delta k. \tag{15}$$

From Theorem 1, it holds that

$$J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_k(\cdot)) \leq J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{-1}(\cdot)), \forall k : t_k + T \leq \bar{t}_f^{k-1}$$

which combined with (15) gives

$$\varepsilon \delta k \leq J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{-1}(\cdot)) \iff k \leq \frac{J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{-1}(\cdot))}{\varepsilon \delta}, \tag{16}$$

which completes the proof. $\square$

Corollary 1. (**Convergence to terminal state**). Under Assumption 1, the terminal state $\boldsymbol{x}_f$ will be reached in finite time.

Proof. Using Theorem 2, the terminal time t_f when the terminal state $\boldsymbol{x}_f$ is reached is upper bounded by

$$t_f \leq t_0 + \delta k_{\max} + T, \tag{17}$$

where $k_{\max}$ is upper bounded in (14) and T is the user-defined RHP horizon length in (7). $\square$

4. A PRACTICAL ALGORITHM

In this section, a reformulation of the RHP problem in the previous section is introduced to handle a piecewise continuous nominal control trajectory. The new formulation is connected to the theory in Section 3 to show that recursive feasibility, non-increasing objective function value and convergence to the terminal state still can be guaranteed. Finally, an algorithm is outlined which summarizes all steps in the proposed RHP approach.

4.1 Solving the receding horizon planning problem

A common approach to solve OCPs such as the RHP problem in (7) is to use direct methods for optimal control. In these methods, the continuous problem is discretized and cast as a standard NLP. This is typically achieved by using a piecewise continuous control signal (Diehl et al., 2006). The discretized problem can then be solved using

standard methods for nonlinear optimization such as SQP or nonlinear interior point methods (Nocedal and Wright, 2006). These solvers can be interfaced through a standard solver interface such as CasADi (Andersson et al., 2018a), which can be used when all involved functions in (7) are (at least) continuously differentiable everywhere.

In practice, it is desirable to use nominal trajectories in (7) where the control signal is piecewise continuous. As an example, this is the case when a lattice-based motion planner is used to compute a nominal trajectory using motion primitives computed by applying direct optimal control techniques (Bergman et al., 2019a). The problem of using a piecewise continuous nominal control signal trajectory is that the terminal manifold, defined by $\bar{\boldsymbol{x}}_{k-1}(\tau)$, and the cost-to-go function $\Psi_k(\tau)$ in (7) are piecewise continuously differentiable with respect to the timing variable τ. This follows from that

$$\begin{aligned} \frac{\mathrm{d}\bar{\boldsymbol{x}}_{k-1}}{\mathrm{d}\tau} &= \dot{\bar{\boldsymbol{x}}}_{k-1}(\tau) = f(\bar{\boldsymbol{x}}_{k-1}(\tau), \bar{\boldsymbol{u}}_{k-1}(\tau)), \\ \frac{\mathrm{d}\Psi_k}{\mathrm{d}\tau} &= -\ell(\bar{\boldsymbol{x}}_{k-1}(\tau), \bar{\boldsymbol{u}}_{k-1}(\tau)), \end{aligned} \tag{18}$$

explicitly depend on the piecewise continuous control signal trajectory $\bar{\boldsymbol{u}}_{k-1}(\tau)$. Hence, in this case it is not possible to directly use standard solver interfaces. One possibility is to modify the solver and/or solver interface, which is out of scope in this work. Another possibility, which is used in this paper and will further be described in the next sections, is to adjust the problem formulation while aiming at preserving the theoretical guarantees proved in Section 3.2.

4.2 Adjusted receding horizon planning formulation

One approach to deal with a piecewise continuous nominal control trajectory is to use a variable horizon length T_k in each RHP iteration, and select the value of the timing parameter τ_k in (7) in a separate step. This means that the RHP problem in (7) can be reformulated as:

$$\begin{aligned} \underset{\boldsymbol{u}_k(\cdot),\, T_k}{\text{minimize}} \quad & J = \int_{t_k}^{t_k+T_k} \ell(\boldsymbol{x}_k(t), \boldsymbol{u}_k(t)) \mathrm{d}t \\ \text{subject to} \quad & \boldsymbol{x}_k(t_k) = \boldsymbol{x}_{\text{cur}}, \\ & \boldsymbol{x}_k(t_k + T_k) = \bar{\boldsymbol{x}}_{k-1}(\tau_k) \\ & \dot{\boldsymbol{x}}_k(t) = f(\boldsymbol{x}_k(t), \boldsymbol{u}_k(t)), \\ & \boldsymbol{x}_k(t) \in \mathcal{X}_{\text{free}}, \quad \boldsymbol{u}_k(t) \in \mathcal{U}. \end{aligned} \tag{19}$$

Here, the difference compared to (7) is that T_k is added as a decision variable, and τ_k is removed from being a decision variable and is instead considered as a parameter to the RHP problem. Since τ_k is no longer a decision variable, it is not an issue with using piecewise continuously differentiable functions $\bar{\boldsymbol{x}}_{k-1}(\cdot)$ and $\Psi_k(\cdot)$. This new problem formulation reduces the terminal state manifold to a single state. Furthermore, the cost-to-go function $\Psi_k(\cdot)$ does not need to be explicitly taken into account since the terminal state, and hence also the cost along the remaining nominal solution, is already selected before (19) is solved. By assuming a piecewise continuous input over each planning interval $[t_k, t_{k+1}]$, the problem can thus be discretized using direct optimal control methods and solved using standard NLP interfaces.

4.3 Feasibility, optimality and convergence

The theoretical results in Section 3.2 neglected that the RHP problem is to be discretized when solved using direct optimal control techniques. This discretization introduces the possibility of loosing recursive feasibility (in contrast to the theoretical setup in Lemma 1) since it is not guaranteed that the time-shifted input in (11) is possible to represent in the discretized version. Even if the problem turns out to be feasible, it could be the case that Theorem 1 does not hold, i.e., the new solution has a higher objective function value than the previously optimized solution. Here, we show how to obtain a practical implementation with the properties already guaranteed for the somewhat simplified theoretical setup in Section 3.

At RHP iteration $k-1$, $(\bar{\boldsymbol{x}}_{k-1}(t), \bar{\boldsymbol{u}}_{k-1}(t))$ is executed during the time interval $t \in [t_{k-1}, t_k]$. Since both model errors and external disturbances are assumed to be zero, the state at t_k will be $\bar{\boldsymbol{x}}_{k-1}(t_k)$. By setting $\boldsymbol{x}_{\text{cur}} = \bar{\boldsymbol{x}}_{k-1}(t_k)$ and a desired value of τ_k in (19), the solution at RHP iteration k (if any exists) will be given by $(\boldsymbol{u}_k^\star(\cdot), T_k^\star)$. If the problem is feasible, a new candidate nominal control is to use:

$$\bar{\boldsymbol{u}}_{\text{can}}(t) = \begin{cases} \bar{\boldsymbol{u}}_{k-1}(t), & t \in [t_0, t_k) \\ \boldsymbol{u}_k^\star(t), & t \in [t_k, t_k + T_k^\star) \\ \bar{\boldsymbol{u}}_{k-1}(t + \Delta t_k), & t \in [t_k + T_k^\star, \bar{t}_f^{\text{can}}] \end{cases} \quad (20)$$

where $\Delta t_k = \tau_k - (t_k + T_k^\star)$. In order to guarantee a result similar to Theorem 1, the candidate solution is explicitly benchmarked against the old one $\bar{\boldsymbol{u}}_{k-1}(\cdot)$. If the total objective function value is improved by using the new candidate, i.e.

$$J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{\text{can}}(\cdot)) < J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{k-1}(\cdot)) \quad (21)$$

the nominal trajectory is updated:

$$\big(\bar{\boldsymbol{u}}_k(\cdot), \bar{\boldsymbol{x}}_k(\cdot), \bar{t}_f^k\big) = \Big(\bar{\boldsymbol{u}}_{\text{can}}(\cdot), \ \bar{\boldsymbol{x}}_{\text{can}}(\cdot), \ \bar{t}_f^{k-1} - \Delta t_k\Big), \quad (22)$$

where $\bar{\boldsymbol{x}}_{\text{can}}(\cdot)$ can be computed analogously to $\bar{\boldsymbol{u}}_{\text{can}}(\cdot)$ in (20). Otherwise, the previously optimized solution $\bar{\boldsymbol{u}}_{k-1}(\cdot)$ is reused, which still represents a feasible solution to $\boldsymbol{x}_f$. Hence, a practically useful approach that provides similar guarantees as in Lemma 1 and Theorem 1 is obtained using (20) and (21). Another required property is to ensure that the approach converges to the terminal state $\boldsymbol{x}_f$. Since the timing variable τ_k is updated before and kept fixed during each RHP iteration (as described in Section 4.2), progress towards $\bar{t}_f^{k-1}$ is required for convergence. A sufficient condition for progress is

$$\tau_{k+1} \geq \tau_k + \varepsilon_\tau, \quad (23)$$

which means that $\tau_k = \bar{t}_f^{k-1} < \infty$ will be selected after a finite number of RHP iterations, implying that $\boldsymbol{x}_f$ is used as terminal state in (19) and hence eventually reached.

4.4 Algorithm

The resulting RHP algorithm for motion planning is outlined in Algorithm 1. Before explaining the steps, note that state and control signal trajectories, i.e. $\boldsymbol{x}(\cdot)$ and $\boldsymbol{u}(\cdot)$ in Algorithm 1, are written as $\boldsymbol{x}$ and $\boldsymbol{u}$ for notational brevity.

The inputs to the algorithm are given by the initial and terminal states, a desired planning horizon T, the time between two consecutive RHP iterations δ (which together define the number of discretization points $N = T/\delta$), and the current representation of $\mathcal{X}_{\text{free}}$. A motion planner is

---

Algorithm 1 Receding horizon planning

1: **Input**: $\boldsymbol{x}_0, \boldsymbol{x}_f, T, \delta, \mathcal{X}_{\text{free}}$
2: $(\bar{\boldsymbol{x}}_{-1}, \bar{\boldsymbol{u}}_{-1}, \bar{t}_f) \leftarrow$ Motion planner$(\boldsymbol{x}_0, \boldsymbol{x}_f, \mathcal{X}_{\text{free}})$
3: $\tau_0 \leftarrow t_0 + T, \quad T_0^{\text{init}} \leftarrow \tau_0 - t_0$
4: $(\boldsymbol{x}_0^{\text{init}}, \boldsymbol{u}_0^{\text{init}}) \leftarrow$ resample$(\bar{\boldsymbol{u}}_{-1}, \bar{\boldsymbol{x}}_{-1}, \delta)$
5: **while** $\tau_k \neq \tau_{k-1}$ **do**
6:     Set $\boldsymbol{x}_{\text{cur}} = \bar{\boldsymbol{x}}_{k-1}(t_k)$ in (19)
7:     $(\boldsymbol{u}_k^\star, T_k^\star) \leftarrow$ Solve (19) using $\boldsymbol{u}_k^{\text{init}}, \boldsymbol{x}_k^{\text{init}}, T_k^{\text{init}}$ and τ_k
8:     **if** $J(\boldsymbol{x}_{\text{cur}}, \boldsymbol{u}_k^\star, T_k^\star) < \infty$ **then**
9:         $\Delta t_k \leftarrow \tau_k - (t_k + T_k^\star)$
10:         $(\bar{\boldsymbol{u}}_{\text{can}}, \bar{\boldsymbol{x}}_{\text{can}}) \leftarrow$ get_cand$(\bar{\boldsymbol{u}}_{k-1}, \bar{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k^\star, \Delta t_k)$
11:         **if** $J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}}_{\text{can}}) < J_{\text{tot}}(\boldsymbol{x}_0, \bar{\boldsymbol{u}})$ **then**
12:             Update solution:
            $(\bar{\boldsymbol{u}}_k, \bar{\boldsymbol{x}}_k) \leftarrow (\bar{\boldsymbol{u}}_{\text{can}}, \bar{\boldsymbol{x}}_{\text{can}})$
            $\bar{t}_f^k \leftarrow \bar{t}_f^{k-1} - \Delta t_k$
13:         **else**
14:             $(\bar{\boldsymbol{u}}_k, \bar{\boldsymbol{x}}_k, \bar{t}_f^k) \leftarrow (\bar{\boldsymbol{u}}_{k-1}, \bar{\boldsymbol{x}}_{k-1}, \bar{t}_f^{k-1})$
15:         **end if**
16:     **else**
17:         $(\bar{\boldsymbol{u}}_k, \bar{\boldsymbol{x}}_k, \bar{t}_f^k) \leftarrow (\bar{\boldsymbol{u}}_{k-1}, \bar{\boldsymbol{x}}_{k-1}, \bar{t}_f^{k-1})$
18:     **end if**
19:     Send nominal trajectory to controller :
        send_reference$(\bar{\boldsymbol{u}}_k, \bar{\boldsymbol{x}}_k)$
20:     Update receding horizon terminal constraint:
        $\tau_{k+1} \leftarrow$ update_timing$(t_{k+1}, T, \bar{t}_f^k)$
21:     Initialization for next iteration:
        $T_{k+1}^{\text{init}} \leftarrow \tau_{k+1} - t_{k+1}$
        $\boldsymbol{x}_{k+1}^{\text{init}}, \boldsymbol{u}_{k+1}^{\text{init}} \leftarrow$ resample$(\bar{\boldsymbol{u}}_k, \bar{\boldsymbol{x}}_k, T_{k+1}^{\text{init}}/N)$
22:     Set $k \rightarrow k + 1$
23: **end while**

---

then used on Line 2 to compute a nominal trajectory. To obtain the best overall performance, the nominal trajectory should also be computed while minimizing the same objective function value as in (4) (Bergman et al., 2019b), since the RHP iterations only perform local improvements of the nominal trajectory.

For each RHP iteration k, the problem in (19) is solved from $\boldsymbol{x}_{\text{cur}} = \bar{\boldsymbol{x}}_{k-1}(t_k)$ starting from a provided initialization (discussed further down in this section) and a selected value of τ_k. If this problem is feasible, a new candidate solution is found using (20). If this candidate has a lower full horizon objective function value (i.e. the inequality in (21) holds), the current candidate is selected as solution. Otherwise, the previous solution is reused. The selected solution is sent on Line 19 to a trajectory-tracking controller.

The timing variable τ_k is updated at Line 20 in Algorithm 1. The result in (23) only requires an update policy such that $\tau_{k+1} \geq \tau_k + \varepsilon_\tau$. One policy that satisfies this requirement is:

$$\tau_{k+1} = \min\big(\bar{t}_f^k, t_{k+1} + T\big), \quad (24)$$

since $t_{k+1} + T = \tau_k + \delta$. This means that the terminal state at the next RHP iteration is selected using the user-defined desired planning horizon T in Algorithm 1.

Finally, the solver initialization for the next RHP iteration is done on Line 21 in Algorithm 1. First, T_k is initialized according to the predicted length, i.e., $T_{k+1}^{\text{init}} = \tau_{k+1} - t_{k+1}$. Then, the previous full horizon solution is resampled to be compatible with T_{k+1}^{init}. Assuming a piecewise constant control signal and a multiple-shooting discretization strategy,

one possible resampling of $(\bar{\boldsymbol{x}}_k(\cdot), \bar{\boldsymbol{u}}_k(\cdot))$ is

$$\begin{aligned}
\boldsymbol{u}_{k+1}^{\mathrm{init}}(t_j) &= \bar{\boldsymbol{u}}_k(t_j), \quad \forall j \in [k+1, k+1+N], \\
\boldsymbol{x}_{k+1}^{\mathrm{init}}(t_j) &= \bar{\boldsymbol{x}}_k(t_j), \quad \forall j \in [k+1, k+2+N],
\end{aligned} \quad (25)$$

where N represents the number of discretization points (given by T/δ), and $t_j = t_0 + j\delta^{\mathrm{init}}$, with $\delta^{\mathrm{init}} = T_{k+1}^{\mathrm{init}}/N$. The RHP iterations are solved until $\tau_k = \tau_{k-1}$, which means that $\boldsymbol{x}_f$ has been used as terminal state in (19).

5. SIMULATION STUDY

In this section, the proposed optimization-based RHP approach presented in Section 4 is evaluated in two challenging parking problem scenarios for a truck and trailer system. To evaluate the proposed RHP approach, a lattice-based motion planning algorithm is employed in a first step to compute nominal trajectories using a library of precomputed motion primitives. The lattice-based planner is implemented in C++, while the optimization-based RHP approach is implemented in Python using CasADi together with the warm-start friendly SQP solver WORHP (Büskens and Wassel, 2013).

5.1 Vehicle model

The truck and trailer system is a general 2-trailer with car-like truck (Altafini et al., 2002; Ljungqvist et al., 2019). The system consists of three vehicle segments: a car-like truck, a dolly and a semitrailer. The state vector for the system is given by

$$\begin{aligned}
\boldsymbol{x} &= \begin{bmatrix} \boldsymbol{q}^T & \alpha & \omega & v_1 & a_1 \end{bmatrix}^T \\
\boldsymbol{q} &= \begin{bmatrix} x_3 & y_3 & \theta_3 & \beta_3 & \beta_2 \end{bmatrix}^T
\end{aligned} \quad (26)$$

where (x_3, y_3) and θ_3 represent the position and orientation of the semitrailer, respectively, while β_3 and β_2 denote the joint angles between the semitrailer and the truck. Finally, α and ω are the truck's steering angle and steering angle rate, respectively, while v_1 and a_1 are the longitudinal velocity and acceleration of the truck. Assuming low-speed maneuvers, the truck and trailer system can compactly be modeled as (Ljungqvist et al., 2019):

$$\begin{aligned}
\dot{\boldsymbol{q}} &= v_1 f(\boldsymbol{q}, \alpha), \\
\dot{\alpha} &= \omega, \quad \dot{\omega} = u_\omega, \\
\dot{v}_1 &= a_1 \quad \dot{a}_1 = u_a.
\end{aligned} \quad (27)$$

The control signal to the truck and trailer system is $\boldsymbol{u}^T = [u_\omega \ u_a]$. The vehicle's geometry coincides with the one used in Ljungqvist et al. (2019). The control signal and the vehicle states are constrained as

$$\begin{array}{llll}
|\beta_3| \le 0.87, & |\beta_2| \le 0.87, & |\alpha| \le 0.73, & |\omega| \le 0.8, \\
|v_1| \le 1.0, & |a_1| \le 1.0, & |u_\omega| \le 10, & |u_a| \le 40,
\end{array}$$

and the cost function is chosen as

$$\ell(\boldsymbol{x}, \boldsymbol{u}) = 1 + \frac{1}{2}\left(\alpha^2 + 10\omega^2 + a_1^2 + \boldsymbol{u}^T\boldsymbol{u}\right), \quad (28)$$

which is used both in the lattice-based planner *and* the proposed optimization-based RHP approach as suggested in Bergman et al. (2019b).

5.2 Lattice-based motion planner

As previously mentioned, a lattice-based planner is used in a first step to compute a nominal trajectory to the terminal state. The lattice-based planner uses a discretized state space $\mathcal{X}_d$ and a library of precomputed motion primitives $\mathcal{P}$. During online planning, a nominal trajectory to the terminal state is computed using A* graph search together

with a precomputed free-space heuristic look-up table (HLUT) (Knepper and Kelly, 2006). In this work, we use a similar state-space discretization $\mathcal{X}_d$ as in Ljungqvist et al. (2019), where the position of the semitrailer is discretized to a uniform grid with resolution $r = 1$ m and the orientation of the semitrailer is irregularly discretized $\theta_3 \in \Theta$ into $|\Theta| = 16$ different orientations. It is done to be able to compute short straight trajectories from each $\theta_3 \in \Theta$ (Pivtoraiko et al., 2009). One difference compared to Ljungqvist et al. (2019) is that the longitudinal velocity is here also discretized as $v_1 \in \mathcal{V} = \{-1, 0, 1\}$. All other vehicle states are constrained to zero for all discrete states in $\mathcal{X}_d$ as was done in Ljungqvist et al. (2019). Note, however, that on the trajectory between two states in $\mathcal{X}_d$, the system is free to take any feasible state.

The motion primitive set $\mathcal{P}$ is computed offline using the framework presented in Bergman et al. (2019a) and consists of straight, parallel and heading change maneuvers between discrete states in $\mathcal{X}_d$. Velocity changes between discrete states are only allowed during straight motions. At each discrete state with nonzero velocity, heading change maneuvers are computed to the eight closest adjacent headings in Θ, and parallel maneuvers ranging from ± 10 m with 1 m resolution. The final motion primitive set $\mathcal{P}$ consists of 1184 motion primitives. More details of the lattice-based planner is found in Bergman et al. (2019a).

5.3 Simulation results

The proposed optimization-based RHP approach is evaluated on a reverse parking scenario (see Fig. 2) and a parallel parking scenario (see Fig 4). The obstacles and vehicle bodies are described by bounding circles (LaValle, 2006). In all simulations, the time between two consecutive RHP iterations is $\delta = 0.5$ s. During the simulations, it is assumed that a trajectory-tracking controller is used to follow the computed trajectories with high accuracy between each RHP iteration, however the controller design is out of the scope in this work.

The results for the reverse parking scenario are presented in Fig. 2 and Table 1. As shown in Fig. 2b, the average difference in objective function value ΔJ_{tot} increases as the planning horizon grows. The maximum achievable improvement is 26.5% compared to the nominal solution computed by the lattice-based planner. However, extending the planning horizon beyond $T = 60$ s only leads to a minor improvement. More precisely, if the full horizon (FH) in (4) is improved in a single iteration as done in Bergman et al. (2019b) (i.e. not using a *receding* horizon approach), only an additional improvement of 3.5% is obtained. Furthermore, the average computation time for one RHP iteration $\bar{t}_{\mathrm{RHP}}$ grows with longer planning horizon (especially for $T > 100$ s), which is mainly due to increased problem dimension of the resulting NLP. Since the time needed to execute the trajectory is included in the cost function (28), a practically relevant performance measure is the total time to reach the terminal state t_{tot}, which is the computation time before trajectory execution can start, i.e. the latency time, plus the trajectory execution time. When the nominal solution is improved using the RHP algorithm, the additional latency time Δt_{lat} depends only on the computation time for the first RHP iteration, since the remaining improvements are done during execution. In Table 1, it is shown that the average difference in total time $\Delta\bar{t}_{\mathrm{tot}}$ between using and not using the RHP algorithm obtains its minimum at a planning
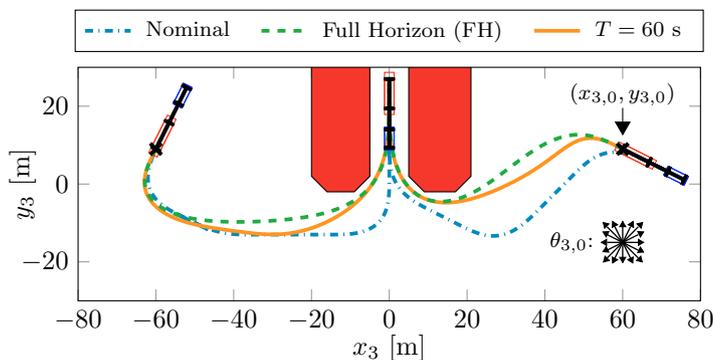
(a) Reverse parking scenario
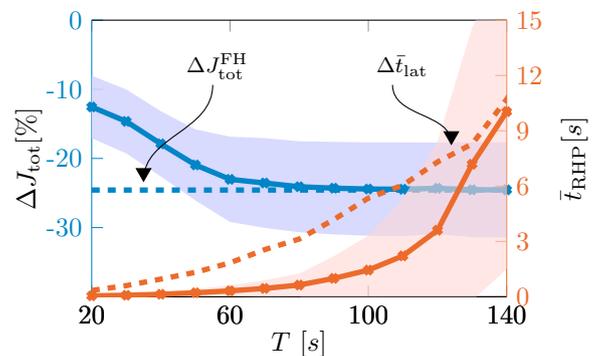
(b) Improvement and computation time vs. planning horizon

Fig. 2. (a): Reverse parking scenario from 32 different initial states. The nominal path (dashdotted) compared to the paths after applying the RHP algorithm using $T = 60$ (solid) and the path using full horizon (FH) improvement (dashed). (b): The average difference in objective function value ΔJ_{tot}, and the average computation time per RHP iteration $\bar{t}_{RHP}$ using different planning horizons T in Algorithm 1. The shaded area represents $\pm$ one standard deviation. Finally, ΔJ_{tot}^{FH} (dashed blue) represents average difference in objective function value using FH improvement, and $\Delta \bar{t}_{lat}$ (dashed red) is the average difference in latency time.

Table 1. Summary of results from the reverse parking scenario in Fig.2. See Fig. 2 and Fig. 4 for a description of the variables.

T [s]	20	40	60	80	120	FH
ΔJ_{tot} [%]	-12.5	-14.4	-23.0	-24.1	-26.3	-26.5
$\bar{t}_{RHP}$ [s]	0.05	0.14	0.32	0.64	3.6	14.0
$\Delta \bar{t}_{lat}$ [s]	0.34	0.96	1.8	3.1	7.3	14.0
$\Delta \bar{t}_{tot}$ [s]	-16.6	-23.5	-30.2	-30.3	-26.2	-22.8

horizon of $60 - 80$ s in this scenario. Using a planning horizon in this interval, the vehicle will in average reach the terminal state more than 30 s faster (latency time + motion execution time) than if the nominal trajectory is planned and executed without improvement.

The results for the parallel parking scenario (Fig. 4 and Table 2) are similar to the ones for the reverse parking scenario. The main differences are that the average decrease in total time $\Delta \bar{t}_{tot}$ and total objective function value ΔJ_{tot} are even more significant in this scenario, with a maximum objective function value improvement of more than 40%. The reason for this is because the lattice-based planner computes a nominal trajectory that is further away from a locally optimal solution due to the confined environment, which leaves large possibilities for improvement to the RHP algorithm. One illustrative example of this is shown in Fig. 3, where it can be seen that the terminal time is nearly halved compared to the nominal solution. Moreover, as can be seen in Table 2 and Fig. 4b, also in this example $\Delta \bar{t}_{tot}$ and ΔJ_{tot} are decreasing rapidly with increased planning horizon until $T = 60 - 80$ s. Beyond that, only a minor additional decrease in ΔJ_{tot} is obtained (full horizon: 2.9%), whereas $\Delta \bar{t}_{tot}$ starts to increase due to an increased average computation time of the first RHP iteration. As a result, in this scenario the vehicle will in average reach the terminal state 54 s faster (latency time + motion execution time) using the proposed RHP approach with planning horizon of $T = 80$ s compared to when the nominal trajectory is planned and executed.

Table 2. Summary of results from the parallel parking scenario in Fig.4. See Fig. 2 and Fig. 4 for a description of the variables.

T [s]	20	40	60	80	120	FH
ΔJ_{tot} [%]	-24.9	-35.2	-40.8	-41.7	-43.4	-43.7
$\bar{t}_{RHP}$ [s]	0.09	0.29	0.77	2.0	10.4	17.0
$\Delta \bar{t}_{lat}$ [s]	0.35	0.73	2.0	3.5	12.3	17.0
$\Delta \bar{t}_{tot}$ [s]	-32.6	-45.7	-53.7	-54.0	-45.9	-44.1

6. CONCLUSIONS AND FUTURE WORK

This paper introduces a new two-step trajectory planning algorithm built on a combination of a search-based motion planning algorithm and an optimization-based receding horizon planning (RHP) algorithm. While the motion planning algorithm quickly can compute a feasible, but often suboptimal, solution taking combinatorial aspects of the problem into account, the RHP algorithm based on direct optimal control techniques iteratively improves the solution quality towards the one typically achieved using direct optimal control. The receding horizon setup makes it possible for the user to conveniently trade off solution time and latency against solution quality. By exploiting the nominal dynamically feasible trajectory, a terminal manifold and a cost-to-go estimate are obtained, which make it possible to provide theoretical guarantees on recursive feasibility, non-increasing objective function value and convergence to the terminal state. These guarantees and the performance of the proposed method are successfully verified in a set of challenging trajectory planning problems for a truck and trailer system, where the proposed method is shown to significantly improve the nominal solution already for short receding planning horizons.

Future work includes to modify the proposed receding horizon planner such that it can be applied in dynamic environments. Another extension is to improve real-time performance by using ideas from fast MPC.
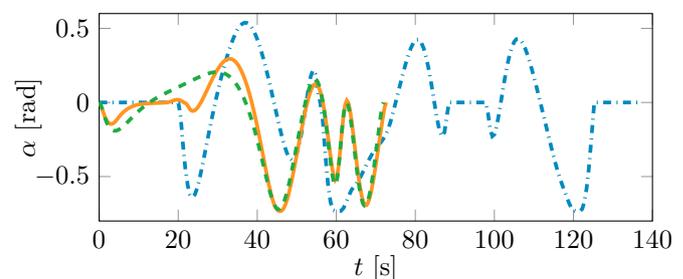

Fig. 3. The resulting steering angle trajectories for the highlighted example in Fig 4a.

(a) Parallel parking scenario

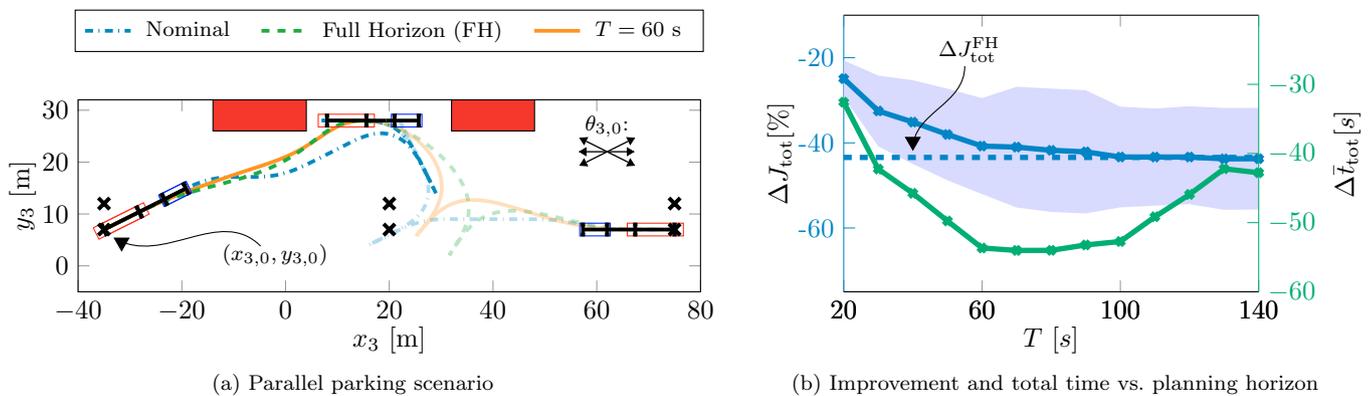(b) Improvement and total time vs. planning horizon

Fig. 4. (a): Parallel parking scenario from 36 different initial states. The nominal solution (dashdotted) is compared with the paths after applying the RHP algorithm using $T = 60$ (solid) and the path using full horizon improvement (dashed). (b): The average difference in objective function value ΔJ_{tot}, and the average difference in total time $\Delta \bar{t}_{\text{tot}}$, i.e., trajectory execution time + computation time for the first RHP iteration, using different planning horizons T in Algorithm 1. The shaded area represents $\pm$ one standard deviation.

REFERENCES

Altafini, C., Speranzon, A., and Johansson, K.H. (2002). Hybrid control of a truck and trailer vehicle. In *Hybrid Systems: Computation and Control*, 21–34. Springer.

Andersson, J.A.E. et al. (2018a). CasADi – A software framework for nonlinear optimization and optimal control. *Math. Programming Computation*.

Andersson, O. et al. (2018b). Receding-horizon lattice-based motion planning with dynamic obstacle avoidance. In *Proceedings of the 57th IEEE Conference on Decision and Control*.

Andreasson, H., Saarinen, J., Cirillo, M., Stoyanov, T., and Lilienthal, A.J. (2015). Fast, continuous state path smoothing to improve navigation accuracy. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 662–669.

Banzhaf, H., Berinpanathan, N., Nienhüser, D., and Zöllner, J.M. (2018). From G^2 to G^3 continuity: Continuous curvature rate steering functions for sampling-based nonholonomic motion planning. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, 326–333.

Bergman, K., Ljungqvist, O., and Axehill, D. (2019a). Improved optimization of motion primitives for motion planning in state lattices. 2019 IEEE Intelligent Vehicles Symposium (IV).

Bergman, K., Ljungqvist, O., and Axehill, D. (2019b). Improved path planning by tightly combining lattice-based path planning and optimal control. *Accepted for publication in IEEE Transactions on Intelligent Vehicles*. Pre-print available at arXiv: https://arxiv.org/abs/1903.07900.

Büskens, C. and Wassel, D. (2013). The ESA NLP solver WORHP. In G. Fasano and J.D. Pintr (eds.), *Modeling and Optimization in Space Engineering*, volume 73, 85–110. Springer New York.

Diehl, M., Bock, H.G., Diedam, H., and Wieber, P.B. (2006). Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, 65–93. Springer.

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2010). Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5), 485–501.

Hart, P.E., Nilsson, N.J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Karaman, S. and Frazzoli, E. (2013). Sampling-based optimal motion planning for non-holonomic dynamical systems. In *2013 IEEE International Conference on Robotics and Automation*, 5041–5047.

Knepper, R.A. and Kelly, A. (2006). High performance state lattice planning using heuristic look-up tables. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3375–3380.

Kuwata, Y., Schouwenaars, T., Richards, A., and How, J. (2005). Robust constrained receding horizon control for trajectory planning. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 6079.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, UK.

Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C.J., and Kumar, V. (2017). Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2(3), 1688–1695.

Ljungqvist, O. et al. (2019). A path planning and path-following control framework for a general 2-trailer with a car-like tractor. *Journal of field robotics*, 36(8), 1345–1377.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer.

Paden, B., Čáp, M., Yong, S.Z., Yershov, D., and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1), 33–55.

Pivtoraiko, M., Knepper, R.A., and Kelly, A. (2009). Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3), 308–333.

Schouwenaars, T., How, J., and Feron, E. (2004). Receding horizon path planning with implicit safety guarantees. In *Proceedings of the 2004 American Control Conference*, volume 6, 5576–5581. IEEE.

Stoneman, S. and Lampariello, R. (2014). Embedding nonlinear optimization in RRT* for optimal kinodynamic planning. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, 3737–3744.

Werling, M., Kammel, S., Ziegler, J., and Gröll, L. (2012). Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3), 346–359.