robotics

MDPI

Article

A Fast and Close-to-Optimal Receding Horizon Control for Trajectory Generation in Dynamic Environments

Khoi Hoang-Dinh [1,*,†], Marion Leibold [2,†] and Dirk Wollherr [2,†]

[1] Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City, No. 12 Nguyen Van Bao, Ward 4, Go Vap District, Ho Chi Minh City 70000, Vietnam
[2] Automatic Control Engineering, Technische Universität München, 80333 München, Germany; marion.leibold@tum.de (M.L.); dw@tum.de (D.W.)
* Correspondence: hoangdinhkhoi@iuh.edu.vn
† These authors contributed equally to this work.

Abstract: This paper presents a new approach for the optimal trajectory planning of nonlinear systems in a dynamic environment. Given the start and end goals with an objective function, the problem is to find an optimal trajectory from start to end that minimizes the objective while taking into account the changes in the environment. One of the main challenges here is that the optimal control sequence needs to be computed in a limited amount of time and needs to be adapted on-the-fly. The control method presented in this work has two stages: the first-order gradient algorithm is used at the beginning to compute an initial guess of the control sequence that satisfies the constraints but is not yet optimal; then, sequential action control is used to optimize only the portion of the control sequence that will be applied on the system in the next iteration. This helps to reduce the computational effort while still being optimal with regard to the objective; thus, the proposed approach is more applicable for online computation as well as dealing with dynamic environments. We also show that under mild conditions, the proposed controller is asymptotically stable. Different simulated results demonstrate the capability of the controller in terms of solving various tracking problems for different systems under the existence of dynamic obstacles. The proposed method is also compared to the related indirect optimal control approach and sequential action control in terms of cost and computation time to evaluate the improvement of the proposed method.

Keywords: optimal control; trajectory generation; robotics; receding horizon control; model predictive control; dynamic environments; simulation results

Citation: Hoang-Dinh, K.; Leibold, M.; Wollherr, D. A Fast and Close-to-Optimal Receding Horizon Control for Trajectory Generation in Dynamic Environments. *Robotics* **2022**, *11*, 72. https://doi.org/10.3390/robotics11040072

Academic Editor: Dario Richiedei

Received: 30 May 2022
Accepted: 4 July 2022
Published: 6 July 2022

1. Introduction

Trajectory planning in robotics and automation has attracted a great deal of attention recently due to the new demands in this area. Besides reaching the goal, it is crucial that the controlled robot is able to react to highly dynamic environments, e.g., avoiding vehicles and pedestrians in the case of autonomously driving cars, or avoiding human co-workers to provide safety in the case of humans and robots working in the same workspace. This requires the robot to adapt rapidly to changing situations. Furthermore, it is desired that, besides safety, also other demands are considered, such as optimizing energy and human comfort, etc. Solving trajectory planning problems while taking all of these aspects into consideration is not a trivial task.

In general, the trajectory planning problem is either solved on a kinematic or dynamic level. On the kinematic level, the outcome of trajectory planning is a set of waypoints; each consists of a time stamp and the position/velocity/acceleration of the system. Then, a controller, i.e., a PID controller, is used to generate a control signal applied to the system. In this area, different planning techniques were first presented in automated vehicle demonstrations [1–3]. One of the first techniques was the use of interpolating curve planners, i.e., with the use of clothoid paths in the Eureka Prometheus Project [4] between 1987 and 1994, where the transitions between linear parts and curves are achieved with

a linear change in curvature. However, this method is very time-consuming and results in a continuous but not smooth path. Later, motion planners based on splines emerged, as in the ARGO Project [5], which have a low computational cost but are not optimal with regard to curvature minimization. Then, graph-search-based planners such as D* were used, as in the Darpa PerceptOR program [6,7], where a drawback is that the resulting path is not continuous. At around the same time, sample-based methods such as rapidly exploring random tree (RRT) [8] were introduced and were commonly used later on as motion planners in a large variety of applications, from autonomous vehicles [9,10] to articulated robots [11,12] and multi-agent systems [13]. In [14], the author introduces a path planning framework using a sample-based approach for articulated robots in the presence of moving obstacles. In recent research, different learning methods have been used in combination with sample-based approaches to improve the efficiency of searching the free-collision path [15,16]. Besides sampling-based methods, optimization-based approaches have also been developed to generate smooth trajectories by minimizing cost functions with regard to velocity/acceleration/jerk terms, as presented in [17]. Nevertheless, these algorithms all compute paths that can be tracked by a controller but not a control force/torque that can be applied on the robot; the constraints on the physical behaviors of the robot cannot be considered in these planning methods.

Differing from sampling-based motion planners, optimal control approaches take the physical behaviors of the robot into account and compute a control input with regard to a pre-defined cost function, which results in an optimal trajectory. The cost functions can be utilized to describe different demands, such as human comfort, optimizing energy, etc.; thus, optimal control methods are very suitable for the aforementioned trajectory planning problem. Most of the optimal control methods are categorized into indirect [18] and direct [19] methods. Indirect methods look at the necessary conditions of optimality of the infinite OCP to derive a boundary value problem (BVP) in ordinary differential equations (ODE). In contrast, direct methods transform the original infinite OCP into a finite nonlinear programming problem and then solve it. However, due to extensive computational effort, these methods are mainly used to compute the trajectories in the offline case and therefore are unable to react to dynamic environments [20]. To overcome this problem, different modifications have emerged in which some pre-computations are performed offline and then used to reduce the computation time in the online phase. For example, in [21], a finite number of global optimal solutions is computed offline; then, they are generalized using support vector machines or Gaussian process regression and used as training data in the online phase. Similarly, machine learning and motion primitives are used in [22,23] to learn precomputed optimal trajectories. However, in real dynamic environments, an infinite number of cases can occur; thus, it is difficult to cover all possibilities with a finite number of precomputed solutions. A solution to deal with dynamic environments is the use of Dynamic Motion Primitives (DMP) [24], where the parameters of the DMP can be adapted online and therefore react to the dynamic environment. However, in this method, the optimality is lost due to the deformation process and therefore the trajectory is not optimal.

Recently, NMPC [25,26] has received a great deal of attention and has been applied in several applications [27–31]. One of the main reasons is the development of different numerical toolboxes and computers with powerful CPUs that are able to solve optimal control problems efficiently. NMPC is a feedback optimal control framework, which basically solves an optimal control problem over a finite receding horizon. Then, only the first interval of the computed control signal is applied until new state measurements are available. After this, the horizon is shifted ahead for one interval and the procedure repeats. The major advantages of NMPC are its fast computation time compared to the original optimal control approaches in [18,19] and its capability of considering dynamic environments. Thus, NMPC has attracted a lot of attention during the last decade. Considerable progress has been made in term of algorithms and software implementations that are able to reduce the computational time of NMPC significantly. In [32], the authors used MPC with a simplified model to find an optimal reaction force profile for a dynamic legged

locomotion system and used a simple PID controller to compute the joint torque, position, and velocity commands based on the reaction forces computed from MPC. In [33,34], the authors proposed a scheme with a limited number of iterations to obtain an approximate solution. This reduces the computation time but the the solution is only suboptimal. For obstacle avoidance in dynamic environments, the authors in [35] developed a robust MPC approach to deal with moving obstacles, but the model is only on a kinematic level, while in [36–38] the authors utilized a simple model of the system to reduce the computational effort. In terms of implementation, the ACADO toolkit [39] is a very strong numerical NMPC solver that can handle a wide range of applications and problems and has been used in different research [40,41]. However, even with the strength of powerful computers, the computation time of NMPC is still significant, especially for nonlinear and complex models, such as an articulated robot or a car-like system. Therefore, most of the works only are only successful with simple or slow systems in a static environment.

SAC was introduced in [42] as a model-based algorithm that is able to compute a sub-optimal control signal for nonlinear systems. It uses the same concept of receding horizon but the main difference between SAC and NMPC, which is also the selling point of SAC, is that it derives a closed-form expression of only one interval of the control signal, which will be applied to the system for a short duration (while NMPC still computes the control signal over the whole horizon). This short duration is usually chosen as equal to a single step size of the control signal. Hence, unlike NMPC, which requires numerical solvers, SAC can derive an analytical expression of this interval. This means that SAC performs faster than NMPC and other optimal control approaches in general, which makes it a promising candidate for online applications. However, this promising analytical solution is only obtainable if there are no constraints in the control problem, the reason being that SAC uses a different method called mode insert gradient [43], which measures the first-order sensitivity of the cost function when the control signal is applied for a short duration. The author then smartly selects an auxiliary cost function, which utilizes this sensitivity formulation such that the analytical solution can be obtained. However, this procedure by default neglects the possibility of adding constraints into the problem. This limits SAC from applications that require additional constraints, i.e., target or final constraints, which are necessary for trajectory planning tasks. Even though these constraints can be formulated as part of the cost function, there is no guarantee that they can be fulfilled.

Inspired by SAC, the proposed method, TC-SAC, was first introduced in [44], and is able to compute an optimal solution very quickly and, in addition, can handle target constraints for trajectory planning tasks. TC-SAC consists of two steps: at first, a first-order gradient approach is used to tackle target constraints for the planning problem and generate an initial guess; after this, SAC is used to improve it further with regard to the cost function. This makes it a promising candidate for real-time optimal control in a dynamic environment. In this work, we extend the method in [44] and evaluate TC-SAC in a broader range of applications. The contribution of this work is as follows:

- We extend the TC-SAC method to cover the cases where target constraints might be violated;
- Different comparisons between TC-SAC, SAC, and indirect optimal control methods are given to show the improvement of the proposed method;
- We show that TC-SAC is able to deal with dynamic environments, which involves avoiding obstacles in our case, and can be applied in different systems without lots of modifications;
- The stability proof of the proposed method is given and discussed.

The remainder of this work is structured as follows. In Section 2, the design of the proposed method is outlined: first, the trajectory tracking problem is formulated in Section 2.1, and then, in Section 2.2, we introduce the basic concept of TC-SAC, as well as the detailed mathematical formulation of the algorithm. Then, the results of the simulation with TC-SAC are given in Section 3. We discuss the stability proof of TC-SAC in Section 4 and conclude our work with the discussion and outlook in Section 5.

2. Materials and Methods

2.1. Problem Formulation

In general, the trajectory tracking task can be formulated as an OCP in the receding horizon, where the terminal constraint is used for the trajectory generation task. For the OCP, a dynamic system with n states and m control inputs is considered and described by a set of ordinary differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)),\ \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1}$$

with $\mathbf{f} : \mathbb{R}^{n+m+1} \rightarrow \mathbb{R}^n$ being nonlinear in state $\mathbf{x} \in \mathbb{R}^n$ and control input $\mathbf{u} \in \mathbb{R}^m$. The initial state of the system is denoted by $\mathbf{x}_0$ and the target configuration $\mathbf{x}_\mathrm{d}(t)$ denotes the final goal. The cost functional

$$J_1 = m_\mathrm{f}(\mathbf{x}(t_\mathrm{f})) + \int_{t_0}^{t_\mathrm{f}} l_1(t, \mathbf{x}(t), \mathbf{u}(t))dt \tag{2}$$

with the performance cost $l_1(\cdot)$ and the terminal cost $m_\mathrm{f}(\cdot)$ are used to measure the performance in the horizon $[t_0, t_\mathrm{f}]$. The optimal control problem over a receding prediction horizon T_P is given by

$$\min_{\mathbf{u}}\ J_1 = m_\mathrm{f}(\mathbf{x}(t_\mathrm{f})) + \int_{t_0}^{t_\mathrm{f}} l_1 dt \tag{3a}$$

$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)),\ \mathbf{x}(t_0) = \mathbf{x}_0,\ \mathbf{x} \in \mathbb{R}^n \tag{3b}$$

$$\mathbf{\Phi} = \begin{pmatrix} x_1(t_\mathrm{f}) - x_{\mathrm{d},1}(t_\mathrm{f}) \\ \vdots \\ x_q(t_\mathrm{f}) - x_{\mathrm{d},q}(t_\mathrm{f}) \end{pmatrix} = 0,\ q \leq n, \tag{3c}$$

where $t_0 = t_\mathrm{cur}$ denotes the current time, $t_\mathrm{f} = t_\mathrm{cur} + T_\mathrm{P}$ is the final time at the end of the prediction horizon, and $(x_{\mathrm{d},1}(t_\mathrm{f}), \ldots, x_{\mathrm{d},q}(t_\mathrm{f}))$ defines q constrained state at the end of the prediction horizon t_f. $\mathbf{\Phi}$ is a column vector that represents the set of q terminal constraints at t_f.

In this paper, the dynamic system in (1) is assumed to be given in control-affine form, i.e.,

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)) = \mathbf{g}(t, \mathbf{x}(t)) + \mathbf{h}(t, \mathbf{x}(t))\mathbf{u}(t) \tag{4}$$

with $\mathbf{f}$ being nonlinear with respect to state $\mathbf{x}$ and linear in control input $\mathbf{u}$.

2.2. Target-Constrained Sequential Action Control

This section outlines the proposed approach, called Target-Constrained Sequential Action Control (TC-SAC) [44], for trajectory planning tasks. The idea of TC-SAC is to utilize the advantage of SAC in terms of fast computation time and to extend the original method with an additional controller to tackle constraints. The overall structure of TC-SAC is given in Figure 1.

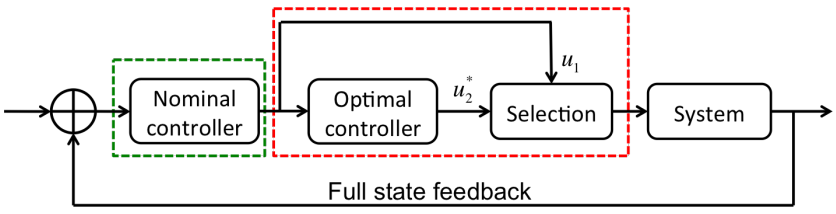

Figure 1. Overview of the controller scheme. First, a nominal control input is computed with the first-order gradient algorithm (green box). After this, the first portion of the control input is updated with SAC (red box).

The proposed method consists of two parts:

1. A nominal controller based on the first-order gradient algorithm (FOGA) [45];
2. An optimal controller based on Sequential Action Control (SAC).

In TC-SAC, an initial guess is first computed by an indirect optimal control method, which is FOGA in our case, in order to consider terminal constraints. Due to time limitations, FOGA is only run for one iteration to obtain the nominal control $\mathbf{u}_1$. Obviously, with one iteration, $\mathbf{u}_1$ is not yet close to the optimal solution. The first interval of $\mathbf{u}_1$ is then improved by SAC since it will be applied on the system in the next iteration. As mentioned, SAC utilizes the concept of mode insert gradient [43] and a proper selection of an additional auxiliary cost function to derive an analytical solution for the optimal control $\mathbf{u}_2^*$, which improves the performance over $\mathbf{u}_1$. Note that $\mathbf{u}_2^*$ differs from $\mathbf{u}_1$ only by the first portion, while the rest remains the same. Then, in the Selection step (see Figure 1), $\mathbf{u}_2^*$ is compared to $\mathbf{u}_1$ in terms of performance and terminal constraint costs. If the comparison shows the improvement of $\mathbf{u}_2^*$, then it is applied to the system for the next iteration.

Looking in depth into the difference between TC-SAC and SAC, our method has a better choice of the nominal controller $\mathbf{u}_1$. To be precise, the original SAC only computes one interval of the control signal $\mathbf{u}_2^*$ per iteration while assuming $\mathbf{u}_1 \equiv 0$. In the case of TC-SAC, instead, $\mathbf{u}_1$ is updated every iteration. Therefore, TC-SAC always improves the optimality of the control signal over time, while SAC does not. The update of $\mathbf{u}_1$ also plays a crucial role for TC-SAC to incorporate target constraints for trajectory tracking tasks and is part of the stability proof that will be discussed later in Section 4. Clearly, this update requires more computational effort than SAC, but it is negligible compared to the benefits that it provides. Next, both parts of our approach will be outlined and explained in detail.

2.2.1. First-Order Gradient Algorithm (FOGA)

The first part of the controller serves the purpose of incorporating constraints, i.e., target constraints, into the OCP, which are crucial for trajectory generation tasks. Since the theoretical background of SAC uses the co-state equation from the Pontryagin principle [46] as part of the calculation (see Section 2.2.2 for more details), FOGA is selected as the solver here to utilize this equation to reduce the amount of steps needed for implementation. FOGA then solves problem (3) for one iteration to find an initial guess. Since this is an optimal control problem with equality constraints at the final state, the idea from [45] is used. First, the dynamic constraint is adjoined to the performance equation l_1 by introducing time-varying Lagrange multiplier vector $\boldsymbol{\rho}$, whose elements are called the co-states of the system. This constructs the Hamiltonian $\mathbb{H} \in \mathbb{R}^1$ defined for all $t \in [t_0, t_\mathrm{f}]$:

$$\mathbb{H}(\mathbf{x}, \mathbf{u}, \boldsymbol{\rho}, t) = \boldsymbol{\rho}^\mathsf{T} \mathbf{f}(\mathbf{x}, \mathbf{u}) + l_1(\mathbf{x}, \mathbf{u}) \tag{5}$$

Following Pontryagin's maximum principle [46], the co-state equation

$$\dot{\boldsymbol{\rho}} = -\left(\frac{\partial \mathbb{H}}{\partial \mathbf{x}}\right)^\mathsf{T} = -\left(\frac{\partial l_1}{\partial \mathbf{x}}\right)^\mathsf{T} - \left(\frac{\partial \mathbf{f}_1}{\partial \mathbf{x}}\right)^\mathsf{T} \boldsymbol{\rho} \tag{6}$$

must be satisfied. Solving (6) requires a terminal condition, which is usually chosen as $\boldsymbol{\rho}(t_\mathrm{f}) = \left(\frac{\partial m_\mathrm{f}}{\partial \mathbf{x}}(t_\mathrm{f})\right)^\mathsf{T}$ if the state $\mathbf{x}$ is not fixed at t_f. In our case, this terminal condition is slightly modified to consider the terminal constraints in (3c):

$$\boldsymbol{\rho}_i(t_\mathrm{f}) = \begin{cases} 0, & i = 1, \cdots, q, \\ \left(\frac{\partial m_\mathrm{f}}{\partial \mathbf{x}_i}\right)^\mathsf{T}_{t=t_\mathrm{f}}, & i = q+1, \cdots, n, \end{cases} \tag{7}$$

with n and q as defined above. Next, the matrix of influence functions $\mathbf{R} \in \mathbb{R}^{n \times q}$ is introduced

$$\dot{\mathbf{R}} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^\mathsf{T} \mathbf{R}, \tag{8}$$

where

$$\mathbf{R}_{ij}(t_f) = \begin{cases} 1, & i = j, \quad i = 1, \cdots, n, \\ 0, & i \neq j, \quad j = 1, \cdots, q. \end{cases} \tag{9}$$

By defining the matrix $\mathbf{R}$, we are able to predict how changes in the control input, $\delta\mathbf{u}(t)$, affect the cost function J_1 and the q terminal constraints in $\mathbf{\Phi}$ by the following; see [45] for more details.

$$\delta J_1 = \int_{t_0}^{t_f} \left(\boldsymbol{\rho}^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} + \frac{\partial l_1}{\partial \mathbf{u}} \right) \delta\mathbf{u}(t) dt \tag{10}$$

$$\delta\mathbf{\Phi} \triangleq \begin{bmatrix} \delta x_1 \\ \vdots \\ \delta x_q \end{bmatrix}_{t=t_f} = \int_{t_0}^{t_f} \mathbf{R}^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \delta\mathbf{u}(t) dt \tag{11}$$

Now, we want to minimize (10) s.t. constraints (11). However, both have linearized relations with regard to $\delta\mathbf{u}$, so there is no minimum for δJ_1. A simple method to create a minimum is to add a quadratic integral penalty function in $\delta\mathbf{u}$ to (10)

$$\delta J_{\text{ex}} = \delta J_1 + \frac{1}{2} \int_{t_0}^{t_f} (\delta\mathbf{u})^\mathsf{T} \mathbf{W} \delta\mathbf{u} \, dt, \tag{12}$$

where $\mathbf{W}(t) \in \mathbb{R}^{m \times m}$ is an arbitrary positive-definite weighting matrix. The problem then becomes a minimization problem of δJ_{ex} subject to (11). Adjoining (11) to (12) with another constant Lagrange multiplier $\boldsymbol{\nu}$, one obtains

$$\delta \bar{J} = \delta J_{\text{ex}} + \boldsymbol{\nu}^\mathsf{T} \left[\int_{t_0}^{t_f} \mathbf{R}^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \delta\mathbf{u}(t) dt - \delta\mathbf{\Phi} \right]. \tag{13}$$

If we neglect the change in coefficients, the first derivative of (13) is given by

$$\delta(\delta \bar{J}) = \int_{t_0}^{t_f} \left[\frac{\partial l_1}{\partial \mathbf{u}} + (\boldsymbol{\rho} + \mathbf{R}\boldsymbol{\nu})^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} + (\delta\mathbf{u})^\mathsf{T} \mathbf{W} \right] \delta(\delta\mathbf{u}) dt \tag{14}$$

Setting (14) to be zero, one can find a solution of

$$\delta\mathbf{u} = -\mathbf{W}^{-1} \left[\frac{\partial l_1}{\partial \mathbf{u}} + (\boldsymbol{\rho} + \mathbf{R}\boldsymbol{\nu})^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right]^\mathsf{T} \tag{15}$$

that minimizes (13). Substituting this into (11), we find that

$$\delta\mathbf{\Phi} = -\mathbf{I}_{\mathbf{\Phi J}} - \mathbf{I}_{\mathbf{\Phi\Phi}} \boldsymbol{\nu} \tag{16}$$

where $\mathbf{I}_{\mathbf{\Phi\Phi}} \in \mathbb{R}^{n \times n}$ and $\mathbf{I}_{\mathbf{\Phi J}} \in \mathbb{R}^{1 \times n}$ are computed by

$$\mathbf{I}_{\mathbf{\Phi\Phi}} = \int_{t_0}^{t_f} \mathbf{R}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \mathbf{W}^{-1} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right)^\mathsf{T} \mathbf{R} dt \tag{17}$$

and

$$\mathbf{I}_{\mathbf{J\Phi}} = \mathbf{I}_{\mathbf{\Phi J}}^\mathsf{T} = \int_{t_0}^{t_f} \left(\boldsymbol{\rho} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} + \frac{\partial l_1}{\partial \mathbf{u}} \right) \mathbf{W}^{-1} \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right)^\mathsf{T} \mathbf{R} dt \tag{18}$$

Assuming that $\mathbf{I}_{\mathbf{\Phi\Phi}}$ is non-singular, we can solve (16) for the value of $\boldsymbol{\nu}$

$$\boldsymbol{\nu} = -\mathbf{I}_{\mathbf{\Phi\Phi}}^{-1} (\delta\mathbf{\Phi} + \mathbf{I}_{\mathbf{\Phi J}}) \tag{19}$$

with $\delta\mathbf{\Phi} = -\epsilon\mathbf{\Phi}[\mathbf{x}(t_f)]$ and a constant $\epsilon \in (0, 1]$. Note that the existence of the inverse of $\mathbf{I}_{\mathbf{\Phi\Phi}}$ is the controllability condition. If $\mathbf{I}_{\mathbf{\Phi\Phi}}^{-1}$ does not exist, it is not possible to control the

system with $\mathbf{u}(t)$ to satisfy one or more of the terminal conditions (see Appendix B2 in [45] for more details). Finally, the new control $\mathbf{u}_1$ is updated to

$$\mathbf{u}_1 = \mathbf{u}_{1,\text{old}} + \delta\mathbf{u} \tag{20}$$

with the old control input $\mathbf{u}_{1,\text{old}}$ and the update $\delta\mathbf{u}$ from (15).

In summary, the procedure of FOGA follows these steps:

1. Initialize a set of control input $\mathbf{u}(t)$;
2. Forward integrate (4) with the initial conditions $\mathbf{x}(t_0)$ and the initial guess of the control input from step 1;
3. Determine the co-state vector $\boldsymbol{\rho}$ and the matrix of influence functions $\mathbf{R}$ by backward integration through (6) and (8) with the terminal conditions (7) and (9);
4. Calculate $\mathbf{I}_{\boldsymbol{\Phi\Phi}}$, $\mathbf{I}_{\boldsymbol{\Phi J}}$ through the integrals (17), (18) simultaneously with step 3;
5. Determine $\boldsymbol{\nu}$ from (19) and compute an estimation of $\delta\mathbf{u}$ via (15);
6. Update the control input $\mathbf{u}_1$ using (20).

Remark: In the standard OCP, steps (2)–(6) are repeated until the optimal solution is found. However, since we pursue fast computation and real-time capability, these steps are only performed once. Obviously, $\mathbf{u}_1$ is not yet close to the optimal solution. SAC is then used to further improve the performance without sacrificing the computation time.

2.2.2. Sequential Action Control

The motivation of using Sequential Action Control (SAC) is that, instead of computing the control input for the whole prediction horizon, it is more crucial to consider the next interval since it will be applied to the system first. On the other hand, in dynamic environments, the predicted controller is affected by moving obstacles or might be completely changed if the final goal changes on-the-fly. SAC therefore aims to improve the control input for only the next interval, but still uses the same prediction horizon $[t_0, t_\text{f}]$ for the performance cost evaluation.

Assuming that the nominal controller $\mathbf{u}_1$ is obtained using FOGA, described in Section 2.2.1, we want to find a control $\mathbf{u}_2^*$, denoted as the optimal control, that further improves the cost function (2) with regard to the dynamic system (4). SAC then computes a triplet consisting of the control value $\mathbf{u}_2^*$, its application time τ_m, and the application duration λ. This triplet is called an action, and the control signal can be written as

$$\mathbf{u}(t) = \begin{cases} \mathbf{u}_1, & t \notin \left[\tau_m - \frac{\lambda}{2}, \tau_m - \frac{\lambda}{2}\right] \\ \mathbf{u}_2^*, & t \in \left[\tau_m - \frac{\lambda}{2}, \tau_m - \frac{\lambda}{2}\right] \end{cases} \tag{21}$$

with the nominal controller $\mathbf{u}_1$ and the optimal controller $\mathbf{u}_2^*$. This can be interpreted as a switching controller, where SAC switches between two modes. These two modes are given by

$$\mathbf{f}_1(t) \triangleq \mathbf{f}(\mathbf{x}(t), \mathbf{u}_1(t)) \tag{22}$$

for the nominal controller $\mathbf{u}_1$ and

$$\mathbf{f}_2(t, \tau_m) \triangleq \mathbf{f}(\mathbf{x}(t), \mathbf{u}_2^*(\tau_m)) \tag{23}$$

for the optimal controller $\mathbf{u}_2^*$. In this paper, the application time τ_m is deterministic with $\tau_m = t_\text{cur} + \tau_\text{sample}/2$, where t_cur denotes the current time and τ_sample denotes the sampling time. Now, recall that our aim is to improve the cost function (2) with the new controller $\mathbf{u}_2^*$ applied within the duration λ. We then rely on the mode insert gradient [43], which evaluates the first-order sensitivity of the cost (2)

$$\frac{dJ_1}{d\lambda^+}(\tau_m) = \boldsymbol{\rho}(\tau_m)^\mathsf{T}(\mathbf{f}_2(\cdot, \tau_m) - \mathbf{f}_1(\cdot, \tau_m)) \tag{24}$$

This equation measures how the cost is influenced by varying the length λ of the application of the optimal control $\mathbf{u}_2^*$. The co-state $\boldsymbol{\rho}$ in (24) is computed based on (6) and (7). To reduce the cost J_1, (24) should be driven to a desired negative value $\alpha_d \in \mathbb{R}^-$. This can be done by simply introducing an auxiliary cost function

$$\begin{aligned} l_2(s) &= l_2(\mathbf{x}(s), \mathbf{u}_1(s), \mathbf{u}_2(s), \boldsymbol{\rho}(s)) \\ &= \frac{1}{2}\left[\frac{\mathrm{d}J_1}{\mathrm{d}\lambda^+} - \alpha_d\right]^2 + \frac{1}{2}\|\mathbf{u}_2(s)\|_{\mathbf{S}}^2 \end{aligned} \tag{25}$$

with $\mathbf{S} > 0$ and $\|\mathbf{u}_2(s)\|_{\mathbf{S}}^2 = \mathbf{u}_2(s)^T\mathbf{S}\mathbf{u}_2(s)$. Solving the minimization problem of (25) results in a control that achieves the desired sensitivity α_d. For models in control-affine form (4), the solution for this minimization is given analytically by

$$\mathbf{u}_2^* = (\boldsymbol{\Lambda} + \mathbf{S}^\mathsf{T})^{-1}[\boldsymbol{\Lambda}\mathbf{u}_1 + \mathbf{h}(\mathbf{x})^\mathsf{T}\boldsymbol{\rho}\alpha_d] \tag{26}$$

with $\boldsymbol{\Lambda} = \mathbf{h}(\mathbf{x})^\mathsf{T}\boldsymbol{\rho}\boldsymbol{\rho}^\mathsf{T}\mathbf{h}(\mathbf{x})$.

2.2.3. Extended Sequential Action Control with Target Constraints

One problem that arises from the theoretical background of SAC, presented in Section 2.2.2, is that it cannot handle constraints, and hence using SAC solely in the next step can lead to the violation of the constraint (3c). To prevent this occurrence, we extend the the original SAC method by using the knowledge from FOGA. We redefine the auxiliary cost function in (25) as

$$\begin{aligned} l_2^{\text{ext}}(s) &= l_2(\mathbf{x}(s), \mathbf{u}_1(s), \mathbf{u}_2(s), \boldsymbol{\rho}(s)) \\ &= \frac{1}{2}\left[\frac{\mathrm{d}J_1}{\mathrm{d}\lambda^+} - \alpha_d\right]^2 + \frac{1}{2}\left[\frac{dl_c}{d\lambda^+} - \alpha_c\right]^2 + \frac{1}{2}\|\mathbf{u}_2(s)\|_{\mathbf{S}}^2 \end{aligned} \tag{27}$$

where $l_c = \frac{1}{2}\boldsymbol{\Phi}^\mathsf{T}\mathbf{Q}_c\boldsymbol{\Phi}$, $\mathbf{Q}_c \in \mathbb{R}^{q\times q}$ is a positive definite matrix and $\boldsymbol{\Phi}$ is the vector of target constraints defined in (3c). l_c can be interpreted as the additional constraint cost and therefore $\frac{dl_c}{d\lambda^+}$ measures how this constraint cost is influenced by varying the length λ of the optimal control $\mathbf{u}_2^*$. Hence, solving the new minimization problem of (27) also drives this sensitivity to a negative value $\alpha_c \in \mathbb{R}^-$. This leads to the reduction of l_c, which then helps SAC to prevent the constraints from being violated.

The only problem now is the evaluation of $\frac{dl_c}{d\lambda^+}$. We have

$$\frac{dl_c}{d\lambda^+}(\tau_m) = \frac{dl_c}{d\mathbf{x}(t_\mathrm{f})}\frac{d\mathbf{x}(t_\mathrm{f})}{d\lambda^+}(\tau_m) = \boldsymbol{\Phi}^\mathsf{T}\mathbf{Q}_c\frac{d\mathbf{x}(t_\mathrm{f})}{d\lambda^+}(\tau_m) \tag{28}$$

From (11), we have

$$\begin{aligned} \delta\boldsymbol{\Phi} = \delta\mathbf{x}(t_\mathrm{f}) &= \int_{t_0}^{t_\mathrm{f}} \mathbf{R}^\mathsf{T}\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\delta\mathbf{u}(t)dt \\ &= \int_{\tau_m}^{\tau_m+\delta\lambda} \mathbf{R}^\mathsf{T}\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}_2 - \mathbf{u}_1)dt \end{aligned} \tag{29}$$

where $[\tau_m, \tau_m + \delta\lambda]$ is the infinitesimal duration where $\mathbf{u}_2$ is applied. As $\delta\lambda \to 0$, (29) can be written as

$$d\mathbf{x}(t_\mathrm{f}) = \mathbf{R}^\mathsf{T}\frac{d\mathbf{f}}{d\mathbf{u}}(\tau_m)(\mathbf{u}_2 - \mathbf{u}_1)d\lambda \tag{30}$$

or

$$\frac{d\mathbf{x}(t_\mathrm{f})}{d\lambda^+}(\tau_m) = \mathbf{R}^\mathsf{T}\mathbf{h}(\tau_m)(\mathbf{u}_2 - \mathbf{u}_1) \tag{31}$$

Substituting this into (28) and solving the minimization problem of (27) in the same way as in Section 2.2.2, we obtain a new analytical solution of $\mathbf{u}_2^*$

$$\mathbf{u}_2^* = (\mathbf{\Lambda}_1 + \mathbf{\Lambda}_2 + \mathbf{S})^{-1}(\mathbf{\Lambda}_1\mathbf{u}_1 + \mathbf{\Lambda}_2\mathbf{u}_1 + \mathbf{h}^\mathsf{T}\boldsymbol{\rho}\alpha_d + \mathbf{h}^\mathsf{T}\mathbf{R}_c^\mathsf{T}\alpha_c) \tag{32}$$

with $\mathbf{\Lambda}_1 = \mathbf{h}^\mathsf{T}\boldsymbol{\rho}\boldsymbol{\rho}^\mathsf{T}\mathbf{h}$, $\mathbf{\Lambda}_2 = \mathbf{h}^\mathsf{T}\mathbf{R}_c^\mathsf{T}\mathbf{R}_c\mathbf{h}$, $\mathbf{R}_c = \mathbf{\Phi}^\mathsf{T}\mathbf{Q}_c\mathbf{R}^\mathsf{T}$.

In this work, the application time λ is set to be equal to τ_{sample} such that $\mathbf{u}_2^*$ is applied for one interval. Additionally, $\mathbf{u}_2^*$ is only applied if it results in both a smaller performance cost J_1 and terminal constraint cost l_c over the prediction horizon T_P compared to the nominal control $\mathbf{u}_1$. This is considered as the Selection procedure in the control scheme, as illustrated in Figure 1.

In order to obtain a feasible solution, a boundary of control signals is given by box constraints, i.e., $U = [u_{\min}, u_{\max}]^m$, by simply saturating the control output computed above. Overall, Algorithm 1 outlines the general structure of the proposed method.

Algorithm 1 TC-SAC.

Initialize $\mathbf{x}_0$, $\mathbf{x}_\text{d}$, current time t_{curr}, prediction horizon T_P, sampling time τ_{sample}, end time T_{end}, initial guess for nominal control $\mathbf{u}_1$.

while $t_{\text{curr}} < T_{\text{end}}$ **do**
  if $t_{\text{curr}} \geq \tau_m$ **then**
    $\tau_m = t_{\text{curr}} + \tau_{\text{sample}}/2$
    Simulate $(\mathbf{x}, \boldsymbol{\rho}, \mathbf{R})$ using $\mathbf{u}_1$ for $t \in [t_0, t_\text{f}]$
    Compute $\boldsymbol{I}_{\psi\psi}$, $\boldsymbol{I}_{J\psi}$ as in (17) and (18), $\boldsymbol{\nu}$ by (19), change in control $\delta\mathbf{u}$ by (15)
    Update $\mathbf{u}_1 \leftarrow \mathbf{u}_1 + \delta\mathbf{u}$
    Saturate $\mathbf{u}_1$ to $[u_{\min}, u_{\max}]$
    Simulate $(\mathbf{x}, \boldsymbol{\rho})$ using new $\mathbf{u}_1$ for $t \in [t_0, t_f]$
    Compute new costs $J_{1,\text{init}}, l_{c,\text{init}}$
    Specify α_d
    Compute $\mathbf{u}_2^*$ from (26) and saturate it to $[u_{\min}, u_{\max}]$
    Initialize $k = 0$, $J_{1,\min} \leftarrow J_{1,\text{init}}$, $J_{1,\text{SAC}} \leftarrow \infty$, $l_{c,\min} \leftarrow l_{c,\text{init}}$, $l_{c,\text{SAC}} \leftarrow \infty$
    while $k < k_{\max}$ **do**
      $\lambda = \tau_{\text{sample}}$
      $(\tau_0, \tau_f) = (\tau_m - \frac{\lambda}{2}, \tau_m + \frac{\lambda}{2})$
      Re-simulate $(\mathbf{x}, \boldsymbol{\rho})$ applying control (21)
      Compute new cost $J_{1,\text{SAC}}, l_{c,\text{SAC}}$
      $k = k + 1$
      if $J_{1,\text{SAC}} < J_{1,\text{init}}$ **and** $l_{c,\text{SAC}} < l_{c,\text{init}}$ **then**
        $J_{1,\min} \leftarrow J_{1,\text{SAC}}, l_{c,\min} \leftarrow l_{c,\text{SAC}}$
      end if
    end while
    if $J_{1,\text{SAC}} < J_{1,\text{init}}$ **and** $l_{c,\text{SAC}} < l_{c,\text{init}}$ **then**
      $\mathbf{u}_1 \leftarrow \mathbf{u}_2^*$ for $t \in (\tau_m - \frac{\lambda}{2}, \tau_m + \frac{\lambda}{2})$
    end if
  else
    Apply control $\mathbf{u}_1$
  end if
  $(t_0, t_f) = (t_0 + \tau_{\text{sample}}, t_0 + T_{\text{pre}} + \tau_{\text{sample}})$
end while

3. Results

This section presents different examples to highlight the improvement of the proposed approach. We apply our method on the reaching task and trajectory tracking task for the 2DOF robotic arm and compare the results with the original SAC. An interesting question that arises from a theoretical point of view is the extent to which the extended SAC step in Section 2.2.3 affects the performance of the system. Therefore, two additional controllers

are added for comparison: first, only the extended SAC with target constraint is used to compare to SAC, and second, FOGA with multi-steps is used to compare to TC-SAC.

3.1. Reaching Motion Task

In this task, we evaluate the performance of the proposed approach on a 2DOF robotic arm in the vertical plane, as shown in Figure 2. The full dynamics of the robot are considered, including friction and gravitation. A detailed mathematical explanation of this model can be found in [47]. The task of the robot is to reach a pre-defined position. The performance cost is chosen to penalize the error between the current and desired states

$$l_1(t) = (\mathbf{x}(t) - \mathbf{x}_d)^T \mathbf{Q} (\mathbf{x}(t) - \mathbf{x}_d), \tag{33}$$

where $\mathbf{Q} \succeq 0$ is the weighting matrix. The target constraint is also set to be the desired position. The prediction horizon is chosen $T_\mathrm{p} = 0.3$ s and the sampling rate is set to 1kHz. The initial position of the robot is $\mathbf{x}_0 = (-\frac{\pi}{2}, 0, 0, 0)$ and the control $\mathbf{u}$ is bounded by $\mathbf{u} \in [-50, 50]$ Nm. The sensitivity α_d is chosen proportional to the cost J_1, $\alpha_\mathrm{d} = \omega_{\alpha_d} J_1$ with $\omega_{\alpha_d} = -10$. Similarly, the sensitivity α_c is set to be $\alpha_c = \omega_{\alpha_c} l_c$ with $\omega_{\alpha_c} = -5$.

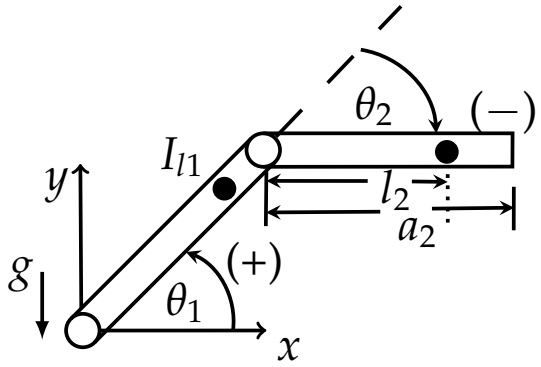

Figure 2. Setup of the two-degrees-of-freedom robot used for the simulation.

3.1.1. Upright as Desired Position

First, the results are evaluated in the case of $\mathbf{x}_\mathrm{d}$ being the upright position, $\mathbf{x}_\mathrm{d} = (\frac{\pi}{2}, 0, 0, 0)$. The weighting matrix for the performance cost (33) is $\mathbf{Q} = \mathrm{diag}([100, 50, 0.0001, 0.0001])$. Having the first two values of $\mathbf{Q}$ much higher than the latter means that the position errors are penalized more heavily than the velocity errors, which is necessary for tracking tasks. The weighting matrix for l_c in (27) is $\mathbf{Q}_c = \mathrm{diag}([10, 1000, 10, 10])$. For FOGA, we run in two iterations to allow a fair comparison to TC-SAC, in terms of cost improvement and computation time. The simulation is run for 1.5 s and the result is shown in Figure 3. All methods succeed in controlling and stabilizing the robot. However, TC-SAC and FOGA need only 0.5 s to converge to $\mathbf{x}_d$, while SAC and the extended SAC take more than 1 s to converge. The overall cost is shown in Table 1. Furthermore, each method is run 100 times to obtain the average computation time. Note that the code is run on the Matlab environment so the computation time is only used to evaluate the speed of these methods relatively.

Comparing between SAC and the extended SAC, the new sensitivity term helps to improve the constraint in the states of the robot. This can be seen clearly in θ_2, where the extended SAC keeps the deviation between the current and desired state at a smaller value compared to SAC. This shows the effectiveness of the extended SAC in terms of tightening the target constraints. Comparing between TC-SAC and FOGA, the performance of both is quite similar, but the computation time of TC-SAC is 34 s less than FOGA with two iterations. This is because one iteration of SAC takes much less time than one iteration of FOGA, since SAC has a direct analytical solution. Therefore, TC-SAC is preferable since it can achieve the same performance in a shorter amount of time.

Table 1. Total cost and computation time.

	SAC	Extended SAC	TC-SAC	FOGA
Total cost	316.74	291.02	152.21	150.60
Computation time	18.18 s	19.60 s	89.82 s	124.63 s

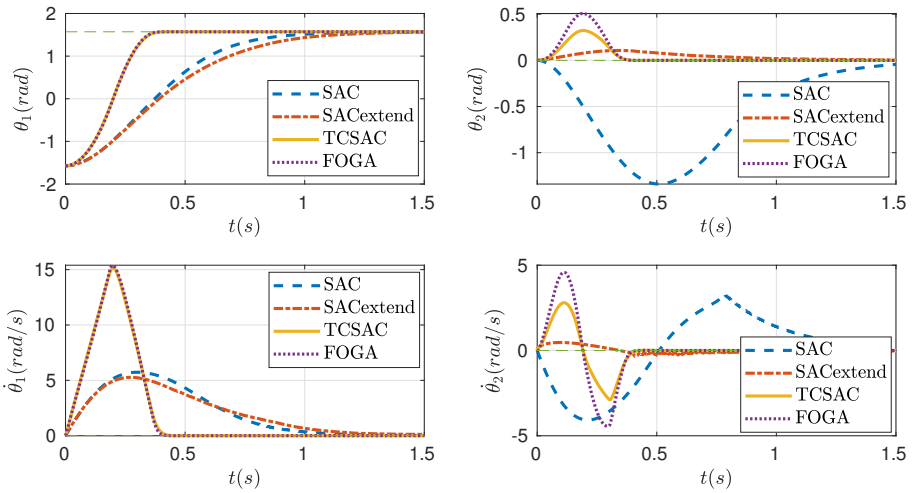

Figure 3. States of 2DOF robotic arm when the designed position is upright.

3.1.2. Arbitrary Position

In this section, we want to analyze the performance of the methods in the case of any arbitrary desired position. In this case, the controller has to compensate the gravitational force, which is zero if the robot is at the upright position (equilibrium point). This highlights the role of the target constraint in TC-SAC and FOGA in terms of convergence. The desired position is set to be $\mathbf{x}_d = (\frac{3\pi}{4}, \frac{-3\pi}{4}, 0, 0)$ and other parameters are set to be the same as in the case of the upright position. The simulation is run for 2 s and the result is shown in Figure 4. It can be seen that TC-SAC and FOGA quickly converge to the desired position in approximately 0.5 s, while both SAC and the extended SAC methods cannot converge to the desired position in θ_1 and θ_2. These offsets appear in any desired position that is not the equilibrium point. Therefore, TC-SAC and FOGA are preferable for reaching/tracking tasks, even though the computation time is longer.

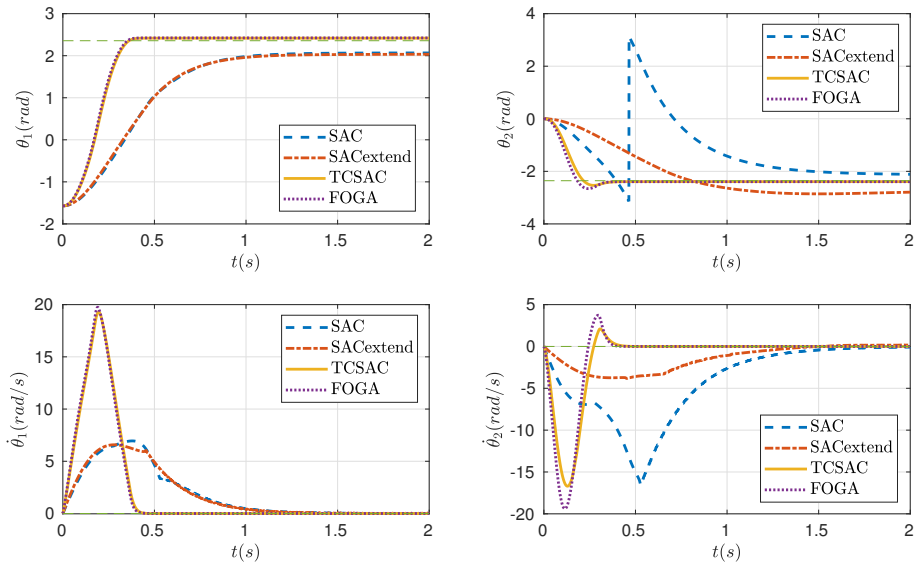

Figure 4. States of 2DOF robotic arm in the case of arbitrary desired position.

3.2. Tracking an Ellipse Trajectory

Moving one step further, we evaluate the methods in a trajectory tracking task. Since SAC and the extended SAC are incapable of reaching an arbitrary position, they are excluded from this task. The trajectory is set to be an ellipse where the radii along the x-axis and y-axis are 0.3 m and 0.18 m, respectively. For the sake of simplicity, the robot follows the ellipse with a constant velocity and the whole ellipse takes 3 s to finish. The starting position is set to be upright. In this simulation, the weighting matrices $\mathbf{Q}, \mathbf{Q}_c$ are set to be $\mathbf{Q} = \text{diag}([100, 50, 10, 10])$ and $\mathbf{Q}_c = \text{diag}([10, 100, 0.1, 0.1])$. Other parameters remain the same. Figure 5 shows the tracking performance between TC-SAC and FOGA in the XY graph, while Figure 6 shows this in each state of the robot. Figure 5 also illustrates the configuration of the robot at different positions along the ellipse.

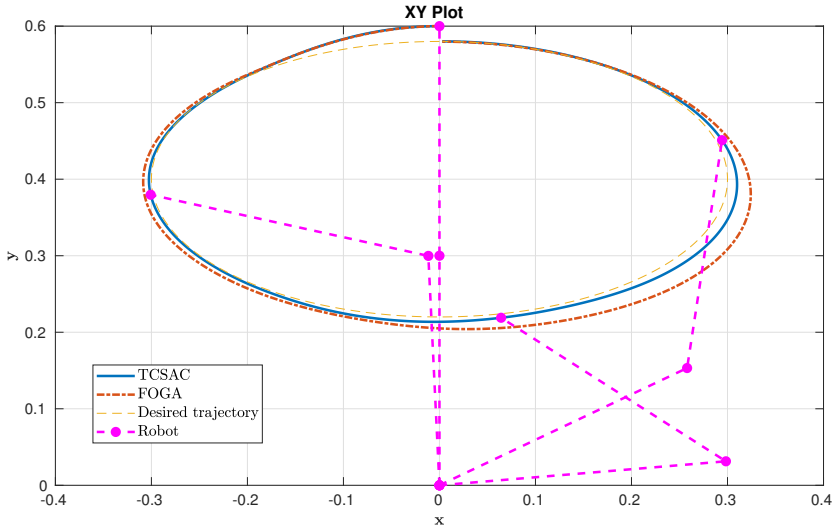

Figure 5. Tracking performance of TC-SAC and FOGA.

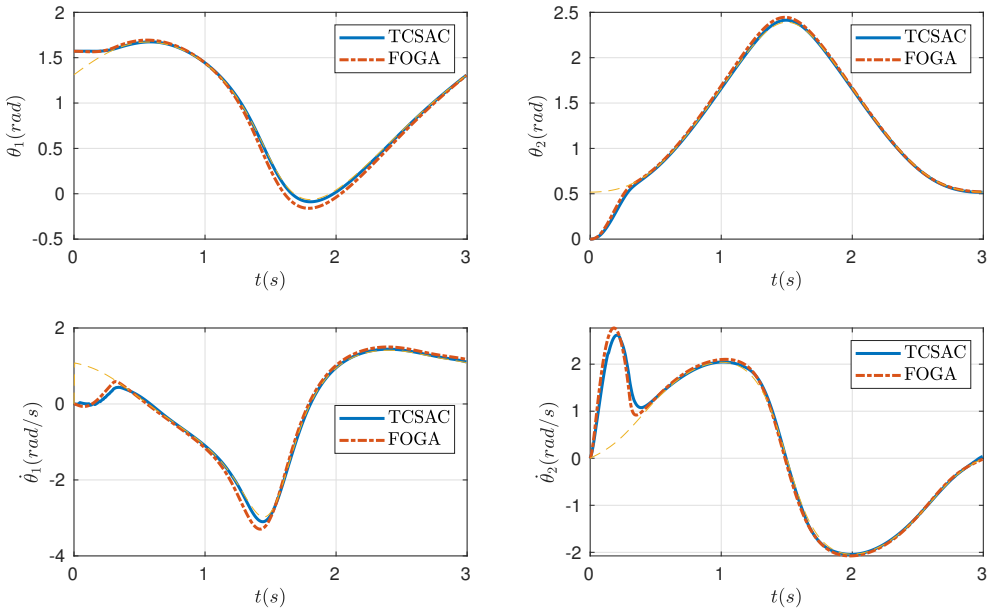

Figure 6. States of 2DOF robotic arm in case of tracking an ellipse trajectory.

It can be seen that both controllers are able to track the given trajectory; however, TC-SAC performs better than FOGA in this task, especially on the right side of the ellipse. From a mathematical point of view, TC-SAC can be interpreted as a combination of one iteration of FOGA plus an update from SAC afterward. Hence, it can be said that the SAC

step improves the cost further compared to the second iteration step of FOGA, and it is achieved in a much shorter amount of time. This highlights the advantage of TC-SAC in the case where the controller needs to be computed rapidly due to time restrictions, i.e., in dynamic environments. TC-SAC helps to improve the overall cost substantially, without losing too much computation time. Figure 7 also shows the control signal of TC-SAC on the first and second joint of the robot. The last graph in Figure 7 displays at which time step SAC is activated (represented as 1), which indicates when the control signal computed by SAC reduces the cost. By looking at how often SAC overtakes FOGA ($\mathbf{u}_2^*$ is applied instead of $\mathbf{u}_1$), we can justify the effectiveness of the additional SAC step in TC-SAC. It can be seen that $\mathbf{u}_2^*$ is used most of the time, which means that SAC often improves the solution of FOGA ($\mathbf{u_1}$). This proves the effectiveness of this additional step since the computation of SAC is fast. In conclusion, TC-SAC is preferable in applications where the controller needs to be computed quickly and optimally. In the next section, we will show that TC-SAC is also applicable in different applications.

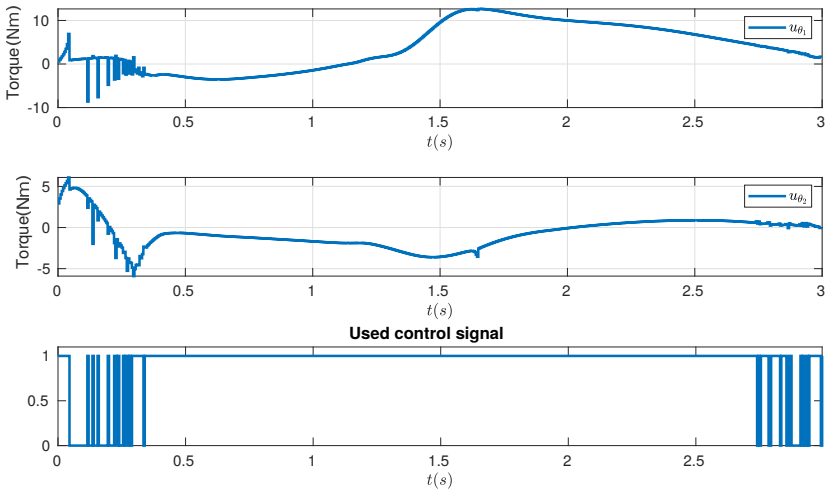

Figure 7. Control signal of the 2DOF robotic arm.

3.3. Trajectory Tracking in Dynamic Environment of a Car-Like System

Autonomous driving recently has received a lot of attention from industry and researchers. In the field of autonomous cars, it is crucial that the controlled car is able to react to highly dynamic environments. Since other participants in traffic cannot be predicted beforehand, the car has to adapt rapidly to changing situations. This section presents the implementation of TC-SAC for the trajectory tracking task, with an emphasis on obstacle avoidance in both static and dynamic cases. We use the single-track model (also known as the bicycle model [2,28,48–50]), as shown in Figure 8, as this model is widely used to represent vehicles.

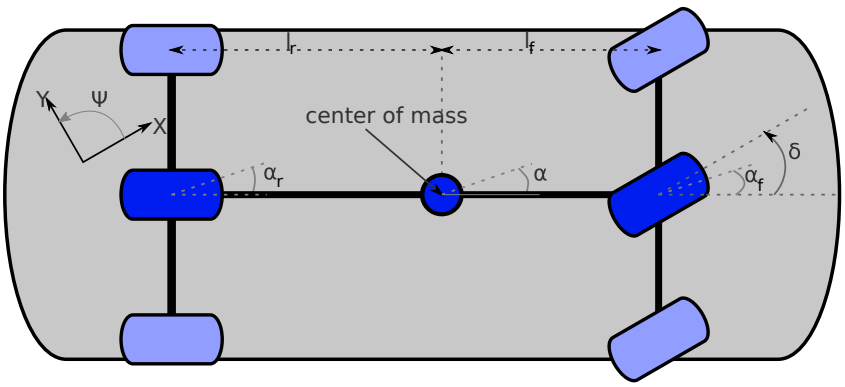

Figure 8. Schematic diagram of the single-track model used in this work.

The single-track model used in this work is nonlinear, with 7 states and 2 control input

$$\mathbf{x} = \begin{bmatrix} X, Y, \Psi_c, v, \alpha, \dot{\Psi}_c, \delta \end{bmatrix} \tag{34}$$

$$\mathbf{u} = \begin{bmatrix} \dot{\delta}_{\text{input}} \\ M \end{bmatrix} \tag{35}$$

with

$$\dot{\delta} = \dot{\delta}_{\text{input}} \tag{36}$$

with the x and y-coordinates X and Y, the orientation Ψ_c, the velocity v, the side slip angle α, the change in orientation $\dot{\Psi}_c$, the steering angle δ, and the applied torque M. The last state in x is the additional state added so that the single-track model is in control-affine form. Further details about this car model are given in Appendix A.

A Lissajous curve with a ratio of the frequencies of $\frac{2}{3}$ and a phase shift of $\frac{\pi}{2}$ is used for this tracking task.

$$\begin{aligned} x_{\text{ref}}(t) &= 50 \sin(0.05\pi t) \\ y_{\text{ref}}(t) &= 50 \cos(0.075\pi t) \\ v_{\text{ref}}(t) &= \sqrt{\dot{x}_{\text{ref}}^2(t) + \dot{y}_{\text{ref}}^2(t)} \end{aligned} \tag{37}$$

This reference trajectory is chosen since it results in different curvature and velocity at every point, which makes it challenging for tracking. The velocity varies between approximately $3.65\frac{\text{m}}{\text{s}}$ and $14.16\frac{\text{m}}{\text{s}}$, which does not exceed the limit of the dynamic of the car. To be able to avoid the obstacles, an additional term is added into the cost function, which is defined as

$$l_{1,\text{avoid}} = \begin{cases} C_{\text{obstacle}} \cdot (\text{dist}(\cdot) - r)^2, & \text{dist}(\cdot) < r \\ 0, & \text{dist}(\cdot) > r \end{cases} \tag{38}$$

with the distance between the car from the center to an obstacle $\text{dist}(\cdot) = \text{dist}(x, y, x_c, y_c) = \sqrt{(x - x_c)^2 + (y - y_c)^2}$, the weighting factor C_{obstacle}, and the obstacle radius r. The overall performance cost l_1 is then given by $l_1 = l_{1,\text{track}} + l_{1,\text{avoid}}$. Since obstacle avoidance is most crucial for the safety of passengers and other traffic participants, C_{obstacle} is set to 10^6 and therefore it is considered with a much higher weighting factor than the tracking task.

First, we test our method in the case of trajectory tracking with static obstacle avoidance. It is assumed that the obstacles are 2 m in diameter, which is close to the width of real cars. However, since avoidance cannot completely be guaranteed with soft constraints, a safety margin of 1 m around the obstacle is added, which leads to an obstacle radius r = 2 m to increase safety. The position of the obstacles is given in Figure 9 and marked by the circles.

It can be seen from the plot that the car successfully avoids all obstacles, while it is still able to track the reference trajectory. Looking at the cost measurement, there are two peaks at around 51 s and 91 s, which are the obstacles in the top right and bottom right of the curve. This means that the obstacles actually interfere with the safety margin around the car, although there is no collision. Therefore, choosing a proper safety margin is very important to ensure safety.

For dynamic obstacle avoidance, again, the Lissajous curve is chosen as the reference trajectory. The prediction horizon is increased to 2 s to consider the more difficult task of dynamic obstacle avoidance. 13 obstacles are included for the scenario: one at the center, which is static, 6 on a circular trajectory with a radius of 40 m, and 6 on a circular trajectory with 50 m radius. Both circles have the same center as the Lissajous curve. The obstacles are equally distributed on these two circles and take 20 s for a full lap. It is assumed that the future path of the obstacle is known by the car and therefore can be considered exactly by TC-SAC. The safety margin around the obstacles is increased to 4 m.

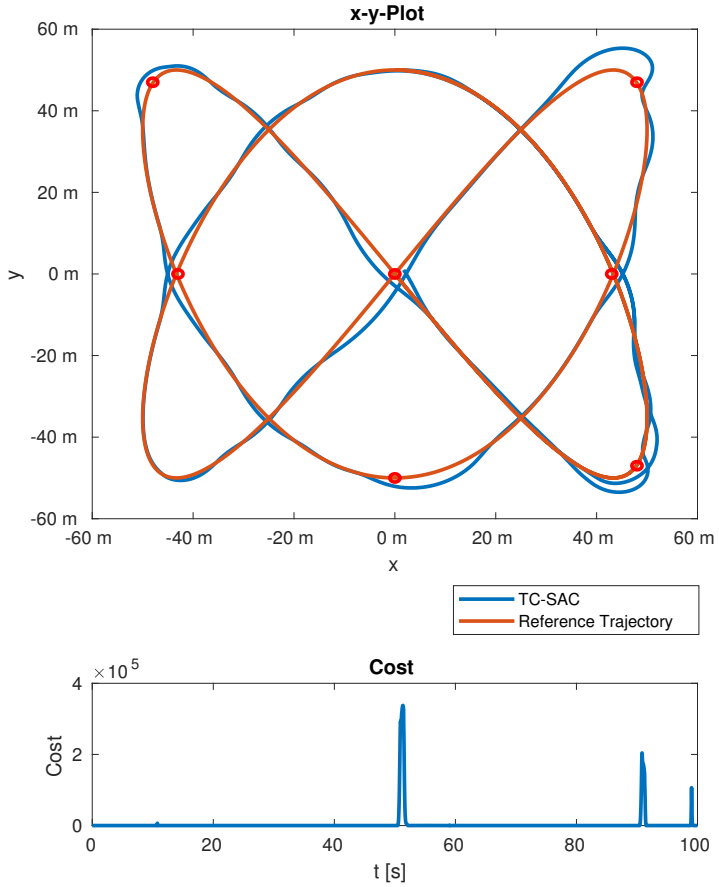

Figure 9. Performance of TC-SAC on a Lissajous curve as reference trajectory with avoidance of static obstacles.

A short frame of the dynamic behavior is given in Figure 10, which shows the scenario at time stamps between 7 s and 10 s with 1 s between the single pictures.

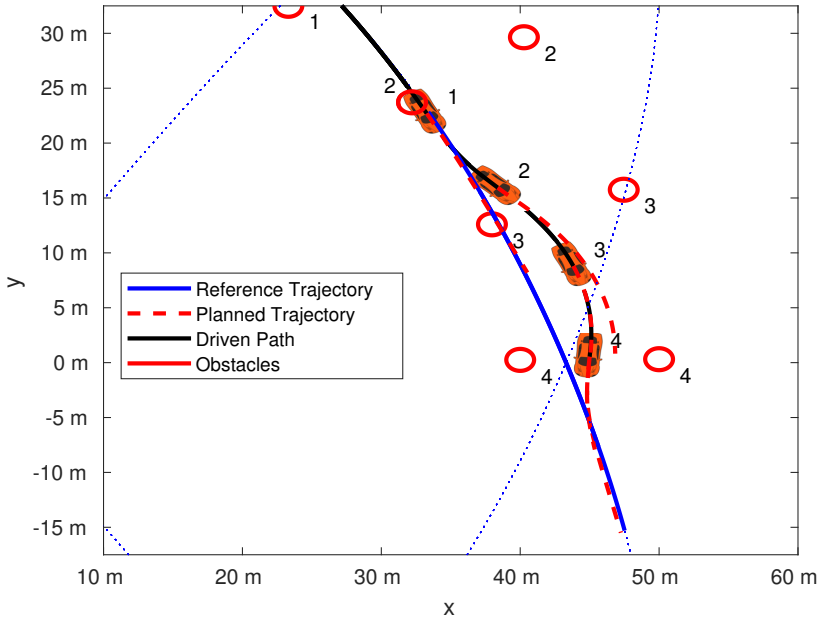

Figure 10. Dynamic obstacle avoidance. The position at different time steps is indicated by the number.

It can be seen that the car is pushed away and afterwards is able to return back to the reference trajectory. A video as proof of the simulated result has been uploaded at https://github.com/khoilsr/tcsac_trajectory_generation (accessed on 30 May 2022). This scenario can be seen as a proof of concept since it creates many different situations of obstacle avoidance, i.e., obstacles from the front, back, side, and successive obstacle avoidance situations, before the car could reach the reference trajectory again. Thus, this scenario is not close to situations that might occur in reality, but more difficult. This also shows the capability of TC-SAC in rapidly solving optimal control problems in different applications.

4. Stability Analysis

The class of systems to be controlled is described by the following general nonlinear set of ODEs

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \mathbf{x}(0) = \mathbf{x}_0 \tag{39}$$

with state vector $\mathbf{x}(t) \in \mathbb{R}^n$, input vector $\mathbf{u}(t) \in \mathbb{R}^m$. We also assume that

1. $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is twice continuously differentiable and $\mathbf{f}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$—thus, $\mathbf{0} \in \mathbb{R}^n$ is an equilibrium of the system;
2. system (39) has a unique solution for any initial condition $\mathbf{x}_0 \in \mathbb{R}^n$ and any piecewise continuous $\mathbf{u}(\cdot) \in \mathbf{R}^m$.

We then analyze and discuss the stability of TC-SAC, presented in Figure 1. For the sake of simplicity, we consider the stability of the system around the origin $\mathbf{x} = \mathbf{0}$. The stability of an arbitrary position is achieved similarly by shifting the state and control signals such that this arbitrary position becomes the origin. In addition, to keep the notation the same as in the literature, the objective cost function is denoted as V, which is equivalent to the notation J_1 presented in previous sections. Recall the OCP with target constraints

$$\min_{\mathbf{u}} \; V(\mathbf{x}, t; \mathbf{u}) = \int_0^T L(\mathbf{x}, \mathbf{u}, t) dt \tag{40a}$$

$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)), \; \mathbf{x}(0) = \mathbf{x}_0, \tag{40b}$$

$$\mathbf{x}(T) = \mathbf{0} \tag{40c}$$

where L has a quadratic form:

$$L(\mathbf{x}, \mathbf{u}, t) = \frac{1}{2}\left[\mathbf{x}^\mathsf{T}(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^\mathsf{T}(t)\mathbf{R}\mathbf{u}(t)\right] \tag{41}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are a positive definite matrix.

Here, let $\mathbf{x}(\cdot; \mathbf{x}_0, 0)$ denote the corresponding trajectory of (39) with initial condition $\mathbf{x}(0) = \mathbf{x}_0$ and $\hat{\mathbf{u}}(\cdot; \mathbf{x}_0, 0)$ denote the optimal control sequence that minimizes the objective function $V(\mathbf{x}_0, 0; \mathbf{u})$. From a methodological point of view, TC-SAC is a combination of FOGA in the first step and the extended SAC in the subsequent one. Hence, an intuitive method to analyze the stability of TC-SAC is to establish the stability conditions for FOGA first, and then the stability of TC-SAC can be concluded after this. Furthermore, TC-SAC uses the same concept of receding horizon as MPC; therefore, it is straightforward to derive the stability conditions of TC-SAC from the stability literature of MPC. In the following, we first look at the stability conditions of FOGA in the absence of the extended SAC in Section 4.1. Then, we discuss the stability of TC-SAC in Section 4.2.

4.1. Stability of FOGA

There have been several works that investigate and deploy sufficient conditions for the closed-loop MPC system to be stable. An overview of most of the works in this area can be found in [51]. Specifically, for our problem in (40), we are seeking the stability conditions of a receding horizon control with terminal constraint for nonlinear continuous systems. The constraint imposed at terminal time T provides a relatively simple procedure to establish

the stability of the closed-loop system. In fact, Chen and Shaw [52] derived sufficient conditions for the closed-loop receding horizon control of (40) to be asymptotically stable. These conditions are described by the following assumptions.

Assumption 1. *There exists an optimal control function* $\hat{\mathbf{u}}(\cdot; \mathbf{x}_0, 0)$*, which gives the minimal cost* $\hat{V}(\mathbf{x}_0, 0; \mathbf{u})$ *and satisfies the terminal constraint* (40c).

Assumption 2. *The optimal cost* $\hat{V}(\mathbf{x}_0, 0; \mathbf{u})$ *satisfies the following conditions for any* $T > 0$

(a) $\hat{V}(\mathbf{x}_0, 0; \mathbf{u}) = 0$ *and* $\hat{V}(\mathbf{x}, t; \mathbf{u}) > 0$ *for* $\mathbf{x} \neq \mathbf{0}$*;*

(b) $\hat{V}(\mathbf{x}, t; \mathbf{u}) \to \infty$ *when* $\|\mathbf{x}\| \to \infty$*;*

(c) $\frac{\partial \hat{V}(\mathbf{x}, t; \mathbf{u})}{\partial \mathbf{x}}$ *exists for any* $\mathbf{x}$*.*

Theorem 1. *Suppose that Assumptions 1 and 2 are satisfied; then, for any fixed* $T > 0$*, the closed-loop system is asymptotically stable at the origin.*

Proof. See Appendix B. □

Assumption 2 can be fulfilled easily by having the cost function in quadratic form, i.e.,

$$V(\mathbf{x}, t; \mathbf{u}) = \frac{1}{2} \int_t^{t+T} \left(\|\mathbf{x}(\tau)\|_{\mathbf{Q}}^2 + \|\mathbf{u}(\tau)\|_{\mathbf{R}}^2 \right) d\tau \tag{42}$$

Assumption 1 is, however, more difficult to be satisfied. It requires the optimal solution of (40), in which the terminal constraint (40c) also has to be satisfied for each receding horizon. We will first prove that there exists an optimal solution that satisfies the terminal constraint when using FOGA as the solver.

Recall the changes in the cost function J_1 and the terminal constraints $\mathbf{\Phi}$

$$\delta J_1 = \int_{t_0}^{t_\mathrm{f}} \left(\boldsymbol{\rho}^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} + \frac{\partial l_1}{\partial \mathbf{u}} \right) \delta \mathbf{u}(t) dt \tag{43}$$

$$\delta \mathbf{\Phi} \triangleq \begin{bmatrix} \delta x_1 \\ \vdots \\ \delta x_q \end{bmatrix}_{t=t_\mathrm{f}} = \int_{t_0}^{t_\mathrm{f}} \mathbf{R}^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \delta \mathbf{u}(t) dt \tag{44}$$

and adjoining (44) to (43), we obtain

$$\delta J_1 + \boldsymbol{\nu} \delta \mathbf{\Phi} = \int_{t_0}^{t_\mathrm{f}} \left[\frac{\partial l_1}{\partial \mathbf{u}} + (\boldsymbol{\rho} + \mathbf{R}\boldsymbol{\nu})^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right] \delta \mathbf{u}(t) dt \tag{45}$$

where $\boldsymbol{\nu}$ is also a Lagrange multiplier. The equation (45) represents the adjoined cost of the OCP. With the solution of $\delta \mathbf{u}(t)$ in (15), we have

$$\delta J_1 + \boldsymbol{\nu} \delta \mathbf{\Phi} = -\mathbf{W}^{-1} \int_{t_0}^{t_\mathrm{f}} \left\| \frac{\partial l_1}{\partial \mathbf{u}} + (\boldsymbol{\rho} + \mathbf{R}\boldsymbol{\nu})^\mathsf{T} \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right\|^2 dt < 0 \tag{46}$$

which is negative unless the integrand vanishes over the whole integration interval. The Lagrange multiplier $\boldsymbol{\nu}$ is computed as

$$\boldsymbol{\nu} = -\mathbf{I}_{\mathbf{\Phi}\mathbf{\Phi}}^{-1} (\delta \mathbf{\Phi} + \mathbf{I}_{\mathbf{\Phi J}}) \tag{47}$$

with $\delta \mathbf{\Phi} = -\epsilon \mathbf{\Phi}[\mathbf{x}(t_\mathrm{f})]$ and a constant $\epsilon \in (0, 1]$. The choice of $\delta \mathbf{\Phi}$ guarantees that the constructed control signal $\mathbf{u}(t)$ drives the system close to the desired state at the final time t_f. Thus, the terminal constraints can be satisfied after a finite amount of iterations. Since the system is fully controllable, $\mathbf{I}_{\mathbf{\Phi}\mathbf{\Phi}}^{-1}$ exists; thus, there always exists $\boldsymbol{\nu}$ and $\delta \mathbf{u}$ such that

the cost function is decreased and the terminal constraints are satisfied. As the optimal solution is approached and $\delta\mathbf{\Phi} = 0$, it is clear that

$$\boldsymbol{\nu} \to -\mathbf{I}_{\mathbf{\Phi}\mathbf{\Phi}}^{-1}\mathbf{I}_{\mathbf{\Phi J}} \tag{48}$$

$$\frac{\partial l_1}{\partial \mathbf{u}} + (\boldsymbol{\rho} + \mathbf{R}\boldsymbol{\nu})^\top \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \to 0 \tag{49}$$

FOGA is therefore able to construct an optimal control signal $\mathbf{u}(t)$ that minimizes the cost and satisfies terminal constraints. Hence, Assumption 1 is satisfied.

A problem that arises here is that Assumption 2 uses the optimal cost $\hat{V}(\mathbf{x}; T)$ as the Lyapunov function to establish the closed-loop stability. This requires FOGA to be performed repeatedly until the optimal solution is found, which is not practical in our case since we want to achieve fast computation for online capability. However, this optimal solution is actually not necessary. Indeed, we will show that, with a proper "warm start", the closed-loop system is still stable even if the cost is not optimal.

Definition 1 (Warm start)**.** *An admissible warm start,* $\tilde{\mathbf{u}}$*, must steer the current state* $\mathbf{x}$ *to the origin, i.e., satisfy terminal constraint* $\mathbf{x}(T) = \mathbf{0}$*.*

A warm start needs to satisfy the constraints but does not have to be optimal; hence, it can be acquired much faster. Furthermore, a warm start only needs to be computed once and can be done offline. In our approach, a warm start is achieved by performing FOGA for a couple of iterations. The process can be sped up with a proper choice of ϵ to calculate the Lagrange multiplier $\boldsymbol{\nu}$ in (47). A controller algorithm using a warm start is as follows.

Controller algorithm with warm start

Data: $\mathbf{x}_0 \in \mathbb{R}^n$, $\delta \in (0, \infty)$, where δ is the sampling interval

Initialization: At time $t_0 = 0$, if $\mathbf{x}_0$ is at the origin, i.e., $\mathbf{x}_0 = \mathbf{0}$, meaning that the system is already at the equilibrium, then employ $\mathbf{u} = \mathbf{0}$ to maintain the current state. Else, perform FOGA for a couple of iterations to compute a feasible warm start $\mathbf{u}_0$ for the OCP problem in (40). Apply the control $\mathbf{u}_0$ to the real system over the interval $[t_0, t_0 + \delta]$.

Repeat:

1. At any time t, if $\mathbf{x}(t) = \mathbf{0}$, meaning that the system reaches the origin, employ $\mathbf{u} = \mathbf{0}$. Else:
2. At any time $t_i \triangleq i\delta, i \in \mathbb{N}$:
   - Obtain an admissible control $\mathbf{u}_i'$ as an initial guess

$$\mathbf{u}_i' = \begin{cases} \mathbf{u}_{i-1} & \text{for} \quad t \in [t_i, t_i + T - \delta] \\ \mathbf{0} & \text{for} \quad t \in [t_i + T - \delta, t_i + T] \end{cases} \tag{50}$$

   - Compute an admissible control horizon that is better than the preceding control horizon in the sense that

$$V(\mathbf{x}_i, t_i, \mathbf{u}_i) \leq V(\mathbf{x}_i, t_i, \mathbf{u}_i') \tag{51}$$

   - Apply the control $\mathbf{u}_i$ to the real system over the interval $[t_i, t_i + \delta]$

The stability proof of MPC with the warm start is presented in [53]. With a choice of admissible warm starts, the controller is asymptotically stable. The terminal constraints are satisfied from the beginning and the cost is improved over time. Even in the situation where the cost increases due to numerical errors, we simply return to the warm start of the previous iteration and continue from there.

4.2. Stability of TC-SAC

As mentioned, TC-SAC can be interpreted as a combination of FOGA in the first step and the extended SAC in the subsequent one. Since FOGA was proven to be stable in

Section 4.1, we only need to confirm that the extended SAC in the second step does not violate the stability property of FOGA. In other words, we need to fulfill two conditions: (1) the performance cost (40a) is decreased when applying the extended SAC, and (2) the terminal constraint (40c) is not violated.

It can be easily seen that both of the conditions are guaranteed naturally by the procedure of TC-SAC presented in Section 2.2. From a methodological point of view, TC-SAC only applies the control signal computed by the extended SAC if it results in a smaller value in both performance cost $V(\mathbf{x}, t; \mathbf{u})$ and terminal cost $l_c = \frac{1}{2}\mathbf{\Phi}^\mathsf{T}\mathbf{Q}_c\mathbf{\Phi}$, when compared to FOGA. The reduction in $V(\mathbf{x}, t; \mathbf{u})$ guarantees that condition 1 is satisfied. Similarly, the reduction in terminal cost l_c means that the extended SAC drives the system closer to the origin at time T, thus fulfilling condition 2. If any of the conditions is not met, the control signal computed by FOGA is used instead. In both cases, we ensure that TC-SAC inherits the stability property of FOGA and, therefore, we can conclude that TC-SAC is asymptotically stable.

5. Discussion and Conclusions

In this paper, we presented a fast and close-to-optimal control method called TC-SAC. We also evaluated the proposed method in different scenarios and situations to test its capability. In detail, TC-SAC is able to fulfill most of the reaching/tracking tasks, even in critical scenarios such as the Lissajous trajectory. Furthermore, it was shown that TC-SAC is able to avoid static and dynamic obstacles efficiently with only soft constraints, i.e., with obstacle avoidance costs. In terms of computational effort, TC-SAC is able to find a sub-optimal solution that is close to the optimal one, without significantly affecting the computation time. Therefore, our proposed method has a lot of potential for applications that require online capability and performance costs to be optimal. Note that TC-SAC is a model-based control method; hence, it can be applied to different linear/nonlinear systems, and is not only limited to the applications presented in this paper. Moreover, the stability proof presented in this work shows TC-SAC to be a reliable controller when it comes to safety aspects. The only downside is that TC-SAC is not able to consider inequality constraints, which might limit its use to systems that have strict requirements in this type of constraint.

Although the results in this work were only achieved in simulation, we believe that this method can be easily transferred to real systems due to its high potential. For real-time applications, the algorithm needs to be written in a system programming level such as C programming, so that it can be computed as quickly as possible. The prediction horizon T_p also needs to be properly selected to achieve good results in terms of optimality, without losing too much of the computational effort. For complex systems such as high-DOF or humanoid robots, a simplified model might be needed to reduce the computation time. In addition, since our approach is model-based, a precise model of the system will play an important role in the quality of the controller. Thus, for systems with unknown parameters, a model identification procedure needs to be performed first.

As for the future work of this paper, TC-SAC will be first implemented on a remote control car system and on a 2DOF robot for testing the trajectory tracking problems and obstacle avoidance behaviors. We also plan to investigate further the influence of the inaccuracy between the real and simulated models on the controller and the robustness of TC-SAC in the existence of noise and disturbances.

Author Contributions: Conceptualization, K.H.-D., M.L. and D.W.; methodology, K.H.-D. and M.L.; software, K.H.-D.; validation, K.H.-D., M.L. and D.W.; formal analysis, K.H.-D.; investigation, K.H.-D. and M.L.; resources, K.H.-D.; data curation, K.H.-D. and D.W.; writing—original draft preparation, K.H.-D. and M.L.; writing—review and editing, M.L. and D.W.; visualization, K.H.-D.; supervision, M.L. and D.W.; project administration, D.W.; funding acquisition, K.H.-D., M.L. and D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The codes, videos and data related to this work are available at https://github.com/khoilsr/tcsac_trajectory_generation (accessed on 30 May 2022).

Acknowledgments: This research was partially conducted at the Chair of Automatic Control Engineering, Technical University of Munich.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TC-SAC	Target-Constrained Sequential Action Control
SAC	Sequential Action Control
NMPC	Nonlinear Model Predictive Control
MPC	Model Predictive Control
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
FOGA	First-Order Gradient Algorithm
DOF	Degree of Freedom

Appendix A. Vehicle Dynamic Model

For car-like systems, a wide range of model complexities exist. One of the most complex representations of a car-like system is the multi-body model, which can very accurately describe the overall physical behavior of the car [54]. However, this level of complexity does not suit the requirements for online application due to the excessive computational load. The model presented in this work is a simplified single-track model [55], which is commonly used in studies of different controllers/methods related to car-like systems due to its simplicity and accurate representation of the dynamic behaviors of a car. The state-space model is as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\Psi}_c \\ \dot{v} \\ \dot{\alpha} \\ \ddot{\Psi}_c \end{bmatrix} = \begin{bmatrix} v\cos(\alpha + \Psi_c) \\ v\sin(\alpha + \Psi_c) \\ \dot{\Psi}_c \\ a_{\text{ave}} \\ s_{\text{ave}} \\ \omega_{\text{ave}} \end{bmatrix} \tag{A1}$$

with the x and y-coordinates X and Y, the orientation Ψ_c, the velocity v, the side slip angle α and the change in orientation $\dot{\Psi}_c$. The control input is given by $\mathbf{u} = \begin{bmatrix} \delta \\ M \end{bmatrix}$, with the steering angle δ and the applied torque M. a_{ave}, s_{ave} and ω_{ave} are given by:

$$\begin{aligned} a_{\text{ave}} = \frac{1}{m}(&F_{f,X}\cos(\alpha - \delta) + F_{r,X}\cos(\alpha) + F_{f,Y}\sin(\alpha - \delta) \\ &+ F_{r,Y}\sin(\alpha) - 0.5 c_w \rho_L A v^2) \end{aligned} \tag{A2}$$

$$\begin{aligned} s_{\text{ave}} = \frac{1}{mv}(&-F_{f,X}\sin(\alpha - \delta) - F_{r,X}\sin(\alpha) \\ &+ F_{f,Y}\cos(\alpha - \delta) + F_{r,Y}\cos(\alpha) - mv\dot{\Psi}_c) \end{aligned} \tag{A3}$$

$$\omega_{\text{ave}} = \frac{F_{f,X} l_f \sin(\delta) + F_{f,Y} l_f \cos(\delta) - F_{r,Y} l_r}{\Theta_Z} \tag{A4}$$

with the drag coefficient c_w, the density of the air ρ_L and the front surface A of the car and the inertia Θ_Z of the car. $F_{f/r,Y}$ in (A2)–(A4) are the lateral tire forces obtained by using Pacejka's magic tire formula

$$F_{f/r,Y} = D_Y \sin\Big(C_Y \operatorname{atan}\Big(B_Y \alpha_{f/r} - E_Y\Big(B_Y \alpha_{f/r} - \operatorname{atan}\Big(B_Y \alpha_{f/r}\Big)\Big)\Big)\Big) \tag{A5}$$

with $B_Y = \frac{K_Y F_{f/r,Z}}{C_Y D_Y}$, $D_Y = \mu_Y F_{f/r,Z}$ and C_Y, E_Y, K_Y are the constants. μ_Y is the friction coefficient in the lateral direction, $F_{f/r,Z}$ is the tire load on the front and rear axis and $\alpha_{f/r}$ is the side slip angle on the front and rear axis. $F_{f/r,Z}$ is given by

$$F_{f/r,Z} = m_c g \frac{l_{r/f}}{l_f + l_r} \tag{A6}$$

For the longitudinal forces $F_{f,X}$ in (A2)–(A4), they are simplified as

$$F_{f,X} = F_{r,X} = \frac{M}{r} \tag{A7}$$

with the torque M as the control input for the system and r as the radius of the tires.

Model Modification: The single-track model presented above is not linear with respect to δ, the steering angle, where it is used in trigonometrical functions as in (A4). Since TC-SAC requires the system to be in control-affine form, as described in (4), a simple way to correct this is to change the steering angle from a control input to a state and introduce the derivative of it, $\dot{\delta}_{\text{input}}$, as the new control input.

After these changes are applied, the state-space model changes to:

$$\mathbf{x} = \begin{bmatrix} X \\ Y \\ \Psi_c \\ v \\ \alpha \\ \dot{\Psi}_c \\ \delta \end{bmatrix}; \dot{\mathbf{x}} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\Psi}_c \\ \dot{v} \\ \dot{\alpha} \\ \ddot{\Psi}_c \\ \dot{\delta} \end{bmatrix} \tag{A8}$$

with the control input

$$\mathbf{u} = \begin{bmatrix} \dot{\delta}_{\text{input}} \\ M \end{bmatrix} \tag{A9}$$

with

$$\dot{\delta} = \dot{\delta}_{\text{input}} \tag{A10}$$

and the steering angle δ still can be controlled directly. All other equations of the state-space model remain the same.

With these changes, $\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u})$ can be written in a control-affine form with:

$$\mathbf{h}(t, \mathbf{x}(t)) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{\cos(\alpha)+\cos(\alpha-\delta)}{mr} \\ 0 & \frac{-\sin(\alpha)+\sin(\alpha-\delta)}{mrv} \\ 0 & \frac{l_f \sin(\delta)}{r\Theta} \\ 1 & 0 \end{bmatrix} \tag{A11}$$

and

$$\mathbf{g}(t, \mathbf{x}(t)) = \begin{bmatrix} v\cos(\Psi_c + \alpha) \\ \vdots \\ 0 \end{bmatrix} \tag{A12}$$

Although the input $\dot{\delta}$ serves mainly for the reason of control-affine form, it also improves the smoothness of the steering angle, i.e., a jump in the derivative results in a linear change in the signal. This makes the controller more realistic when applied on a real system, since the steering angle will be applied, not the derivative of it. However, the drawback of the modification is that, since the steering angle is not a control input, it cannot be saturated afterwards. In detail, it cannot be limited to realistic values, i.e., 35–45°. To overcome this problem, u_δ is changed as follows:

$$\dot{\delta} = u_\delta - k\delta \tag{A13}$$

with the constant $k \in \mathbb{R}^+$. Note that now u_δ has no direct physical meaning. Through this change, $\mathbf{h}(t, \mathbf{x}(t))$ is not affected, while $\mathbf{g}(t, \mathbf{x}(t))$ changes to

$$\mathbf{g}(t, \mathbf{x}(t)) = \begin{bmatrix} v\cos(\Psi_c + \alpha) \\ \vdots \\ -k \end{bmatrix} \tag{A14}$$

and the control input $\mathbf{u}$ to

$$\mathbf{u} = \begin{bmatrix} u_\delta \\ M \end{bmatrix} \tag{A15}$$

For the maximum and minimum steering angle, $\delta_{\max} = -\delta_{\min}$ holds, as, for the maximum and minimum control input, $u_{\delta,\max} = -u_{\delta,\min}$ holds. k, therefore, has to be chosen depending on the desired maximum steering angle $\delta_{\max}$ and the maximum control input $u_{\delta,\max}$ for it. Thus, k is determined with:

$$k = \frac{u_{\delta,\max}}{\delta_{\max}} \tag{A16}$$

Appendix B. Proof of Theorem 1

Here, we summarize the proof given in [56] for the case of nonlinear systems with terminal constraints. The idea is to show that the optimal cost function $\hat{V}(\mathbf{x}, t; \mathbf{u})$ can be used as a Lyapunov function for the receding horizon control. Let $\mathbf{x}^*$ and $\mathbf{u}^*$ denote the receding horizon strategy when the initial state is $\mathbf{x}_0$ at $t = 0$. We wish to evaluate $(d/dt)\hat{V}(\mathbf{x}^*(t))$ at an arbitrary, fixed instant of time. By definition,

$$\begin{aligned} \hat{V}(\mathbf{x}^*(t)) = \frac{1}{2}\int_t^{t+\Delta t} \left[\hat{\mathbf{x}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{x}}(\tau) + \hat{\mathbf{u}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{u}}(\tau)\right] d\tau \\ + \frac{1}{2}\int_{t+\Delta t}^{t+T} \left[\hat{\mathbf{x}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{x}}(\tau) + \hat{\mathbf{u}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{u}}(\tau)\right] d\tau \end{aligned} \tag{A17}$$

where $\hat{\mathbf{x}}(\tau) \triangleq \hat{\mathbf{x}}(\tau; \mathbf{x}^*(t), t)$ and $\hat{\mathbf{u}}(\tau) \triangleq \hat{\mathbf{u}}(\tau; \mathbf{x}^*(t), t)$ are the optimal solution for the OCP in (40). Consider a control $\tilde{\mathbf{u}} : [t + \Delta t, t + T + \Delta t]$ defined as follows:

$$\tilde{\mathbf{u}}(\tau) \triangleq \begin{cases} \hat{\mathbf{u}}(\tau; \mathbf{x}^*(t), t) & \text{for} \quad \tau \in [t + \Delta t, t + T] \\ 0 & \text{for} \quad \tau \in (t + T, t + T + \Delta t]. \end{cases} \tag{A18}$$

Let $\tilde{\mathbf{x}}(\cdot) = \tilde{\mathbf{x}}(\cdot; \hat{\mathbf{x}}(t+\Delta t), t+\Delta t)$ denote the corresponding trajectory with initial condition $\tilde{\mathbf{x}}(t+\Delta t) = \hat{\mathbf{x}}(t+\Delta t; \mathbf{x}^*(t), t)$. Clearly,

$$\tilde{\mathbf{x}}(\tau) = \begin{cases} \tilde{\mathbf{x}}(\tau; \mathbf{x}^*(t), t) & \text{for} \quad \tau \in [t+\Delta t, t+T] \\ 0 & \text{for} \quad \tau \in (t+T, t+T+\Delta t) \end{cases} \tag{A19}$$

because $\tilde{\mathbf{x}}(t+T; \hat{\mathbf{x}}(t+\Delta t), t+\Delta t) = \hat{\mathbf{x}}(t+T; \mathbf{x}^*(t), t) = 0$ and $\tilde{\mathbf{u}} = 0$ for $\tau > t+T$. Since $\tilde{\mathbf{u}}$ is not necessarily optimal, it follows that

$$\begin{aligned} \hat{V}(\mathbf{x}^*(t)) &= \frac{1}{2}\int_t^{t+\Delta t} \left[\hat{\mathbf{x}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{x}}(\tau) + \hat{\mathbf{u}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{u}}(\tau)\right] d\tau + V(\hat{\mathbf{x}}(t+\Delta t), t+\Delta t; \tilde{\mathbf{u}}) \\ &\geq \frac{1}{2}\int_t^{t+\Delta t} \left[\hat{\mathbf{x}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{x}}(\tau) + \hat{\mathbf{u}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{u}}(\tau)\right] d\tau + \hat{V}(\hat{\mathbf{x}}(t+\Delta t)) \end{aligned} \tag{A20}$$

so that

$$\hat{V}(\hat{\mathbf{x}}(t+\Delta t)) - \hat{V}(\mathbf{x}^*(t)) \leq -\frac{1}{2}\int_t^{t+\Delta t} \left[\hat{\mathbf{x}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{x}}(\tau) + \hat{\mathbf{u}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{u}}(\tau)\right] d\tau \tag{A21}$$

Since $\hat{V}$ is continuously differentiable, it follows from the Mean Value Theorem that

$$\begin{aligned} \frac{\hat{V}(\hat{\mathbf{x}}(t+\Delta t)) - \hat{V}(\mathbf{x}^*(t))}{\Delta t} &= \nabla_{\mathbf{x}}\hat{V}(\mathbf{x}^*(t) \\ &\quad + \theta(\Delta t)(\hat{\mathbf{x}}(t+\Delta t) - \mathbf{x}^*(t)))\frac{(\hat{\mathbf{x}}(t+\Delta t) - \mathbf{x}^*(t))}{\Delta t} \end{aligned} \tag{A22}$$

for $\theta(\Delta t) \in (0,1)$. Since

$$\mathbf{x}^*(t) = \hat{\mathbf{x}}(t; \mathbf{x}^*(t), t) = \hat{\mathbf{x}}(t), \mathbf{u}^*(t) = \hat{\mathbf{u}}(t; \mathbf{u}^*(t), t) = \hat{\mathbf{u}}(t) \tag{A23}$$

and $\hat{\mathbf{u}}$ is continuous at t, we have that

$$\lim_{\Delta t \to 0_+} \frac{\hat{\mathbf{x}}(t+\Delta t) - \mathbf{x}^*(t)}{\Delta t} = f(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t)) = f(\mathbf{x}^*(t), \mathbf{u}^*(t)). \tag{A24}$$

Since $\nabla_{\mathbf{x}}\hat{V}$ and $\mathbf{x}$ are continuous, then from (A20), (A23) and (A24), it follows that

$$\lim_{\Delta t \to 0_+} \frac{\hat{V}(\hat{\mathbf{x}}(t+\Delta t)) - \hat{V}(\mathbf{x}^*(t))}{\Delta t} = \nabla_{\mathbf{x}}\hat{V}(\mathbf{x}^*(t))f(\mathbf{x}^*(t), \mathbf{u}^*(t)). \tag{A25}$$

By continuity of $\hat{\mathbf{x}}$ and $\hat{\mathbf{u}}$ and the Mean Value Theorem for integrals, we also have

$$\begin{aligned} &\lim_{\Delta t \to 0_+} -\frac{1}{2\Delta t}\int_t^{t+\Delta t} \left[\hat{\mathbf{x}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{x}}(\tau) + \hat{\mathbf{u}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{u}}(\tau)\right] d\tau \\ &\qquad \leq \lim_{\Delta t \to 0_+} -\frac{1}{2\Delta t}\int_t^{t+\Delta t} \left[\hat{\mathbf{x}}^\mathsf{T}(\tau)\mathbf{Q}\hat{\mathbf{x}}(\tau)\right] d\tau = -\frac{1}{2}\left[\mathbf{x}^*(t)^\mathsf{T}\mathbf{Q}\mathbf{x}^*(t)\right]. \end{aligned} \tag{A26}$$

Dividing both sides of (A21) by $\Delta t > 0$ and taking the limit as $\Delta t \to 0_+$ yields

$$\nabla_{\mathbf{x}}\hat{V}(\mathbf{x}^*(t))f(\mathbf{x}^*(t), \mathbf{u}^*(t)) \leq -\frac{1}{2}\left[\mathbf{x}^*(t)^\mathsf{T}\mathbf{Q}\mathbf{x}^*(t)\right] < 0. \tag{A27}$$

Hence,

$$(d/dt)\hat{V}(\mathbf{x}^*(t)) = \nabla_{\mathbf{x}}\hat{V}(\mathbf{x}^*(t))f(\mathbf{x}^*(t), \mathbf{u}^*(t)) < 0 \tag{A28}$$

unless $\mathbf{x}^*(t) = 0$. Hence, the system is asymptotically stable.

References

1. González, D.; Pérez, J.; Milanés, V.; Nashashibi, F. A Review of Motion Planning Techniques for Automated Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1135–1145. [CrossRef]
2. Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55. [CrossRef]
3. Katrakazas, C.; Quddus, M.; Chen, W.H.; Deka, L. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transp. Res. Part Emerg. Technol.* **2015**, *60*, 416–442. [CrossRef]
4. Behringer, R.; Muller, N. Autonomous road vehicle guidance from autobahnen to narrow curves. *IEEE Trans. Robot. Autom.* **1998**, *14*, 810–815. [CrossRef]
5. Piazzi, A.; Bianco, C.G.L.; Bertozzi, M.; Fascioli, A.; Broggi, A. Quintic G2-splines for the iterative steering of vision-based autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2002**, *3*, 27–36. [CrossRef]
6. Kelly, A.; Stentz, A.; Amidi, O.; Bode, M.; Bradley, D.; Diaz-Calderon, A.; Happold, M.; Herman, H.; Mandelbaum, R.; Pilarski, T.; et al. Toward reliable off road autonomous vehicles operating in challenging environments. *Int. J. Robot. Res.* **2006**, *25*, 449–483. [CrossRef]
7. Krotkov, E.; Fish, S.; Jackel, L.; McBride, B.; Perschbacher, M.; Pippine, J. The DARPA PerceptOR evaluation experiments. *Auton. Robot.* **2007**, *22*, 19–35. [CrossRef]
8. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 2, pp. 995–1001. [CrossRef]
9. Kuwata, Y.; Teo, J.; Fiore, G.; Karaman, S.; Frazzoli, E.; How, J.P. Real-Time Motion Planning With Applications to Autonomous Urban Driving. *IEEE Trans. Control. Syst. Technol.* **2009**, *17*, 1105–1118. [CrossRef]
10. Jo, K.; Lee, M.; Kim, D.; Kim, J.; Jang, C.; Kim, E.; Kim, S.; Lee, D.; Kim, C.; Kim, S.; et al. Overall reviews of autonomous vehicle a1-system architecture and algorithms. *IFAC Proc. Vol.* **2013**, *46*, 114–119. [CrossRef]
11. Kuffner, J.J.; Kagami, S.; Nishiwaki, K.; Inaba, M.; Inoue, H. Dynamically-Stable Motion Planning for Humanoid Robots. *Auton. Robot.* **2002**, *12*, 105–118. [CrossRef]
12. Stilman, M. Global Manipulation Planning in Robot Joint Space With Task Constraints. *IEEE Trans. Robot.* **2010**, *26*, 576–584. [CrossRef]
13. Tsymbal, O.; Mercorelli, P.; Sergiyenko, O. Predicate-Based Model of Problem-Solving for Robotic Actions Planning. *Mathematics* **2021**, *9*, 3044. [CrossRef]
14. Cefalo, M.; Oriolo, G. A general framework for task-constrained motion planning with moving obstacles. *Robotica* **2019**, *37*, 575–598. [CrossRef]
15. Xiong, C.; Zhou, H.; Lu, D.; Zeng, Z.; Lian, L.; Yu, C. Rapidly-Exploring Adaptive Sampling Tree*: A Sample-Based Path-Planning Algorithm for Unmanned Marine Vehicles Information Gathering in Variable Ocean Environments. *Sensors* **2020**, *20*. [CrossRef]
16. Sangiovanni, B.; Incremona, G.P.; Piastra, M.; Ferrara, A. Self-Configuring Robot Path Planning With Obstacle Avoidance via Deep Reinforcement Learning. *IEEE Control Syst. Lett.* **2021**, *5*, 397–402. [CrossRef]
17. Ziegler, J.; Bender, P.; Schreiber, M.; Lategahn, H.; Strauss, T.; Stiller, C.; Dang, T.; Franke, U.; Appenrodt, N.; Keller, C.G.; et al. Making Bertha Drive—An Autonomous Journey on a Historic Route. *IEEE Intell. Transp. Syst. Mag.* **2014**, *6*, 8–20. [CrossRef]
18. Von Stryk, O.; Bulirsch, R. Direct and indirect methods for trajectory optimization. *Ann. Oper. Res.* **1992**, *37*, 357–373. [CrossRef]
19. Diehl, M.; Bock, H.G.; Diedam, H.; Wieber, P.B. Fast direct multiple shooting algorithms for optimal robot control. In *Fast Motions in Biomechanics and Robotics*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 65–93.
20. Rao, A. A Survey of Numerical Methods for Optimal Control. *Adv. Astronaut. Sci.* **2010**, *135*, 497–528.
21. Lampariello, R.; Nguyen-Tuong, D.; Castellini, C.; Hirzinger, G.; Peters, J. Trajectory planning for optimal robot catching in real-time. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3719–3726. [CrossRef]
22. Werner, A.; Trautmann, D.; Lee, D.; Lampariello, R. Generalization of optimal motion trajectories for bipedal walking. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 1571–1577. [CrossRef]
23. Apostolopoulos, S.; Leibold, M.; Buss, M. Online motion planning over uneven terrain with walking primitives and regression. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 3799–3805. [CrossRef]
24. Weitschat, R.; Haddadin, S.; Huber, F.; Albu-Schäffer, A. Dynamic optimality in real-time: A learning framework for near-optimal robot motions. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 5636–5643. [CrossRef]
25. Rawlings, J.B.; Mayne, D.Q.; Diehl, M.M. *Model Predictive Control: Theory, Computation and Design*; Nob Hill Publishing, LLC: Goleta, CA, USA, 2017.
26. Allgöwer, F.; Zheng, A. *Nonlinear Model Predictive Control*; Birkhäuser: Basel, Switzerland, 2012; Volume 26.
27. Qin, S.J.; Badgwell, T.A. An Overview of Nonlinear Model Predictive Control Applications. In *Nonlinear Model Predictive Control*; Allgöwer, F., Zheng, A., Eds.; Birkhäuser: Basel, Switzerland, 2000; pp. 369–392.

28. Verschueren, R.; Zanon, M.; Quirynen, R.; Diehl, M. Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm. In Proceedings of the Control Conference (ECC), 2016 European, Aalborg, Denmark, 29 June–1 July 2016; pp. 141–147.
29. Obayashi, M.; Uto, K.; Takano, G. Appropriate overtaking motion generating method using predictive control with suitable car dynamics. In Proceedings of the Decision and Control (CDC), 2016 IEEE 55th Conference, Las Vegas, NV, USA, 12–14 December 2016, pp. 4992–4997.
30. Madås, D.; Nosratinia, M.; Keshavarz, M.; Sundström, P.; Philippsen, R.; Eidehall, A.; Dahlén, K.M. On path planning methods for automotive collision avoidance. In Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, QLD, Australia, 23–26 June 2013; pp. 931–937. [CrossRef]
31. Li, X.; Sun, Z.; Zhu, Q.; Liu, D. A unified approach to local trajectory planning and control for autonomous driving along a reference path. In Proceedings of the 2014 IEEE International Conference on Mechatronics and Automation, Tianjin, China, 3–6 August 2014; pp. 1716–1721. [CrossRef]
32. Kim, D.; Carlo, J.D.; Katz, B.; Bledt, G.; Kim, S. Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control. *arXiv* **2019**, arXiv:1909.06586.
33. Ohtsuka, T. A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica* **2004**, *40*, 563–574. [CrossRef]
34. Feller, C.; Ebenbauer, C. A stabilizing iteration scheme for model predictive control based on relaxed barrier functions. *Automatica* **2016**, *80*, 328–339. [CrossRef]
35. Soloperto, R.; Köhler, J.; Allgöwer, F.; Müller, M.A. Collision avoidance for uncertain nonlinear systems with moving obstacles using robust Model Predictive Control. In Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019; pp. 811–817. [CrossRef]
36. Zhu, H.; Alonso-Mora, J. Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments. *IEEE Robot. Autom. Lett.* **2019**, *4*, 776–783. [CrossRef]
37. Li, S.; Li, Z.; Yu, Z.; Zhang, B.; Zhang, N. Dynamic Trajectory Planning and Tracking for Autonomous Vehicle With Obstacle Avoidance Based on Model Predictive Control. *IEEE Access* **2019**, *7*, 132074–132086. [CrossRef]
38. Li, W.; Xiong, R. Dynamical Obstacle Avoidance of Task- Constrained Mobile Manipulation Using Model Predictive Control. *IEEE Access* **2019**, *7*, 88301–88311. [CrossRef]
39. Houska, B.; Ferreau, H.; Diehl, M. ACADO Toolkit—An Open Source Framework for Automatic Control and Dynamic Optimization. *Optim. Control. Appl. Methods* **2011**, *32*, 298–312. [CrossRef]
40. Kamel, M.; Alexis, K.; Achtelik, M.; Siegwart, R. Fast nonlinear model predictive control for multicopter attitude tracking on SO(3). In Proceedings of the 2015 IEEE Conference on Control Applications (CCA), Sydney, NSW, Australia, 21–23 September 2015; pp. 1160–1166. [CrossRef]
41. Zanelli, A.; Horn, G.; Frison, G.; Diehl, M. Nonlinear Model Predictive Control of a Human-sized Quadrotor. In Proceedings of the 2018 European Control Conference (ECC), Limassol, Cyprus, 12–15 June 2018; pp. 1542–1547. [CrossRef]
42. Ansari, A.; Murphey, T.D. Sequential Action Control: Closed-Form Optimal Control for Nonlinear Systems. *IEEE Trans. Robot.* **2016**, *32*, 1196 – 1214. [CrossRef]
43. Egerstedt, M.; Wardi, Y.; Axelsson, H. Transition-time optimization for switched-mode dynamical systems. *IEEE Trans. Autom. Control* **2006**, *51*, 110–115. [CrossRef]
44. Dinh, K.H.; Weiler, P.; Leibold, M.; Wollherr, D. Fast and close to optimal trajectory generation for articulated robots in reaching motions. In Proceedings of the 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Munich, Germany, 3–7 July 2017; pp. 1221–1227. [CrossRef]
45. Bryson, A.E.; Ho, Y.C. *Applied Optimal Control Optimization, Estimation and Control*; Wiley: Hoboken, NJ, USA, 1975; pp. 221–228.
46. Pontryagin, L. *Mathematical Theory of Optimal Processes*, English ed.; Classics of Soviet Mathematics; CRC Press: Boca Raton, FL, USA, 1987.
47. Mahil, S.M.; Al-Durra, A. Modeling analysis and simulation of 2-DOF robotic manipulator. In Proceedings of the 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS), Abu Dhabi, United Arab Emirates, 16–19 October 2016; pp. 1–4. [CrossRef]
48. Rucco, A.; Notarstefano, G.; Hauser, J. Dynamics exploration of a single-track rigid car model with load transfer. In Proceedings of the Decision and Control (CDC), 2010 49th IEEE Conference, Atlanta, Georgia, USA, 15–17 December 2010. 2010, pp. 4934–4939.
49. Rubin, D.; Arogeti, S. Vehicle yaw stability control using rear active differential via sliding mode control methods. In Proceedings of the Control & Automation (MED), 2013 21st Mediterranean Conference, Chania, Crete, Greece, 25–28 June 2013; pp. 317–322.
50. Liu, W.; Li, Z.; Li, L.; Wang, F.Y. Parking Like a Human: A Direct Trajectory Planning Solution. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 3388–3397. [CrossRef]
51. Mayne, D.; Rawlings, J.; Rao, C.; Scokaert, P. Constrained model predictive control: Stability and optimality. *Automatica* **2000**, *36*, 789–814. [CrossRef]
52. Chen, C.; Shaw, L. On receding horizon feedback control. *Automatica* **1982**, *18*, 349–352. [CrossRef]
53. Michalska, H.; Mayne, D. Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. Autom. Control* **1993**, *38*, 1623–1633. [CrossRef]
54. Althoff, M.; Koschi, M.; Manzinger, S. CommonRoad: Composable benchmarks for motion planning on roads. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 719–726. [CrossRef]

55. de Souza Mendes, A.; Meneghetti, D.D.R.; Ackermann, M.; de Toledo Fleury, A. *Vehicle Dynamics-Lateral: Open Source Simulation Package for MATLAB*; Technical report; SAE Technical Paper; SAE: Warrendale, PA, USA, 2016.
56. Mayne, D.Q.; Michalska, H. Receding horizon control of nonlinear systems. *IEEE Trans. Autom. Control* **1990**, *35*, 814–824. [CrossRef]