AROC: A Toolbox for Automated Reachset Optimal Controller Synthesis

Niklas Kochdumper, Felix Gruber, Bastian Schürmann,
Victor Gaßmann, Moritz Klischat, Matthias Althoff
Technische Universität München, Germany

ABSTRACT

We present a MATLAB toolbox for Automated Reachset Optimal Control (AROC) that automatically synthesizes verified controllers for solving reach-avoid problems using reachability analysis. The toolbox implements two different types of control approaches: When using our verified model predictive controller, a feasible control law is constructed and verified on-the-fly during online application of the system. For motion-primitive-based control, on the other hand, controllers for many motion primitives are synthesized offline and then used for online motion planning with a maneuver automaton. Since our toolbox considers general nonlinear systems with input constraints, state constraints, and bounded disturbances, it is applicable to a very broad class of systems, as we demonstrate with several numerical examples. AROC is available at https://aroc.in.tum.de.

CCS CONCEPTS

• **Computing methodologies** → **Computational control theory**; **Motion path planning**.

KEYWORDS

Reach-avoid problems, controller synthesis, reachability analysis, maneuver automata, model predictive control.

ACM Reference Format:
Niklas Kochdumper, Felix Gruber, Bastian Schürmann,, Victor Gaßmann, Moritz Klischat, Matthias Althoff. 2021. AROC: A Toolbox for Automated Reachset Optimal Controller Synthesis. In *24th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '21), May 19–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3447928.3456703

1 INTRODUCTION

Most control tasks for cyber-physical systems can be modeled as reach-avoid problems, where the goal is to steer all states inside an initial set $\mathcal{R}_0$ as close as possible to a desired final state x_f while avoiding static and/or dynamic obstacles (see Fig. 1). Especially for safety-critical applications, such as autonomous driving or robotic surgery, it is crucial that reach-avoid problems are solved provably correct. In this work, we present our toolbox AROC, which automatically synthesizes verified controllers for solving reach-avoid problems using reachability analysis.

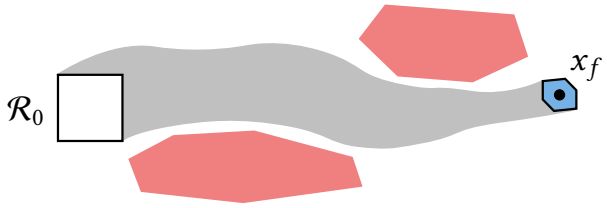

Figure 1: Schematic visualization of a reach-avoid problem, where obstacles are depicted in red, the reachable set of the controlled system is depicted in gray, and the final reachable set is depicted in blue.

1.1 State of the Art

Approaches for solving reach-avoid problems can mainly be divided into three categories: abstraction-based control, model predictive control (MPC), and motion-primitive-based control. Abstraction-based control approaches [16, 20, 43] usually abstract the system by finite-state automata, where reachability analysis is often used to determine valid transitions between the discrete states of the automaton. Based on this abstraction, controllers that are guaranteed to satisfy complex temporal logic specifications can be synthesized. However, since the abstraction by a finite-state automaton relies on a subdivision of the state space, the computational complexity for abstraction-based approaches often grows exponentially with the system dimension.

For model predictive control, there exist two different types of approaches: in implicit MPC [4, 6], optimal control inputs are computed on-the-fly during online application by solving an optimization problem. However, this is in general computationally demanding, so that implicit MPC is often only applicable for "slow" systems. To avoid this shortcoming, explicit MPC [9, 27, 29] computes optimal control inputs offline, and then uses a mapping function to assign an optimal control input to the current state during online application. Since this technique, however, usually requires to partition the state space, it suffers from the curse of dimensionality. For tube-based MPC [22, 30, 31, 42], which can be implicit or explicit, an additional tracking controller is applied to keep the system in a tube around the optimized trajectory, making the approach robust against disturbances.

The last group considered are motion-primitive-based control algorithms: The approach in [39] uses LQR-trees consisting of LQR-stabilized trajectories, where the regions of attractions for the single LQR controllers are computed with sum-of-squares techniques. Regions of attraction are also used in [21] and [40], where [21]

HSCC '21, May 19–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8339-4/21/05…$15.00
https://doi.org/10.1145/3447928.3456703

constructs a maneuver automaton from LQR stabilized trajectories, and [40] combines rapidly-exploring random trees (RRT) with stabilizing feedback controllers. Reachability-based trajectory design [17] uses parameterized reachable sets to determine collision-free motion plans with polynomial optimization. Moreover, the approach in [33] applies satisfiability modulo theory techniques to plan motions that satisfy temporal logic specifications, and the method in [34] proposes a hybrid controller for robust control based on motion primitives. Two drawbacks of motion-primitive-based planners are that usually a very large number of motion primitives is required, and that planning trajectories through narrow passages is often problematic since the available action space is restricted to discrete motions.

There already exist several tools for controller synthesis: Tools for abstraction-based control are CoSyMA [24], PESSOA [23], ROCS [19], SCOTS [32], and TuLiP [41], where CoSyMA, PESSOA, ROCS, and SCOTS consider nonlinear systems, and TuLip considers piecewise affine systems. Many toolboxes for designing and simulating implicit and explicit MPC controllers have been developed, e.g., the ACADO toolkit [14], the Control Toolbox [12], MATMPC [7], OptCon [25], OPTIPLAN [15], ParNMPC [10], and SolACE [3], but only few tools are able to verify correctness, like e.g., MUP [26] and CODEV [5]. For motion-primitive-based planning, many tools apply RRTs, e.g., DryVR 2.0 [28] and the Open Motion Planning Library [8]. Unlike most existing tools for MPC and motion-primitive-based control, AROC is able to guarantee robust safety for the synthesized controllers despite disturbances. Furthermore, most of the algorithms implemented in AROC scale better with the system dimension than abstraction-based control approaches, and are therefore well-suited for high-dimensional systems.

1.2 Overview

In this paper we present the toolbox AROC, which implements a) the reachset MPC approach in [38], b) the three motion-primitive-based control approaches *optimization-based control* [37], *convex interpolation control* [35], and *generator space control* [36], and c) a maneuver automaton for efficient online motion planning according to [13]. The paper is structured as follows: We first formally define reach-avoid problems in Sec. 2, before we explain the reachset MPC approach in Sec. 3. Next, in Sec. 4, we describe online motion planning with maneuver automata, and show how feedback controllers for motion primitives can be synthesized using the motion-primitive-based control algorithms. In Sec. 5 we provide a short description of the most important aspects of AROC as well as a code example, and finally demonstrate the performance of our novel toolbox on several benchmark systems in Sec 6.

2 PROBLEM FORMULATION

The AROC toolbox considers general nonlinear disturbed systems with input and state constraints defined by the differential equation

$$\dot{x}(t) = f(x(t), u(t), w(t)), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the vector of system states, $u(t) \in \mathbb{R}^m$ is the vector of control inputs, $w(t) \in \mathbb{R}^q$ is the vector of disturbances, $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q \to \mathbb{R}^n$ is a Lipschitz continuous function, and $t \in \mathbb{R}^+$ is the time. The disturbances are bounded by a compact set $w(t) \in \mathcal{W} \subset \mathbb{R}^q$, and the system has to satisfy the input constraints defined by the convex set $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ as well as the state constraints defined by the convex set $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$.

Given a nonlinear system defined as in (1), the goal is to solve a reach-avoid problem, where all states inside an initial set $x(0) \in \mathcal{R}_0 \subset \mathbb{R}^n$ should be steered as close as possible to a desired final state $x_f \in \mathbb{R}^n$ while avoiding collisions with unsafe sets (see Fig. 1). The AROC toolbox automatically synthesizes a suitable control law $u_c(x(t), t)$. Let us denote the solution to the closed-loop system at time t for an initial state $x_0 = x(0)$ by $\xi(t, x_0, u_c(\cdot), w(\cdot))$. In order to verify the controller, we use reachability analysis:

Definition 1. *(Reachable Set) The reachable set of the controlled system at time t is*

$$\mathcal{R}_{u_c(\cdot)}(t) = \Big\{ \xi\big(t, x_0, u_c(\cdot), w(\cdot)\big) \;\Big|\; x_0 \in \mathcal{R}_0, \; \forall \tau \in [0, t] : \; w(\tau) \in \mathcal{W} \Big\},$$

where $\mathcal{R}_0 \subset \mathbb{R}^n$ is the inital set and $\mathcal{W} \subset \mathbb{R}^q$ the set of disturbances.

Let us first present the verified synthesis of model predictive controllers.

3 MODEL PREDICTIVE CONTROL

AROC implements the model predictive controller in [38], which is visualized in Fig. 2. To ensure safety for an infinite time horizon, [38] considers a terminal region $\mathcal{T} \subset \mathbb{R}^n$ around a desired final state $x_f \in \mathbb{R}^n$ for which a stabilizing controller is known. The synthesized control law guarantees that the terminal region is reached in finite time and combines a feed-forward part with a tracking controller:

$$u_c(x(t), t) = u_{ff}(t) + K(x(t) - x_{ff}(t)),$$

where the piecewise constant control inputs $u_{ff}(t)$ for the feed-forward trajectory $x_{ff}(t)$ are determined by solving an optimal control problem, and $K \in \mathbb{R}^{m \times n}$ is the feedback matrix of a linear-quadratic regulator (LQR) [18, Chapter 3.3] of the linearized system.

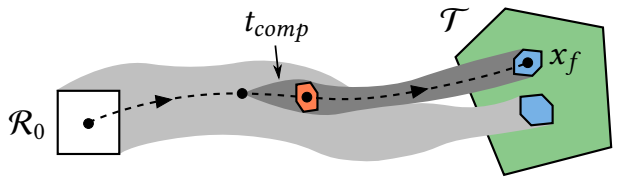

Figure 2: Schematic visualization of the reachset model predictive control algorithm, where the reachable set for the initial control law is depicted in light gray and the reachable set for the updated control law in dark gray. The predicted set of system states at the allocated computation time is shown in orange.

As common in MPC, the approach in [38] tries to construct a better control law with lower costs each time a new measurement of the system state is obtained. This, however, poses a problem, since the optimization based construction of the new control law as well as its verification with reachability analysis requires a certain computation time t_{comp}. Once the computation is finished, the system

will therefore already be in a different state, impairing safety guarantees. To solve this problem, [38] applies reachability analysis to predict all possible system states at the time when the computation of the new control law is finished, and then uses the set of predicted states for verifying the new control law. If the actual computation time exceeds the allocated time t_{comp}, the computation of the new control law is terminated and the previous control law is used. Since the computation time is therefore explicitly considered, safety can be guaranteed for all times.

4 MOTION-PRIMITIVE-BASED CONTROL

Maneuver automata are a computationally efficient method to solve reach-avoid problems with non-convex and dynamic obstacles online [13]. In AROC, a maneuver automaton can be constructed from a list of motion primitives [11] for which a feasible feedback controller has been synthesized with one of the algorithms described in this section. The automaton then stores information about which motion primitives can be connected to each other. Two motion primitives can be connected if the final reachable set of the first motion primitive is a subset of the initial set of the second motion primitive [13, Eq. (3)], since only then it is guaranteed that the controlled system stays in the reach-tube defined by the concatenated motion primitives. With a maneuver automaton, online motion planning then reduces to the task of solving a standard discrete search problem, as it is visualized in Fig. 3 for the example of an autonomous car.

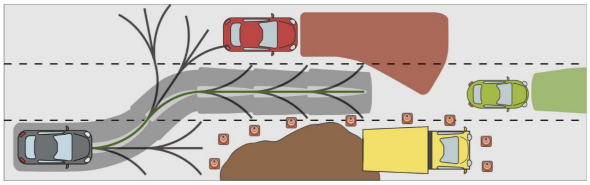

Figure 3: Motion planning with a maneuver automaton, where the reachable set of the car center is shown in light gray, and the space occupied by the whole car in dark gray.

After introducing maneuver automata, we now describe the different algorithms AROC provides for synthesizing control laws for single motion primitives. Our goal is to determine a control law that steers all states inside the initial set $\mathcal{R}_0$ as close as possible to the desired final state $x_f \in \mathbb{R}^n$ at the final time t_f:

$$\min_{u_c(x(t),t)} \ \rho\left(\mathcal{R}_{u_c(\cdot)}(t_f), x_f\right) \quad \text{s.t. input and state constraints are met,} \tag{2}$$

where $\rho(\mathcal{R}_{u_c(\cdot)}(t_f), x_f) \to \mathbb{R}^+$ is a cost function measuring the distance between the states in $\mathcal{R}_{u_c(\cdot)}(t_f)$ and the desired final state x_f. Since in general the optimization problem in (2) cannot be solved exactly, the algorithms presented in this section compute a feasible and close to optimal solution instead. All algorithms first solve an optimal control problem to determine the control inputs $u_{ref}(t)$ for a reference trajectory $x_{ref}(t)$ leading from the center of $\mathcal{R}_0$ to x_f, and then synthesize a feedback controller that tracks this reference trajectory. Furthermore, the time horizon $t \in [0, t_f]$ is divided into $N \in \mathbb{N}^+$ time steps. To obtain a maneuver automaton with many connections, it is crucial that the control algorithms are able to contract the reachable set.

4.1 Optimization-Based Control

Let us first present the optimization-based approach in [37], which optimizes the following control law:

$$u_c(x(t), t) = u_{ref}(t) + K(t)(x(t) - x_{ref}(t)),$$

where the time-varying feedback matrix $K(t) \in \mathbb{R}^{m \times n}$ is constant for each time step. The approach in [37] then determines suitable feedback matrices $K_1, \ldots, K_N \in \mathbb{R}^{m \times n}$ by solving the optimization problem in (2) directly using nonlinear programming. One major advantage of the optimization-based controller is that continuous feedback is used, which makes the approach very robust against disturbances. However, one downside is that solving (2) with nonlinear programming is often computationally expensive. A schematic visualization of the approach is shown in Fig. 4.

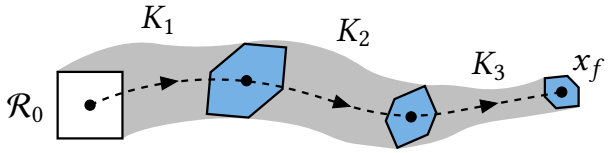

Figure 4: Schemantic visualization of a the optimization-based control algorithm with $N = 3$ time steps.

4.2 Convex Interpolation Control

Another algorithm implemented in AROC is the convex interpolation control approach in [35], where in each time step the following procedure is applied in order to synthesize a suitable control law:

(1) The reachable set at the beginning of the time step is enclosed by a parallelotope.
(2) Optimal control inputs for all vertices of the parallelotope are determined by solving optimal control problems, with the goal of steering the trajectories starting at these vertices as close as possible to the reference trajectory.
(3) Interpolation between the optimal control inputs for the parallelotope vertices as described in [35, Sec. 4] yields the control law $u_c(x(t), t)$.

The resulting nonlinear interpolation control law automatically adapts to nonlinearities in the model, so that the approach in [35] is well-suited for the control of strongly nonlinear systems. However, one disadvantage is that the computational complexity grows exponentially with respect to the system dimension n since a n-dimensional parallelotope has 2^n vertices. A schematic visualization of the convex interpolation control approach is shown in Fig. 5.

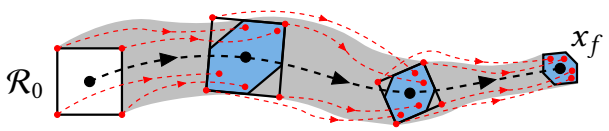

Figure 5: Schemantic visualization of a the convex interpolation control algorithm with $N = 3$ time steps.

4.3 Generator Space Control

The last control algorithm for motion primitives is the generator space controller in [36]. This control algorithm applies the same procedure as the convex interpolation approach in Sec. 4.2, but solves an optimal control problem for each generator vector of the parallelotope, instead of for each vertex. Since a parallelotope has only n generator vectors, this is computationally much more efficient. The control law $u_c(x(t), t)$ is then constructed through superposition of the control inputs for the different generators [36, Eq. (15)]. Since the computation complexity is only polynomial with respect to the system dimension, the approach in [36] is well-suited for high-dimensional systems. One drawback is that the resulting control law is linear, which can be problematic for strongly nonlinear systems. A schematic visualization of the generator space control approach is shown in Fig. 6.

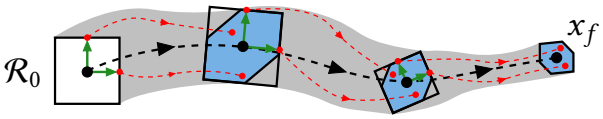

Figure 6: Schematic visualization of a the generator space control algorithm with $N = 3$ time steps.

5 TOOLBOX DESCRIPTION

AROC is implemented in MATLAB, uses the CORA toolbox [1] to compute reachable sets, and the ACADO toolkit [14] to solve optimal control problems. For autonomous driving applications, AROC provides an interface to CommonRoad [2], a framework containing thousands of traffic scenarios, which makes it easy to test motion planning with maneuver automata created in AROC. The syntax for executing controller synthesis is identical for all algorithms, so that different control algorithms can be compared effortlessly. In addition, offline controller synthesis and online execution of the controller are strictly separated in AROC, which makes it simple to implement the online control law on a micro-controller for real-world applications.

Let us now demonstrate the AROC toolbox with a code example. We consider the deliberately simple example of a cart that is coupled to the environment via a damping element and a spring with nonlinear stiffness:

$$\begin{aligned} \dot{x}_1 &= x_2 + w_1 \\ \dot{x}_2 &= (-d \cdot x_2^2 - k \cdot x_1^3 + u)/m + w_2, \end{aligned} \tag{3}$$

where the system states are the position x_1 and the velocity x_2 of the cart, and the system input is a force u acting on the cart. The parameters are the weight of the cart $m = 1\text{kg}$, the damping constant $d = 1\text{kg}\,\text{m}^{-1}$, and the spring stiffness constant $k = 1\text{N}\,\text{m}^{-3}$. The input constraint is $u \in [-14, 14]\text{N}$, and the set of disturbances is $w_1 \in [-0.1, 0.1]\text{m}\,\text{s}^{-1}$ and $w_2 \in [-0.1, 0.1]\text{m}\,\text{s}^{-2}$. Furthermore, we consider the initial set $x_1(0) \in [-0.2, 0.2]\text{m}$ and $x_2(0) \in [-0.2, 0.2]\text{m}\,\text{s}^{-1}$. The goal is to synthesize a controller that steers all states inside the initial set as close as possible to the desired final state $x_f = [2\text{m}, 0\text{m}\,\text{s}^{-1}]^T$ at the final time $t_f = 1\text{s}$.

To solve this problem with AROC we first create a function `cart` that implements the differential equation in (3):

```
function f = cart(x,u,w)

    m = 1; k = 1; d = 1;

    f(1,1) = x(2)+w(1);
    f(2,1) = (-d*x(2)^2-k*x(1)^3+u)/m+w(2);
end
```

Next, we synthesize a feasible controller using the optimization-based control approach in Sec. 4.1 as follows:

```
1  % system parameter
2  Param.R0 = interval([-0.2;-0.2],[0.2;0.2]);
3  Param.xf = [2;0];
4  Param.tFinal = 1;
5  Param.U = interval(-14,14);
6  Param.W = interval([-0.1;-0.1],[0.1;0.1]);
7
8  % algorithm settings
9  Opts.N = 10;
10 Opts.refTraj.Q = diag([5,5]);
11 Opts.refTraj.R = 1e-3;
12
13 % controller synthesis
14 optimizationBasedControl('cart',Param,Opts);
```

In lines 1-6 in the code example above the system parameter, such as the initial set $\mathcal{R}_0$, the set of disturbances $\mathcal{W}$, etc., are defined first. Next, in lines 8-11 algorithm settings for the optimization-based control algorithm are specified. These are the number of time steps N, as well as the state weighing matrix $Q \in \mathbb{R}^{n \times n}$ and the input weighting matrix $R \in \mathbb{R}^{m \times m}$ for the optimal control problem that is solved to obtain the reference trajectory $x_{ref}(t)$. Finally, a suitable controller is synthesized in Line 14. The resulting reachable set of the controlled system is visualized in Fig. 7.

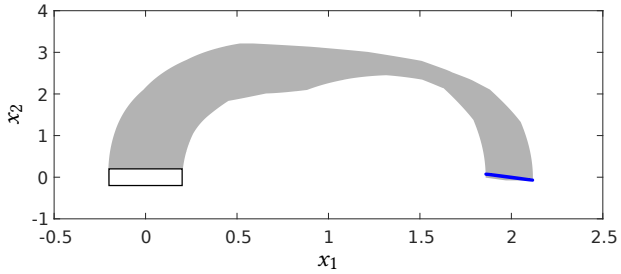

Figure 7: Visualization of the reachable set of the controlled system (gray) for the cart benchmark in (3), where the initial set is depicted in white with a black border, and the final reachable set is depicted in blue.

6 NUMERICAL EXAMPLES

In this section, we demonstrate the capabilities of the AROC toolbox on several benchmark systems. All computations are carried out in MATLAB on a 2.9GHz quad-core i7 processor with 32GB memory.

6.1 Artificial System

First, we consider the 2-dimensional nonlinear disturbed artificial system in [42, Eq. (19)], which we control with the reachset MPC

approach described in Sec. 3. We allocate a computation time of $t_{comp} = 2.5$ seconds, which allows us to run reachset MPC in real-time. The results are visualized in Fig. 8, where we compare AROC with the tube-based MPC approach in [42]. While both approaches calculate very similar optimal trajectories, the reachable set computed with AROC is clearly much tighter than the tube computed with the method in [42].

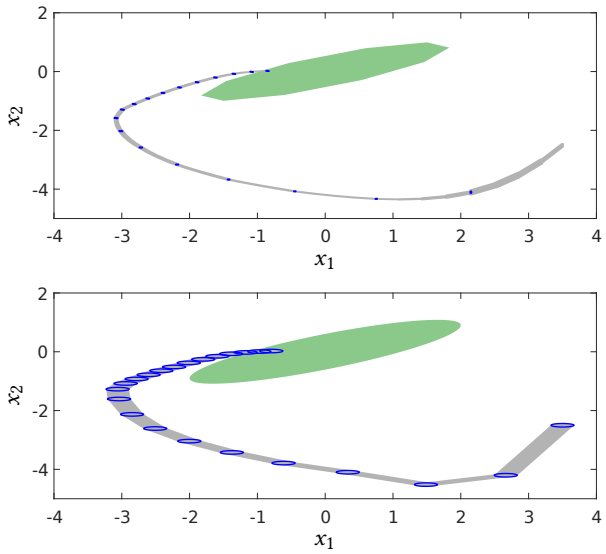

Figure 8: Visualization of the reachable set/tube (gray) for the artificial system in Sec. 6.1 computed with AROC (top) and the tube-based MPC approach in [42] (bottom). The terminal region is shown in green, and the reachable set at the end of each time step in blue. The results for the tube-based approach are taken from [42, Fig. 2].

6.2 Vehicle Platoon

Next, we examine the benchmark in [37, Sec. IV], which describes a platoon with M vehicles. The dimension of the system, which is $n = 2M$, can be increased by adding more vehicles to the platoon. As a motion primitive we consider the same acceleration maneuver as in [37, Sec. IV], for which we synthesize a feasible controller with the convex interpolation control approach described in Sec. 4.2. The input constraints, the state constraints, and the set of disturbances are identical to the ones in [37, Sec. IV]. Tab. 1 shows the computation time of controller synthesis for different system dimensions, where we compare AROC with the abstraction-based tool SCOTS [32]. For higher dimensions SCOTS failed to synthesize a suitable controller in less than 20 minutes, while AROC was still able to construct a controller in reasonable time. The resulting reachable set for a platoon with 4 vehicles is visualized in Fig. 9. Clearly, the controller synthesized with AROC is able to contract the reachable set, so that the shifted final set is a subset of the initial set.

6.3 Autonomous Car

As a final example, we demonstrate motion planning with a maneuver automaton using AROC. For this, we consider the 4-dimensional kinematic single track model of an autonomous car in [35, Eq. (19)]

Table 1: Computation time of controller synthesis for a vehicle platoon with M vehicles in seconds.

Vehicles	M = 1	M = 2	M = 3	M = 4	M = 5
AROC	14.93	33.12	49.25	75.97	169.88
SCOTS	5.19	-	-	-	-

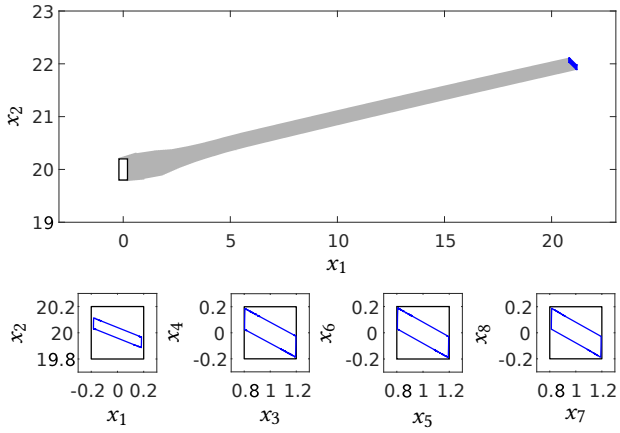

Figure 9: Visualization of the reachable set of the controlled system (top), the shifted final reachable set (bottom, blue), and the initial set (bottom, black) for a vehicle platoon with $M = 4$ vehicles computed with AROC.

with the same input constraints and the same set of disturbances as in [35, Sec. 6]. We construct a maneuver automaton consisting of four motion primitives with the generator space control approach described in Sec. 4.3, and use this maneuver automaton to solve the motion planning problem defined by the CommonRoad [2] traffic scenario *ZAM_Zip-1_19_T-1*. The resulting trajectory planned with the maneuver automaton is visualized in Fig. 10. While the offline construction of suitable controllers for the four motion primitives takes 231 seconds, online motion planning with an A* search algorithm takes only 0.9 seconds for a planning horizon of 9 seconds.

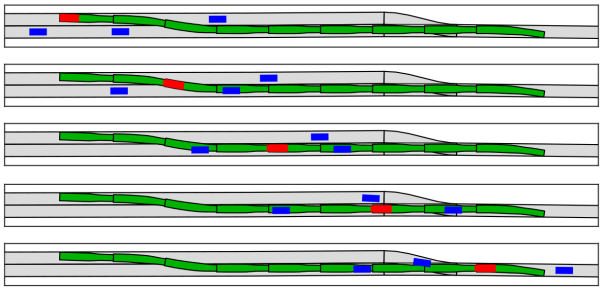

Figure 10: Visualization of the planned trajectory (green), the current position of the autonomous car (red), and the occupancy sets for the other traffic participants (blue) for times 0s, 2s, 4s, 6s and 8s (top to bottom).

7 CONCLUSION

We presented AROC, a novel toolbox for the automated synthesis of robust controllers, which implements one model predictive control algorithm as well as the three motion-primitive-based control

algorithms *optimization-based control*, *convex interpolation control*, and *generator space control*. Since the toolbox considers the very general case of disturbed nonlinear continuous systems with input and state constraints, it is applicable to a very broad class of systems, as we demonstrated by several numerical examples. Moreover, the controllers synthesized by AROC are guaranteed to satisfy input and state constraints despite disturbances since reachability analysis is applied internally for verification. Another advantage is that most of the implemented control algorithms scale well with respect to the system dimension, so that the toolbox is also applicable to high-dimensional systems.

ACKNOWLEDGMENTS

We gratefully acknowledge financial support by the project justIT-SELF funded by the European Research Council (ERC) under grant number 817629, and the German Research Foundation (DFG) project faveAC under grant number AL 1185/5-1.

REFERENCES

[1] M. Althoff. 2015. An Introduction to CORA 2015. In *Proc. of the 1st and 2nd International Workshop on Applied Verification for Continuous and Hybrid Systems*. 120–151.

[2] M. Althoff, M. Koschi, and S. Manzinger. 2017. CommonRoad: Composable Benchmarks for Motion Planning on Roads. In *Proc. of the 28th IEEE Intelligent Vehicles Symposium*. 719–726.

[3] S. Bhonsale and et al. 2016. SolACE: An Open Source Package for Nolinear Model Predictive Control and State Estimation for (Bio)Chemical Processes. *Computer Aided Chemical Engineering* 38 (2016), 1971–1976.

[4] J. M. Bravo, T. Alamo, and E. F. Camacho. 2006. Robust MPC of Constrained Discrete-Time Nonlinear Systems Based on Approximated Reachable Sets. *Automatica* 42 (2006), 1745–1751.

[5] N. Chan and S. Mitra. 2018. CODEV: Automated Model Predictive Control Design and Formal Verification. In *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*. 281–282.

[6] H. Chen, C. Scherer, and F. Allgöwer. 1997. A Game Theoretic Approach to Nonlinear Robust Receding Horizon Control of Constrained Systems. In *Proc. of the 1997 American Control Conference*. 3073–3077.

[7] Y. Chen and et al. 2019. MATMPC–A MATLAB Based Toolbox for Real-time Nonlinear Model Predictive Control. In *Proc. of the 18th European Control Conference*. 3365–3370.

[8] I. A. Şucan, M. Moll, and L. E. Kavraki. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine* 19, 4 (2012), 72–82.

[9] M. de la Pena, A. Bemporad, and C. Filippi. 2004. Robust Explicit MPC Based on Approximate Multi-parametric Convex Programming. In *Proc. of the 43rd IEEE Conference on Decision and Control*. 2491–2496.

[10] H. Deng and T. Ohtsuka. 2020. ParNMPC–A Parallel Optimisation Toolkit for Real-time Nonlinear Model Predictive Control. *To appear in: International Journal of Control* (2020).

[11] E. Frazzoli, M. A. Dahleh, and E. Feron. 2005. Maneuver-Based Motion Planning for Nonlinear Systems With Symmetries. *IEEE Transactions on Robotics* 21, 6 (2005), 1077–1091.

[12] M. Giftthaler and et al. 2018. The Control Toolbox–An Open-Source C++ Library for Robotics, Optimal and Model Predictive Control. In *Proc. of the 2018 International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. 123–129.

[13] D. Heß, M. Althoff, and T. Sattel. 2014. Formal Verification of Maneuver Automata for Parameterized Motion Primitives. In *Proc. of the 2014 International Conference on Intelligent Robots and Systems*. 1474–1481.

[14] B. Houska, H. J. Ferreau, and M. Diehl. 2011. ACADO Toolkit–An Open-source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods* 32, 3 (2011), 298–312.

[15] F. Janeček and et al. 2017. OPTIPLAN: A MATLAB Toolbox for Model Predictive Control with Obstacle Avoidance. In *Proc. of the 20th IFAC World Congress*. 531–536.

[16] M. Kloetzer and C. Belta. 2008. A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications. *IEEE Trans. Automat. Control* 53, 1 (2008), 287–297.

[17] S. Kousik and et al. 2020. Bridging the Gap between Safety and Real-time Performance in Receding-horizon Trajectory Design for Mobile Robots. *The International Journal of Robotics Research* 39, 12 (2020), 1419–1469.

[18] H. Kwakernaak and R. Sivan. 1972. *Linear Optimal Control Systems*. Vol. 1. Wiley.

[19] Y. Li and J. Liu. 2018. ROCS: A Robustly Complete Control Synthesis Tool for Nonlinear Dynamical Systems. In *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*. 130–135.

[20] L. Liu and et al. 2012. Reactive Controllers for Differentially Flat Systems with Temporal Logic Constraints. In *Proc. of the 51st IEEE Conference on Decision and Control*. 7664–7670.

[21] A. Majumdar and R. Tedrake. 2017. Funnel Libraries for Real-time Robust Feedback Motion Planning. *The International Journal of Robotics Research* 36, 8 (2017), 947–982.

[22] D. Mayne and et al. 2011. Tube-based Robust Nonlinear Model Predictive Control. *International Journal of Robust and Nonlinear Control* 21, 11 (2011), 1341–1353.

[23] M. Mazo, A. Davitian, and P. Tabuada. 2010. PESSOA: A Tool for Embedded Controller Synthesis. In *Proc. of the 22nd International Conference on Computer Aided Verification*. 566–569.

[24] S. Mouelhi, A. Girard, and G. Gössler. 2013. CoSyMA: A Tool for Controller Synthesis using Multi-scale Abstractions. In *Proc. of the 16th International Conference on Hybrid Systems: Computation and Control*. 83–88.

[25] Z. K. Nagy. 2008. OptCon–An Efficient Tool for Rapid Prototyping of Nonlinear Model Predictive Control Applications. In *Proc. of the 2008 AIChE Annual Meeting*. 16–21.

[26] J. Oravec and M. Bakošové. 2015. Software for Efficient LMI-based Robust MPC Design. In *Proc. of the 20th International Conference on Process Control*. 272–277.

[27] E. Pistikopoulos. 2009. Perspectives in Multiparametric Programming and Explicit Model Predictive Control. *AIChE Journal* 55, 8 (2009), 1918–1925.

[28] B. Qi and et al. 2018. DryVR 2.0: A Tool for Verification and Controller Synthesis of Black-box Cyber-physical Systems. In *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*. 269–270.

[29] D. M. Raimondo and et al. 2011. A Robust Explicit Nonlinear MPC Controller with Input-to-state Stability Guarantees. In *Proc. of the 18th IFAC World Congress*. 9284–9289.

[30] S. V. Raković and et al. 2012. Parameterized Tube Model Predictive Control. *IEEE Trans. Automat. Control* 57, 11 (2012), 2746–2761.

[31] M. Rubagotti and et al. 2010. Robust Model Predictive Control with Integral Sliding Mode in Continuous-Time Sampled-Data Nonlinear Systems. *IEEE Trans. Automat. Control* 56, 3 (2010), 556–570.

[32] M. Rungger and M. Zamani. 2016. SCOTS: A Tool for the Synthesis of Symbolic Controllers. In *Proc. of the 19th International Conference on Hybrid Systems: Computation and Control*. 99–104.

[33] I. Saha and et al. 2014. Automated Composition of Motion Primitives for Multi-Robot Systems from Safe LTL Specifications. In *Proc. of the 2014 International Conference on Intelligent Robots and Systems*. 1525 – 1532.

[34] R. Sanfelice and E. Frazzoli. 2008. A Hybrid Control Framework for Robust Maneuver-based Motion Planning. In *Proc. of the 2008 American Control Conference*. 2254–2259.

[35] B. Schürmann and M. Althoff. 2017. Convex Interpolation Control with Formal Guarantees for Disturbed and Constrained Nonlinear Systems. In *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*. 121–130.

[36] B. Schürmann and M. Althoff. 2017. Guaranteeing Constraints of Disturbed Nonlinear Systems Using Set-Based Optimal Control in Generator Space. In *Proc. of the 20th IFAC World Congress*. 11515–11522.

[37] B. Schürmann and M. Althoff. 2017. Optimal Control of Sets of Solutions to Formally Guarantee Constraints of Disturbed Linear Systems. In *Proc. of the 2017 American Control Conference*. 2522–2529.

[38] B. Schürmann, N. Kochdumper, and M. Althoff. 2018. Reachset Model Predictive Control for Disturbed Nonlinear Systems. In *Proc. of the 57th IEEE Conference on Decision and Control*. 3463–3470.

[39] R. Tedrake and et al. 2010. LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification. *The International Journal of Robotics Research* 29, 8 (2010), 1038–1052.

[40] A. Weiss and et al. 2017. Motion Planning with Invariant Set Trees. In *Proc. of the 1st IEEE Conference on Control Technology and Applications*. 1625–1630.

[41] T. Wongpiromsarn and et al. 2011. TuLiP: A Software Toolbox for Receding Horizon Temporal Logic Planning. In *Proc. of the 14th International Conference on Hybrid Systems: Computation and Control*. 313–314.

[42] S. Yu and et al. 2013. Tube MPC Scheme Based on Robust Control Invariant Set with Application to Lipschitz Nonlinear Systems. *Systems & Control Letters* 62, 2 (2013), 194–200.

[43] M. Zamani and et al. 2012. Symbolic Models for Nonlinear Control Systems Without Stability Assumptions. *IEEE Trans. Automat. Control* 57, 7 (2012), 1804–1809.