Online Reachability Analysis and Space Convexification for Autonomous Racing

Sergiy Bogomolov
Newcastle University,
Newcastle upon Tyne, United Kingdom
sergiy.bogomolov@ncl.ac.uk

Taylor T. Johnson
Vanderbilt University,
Nashville, USA
taylor.johnson@vanderbilt.edu

Diego Manzanas Lopez
Vanderbilt University,
Nashville, USA
diego.manzanas.lopez@vanderbilt.edu

Patrick Musau
Vanderbilt University,
Nashville, USA
patrick.musau@vanderbilt.edu

Paulius Stankaitis
Newcastle University,
Newcastle upon Tyne, United Kingdom
paulius.stankaitis@ncl.ac.uk

This paper presents an optimisation-based approach for an obstacle avoidance problem within an autonomous vehicle racing context. Our control regime leverages online reachability analysis and sensor data to compute the maximal safe traversable region that an agent can traverse within the environment. The idea is to first compute a non-convex safe region, which then can be convexified via a novel coupled separating hyperplane algorithm. This derived safe area is then used to formulate a nonlinear model-predictive control problem that seeks to find an optimal and safe driving trajectory. We evaluate the proposed approach through a series of diverse experiments and assess the runtime requirements of our proposed approach through an analysis of the effects of a set of varying optimisation objectives for generating these coupled hyperplanes.

1 Introduction

Over the last several years, autonomous racing has actively been pursued as a strategy to explore edge-case scenarios in autonomous driving [16]. Racing scenarios present unique challenges with respect to navigating high speeds and multi-agent interactions. In these contexts, vehicles must be able to operate at the edge of their operating envelopes in close proximity to static and dynamic obstacles. Several competitions have emerged over the last couple of years, such as the Indy Autonomous Challenge (IAC) [16], and the F1TENTH International Autonomous racing competition [34]. Although numerous racing strategies have been proposed over the last several years, head-to-head racing at high speeds remains a challenge. Unlike the time trials that are frequently used as qualification rounds in these competitions [33], head-to-head racing requires designing a regime to be able to predict the future trajectories that reflect the intentions of the other opponents and drive through the track as quickly as possible.

Within the autonomous racing space, one of the most popular frameworks for tackling the racing problem has been formulating and solving an optimisation problem that balances obstacle avoidance and travelling at high velocities [40, 20]. Specifically, the model-predictive control framework (MPC), which finds optimal control commands based on a model of the underlying system, while satisfying a set of constraints is the most widely used approach [18]. Although MPC approaches have enjoyed success

M. Farrell, M. Luckcuck, M. Schwammberger, and
M. Gleirscher (Eds): Fifth International Workshop on
Formal Methods for Autonomous Systems (FMAS 2023)
EPTCS 395, 2023, pp. 95–112, doi:10.4204/EPTCS.395.7

in these settings [40], one of the main limitations exhibited by many approaches is a lack of robust online risk assessment in often dynamic and uncertain environments, particularly around vehicle-to-vehicle interactions. While a lot of progress has been made in this area, collisions still occur due to misplaced estimations of the set of all possible trajectories that the vehicle could pursue [20]. Furthermore, as Katrakazas et al. note "exhaustively calculating and predicting the trajectories of other traffic participants at each epoch incurs a huge computational cost". Currently, many existing approaches treat the vehicle as an isolated entity, and the behavioural models of other participants within the environment have not yet been widely incorporated into the MPC regime [20].

One of the ways that this challenge has been addressed has been through the use of reachability analysis approaches [1]. The idea is to compute the set of states that the other racing agents could occupy in the future, for a fixed time horizon, and plan trajectories for the ego vehicle that avoids this unsafe set [23, 26, 29]. This unsafe set allows for modelling the inherent uncertainty in the behaviour of other agents and for the synthesis of safe racing trajectories [1]. There are two main challenges that arise in these contexts. The first is that over long time horizons, reachability approaches will result in overly conservative behaviours as the set of avoidable states grows. The second is that reachability approaches are typically computationally challenging endeavours, thus leveraging them online is quite challenging. In light of these challenges, the following paper presents a model-predictive control framework leveraging real-time reachability for a 1/10 scale autonomous vehicle test-bed in a multi-agent racing setting modelled after the F1TENTH International Autonomous Racing Competition.

Finally, obtaining a solution to the MPC problem generally entails solving a convex optimisation problem, which guarantees convergence to a globally optimum solution. However, due to the presence of static and dynamic obstacles, the optimal control problem of obstacle avoidance is inherently a non-convex problem. Therefore, to solve this problem efficiently many approaches leverage state-space convexification. In the past, several state-space convexification approaches have been proposed, including region partitioning [28], computing separating hyperplanes [25, 31], and constructing approximations using stored data points [38] (further discussed in Section 2). In our framework, we propose a novel optimisation-based approach for convexifying non-convex state spaces by computing coupled separating hyperplanes. The coupling of separating hyperplanes makes it possible to compute optimal safe and convex regions. However, it comes at the cost of increased computation time. Therefore, in this paper, we investigate the feasibility (e.g., timing constraints) of computing coupled separating hyperplanes in a real-time autonomous racing scenario.

In summary, the contributions of this paper are: (1) we introduce a novel closed-loop model-predictive obstacle avoidance controller that integrates online reachability analysis and an optimisation-based state-space convexification approach, (2) we evaluate this approach across a diverse set of simulation experiments using the F1TENTH simulation platform. These experiments include varying the number of dynamic agents, the number of static and dynamic obstacles, and the racing environment. (3) We present a timing analysis of the state-space convexification approach. (4) Finally, we evaluate our approach against the well-known model-predictive contouring control approach, which has shown great success in obstacle avoidance tasks.

2 Related Work

Researchers have approached the obstacle avoidance problem from two major perspectives. The first strategy involved formulating and solving an optimisation problem. The second regime has typically involved a hierarchical decomposition of path planning and reference tracking. A variety of algorithms

such as artificial potential fields [47], genetic algorithms [45], rapidly-exploring random trees (RRT) [19], fuzzy logic algorithms [32], elastic band theory [12], and rolling window methods [46] have demonstrated success in numerous arenas. A key limitation of many path planning approaches is that they are incapable of respecting kinodynamic constraints, such as bounds on the acceleration, and often the trajectories must be passed to a low-level controller that utilises a higher fidelity dynamics model and respects control constraints [40]. Furthermore, in highly dynamic and uncertain environments, planners must be able to replan sufficiently fast to react appropriately to split-second environmental threats [19]. However, most planners typically do not replan sufficiently rapidly to ensure split-second reactivity to threats [23].

As mentioned previously, MPC approaches have demonstrated great success in generating optimal trajectories that respect kinodynamic constraints and recently researchers have combined these approaches with reachability analysis to generate provably collision-free paths [1, 5, 23, 35, 23]. Within this regime, [1, 35, 26] utilise forward reachability methods in order to eliminate areas of the state space that would result in collisions. While these methods are extremely effective, these approaches must be implemented carefully in order to ensure that the resulting trajectories do not result in overly conservative behaviours [23]. The alternative to these approaches is backward reachability approaches [5, 23] which utilise a target set representing a set of undesirable states, in order to design controllers that can guarantee dynamic and static obstacle avoidance with minimal intervention. However, these approaches are computationally demanding and typically the safety-ensuring control constraints, derived from these methods, are computed and cached offline before being incorporated into an MPC problem [23].

Beyond reachability methods, over the last several years, several space convexification approaches for the obstacle avoidance problem have been proposed. In [39] a feasible convex set for model-predictive control is obtained by computing two parallel time-varying hyperplanes on racetrack borders. However, the resulting hyperplanes do not consider static obstacles or dynamic agents. The works of Mercy et al. [31, 30] and Scholte et al. [41] utilise the concept of separating hyperplanes to compute hyperplanes which separate autonomous systems from convex obstacles. The paper [22] combines the model-predictive control and dynamic agent reachability analysis, and uses IRIS (Iterative Regional Inflation by Semi-definite programming) [11] for a state-space convexification. A similar approach has been introduced in [27] for motion planning. Finally, in [28] two (polar and convex) different types of convexification methods based on region partitioning for obstacle avoidance were proposed. Their convex partitioning regime utilises a convex partitioning algorithm [21] to compute the minimum number of convex regions that are needed to capture non-convex obstacles, whereas the polar partitioning approach derives a safe set by using a minimum number of triangles.

The state-space convexification approaches described above have two main limitations for the racing scenario: they generally aim to compute the largest convex region in the non-convex space (e.g., not necessarily in the travelling direction of the ego vehicle) or are not able to handle non-convex obstacles. In our approach, we also express the problem of computing separating hyperplanes as an optimisation problem, but we are interested in computing a correct set of separating hyperplanes that provide the largest safe convex region in the direction of the ego vehicle. Furthermore, our proposed approach is able to handle non-convex obstacles.

3 Preliminaries

3.1 Model-Predictive Control

Let us suppose we have the following (1) discrete-time system where $x \in X \subseteq \mathbb{R}^n$, $u \in U \subseteq \mathbb{R}^m$ and $t \in \mathbb{N}$.

$$x_{t+1} = f(x_t, u_t) \tag{1}$$

The MPC problem can then be expressed as a finite horizon optimisation problem (1) where a cost function J is being minimised over a finite time horizon N subject to constraints (2.1 - 2.4).

$$J_{t\to t+N}(x_t) = \min_{u_0,\ldots,u_{N-1}} p(x_{t+N}) + \sum_{k=t}^{t+N-1} q(x_k, u_k) \tag{2}$$

$$x_{k+1} = f(x_k, u_k),\ \forall k \in \{t, \ldots, t+N-1\} \tag{2.1}$$

$$x_0 = x_s \tag{2.2}$$

$$x_k \in X,\ \forall k \in \{t, \ldots, t+N-1\} \tag{2.3}$$

$$u_k \in U,\ \forall k \in \{t, \ldots, t+N-1\} \tag{2.4}$$

The cost function J is made up of a stage cost function q and a terminal cost function p which determine the cost of being at the interim state x_k after applying an input u_k, and the cost of being at the final state x_{t+N}. The constraints (2.1 - 2.4) assert that the optimisation problem, given by equation (1), begins from an initial state x_s and that the interim state and control inputs must respect the constraint sets X and U.

If the dynamics and constraints can be formulated as linear expressions, then the MPC problem can be solved efficiently using standard convex optimisation techniques. However, if the dynamics or constraints are nonlinear, then the problem becomes a nonlinear optimisation problem that is much more computationally challenging to solve. On the other hand, allowing for nonlinear dynamics and constraints may permit one to track complex systems with a higher level of fidelity than using linear expressions. Thus, the computational cost must be evaluated against overall system performance [42].

3.2 Reachability Analysis

Reachability analysis is a technique for computing the set of all reachable states of a dynamical system from a set of initial states. The reachable set of R_{t+1} can be defined formally as:

$$R_{t+1}(X_0) = f(X_0, U) \tag{3}$$

where $X_0 \subseteq \mathbb{R}^n$ represents the set of initial states, $U \subseteq \mathbb{R}^m$ represents the input set. More generally, reachability analysis methods aim to construct a *conservative* flowpipe (4) which encompasses all the possible reachable sets of a dynamical system over a time-horizon $[0, T]$. This can be formalised as follows (in practice the union is computed over a discretised interval):

$$R_{[0,T]}(X_0) = \bigcup_{t\in[0,T]} R_t(X_0) \tag{4}$$

Reachability analysis has been widely used in applications that range from the formal verification of systems to problems relating to the safe synthesis of complex systems [2]. The majority of reachability analysis approaches leverage a combination of numerical analysis techniques, graph algorithms, and computational geometry [4, 3], and while in some cases it is possible to derive the exact reachable set of states, for many classes of systems computing the exact reachable set is infeasible. Thus, deriving the reachable set for these classes of systems involves obtaining a sound approximation (i.e., guarantee to

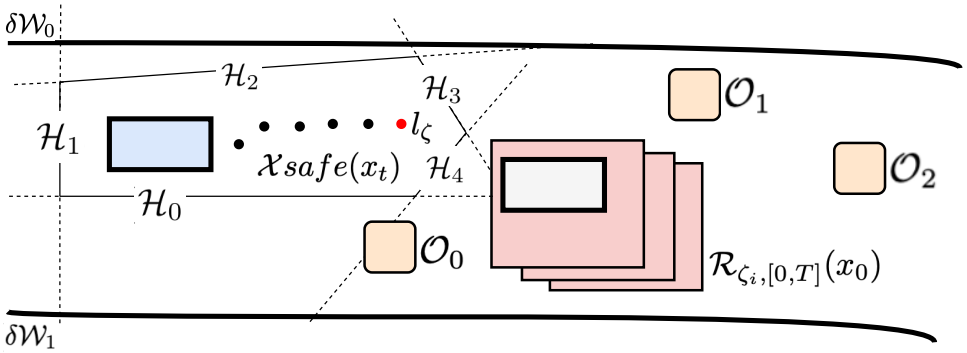


Figure 1: Visualisation of the autonomous racing problem with track boundaries, $\{\delta W_0, \delta W_1\}$, a dynamic opponent described by its reachable set $R_{\zeta_i,[0,T]}(x_0)$ and static obstacles $\{O_0, O_1, O_2\}$. In this figure, the blue rectangle corresponds to the ego vehicle, and the white rectangle corresponds to a dynamic opponent. The main sub-problem is computing an n-number of separating hyperplanes ($H_0 ... H_4$) which jointly create a polyhedron X_{safe}. The computed X_{safe} must contain an ego vehicle and its target location l_ζ as well as not overlap with observable obstacles.

contain a complete reachable set) of this set using a variety of set representations. Consequently, there is an inherent trade-off between the accuracy of the approximation and the time it takes to construct this set. We refer interested readers to the following papers [4, 3] for an in-depth discussion of these techniques.

4 Problem Statement and Space Convexification

4.1 Problem Formulation

In this paper, we consider the general autonomous racing problem (5) where a model predictive controller (2) is tasked with generating a sequence of control inputs $u_{0...T}$ that control a vehicle (1) such that it reaches the terminal state $x_T \in X_f$ starting from an initial state x_s and steering through safe states, where X_{safe} and X_f are the safe states and terminal sets respectively. The goal is to steer the vehicle into the terminal set with the shortest time horizon T.

$$\begin{aligned} &\min_{T,u_0,u_1,..u_{T-1}} p(x_T) + \sum_{k=0}^{T} q(x_k,u_k) \quad s.t. \\ &x_{t+1} = f(x_t,u_t), \quad x_0 = x_s \\ &x_t \in X_{safe}, \quad u_t \in U, \quad x_T \in X_F \end{aligned} \tag{5}$$

In our formulation, the autonomous vehicle operates within a two-dimensional environment $W \subset \mathbb{R}^2$ enclosed by boundaries $\{\delta W_0, \delta W_1, \ldots \delta W_i\}$ as $\delta W \subset W$, among a set of dynamic agents $\zeta = \{\zeta_0, \ldots, \zeta_i\}$ (ζ could be either ego ζ_e or opponent vehicle ζ_o) and static obstacles $\{O_0, O_1, \ldots, O_i\}$ with $O \subset W$. The region of space occupied by a dynamic agent $\zeta_i(x_t) \in W$ in the environment over a time interval $[t,t']$ from its current state x_t is given by its reachable set $R_{\zeta_i,[t,t']}(x_t) \subset W$. Our assumption is that the static and dynamic obstacles are contained within the two-dimensional environment. Furthermore, we refer to opponent vehicles within the racing environment as dynamic agents and refer to all other dynamic entities as dynamic obstacles.

To obtain a globally optimal solution to problem (5), as opposed to a locally optimal solution, the model-predictive control problem requires the state-space X to be convex. However, because of environment borders, static obstacles and dynamic agents, X is generally a non-convex entity. Therefore, the main sub-problem we are addressing in this paper is the computation of the safe, convex and *optimal* state-space X_{safe} in which a safe trajectory starting from x_0 to a target location $l_\zeta \in X_{safe}$ could be generated using an optimisation-based controller for the autonomous system (see Figure 1). The safe region of the state-space X_{safe} can be defined as follows:

$$X_{safe} = \{x \,|\, x \notin (\delta W \cup O \cup \bigcup_{i=0}^{N-1} R_{\zeta_i,[0,T]}(x_0))\} \tag{6}$$

where N is the number of observable dynamic agents. The computation of X_{safe} requires considering only *observable* obstacles, agents and borders. To define observable points we first introduce a notion of the LiDAR sensor which is mounted on the autonomous system and makes it possible to determine the distance to obstacles. The sensor sends M light pulses in an anti-clockwise direction around the autonomous system defined by $\delta\theta$ increments and returns a set of observational points $\{r_0(x_t), ..., r_M(x_t)\}$ where a LiDAR observational point $r_i(x_t) \in \mathbb{R}$ in the direction θ_i can be formally defined in the following way (7):

$$r_i(x_t) = \min_{O_i \in O} \min_{z \in O_i} ||z - \zeta(x_t)||_2 \quad s.t. \quad \text{atan2}(z - \zeta(x_t)) = \theta_i \tag{7}$$

Ranges of the observable LiDAR signals $r_i(x_t)$ can be converted into a two-dimensional point cloud of the W where a single point $p_i(x_t)$ of an agent $\zeta(x_t)$ can be defined as a tuple (8):

$$p_i(x_t) = (\zeta(x_t) + r_i(x_t)\cos\theta_i,\ \zeta(x_t) + r_i(x_t)\sin\theta_i) \tag{8}$$

Now, we can define observable static obstacles of $\zeta(x_t)$ as a set Q_{ob} of LiDAR points within a constant radius distance d from the agent's state $\zeta(x_t)$:

$$\begin{aligned} Q_{ob} = \{q \mid q \in \{p_0, ..., p_{M-1}\} \wedge ||q - \zeta(x_t)||_2 \leq d\} \\ d \in \mathbb{R},\ 0 < d \leq \max(r_0(x_t), ..., r_{M-1}(x_t)) \end{aligned} \tag{9}$$

Furthermore, the observable unsafe space Q_{ob} should include reachable sets of other dynamic agents $\{\zeta_0, \ldots, \zeta_i\}$. However, we are only interested in other dynamic agents which are within some distance $d \in \mathbb{R}^+$ and so we update our definition Q_{ob} to include reachable regions of other *close* dynamic agents (10):

$$Q_{ob}^+ = Q_{ob} \cup \{q \mid q \in \bigcup_{i=0}^{N-1} R_{\zeta_i,[t,t']}(x_t) \ \wedge\ ||q - \zeta_e(x_t)||_2 \leq d\} \tag{10}$$

4.2 Space Convexification via Separating Hyperplanes

This paper proposes a solution for the computation of X_{safe} which is based on the convexification of non-convex state space via separating coupled hyperplanes. A hyperplane $H = \{x \mid a^\top x = b\}$, where $a \in \mathbb{R}^n, b \in \mathbb{R}, a \neq 0$, is a set which splits set $\mathbb{R}^n$ into two halfspaces. Let us also denote H^* (11) as one

of the halfspaces of the hyperplane H. A separating hyperplane H is then said to separate two disjoint convex sets A, B such that $A \subseteq H^+$ and $B \subseteq H^-$ [7].

$$\begin{aligned} &H^* \in \{H^+, H^-\} \quad H^+ \cap H^- = H \\ &H^+ = \{x \mid a^\top x \geq b\} \quad H^- = \{x \mid a^\top x \leq b\} \end{aligned} \tag{11}$$

An intersection of finite halfspaces is a polyhedron P (12):

$$P_H = \{x \mid x \in \bigcap_{i=0}^{N-1} H_i^*\} \tag{12}$$

The idea behind a space convexification via separating coupled hyperplanes is to compute a set of hyperplanes $HS = \{H_0, ..., H_n\}$ such that together they *create* a polyhedron P_{HS} which (1) does not intersect with the set observable obstacles of the ego vehicle and (2) the ego vehicle $\zeta_e(x_t)$ with its target location l_ζ are within the polyhedron at the time t (13):

$$X_{safe} = \{x \mid x \in P_{HS} \wedge P_{HS} \cap Q_{ob}^+ = \emptyset \wedge \zeta_e(x_t) \in P_{HS} \wedge l_{\zeta(x_t)} \in P_{HS}\} \tag{13}$$

The problem of generating a set of separating coupled-hyperplanes HS can be defined as an optimisation or satisfiability problem (14) in which n number of hyperplanes are computed such that: 1) each hyperplane separates a part of observable obstacles from the ego vehicle and its target location and 2) all observable obstacles are separated by separating coupled hyperplanes

$$\begin{aligned} compute \quad &HS = \{H_0, ..., H_i\} \quad s.t. \\ &\forall q_i^{ob+} \in Q_{ob}^+ \Rightarrow \exists H_i \in HS \wedge q_i^{ob+} \in H_i^* \wedge \zeta(x_t), l_{\zeta(x_t)} \in \mathbb{R}^n \setminus H_i^* \end{aligned} \tag{14}$$

The convex and safe polyhedron X_{safe} is the intersection of halfspaces H_i^* of each hyperplane $H_i \in HS$ for which $\zeta_e(x_t) \in H_i^*$ and $l_{\zeta(x_t)} \in H_i^*$ hold. The problem (14) can be expressed as an optimisation problem on the set of hyperplanes HS or polyhedron P_{HS}. One possible *performance* metric could be finding the largest P_{HS} [13, 8].

5 Autonomous Vehicle Control System

5.1 Overview of the Closed-Loop Control System

The closed-loop control system for obstacle avoidance which we propose in this paper combines online reachability analysis and non-linear model-predictive control (visualised in Fig. 2). The control cycle can be divided into four main procedures: sensing, environment data processing and local planning, state-space convexification and solving an optimal control problem.

The control system relies on the LiDAR sensor to obtain and identify the set of observable obstacles and safe regions. We then leverage the Ramer-Douglas-Peucker algorithm [14] to simplify the observed LiDAR data and reduce the noisiness of its measurements. Doing so allows us to reduce the computation time needed to produce a set of coupled separating hyperplanes. The other sensors, namely, odometry measurements and the results of state-estimators, are used to determine the state of the ego vehicle and other agents respectively. In this work, we assume that the state of the ego vehicle and opponent agents

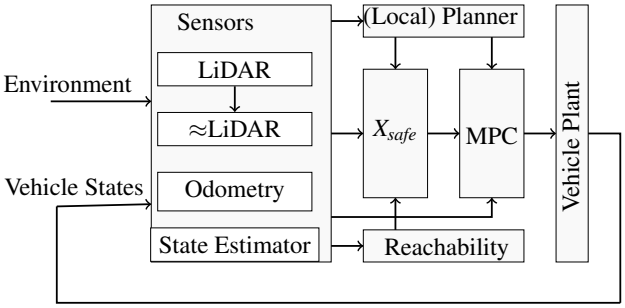


Figure 2: The architecture of the closed-loop control system for obstacle avoidance

are estimated perfectly. Therefore, we use the ground truth data provided by the simulator. This data is then passed to a (local) planner (e.g., Follow-the-Gap [43]) to select a target position. We then use reachability analysis to compute the set of reachable states for all agents within the environment.

The computation of separating coupled hyperplanes, which produces a safe and convex X_{safe}, involves using sensor information, the target location obtained from the local planner, and the set of reachable states of the dynamic agents within the environment. The hyperplanes are then passed to the model-predictive controller, together with the target location and odometry data, which solves an online optimal control problem (5) to determine the optimal inputs for the vehicle.

5.2 Computing Separating Coupled-Hyperplanes

The problem of computing separating coupled hyperplanes, which establishes a convex and safe X_{safe}, can be formulated as an optimisation (or satisfiability) problem. Thus, we present an optimisation-based method for solving (14) in order to separate the observable obstacle set Q_{ob}^{+} from the autonomous system $\zeta_e(x_t)$ and its target location $l_{\zeta_e(x_t)}$ at the state x_t.

In Algorithm 1, we describe the computation of our separating coupled-hyperplanes $H_{0..n}$. First, the set of unsafe states is included in the set Q_{ob}. The set of unsafe states consists of the set of observable obstacles from the LiDAR sensor and the reachable states of the dynamic agents. Using this set, we then make use of the state of the ego vehicle, the target location obtained from the local planner, and in the case case of the constrained optimisation method a predefined number of hyper-planes to formulate an optimisation problem. Furthermore, only obstacles (Algorithm 1 ln. 6) and reachable sets (Algorithm 1 ln. 7-8) within a distance d from the ego vehicle $\zeta_e(x_t)$ are considered in the hyperplane computation.

Constrained Optimisation Method The first approach uses a derivative-free constrained optimisation formulation which utilises a linear approximation of the objective function and optimisation constraints to solve the aforementioned optimisation problem [37]. In the optimisation problem, an individual separating hyperplane $H_n \in \{H_0, .., H_n\}$ is only *responsible* for separating a subset of ${Q^+}_{ob}$ from $\zeta_e(x_t)$ and $l(x_t)$, while the set of all hyperplanes considered should separate the vehicle from ${Q^+}_{ob}$ as a whole.

For each $q_{ob} \in Q_{ob}^{+}$ a separate constraint in the optimisation problem is defined which checks if q_{ob} is separated from the target location and autonomous system $\zeta_e(x_t)$ with some hyperplane H_n. The constrained optimisation method can use different objective functions which characterise how the set of hyperplanes is derived. For example, the optimisation problem could try minimising the distance between each H_n and its associated set of q_{ob}, or simply be expressed as a satisfiability problem with a constant objective function. We present an analysis of different optimisation objective functions for this purpose in the evaluation section.

ALGORITHM 1 The overall algorithm for the computation of separating coupled hyperplanes

Inputs: observable radius distance $d \in \mathbb{R}^+$
Inputs: states of the ego vehicle $\zeta_e(x_t) = \{x_e, y_e\}$ and other dynamic agents $\{\zeta_0, \ldots, \zeta_N\}$
Inputs: static obstacle data $P = \{p_0, ..., p_{N-1}\}$ from the LiDAR (ranges Equation (7))
  Compute target states of the ego vehicle with the local planner $l(x_t) = \{x_t, y_t\}$
  Compute reachable states $R = \bigcup_{i=0}^{N} R_i$ of observable dynamic agents $\{\zeta_0, \ldots, \zeta_N\}$
  Compute $Q_{ob}(x_t)$ by using static obstacle LiDAR data $\{q \mid q \in P \wedge \|q - \zeta_e(x_t)\|_2 \leq d\}$
  Compute $Q^+_{ob}(x_t)$ by combining static and dynamic obstacles $Q_{ob}(x_t) \cup R$
  Encode $q \in Q^+_{ob}, x \in \zeta_e(x_t), x \in l(x_t)$ as constraints of the optimisation problem and solve by using the constrained or bi-level optimisation method
Output: $\{H_0, \ldots, H_n\}$

Bi-level Optimisation Method The problem defined in (14) can also be encoded as a bi-level optimisation problem in (15). The problem is similar to one solved by Deits and Tedrake [11] except we are interested in computing a polygon defined by a minimum number of hyperplanes which contains the largest ellipsoids in the direction of the target location. The [11] maximises ellipsoid in any possible direction, which is not suitable for the racing context, as the most optimal trajectories produced by the MPC will most likely be along the ego vehicle to the target corridor.

The outer part of the problem computes the minimum set of separating hyperplanes (the size of the A matrix's diagonal) that separate obstacle points Q^+_{ob} from the ego vehicle and its target location. The inner part of the bi-level optimisation solves the Chebyshev centre [7] problem[1] by finding the centre q of the largest inscribable ellipsoid with radius R.

$$
\begin{aligned}
&\arg\min_{A,b} ||\mathbf{diag}(A)|| \text{ s.t.} \\
&\quad Aq \geq b,\ \forall q \in Q^+_{ob} \\
&\quad Ax \leq b,\ \forall x \in \zeta_e(x_t) \cup l(x_t) \\
&\qquad \arg\max_{q,R} R \text{ s.t.} \\
&\qquad\quad a_j q + ||A|| R \leq b_j \\
&\qquad\quad R \geq 0
\end{aligned}
\tag{15}
$$

5.3 Reachability Analysis of Dynamic Obstacles

To perform reachability analysis, we first identify a dynamical model of the vehicle and assume models for the dynamic obstacles within its environment.

5.3.1 Dynamic Obstacle Model

The obstacle-tracking problem is a well-studied and challenging topic within the autonomous vehicle, computer vision, and robotics literature [44]. Typically, some assumptions are required in order to con-

[1]The reason for maximising the largest inscribable ellipsoid in contrast to directly maximising the area of the safe polyhedron X_{safe} is efficiency. There are no efficient methods for computing the area of irregular polyhedrons, while the Chebyshev centre problem can be solved sufficiently fast.

strain the tracking problem to suit the context of the application. In our framework, we assume that the dynamic obstacles are described by a two-dimensional kinematic model and a corresponding bounding box. The equations describing the kinematic model are given as follows:

$$\dot{x} = v_x, \dot{y} = v_y$$

where v_x and v_y are the velocities in the x and y direction, respectively. Additionally, we make the assumption that we have access to the position and velocity of the other race participants.

While it is possible to use more sophisticated models to describe the behaviour of the dynamic obstacles within the vehicle's environment, for simplicity we selected a two-dimensional kinematic model. However, it is worth noting that there has been a growth in approaches that perform online parameter estimation for dynamic obstacles within a robot's environment through online system identification [15].

5.3.2 Online Reachability Computation

Using the dynamics models obtained in the previous sections, the crux of the real-time reachability algorithm is computing the set of reachable states $R_{[0,T]}(X_0)$ over a finite time horizon. The algorithm utilised within this work is based on mixed face-lifting, which is part of a class of methods that deal with *flow-pipe construction* or *reachtube computation* [17]. This is done using snapshots of the set of reachable states that are enumerated at successive points in time, as outlined in Equation (3).

In general, it is not possible to obtain the exact reachable set $R_{[0,T]}(X_0)$, so we compute an over-approximation such that the actual system behaviour is contained within the over-approximation [24]. The algorithm utilised in this work utilises n-dimensional hyper-rectangles ("boxes") as the set representation to generate reachtubes [17]. Over long reach-times, the over-approximation error resulting from the use of this representation can be problematic. However, for short reach-times it is ideal in terms of its simplicity and speed [6].

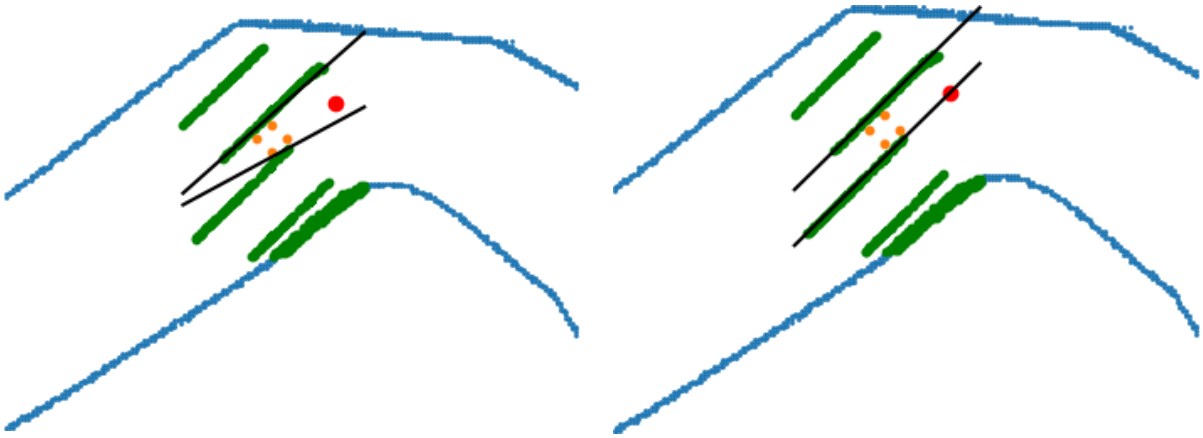
Traditionally, reachability approaches have been executed offline because they are computationally intensive endeavours. However, in [6, 17], Bak et al. and Johnson et al. presented a reachability algorithm, based on the seminal mixed face-lifting algorithm [10], capable of running in real-time on embedded processors. The algorithm is implemented as a standalone C-package that does not rely on sophisticated (non-portable) libraries, recursion, or dynamic data structures and is amenable to the anytime computation model in the real-time scheduling literature. In this regime, each task produces a partial result that is improved upon as more computation time is available, known as an anytime algorithm [17]. We refer readers to the following papers for an in-depth treatment of these procedures [10, 6, 17].

6 Evaluation

In this section, we present a runtime analysis of proposed algorithms for computing separating coupled hyperplanes and an evaluation of the overall control system by using the F1TENTH simulation platform. In the following section, we first describe an optimisation-free method (MPCC) for computing separating hyperplanes, which will be used to compare against our proposed approaches.

6.1 MPCC Optimisation-free Hyperplane Approach

In [25] Liniger et al. tackled the autonomous racing problem via a nonlinear MPC problem that encoded the obstacle avoidance problem by means of a high-level corridor planner based on dynamic programming. The safe corridor that their framework utilised was constructed by projecting the points along



(a) Constrained Optimisation Method

(b) MPCC approach

Figure 3: A snapshot of the artificial overtaking scenario with two opponents represented: combined observable static and dynamic obstacles Q_{ob}^{+} (green points), corners of the ego vehicle (orange points), target point (red point), computed hyperplanes (black) and the boundaries of the racetrack (blue).

the centre line of the track onto the racetrack borders (one for the left border, and one for the right border). Their regime demonstrated success in controlling 1/43 scale race cars, driven at speeds of more than 3 m/s using controllers executing at 50 Hz sampling rate on embedded computing platforms [25]. While their evaluation was limited to environments with static obstacles, we experimented with using such a scheme to obtain the separating hyperplanes framing our MPC problem. We refer readers to the following paper for an in-depth discussion of their approach [25].

6.2 Offline Analysis of Convexification Algorithms

Deploying optimisation-based methods into real-time autonomous control systems requires careful consideration of timing constraints issued by the optimisation method. The computation time of the proposed methods can be affected by the number of obstacle points being considered or in the case of the constrained optimisation method by the selected optimisation cost function. In these experiments, we aim to evaluate the *quality* of separating hyperplanes generated by different approaches and the computation time. The former is assessed by inscribing the largest circle with radius R between generated hyperplanes (the centre of the circle must be between the ego vehicle and its target location) and gives a reasonable size approximation of the X_{safe} in the travelling direction of ego vehicle. In the first set of experiments, we traversed the ego vehicle along a predefined path on one of the two racetracks: Porto (see Figure 5) and Walker without other dynamic agents, and then considered an artificially created overtaking scenario with one and two opponents (visualised in Figure 3). The results of this experiment are summarised in Table 1. The MPCC approach is clearly more time efficient compared to our proposed approaches as it is not an optimisation-based approach. However, it produces a smaller average inscribed circle radius (i.e., smaller X_{safe}), particularly, in the overtaking scenarios with up to 16 per cent smaller R in comparison to the largest averaged R. The bi-level optimisation approach is around tenfold faster than the constrained optimisation approach, as its outer problem is a linear programming problem, which can be efficiently solved even with a larger number of obstacles (problem constraints). However, our experiments show that the bi-level optimisation method does not always produce the largest X_{safe} and in some

cases generates more than two hyperplanes, which would negatively affect solving the MPC problem.

In the second experiment, we increased the number of generated (randomly positioned) obstacle points around an ego vehicle to evaluate our method's scalability with respect to a larger number of obstacles. For the constrained optimisation method we considered three types of objective functions: Hausdorff, Euclidean distances and a satisfiability problem which only requires satisfying optimisation constraints. For each objective function and bi-level optimisation, the number of obstacle points varied from 10 to 2000. The evaluation results are visually shown in Figure 4.

Experiment Scenario	Approach	H	Time (s)	R
Porto (w/o obstacles)	MPCC	2	$\mathbf{6.46 \times 10^{-5}}$	1.29
Porto (w/o obstacles)	Constrained Optimisation	2	0.132	**1.30**
Porto (w/o obstacles)	Bi-level Optimisation	2.16	0.019	1.072
Walker (w/o obstacles)	MPCC	2	$\mathbf{7.65 \times 10^{-5}}$	0.702
Walker (w/o obstacles)	Constrained Optimisation	2	0.12	0.675
Walker (w/o obstacles)	Bi-level Optimisation	2.26	0.019	**0.715**
Overtaking (1 opponent)	MPCC	2	$\mathbf{8.679 \times 10^{-5}}$	0.627
Overtaking (1 opponent)	Constrained Optimisation	2	0.18	0.661
Overtaking (1 opponent)	Bi-level Optimisation	2.13	0.022	**0.755**
Overtaking (2 opponents)	MPCC	2	9.79×10^{-5}	0.465
Overtaking (2 opponents)	Constrained Optimisation	2	0.265	**0.488**
Overtaking (2 opponents)	Bi-level Optimisation	2.59	0.026	0.473

Table 1: Offline evaluation of different methods for computing separating hyperplanes with average computation time, an average inscribed radius R and an average number of hyperplanes H.

6.3 Real-Time Control System Evaluation

Our real-time evaluation of the overall control system (MPC Hype) includes a diverse set of experiments that include changing the number of racing agents present within the racetrack, including additional dynamic obstacles within the racetrack, adding static obstacles onto the racetrack, and changing the racing environment. We compare the performance of our approach against a set of controllers typically utilized within the F1TENTH racing competitions with respect to two metrics that we refer to as *efficiency*, and *safety*. *Efficiency* is the total distance that the F1TENTH vehicle traverses around the track divided by the amount of time it took to do so.[2] *Safety* corresponds to the controller's ability to avoid collisions over a set of experimental runs (i.e., 10 collisions in 20 experiments corresponds to a safety score of 50%). The following controllers were utilised as a local planning mechanism for selecting the target point used in our MPC regime. Additionally, we utilised them as a baseline comparison for our approach.

Pure Pursuit The Pure Pursuit algorithm is a widely used path-tracking algorithm that was originally designed to calculate the arc needed to get a robot back onto a path [9]. It has shown great success in being used in numerous contexts, and in this work, we utilise it to design a controller that allows the F1TENTH vehicle to follow a path along the centre of the racetrack.

[2]This is equivalent to the average speed attained during the experiment.

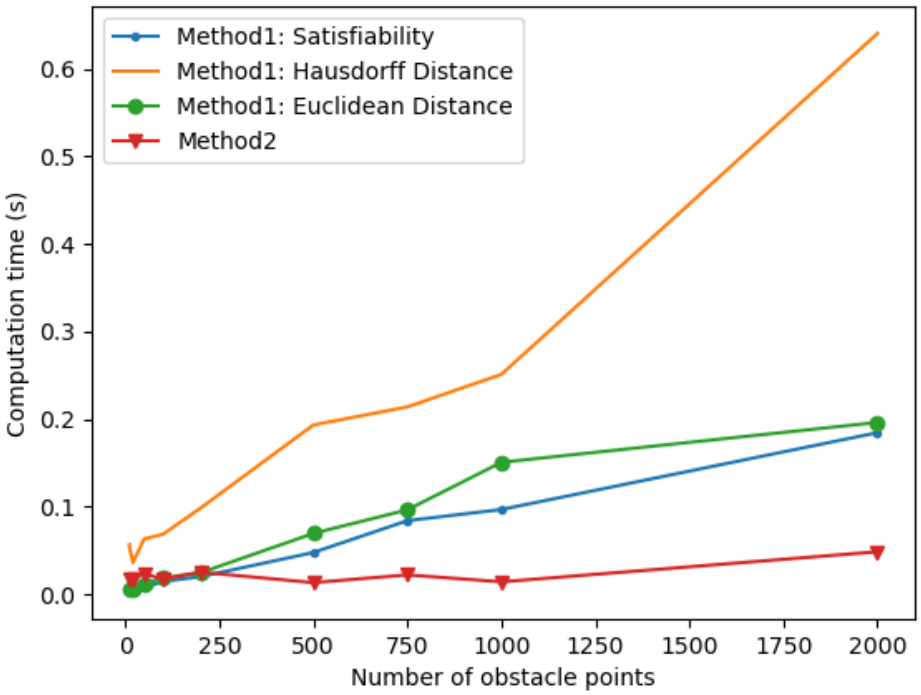


Figure 4: Offline evaluation of separating coupled hyperplane computation time against different numbers of obstacle points (optimisation constraints), different objective functions (method 1 - constrained optimisation approach, method 2 - bi-level optimisation).

Gap Following Obstacle avoidance is an essential component of a successful autonomous racing strategy. Gap following approaches have shown great promise in dealing with dynamic and static obstacles. They are based on the construction of a gap array around the vehicle used for calculating the best heading angle needed to move the vehicle into the centre of the maximum gap [34]. In this work, we utilise a gap following controller called the "disparity extender" by Otterness et al. that won the F1TENTH competition in April of 2019 [36].

Our evaluation included a sizeable diversity of experiments with respect to the number of vehicles present in the racing environment, the presence of static and dynamic obstacles, the racetrack used for the autonomous race, the local planner chosen to select goal points, and the method selected to obtain the separating hyperplanes. Each configuration was evaluated over 30 experimental runs of 60 seconds. Table 2 displays the results of experiments with two and three cars respectively (separated by horizontal line) on a single track without static obstacles (a screenshot of the two agent experiment is shown in Figure 5). In the table that follows, DE corresponds to the disparity extender, PP corresponds to pure pursuit, MPCC corresponds to the approach presented by Liniger et al. [25], and MPC Hype corresponds to the optimisation-based approach presented in this document. Finally, Race Duration corresponds to the amount of time the agents were able to race before a collision occurred.

The results from the experiments suggest that our proposed control system (MPC Hype) can increase autonomous vehicle safety without loss of efficiency (compared to MPCC), especially when the num-

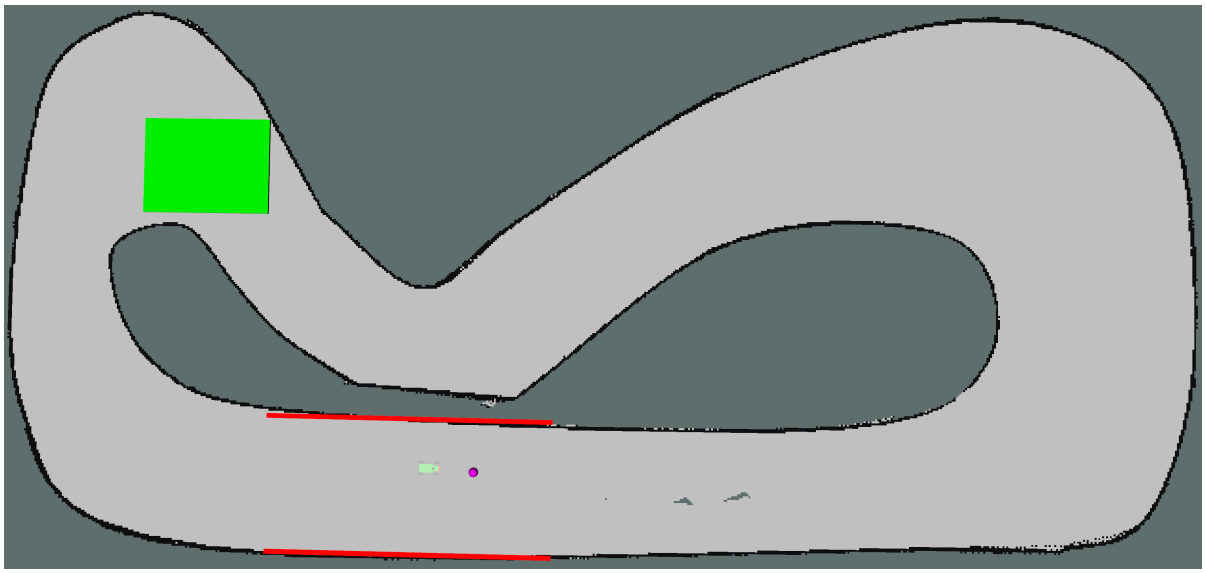


Figure 5: An example of a two-agent racing scenario. The bright green rectangle, represents the reachable set (convex hull) of the opponent vehicle over a $t = 0.5$ second time horizon, while the faded green vehicle represents the ego vehicle. The purple dot corresponds to the target location obtained from the local planner. The red lines are the two parallel half spaces that approximate the traversable region within the racetrack.

Table 2: Performance summary of two-car experiments (without static obstacles): DE (Disparity Extender), PP (Pure Pursuit), MPCC (Model-Predictive Control with Contouring) and MPC Hype (our control system with the constrained optimisation hyperplane computation)

Track	Approach	Local Planner	Ego Efficiency (m/s)	Opponent Efficiency (m/s)	Race Duration (s)	Safety (%)
Porto	DE	DE	5.29	4.65	51.57	38.33
Porto	MPC Hype	DE	0.00	5.27	5.53	0.00
Porto	MPC Hype	PP	3.06	5.18	25.74	13.33
Porto	MPCC	DE	3.00	4.97	7.12	20.00
Porto	MPCC	PP	3.00	5.34	55.14	46.67
Porto	PP	PP	4.70	5.33	60.0	100.00
Porto	DE	DE	5.38	4.10	33.78	28.33
Porto	MPC Hype	DE	1.19	4.50	5.40	0.00
Porto	MPC Hype	PP	2.75	2.96	43.26	30.00
Porto	MPCC	DE	1.66	4.23	5.39	3.33
Porto	MPCC	PP	1.83	4.00	5.37	16.67
Porto	PP	PP	4.70	3.73	57.30	70.00

ber of opponent vehicles is increased. However, results also suggest that the performance of our MPC implementation could be further improved, for example, by improving the MPC cost function to generate better speed profiles in corners. This would also provide us with more evidence of the hyperplane approach when ego velocity is increased. Furthermore, our experimentation setup did not differentiate between different types of collisions, for example, collisions, where the opponent vehicle collided with the back of the ego vehicle and the reverse situation, were treated equally (i.e., counted the same in the safety metric). A more nuanced safety metric with a *blame* factor would provide a better understanding

of our control system performance.

7 Conclusions and Future Work

This paper presented an optimisation-based approach for static and dynamic obstacle avoidance problems within an autonomous vehicle racing context. Our control regime leveraged online reachability analysis and sensor data to compute the maximal safe traversable region that an agent can traverse within the environment. We described a technique for computing a convex safe region via a novel coupled separating hyperplane algorithm. This derived safe area was then used to formulate a nonlinear model-predictive control problem that sought to find an optimal and safe driving trajectory with varying degrees of efficacy. Our experimental evaluation demonstrated that our approach was feasible as an obstacle avoidance strategy. Finally, we assessed the runtime requirements of our proposed approach by analysing the effects of a set of varying optimisation objectives for generating these coupled hyperplanes.

There are a number of future work directions we would like to explore. Firstly, our study did not consider uncertainty in sensors, our future work will seek to include uncertainties arising from the state estimation of opponent vehicles in their reachable set computation. Secondly, future studies would include an analysis against hierarchical control architectures that decompose the obstacle avoidance problem into planning and trajectory tracking. Lastly, we would like to evaluate the proposed approach on the physical F1TENTH platform in order to validate further that our approach admits low resource requirements.

References

[1] M. Althoff & J. M. Dolan (2014): *Online Verification of Automated Road Vehicles Using Reachability Analysis*. *IEEE Transactions on Robotics* 30(4), pp. 903–918, doi:10.1109/TRO.2014.2312453.

[2] Matthias Althoff, Goran Frehse & Antoine Girard (2021): *Set Propagation Techniques for Reachability Analysis*. *Annual Review of Control, Robotics, and Autonomous Systems* 4(1), pp. 369–395, doi:10.1146/annurev-control-071420-081941.

[3] Eugene Asarin, Thao Dang, Goran Frehse, Antoine Girard, Colas Le Guernic & Oded Maler (2007): *Recent progress in continuous and hybrid reachability analysis*. *Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design, CACSD*, pp. 1582–1587, doi:10.1109/CACSD-CCA-ISIC.2006.4776877.

[4] Eugene Asarin, Thao Dang & Antoine Girard (2003): *Reachability Analysis of Nonlinear Systems Using Conservative Approximation*. In Oded Maler & Amir Pnueli, editors: *Hybrid Systems: Computation and Control*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 20–35, doi:10.1007/3-540-36580-X_5.

[5] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani & C. J. Tomlin (2019): *An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments*. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 1758–1765, doi:10.1109/CDC40024.2019.9030133.

[6] S. Bak, T. T. Johnson, M. Caccamo & L. Sha (2014): *Real-Time Reachability for Verified Simplex Design*. In: *2014 IEEE Real-Time Systems Symposium*, pp. 138–148, doi:10.1109/RTSS.2014.21.

[7] Stephen Boyd & Lieven Vandenberghe (2004): *Convex Optimization*. Cambridge University Press, doi:10.1017/CBO9780511804441.

[8] J. S. Chang & C. K. Yap (1986): *A Polynomial Solution for the Potato-Peeling Problem*. *Discrete Comput. Geom.* 1(2), p. 155–182, doi:10.1007/BF02187692.

[9] R. Craig Coulter (1992): *Implementation of the Pure Pursuit Path Tracking Algorithm*. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA.

[10] Thi Xuan Thao Dang (2000): *Verification and Synthesis of Hybrid Systems*. Theses, Institut National Polytechnique de Grenoble - INPG.

[11] Robin Deits & Russ Tedrake (2015): *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*, pp. 109–124. Springer International Publishing, Cham, doi:10.1007/978-3-319-16595-0_7.

[12] Huixu Dong, Ching-Yen Weng, Chuangqiang Guo, Haoyong Yu & I-Ming Chen (2021): *Real-Time Avoidance Strategy of Dynamic Obstacles via Half Model-Free Detection and Tracking With 2D Lidar for Mobile Robots*. *IEEE/ASME Transactions on Mechatronics* 26(4), pp. 2215–2225, doi:10.1109/TMECH.2020.3034982.

[13] Reza Dorrigiv, Stephane Durocher, Arash Farzan, Robert Fraser, Alejandro López-Ortiz, J. Ian Munro, Alejandro Salinger & Matthew Skala (2009): *Finding a Hausdorff Core of a Polygon: On Convex Polygon Containment with Bounded Hausdorff Distance*. In Frank Dehne, Marina Gavrilova, Jörg-Rüdiger Sack & Csaba D. Tóth, editors: *Algorithms and Data Structures*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 218–229, doi:10.1007/978-3-642-03367-4_20.

[14] David H Douglas & Thomas K Peucker (1973): *Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature*. *Cartographica: The International Journal for Geographic Information and Geovisualization* 10(2), pp. 112–122, doi:10.3138/FM57-6770-U75U-7727.

[15] Gowtham Garimella, Matthew Sheckells & Marin Kobilarov (2017): *Robust obstacle avoidance for aerial platforms using adaptive model predictive control*. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5876–5882, doi:10.1109/ICRA.2017.7989692.

[16] Gabriel Hartmann, Zvi Shiller & Amos Azaria (2021): *Autonomous Head-to-Head Racing in the Indy Autonomous Challenge Simulation Race*. *CoRR* abs/2109.05455. arXiv:2109.05455.

[17] Taylor T. Johnson, Stanley Bak, Marco Caccamo & Lui Sha (2016): *Real-Time Reachability for Verified Simplex Design*. *ACM Trans. Embed. Comput. Syst.* 15(2), doi:10.1109/RTSS.2014.21.

[18] Chanyoung Jung, Seungwook Lee, Hyunki Seong, Andrea Finazzi & David Hyunchul Shim (2021): *Game-Theoretic Model Predictive Control with Data-Driven Identification of Vehicle Model for Head-to-Head Autonomous Racing*. *CoRR* abs/2106.04094, doi:10.48550/arXiv.2106.04094.

[19] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli & Seth Teller (2011): *Anytime Motion Planning using the RRT**. In: *2011 IEEE International Conference on Robotics and Automation*, pp. 1478–1483, doi:10.1109/ICRA.2011.5980479.

[20] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen & Lipika Deka (2015): *Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions*. *Transportation Research Part C: Emerging Technologies* 60, pp. 416–442, doi:10.1016/j.trc.2015.09.011.

[21] Mark Keil & Jack Snoeyink (2002): *On The Time Bound For Convex Decomposition Of Simple Polygons*. *International Journal of Computational Geometry & Applications* 12(03), pp. 181–192, doi:10.1142/S0218195902000803.

[22] Shivesh Khaitan, Qin Lin & John M. Dolan (2021): *Safe Planning and Control Under Uncertainty for Self-Driving*. *IEEE Transactions on Vehicular Technology* 70(10), pp. 9826–9837, doi:10.1109/TVT.2021.3108525.

[23] Karen Leung, Edward Schmerling, Mengxuan Zhang, Mo Chen, John Talbot, J Christian Gerdes & Marco Pavone (2020): *On Infusing Reachability-Based Safety Assurance within Probabilistic Planning Frameworks for Human-Robot Vehicle Interactions*. *The International Journal of Robotics Research* 39(10-11), pp. 1326–1345, doi:10.1177/0278364920950795.

[24] Qin Lin, Xin Chen, Aman Khurana & John Dolan (2020): *ReachFlow: An Online Safety Assurance Framework for Waypoint-Following of Self-driving Cars*. In: *International Conference on Intelligent Robots and systems (IROS)*, IROS'2020, IEEE, doi:10.1109/IROS45743.2020.9341122.

[25] Alexander Liniger, Alexander Domahidi & Manfred Morari (2014): *Optimization-based autonomous racing of 1:43 scale RC cars*. *Optimal Control Applications and Methods* 36(5), p. 628–647, doi:10.1002/oca.2123.

[26] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark W. Barrett & Mykel J. Kochenderfer (2019): *Algorithms for Verifying Deep Neural Networks*. CoRR abs/1903.06758, doi:10.48550/arXiv.1903.06758. arXiv:1903.06758.

[27] Changliu Liu, Chung-Yen Lin & Masayoshi Tomizuka (2017): *The Convex Feasible Set Algorithm for Real Time Optimization in Motion Planning*, doi:10.48550/ARXIV.1709.00627. Available at `https://arxiv.org/abs/1709.00627`.

[28] Jiechao Liu, Paramsothy Jayakumar, Jeffrey L. Stein & Tulga Ersal (2018): *A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments*. Vehicle System Dynamics 56(6), pp. 853–882, doi:10.1080/00423114.2017.1399209.

[29] Joseph Lorenzetti, Mo Chen, Benoit Landry & Marco Pavone (2018): *Reach-Avoid Games Via Mixed-Integer Second-Order Cone Programming*. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 4409–4416, doi:10.1109/CDC.2018.8619382.

[30] Tim Mercy, Wannes Van Loock & Goele Pipeleers (2016): *Real-time motion planning in the presence of moving obstacles*. In: 2016 European Control Conference (ECC), pp. 1586–1591, doi:10.1109/ECC.2016.7810517.

[31] Tim Mercy, Ruben Van Parys & Goele Pipeleers (2018): *Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment*. IEEE Transactions on Control Systems Technology 26(6), pp. 2182–2189, doi:10.1109/TCST.2017.2739706.

[32] Siti Hajar Ashikin Mohammad, Muhammad Akmal Jeffril & Nohaidda Sariff (2013): *Mobile robot obstacle avoidance by using Fuzzy Logic technique*. In: 2013 IEEE 3rd International Conference on System Engineering and Technology, pp. 331–335, doi:10.1109/ICSEngT.2013.6650194.

[33] Matthew O'Kelly, Varundev Sukhil, Houssam Abbas, Jack Harkins, Chris Kao, Yash Vardhan Pant, Rahul Mangharam, Dipshil Agarwal, Madhur Behl, Paolo Burgio & Marko Bertogna (2019): *F1/10: An Open-Source Autonomous Cyber-Physical Platform*. CoRR abs/1901.08567, doi:10.48550/arXiv.1901.08567. arXiv:1901.08567.

[34] Matthew O'Kelly, Hongrui Zheng, Dhurv Karthik & Rahul Mangharam (2020): *F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning*. In Hugo Jair Escalante & Raia Hadsell, editors: Post Proceedings of the NeurIPS 2019 Demonstration and Competition Track, Proceedings of Machine Learning Research, PMLR.

[35] Michael Otte & Emilio Frazzoli (2016): *RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning*. The International Journal of Robotics Research 35(7), pp. 797–822, doi:10.1177/0278364915594679.

[36] Nathan Otterness (2019): *The "Disparity Extender" Algorithm, and F1/Tenth*. Available at `https://www.nathanotterness.com/2019/04/the-disparity-extender-algorithm-and.html`.

[37] M. J. D. Powell (2007): *A View of Algorithms for Optimization without Derivatives*. Technical Report DAMTP 2007/NA03, University of Cambridge.

[38] Ugo Rosolia & Francesco Borrelli (2019): *Learning How to Autonomously Race a Car: a Predictive Control Approach*. arXiv:1901.08184.

[39] Ugo Rosolia, Stijn De Bruyne & Andrew G. Alleyne (2017): *Autonomous Vehicle Control: A Nonconvex Approach for Obstacle Avoidance*. IEEE Transactions on Control Systems Technology 25(2), pp. 469–484, doi:10.1109/TCST.2016.2569468.

[40] Tobias Schoels, Luigi Palmieri, Kai Oliver Arras & Moritz Diehl (2019): *An NMPC Approach using Convex Inner Approximations for Online Motion Planning with Guaranteed Collision Freedom*. CoRR abs/1909.08267, doi:10.48550/arXiv.1909.08267.

[41] Eelco Scholte & Mark E. Campbell (2008): *Robust Nonlinear Model Predictive Control With Partial State Information*. IEEE Transactions on Control Systems Technology 16(4), pp. 636–651, doi:10.1109/TCST.2007.912120.

[42] Hiroya Seki, Satoshi Ooyama & Morimasa Ogawa (2002): *Nonlinear Model Predictive Control Using Successive Linearization*. Transactions of the Society of Instrument and Control Engineers 38, pp. 61–66, doi:10.1109/AIM.2017.8014275.

[43] Volkan Sezer & Metin Gokasan (2012): *A novel obstacle avoidance algorithm: "Follow the Gap Method"*. Robotics and Autonomous Systems 60(9), pp. 1123–1134, doi:10.1016/j.robot.2012.05.021.

[44] Alper Yilmaz, Omar Javed & Mubarak Shah (2006): *Object Tracking: A Survey*. ACM Comput. Surv. 38(4), p. 13–es, doi:10.1145/1177352.1177355.

[45] Yang Zeqing, Liu Libing, Tan Zhihong & Liu Weiling (2008): *Application of Adaptive Genetic Algorithm in flexible inspection path planning*. In: 2008 27th Chinese Control Conference, pp. 75–80, doi:10.1109/CHICC.2008.4605656.

[46] Yalong Zhang, Zhenghua Liu & Le Chang (2017): *A new adaptive artificial potential field and rolling window method for mobile robot path planning*. In: 2017 29th Chinese Control And Decision Conference (CCDC), pp. 7144–7148, doi:10.1109/CCDC.2017.7978472.

[47] Liu Zhiyang & Jiang Tao (2017): *Route planning based on improved artificial potential field method*. In: 2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), pp. 196–199, doi:10.1109/CCDC52312.2021.9602174.