2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

An NMPC Approach using Convex Inner Approximations for Online Motion Planning with Guaranteed Collision Avoidance

Tobias Schoels[1,2], Luigi Palmieri[2], Kai O. Arras[2], and Moritz Diehl[1]

***Abstract*— Even though mobile robots have been around for decades, trajectory optimization and continuous time collision avoidance remain subject of active research. Existing methods trade off between path quality, computational complexity, and kinodynamic feasibility. This work approaches the problem using a nonlinear model predictive control (NMPC) framework, that is based on a novel convex inner approximation of the collision avoidance constraint. The proposed Convex Inner ApprOximation (CIAO) method finds kinodynamically feasible and continuous time collision free trajectories, in few iterations, typically one. For a feasible initialization, the approach is guaranteed to find a feasible solution, i.e. it preserves feasibility. Our experimental evaluation shows that CIAO outperforms state of the art baselines in terms of planning efficiency and path quality. Experiments show that it also efficiently scales to high-dimensional systems. Furthermore real-world experiments demonstrate its capability of unifying trajectory optimization and tracking for safe motion planning in dynamic environments.**

I. INTRODUCTION

Several existing mobile robotics applications (e.g. intralogistic and service robotics) require robots to operate in dynamic environments among other agents, such as humans or other autonomous systems. In these scenarios, the reactive avoidance of unforeseen dynamic obstacles is an important requirement. Combined with the objective of reaching optimal robot behavior, this poses a major challenge for motion planning and control and remains subject of active research.

Recently several researchers have tackled the obstacle avoidance problem by formulating and solving optimization problems [1]–[14]. This approach is well suited for finding locally optimal solutions, but generally gives no guarantee of finding the global optimum. Most methods therefore rely on the initialization by an asymptotically optimal sampling-based planner [15]–[17]. A shortcoming of most common trajectory optimization methods is that they are incapable of respecting kinodynamic constraints, e.g. bounds on the acceleration, and typically lack a notion of time in their predictions, [1]–[4]. These approaches are typically limited to the optimization of paths rather than trajectories and impose constraints by introducing penalties.

[1]T. Schoels and M. Diehl are with the Department of Microsystems Engineering, University of Freiburg. {tobias.schoels, moritz.diehl}@imtek.uni-freiburg.de.

[2]T. Schoels, L. Palmieri and K. O. Arras are with Robert Bosch GmbH, Corporate Research, Stuttgart, Germany. {tobias.schoels, luigi.palmieri, kaioliver.arras}@de.bosch.com.

This research was supported by the German Federal Ministry for Economic Affairs and Energy (BMWi) via eco4wind (0324125B) and DyConPV (0324166B), by DFG via Research Unit FOR 2401, and the EU's Horizon 2020 research and innovation program under grant agreement No 732737 (ILIAD).

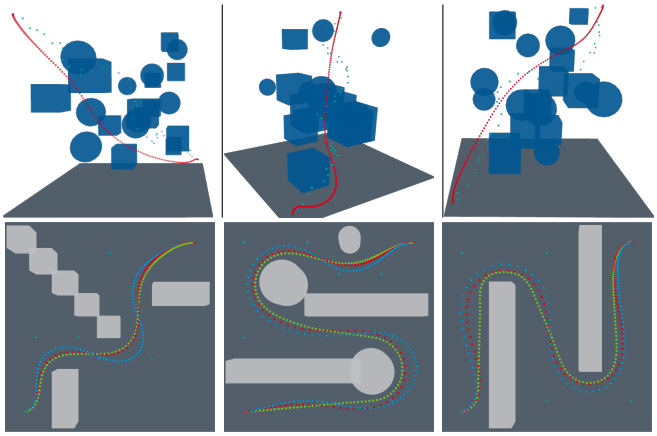


Fig. 1: CIAO trajectories for the Astrobee robot (top row) in red and for a unicycle robot (last row) with three different maximum speeds $v_{\max}$ and corresponding minimum distances $\underline{d}$ (see (7)): green - slow, red - normal, blue - fast. The boxes and spheres represent obstacles, the turquoise dots the reference path. A wider spacing between the dots indicates a higher speed. The start is always located in the bottom and the goal in the top.

The increase of computing power and the availability of fast numerical solvers, as discussed in [18], has given rise to model predictive control (MPC) based approaches, e.g. [3]–[7]. In this framework, an optimal control problem (OCP) is solved in every iteration. These methods succeed in finding kinodynamically [5]–[7] or kinematically [3]–[5] feasible trajectories, but typically use penalty terms in the cost function that offer no safety guarantees [7] or require that obstacles are given as a set of convex hulls [4]. This work presents CIAO, a nonlinear MPC (NMPC) based approach to real-time collision avoidance for single body robots. It preserves feasibility across iterations and uses a novel, convex formulation of the collision avoidance constraint that is compatible with many implementations of the distance function, even discrete ones like distance fields. To the best of authors' knowledge, CIAO is the first real-time capable NMPC approach that guarantees continuous time collision free trajectories and is agnostic of the distance function's implementation. The method's efficacy is demonstrated and evaluated in simulation and real-world using robots with nonlinear, constrained dynamics and state of the art baselines.

II. RELATED WORK

Classical approaches to obstacle avoidance, such as [19]–[23], do neither produce optimal trajectories, nor unify planning and control, nor account for complex robot dynamics.

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

A simple and effective method that is still used in practice, is the elastic-band algorithm [1]. The computed paths, however, are generally non-smooth, i.e. they are not guaranteed to satisfy kinodynamic constraints, nor is a velocity profile computed. Like [9], our approach aims to fix this shortcoming, while building up on the notion of (circular) free regions. But instead of optimizing velocity profile and path separately, we optimize them jointly utilizing an NMPC setup. Like [10] we combine elastic-band with an optimization algorithm, but enforce obstacle avoidance as constraint and provide a method for joint trajectory optimization and tracking.

Popular optimization based methods include CHOMP [3], TrajOpt [4], OBCA [6], and GuSTO [7], which have been shown to find smooth trajectories efficiently. They typically use a simplified system model to compute a path from the current to the goal state (or set), and rely on an additional controller to steer the system along the precomputed path. The proposed method, CIAO, is MPC-based and provides algorithms for both trajectory optimization and receding horizon control (RHC), simultaneously controlling the robot and optimizing its trajectory in real-time.

Frasch et al. [11] propose an MPC setup with boxconstraints to model obstacles and road boundaries. Liniger et al. [12] handle obstacles in a similar way, but apply contouring control. These frameworks do not consider arbitrarily placed obstacles, particularly no moving obstacles, which makes them unsuitable for many applications. In our approach we handle more complex obstacle definitions, modeled according to a generic nonlinear and nonconvex distance function. Also [5], [14] propose RHC to unify trajectory optimization and tracking, but their approaches does not include a strategy for obstacle avoidance as in CIAO.

The recently proposed method GuSTO [7] uses sequential convex programming (SCP), like other common algorithms, e.g. [4], [10]–[12]. SCP requires a full convexification of the originally nonlinear and nonconvex trajectory optimization problem. This is accomplished by linearizing the system model and incorporating paths constraints, including collision avoidance, as penalties in the objective function. In general these approximations may lead to infeasible, i.e. colliding or kinodynamically intractable trajectories. Typically several SCP iterations are required to find a feasible solution. CIAO, on the other hand, solves partially convexified nonlinear programs (NLPs), using a convex inner approximation of the collision avoidance constraint, and finds feasible solutions in less iterations, typically one. The individual iterations are computationally cheaper and feasibility is preserved. Since the dynamical model is accounted for by the NLP-solver, linearization errors are minimized.

Finally CIAO can be considered as a trust region method [24], where the nonlinear state constraints are approximated with a convex inner approximation.

III. PROBLEM FORMULATION

We want to find a kinodynamically feasible, collision free trajectory by formulating and solving a constrained OCP. Kinodynamic feasibility is ensured by using a dynamical model to simulate the robot's behavior and collision avoidance is achieved constraining the robot to positions with a *minimum distance* $\underline{d}$ to all obstacles.

The *occupied set* $\mathscr{O}$ is defined as the set of points in the robot's workspace $\mathscr{W} \subseteq \mathbb{R}^n$ that are occupied by obstacles. The Euclidean distance to the closest obstacle for any point $\mathbf{p} \in \mathscr{W}$ is given by the *distance function* $d_{\mathscr{O}} : \mathscr{W} \to \mathbb{R}$:

$$d_{\mathscr{O}}(\mathbf{p}) = d(\mathbf{p}; \mathscr{O}) = \min_{\mathbf{o} \in \mathscr{O}} \|\mathbf{p} - \mathbf{o}\|_2 .$$

For the sake of simplicity we assume that the robot's shape is contained in an n-dimensional sphere with radius $< \underline{d}$ and center point $\mathbf{p}$. Collision avoidance can now be achieved by requesting that the distance function at the robot's position $\mathbf{p} \in \mathscr{W}$ is at least the minimum distance $\underline{d}$:

$$d_{\mathscr{O}}(\mathbf{p}) \geq \underline{d}, \tag{1}$$

which is equivalent to $\|\mathbf{p} - \mathbf{o}\|_2 \geq \underline{d}, \forall \mathbf{o} \in \mathscr{O}$.

We assume $\underline{d} > 0$ to be fixed from now on, to ensure that points in the occupied set $\mathscr{O}$ do not satisfy (1). A point $\mathbf{p} \in \mathscr{W}$ that satisfies (1) is called '*free*'. Further we define the *safety margin* as the area for which $0 < d_{\mathscr{O}}(\mathbf{p}) < \underline{d}$ holds.

We can now formulate an OCP that enforces collision avoidance as path constraint:

$$\begin{aligned} \min_{\mathbf{x}(\cdot), \mathbf{u}(\cdot)} \quad & \int_0^T l(\mathbf{x}(t), \mathbf{u}(t), \mathbf{r}(t))\, dt \; + l_{\mathrm{T}}(\mathbf{x}(T), \mathbf{r}(T)) \\ \text{s.t.} \quad & \mathbf{x}(0) = \overline{\mathbf{x}}_0, \\ & \mathbf{x}(T) \in \mathbb{X}_{\mathrm{T}}, \\ & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad t \in [0, T], \\ & h(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, \quad t \in [0, T], \\ & d_{\mathscr{O}}(\mathbf{p}(t)) \geq \underline{d}, \quad t \in [0, T], \end{aligned} \tag{2}$$

where $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^{n_{\mathbf{x}}}$ denotes the robot's state, $\mathbf{u}(\cdot) : \mathbb{R} \to \mathbb{R}^{n_{\mathbf{u}}}$ is the vector of controls, $\mathbf{r}(\cdot) : \mathbb{R} \to \mathbb{R}^{n_{\mathbf{x}}+n_{\mathbf{u}}}$ provides reference states and controls, T is the length of the horizon in seconds, the function $l(\cdot)$ denotes the cost at time point t and $l_{\mathrm{T}}(\cdot)$ the terminal cost, $\overline{\mathbf{x}}_0$ is robot's current state, and $\mathbb{X}_{\mathrm{T}} \subseteq \mathbb{R}^{n_{\mathbf{x}}}$ is the set of admissible terminal states. We use the common shorthand $\dot{\mathbf{x}}$ to denote the derivative with respect to time, i.e. $\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial t}$. The function f models the robot's dynamics and h implements a set of path constraints, e.g. physical limitations of the system, and $\mathbf{p}(t) = S_{\mathbf{p}} \cdot \mathbf{x}(t) \in \mathscr{W}$ denotes the robot's position with a selector matrix $S_{\mathbf{p}}$ chosen accordingly.

IV. CONVEX INNER APPROXIMATION (CIAO)

In this section we describe how we solve the OCP presented in (2) by adopting a convex inner approximation of the actual collision avoidance constraint presented in Sec. III.

First we discretize (2) using a direct multiple shooting scheme as proposed by [25]. The resulting NLP is a function of $\mathbf{r}$, $\overline{\mathbf{x}}_0$, and the sampling time Δt. Using the shorthand $\mathbf{x}_k = \mathbf{x}(k \cdot \Delta t),\ k \in \mathbb{Z}$ for cleaner notation, we discretize (2) as:

$$\min_{\mathbf{W}} \quad J(\mathbf{w}, \mathbf{r}) \tag{3a}$$

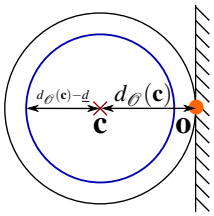
$$\text{s.t.} \quad \mathbf{x}_0 - \overline{\mathbf{x}}_0 = 0, \tag{3b}$$

$$\mathbf{x}_N \in \mathbb{X}_{\mathrm{T}}, \tag{3c}$$

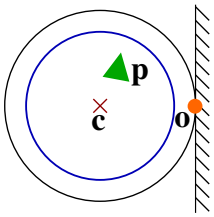
$$\mathbf{x}_{k+1} - F(\mathbf{x}_k, \mathbf{u}_k; \Delta t) = 0, \quad k = 0, \ldots, N-1, \tag{3d}$$

$$h(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \quad k = 0, \ldots, N, \tag{3e}$$

$$d_{\mathscr{O}}(\mathbf{p}_k) \geq \underline{d}, \quad k = 0, \ldots, N, \tag{3f}$$



(a) Example of a free ball around the center $\mathbf{c}$ marked by the red cross for a 2D environment. It is also the center of the circles, the black circle has radius $d_{\mathscr{O}}(\mathbf{c})$, and the blue circle radius $d_{\mathscr{O}}(\mathbf{c})-\underline{d}$.



(b) Example for free ball constraint. The green arrow head depicts the robot's current position $\mathbf{p}$, the orange dot the closest obstacle $\mathbf{o}$, the circles, and the red cross are identical to the ones in 2a.

Fig. 2: The left figure illustrates the free ball concept, the right shows how it can be used as a constraint.

where $\mathbf{w} = [\mathbf{x}_0^\top, \mathbf{u}_0^\top, \ldots, \mathbf{u}_{N-1}^\top, \mathbf{x}_N^\top]^\top \in \mathbb{R}^{n_\mathbf{w}}$ is a vector of optimization variables that contains the stacked controls and states for all N steps in the horizon, similarly $\mathbf{r}$ contains the reference states and controls, $J(\mathbf{w},\mathbf{r}) = \sum_{k=0}^{N-1} l(\mathbf{x}_k,\mathbf{u}_k,\mathbf{r}_k) + l_\mathrm{T}(\mathbf{x}_N,\mathbf{r}_N)$ is the discretized objective, with stage cost l and terminal cost l_T, and F models the discretized robot dynamics. We denote the feasible set of (3) by $\mathscr{F}_{(3)} \subset \mathbb{R}^{n_\mathbf{w}}$.

A. Free Balls: A Convex Inner Approximation of the Obstacle Avoidance Constraint

The actual obstacle avoidance constraint formulated in Eq. (1) is generally nonconvex and nonlinear, which makes it ill-suited for rapid optimization. We propose a convex inner approximation of the constraint that is based on the notion of free balls (FBs), as proposed in [1] and extended by [9]. For cleaner notation we first define the *free set*.

Definition 1: Let $\mathscr{O}$ be the occupied set and $\underline{d} > 0$ be the minimum distance, then the free set $\mathscr{A}$ is defined as

$$\mathscr{A} = \{a \in \mathscr{W} : \|\mathbf{a}-\mathbf{o}\|_2 \geq \underline{d} \quad \forall\, \mathbf{o} \in \mathscr{O}\}.$$

Remark: This definition implies that the free set $\mathscr{A}$ and the occupied set $\mathscr{O}$ are disjunct, i.e. $\mathscr{A} \cap \mathscr{O} = \emptyset$.

We can now formulate an obstacle avoidance constraint by enforcing that the robot's position lies within an n-dimensional ball formed around $\mathbf{c} \in \mathscr{A}$ as shown in Fig. 2.

Definition 2: For an arbitrary free point $\mathbf{c} \in \mathscr{A}$ we define the *free ball* as $\mathscr{A}_\mathbf{c} := \{\mathbf{p} \in \mathscr{W} : \|\mathbf{p}-\mathbf{c}\|_2 \leq d_{\mathscr{O}}(\mathbf{c}) - \underline{d}\}$.

We will now show that a free ball is a convex subset of the free set.

Lemma 1: Let $\mathbf{c} \in \mathscr{A}$ be a free point, then the free ball $\mathscr{A}_\mathbf{c}$ is a convex subset of $\mathscr{A}$, i.e. $\mathbf{c} \in \mathscr{A} \Rightarrow \mathscr{A}_\mathbf{c} \subseteq \mathscr{A}$.

Proof: We will prove this lemma in two steps. First, we observe that the free ball is a norm ball and therefore convex. Second, we show by contradiction that $\mathscr{A}_\mathbf{c} \nsubseteq \mathscr{A} \Rightarrow \mathbf{c} \notin \mathscr{A}$.

Suppose $\exists\, \mathbf{o} \in \mathscr{O}$ and $\mathbf{p} \in \mathscr{A}_\mathbf{c}$ such that $\|\mathbf{p}-\mathbf{o}\|_2 < \underline{d}$. We now apply the triangle inequality and obtain $\|\mathbf{c}-\mathbf{o}\|_2 \leq \|\mathbf{p}-\mathbf{c}\|_2 + \|\mathbf{p}-\mathbf{o}\|_2 < \|\mathbf{p}-\mathbf{c}\|_2 + \underline{d}$, see Fig. 2b. Using the distance function's definition we get $d_{\mathscr{O}}(\mathbf{c}) < \|\mathbf{p}-\mathbf{c}\|_2 + \underline{d}$. Reordering yields $\|\mathbf{p}-\mathbf{c}\|_2 > d_{\mathscr{O}}(\mathbf{c}) - \underline{d}$, which shows that $\mathscr{A}_\mathbf{c} \nsubseteq \mathscr{A}$. ∎

Based on Lem. 1 and Def. 2 we can approximate the collision avoidance constraint by $\|\mathbf{p}-\mathbf{c}\|_2 \leq d_{\mathscr{O}}(\mathbf{c}) - \underline{d}$. This formulation is not differentiable in $\mathbf{p} = \mathbf{c}$ and might pose a problem for gradient based solvers. To prevent this case and linear independence constraint qualification (LICQ) violations at the only feasible point we assume $d_{\mathscr{O}}(\mathbf{c}) > \underline{d}$.This implies that both sides are grater 0, such that we can square both sides and get

$$\|\mathbf{p}-\mathbf{c}\|_2^2 \leq (d_{\mathscr{O}}(\mathbf{c}) - \underline{d})^2. \tag{4}$$

With the constraint formulated in (4) and assuming that LICQ holds for all free ball center points $\mathbf{c}_k$ with $k = 0,\ldots,N$, we can partially convexify the NLP (3). We obtain the CIAO-NLP, which like (3) depends on $\mathbf{r}$, $\bar{\mathbf{x}}_0$, Δt, and additionally the tuple of center points $\mathbf{C} = (\mathbf{c}_0,\ldots,\mathbf{c}_N)$:

$$\min_{\mathbf{w},\mathbf{s}} \quad J(\mathbf{w},\mathbf{r}) + \sum_{k=0}^{N} \mu_k \cdot s_k \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_0 - \bar{\mathbf{x}}_0 = 0, \tag{5b}$$

$$\mathbf{x}_N \in \mathbb{X}_\mathrm{T}, \tag{5c}$$

$$\mathbf{x}_{k+1} - F(\mathbf{x}_k,\mathbf{u}_k;\Delta t) = 0, \quad k = 0,\ldots,N-1, \tag{5d}$$

$$h(\mathbf{x}_k,\mathbf{u}_k) \leq 0, \quad k = 0,\ldots,N, \tag{5e}$$

$$\|\mathbf{p}_k - \mathbf{c}_k\|_2^2 \leq (d_{\mathscr{O}}(\mathbf{c}_k) - \underline{d}_k)^2 + s_k, \quad k = 0,\ldots,N, \tag{5f}$$

$$s_k \geq 0, \quad k = 0,\ldots,N. \tag{5g}$$

This reformulation of the actual NLP (3) is called Convex Inner ApprOximation (CIAO). For numerical stability we include slack variables $\mathbf{s} = [s_0,\ldots,s_N]^\top \in \mathbb{R}^{N+1}$ that are penalized. A point $\mathbf{w}$ is only considered admissible if all slacks are zero, i.e. $\mathbf{s} = 0$ in a vector sense. To ensure that the slacks are only active for problems, that would be infeasible otherwise, the multipliers μ_k have to be chosen sufficiently large, i.e. $\mu_k \gg 1$ for $k = 0,\ldots,N$. The feasible set for optimization variables $\mathbf{w}$ of this NLP depends on $\mathbf{C}$ and is denoted as $\mathscr{F}_{(5)}(\mathbf{C})$, recall that $\mathbf{w} = [\mathbf{x}_0^\top, \mathbf{u}_0^\top, \ldots, \mathbf{u}_{N-1}^\top, \mathbf{x}_N^\top]^\top$ and $\mathbf{p}_k = S_\mathbf{p} \cdot \mathbf{x}_k$.

Note that $d_{\mathscr{O}}(\mathbf{c})$ enters the NLP as a constant ($\mathbf{c}$ is not an optimization variable). Thereby CIAO is compatible any implementation of the distance function, even discrete ones.

Note that for a convex objective $J(\cdot)$, a convex terminal set $\mathbb{X}_\mathrm{T}$, affine dynamics F, and convex path constraints h, the CIAO-NLP (5) is convex.

Further note that for a linear-quadratic objective $J(\cdot)$, affine-quadratic path constraints h, affine dynamics F, and a terminal set $\mathbb{X}_\mathrm{T}$ that can be written as either (i) an affine equality or (ii) an affine-quadratic inequality constraint, CIAO-NLP (5) is a quadratically constrained quadratic program (QCQP). If is also convex, it is a convex QCQP.

Lemma 2: Given $d_{\mathscr{O}}(\mathbf{c}) > \underline{d}\ \forall\, \mathbf{c} \in \mathbf{C} \Rightarrow \mathscr{F}_{(5)}(\mathbf{C}) \subseteq \mathscr{F}_{(3)}$, i.e. each feasible point of the CIAO-NLP (5) is a feasible point of the original NLP (3).

Proof: We observe that (3) and (5) are identical except for the collision avoidance constraint (3f) and (5f). As stated above $\mathbf{s} = 0$ holds for feasible points, thus the slacks $\mathbf{s}$ can be ignored. As shown in Lem. 1 (5f) is a convex inner approximation of (3f), therefore $\mathscr{F}_{(5)}(\mathbf{C}) \subseteq \mathscr{F}_{(3)}$ follows by construction. ∎

B. The CIAO-iteration

We will now introduce the CIAO-iteration, as detailed in Alg. 1. It takes a two step approach that first formulates the CIAO-NLP (5) by finding a tuple of center points $\mathbf{C} = (\mathbf{c}_0, \ldots, \mathbf{c}_N)$ before solving it.

Algorithm 1 the CIAO-iteration

1: **function** CIAO-ITERATION($\mathbf{w}$; $\mathbf{r}$, $\underline{\mathbf{x}}_0$, Δt)
2: $\quad\mathbf{C} \leftarrow (\mathbf{c}_k = S_\mathbf{p} \cdot \mathbf{x}_k$ for $k = 0, \ldots, N)$ ▷ recall $\mathbf{x}_k \in \mathbf{w}$
3: $\quad\mathbf{C}^* \leftarrow (\mathbf{c}^* = \text{MAXIMIZEFB}(\mathbf{c})$ for all $\mathbf{c} \in \mathbf{C})$ ▷ solve (6)
4: $\quad\mathbf{w}^* \leftarrow$ SOLVENLP($\mathbf{w}$; $\mathbf{C}^*$, $\mathbf{r}$, $\underline{\mathbf{x}}_0$, Δt) ▷ solve (5)
5: **end function return** $\mathbf{w}^*$ ▷ return newly found trajectory

In Line 2 we find an initial tuple of center points $\mathbf{C}$. In practice the free balls resulting from these center points are very small and therefore very restrictive, which leaves little room for optimization, especially if the initial guess $\mathbf{w}$ approaches obstacles closely. To overcome this problem we maximize free balls (FBs) (Line 3) by solving the following optimization problem for each $\mathbf{c} \in \mathbf{C}$ and obtain an optimized center point $\mathbf{c}^* = \eta \cdot \mathbf{g} + \mathbf{c}$:

$$\max_{\eta \geq 0} \eta \quad \text{s.t.} \quad d_{\mathscr{O}}(\eta \cdot \mathbf{g} + \mathbf{c}) = \eta + d_{\mathscr{O}}(\mathbf{c}), \tag{6}$$

where $\mathbf{c} \in \mathscr{A}$ is a given initial point, $\mathbf{g} \in \mathbb{R}^n$ is the search direction with $\|\mathbf{g}\|_2 = 1$ and η is the step size. It yields a maximized free ball $\mathscr{A}_{\mathbf{c}^*}$ with radius $r = d_{\mathscr{O}}(\mathbf{c}^*)$ and center point $\mathbf{c}^* = \eta \cdot \mathbf{g} + \mathbf{c}$ for each $\mathbf{c} \in \mathbf{C}$. The optimized center points are collected in the tuple $\mathbf{C}^* = (\mathbf{c}_0^*, \ldots \mathbf{c}_N^*)$. To ensure convergence of Alg.1 we require that the distance function is bounded, i.e. $\exists\, \overline{d} > 0$ such that $d_{\mathscr{O}}(\mathbf{p}) \leq \overline{d}\ \forall\, \mathbf{p} \in \mathscr{W}$.

We will now show that the optimization problem (6) preserves feasibility of the initial guess $\mathbf{w}$ by showing that $\mathscr{A}_{\mathbf{c}^*}$ includes $\mathscr{A}_\mathbf{c}$, i.e. $\mathscr{A}_\mathbf{c} \subseteq \mathscr{A}_{\mathbf{c}^*}$.

Lemma 3: For $\mathbf{c} \in \mathscr{A}$, $\mathbf{g} \in \{g \in \mathbb{R}^n : \|g\|_2 = 1\}$ and $\eta \geq 0$, $d_{\mathscr{O}}(\mathbf{c}^*) = \eta + d_{\mathscr{O}}(\mathbf{c}) \Rightarrow \mathscr{A}_\mathbf{c} \subseteq \mathscr{A}_{\mathbf{c}^*}$ holds with $\mathbf{c}^* = \eta \cdot \mathbf{g} + \mathbf{c}$.

Proof: We will prove this by contradiction, assuming $\exists\, \mathbf{p} \in \mathscr{A}_\mathbf{c}$ s.t. $\mathbf{p} \notin \mathscr{A}_{\mathbf{c}^*}$. Using Def. 2 we can rewrite this as $\|(\eta \cdot \mathbf{g} + \mathbf{c}) - \mathbf{p}\|_2 > d_{\mathscr{O}}(\mathbf{c}^*) - \underline{d}$. Applying the triangle inequality on the left side yields $\|\mathbf{p} - (\mathbf{c} + \eta \cdot \mathbf{g})\|_2 \leq \|\mathbf{p} - \mathbf{c}\|_2 + \|\eta \cdot \mathbf{g}\|_2 = \|\mathbf{p} - \mathbf{c}\|_2 + \eta$ and based on our assumption $\|\mathbf{p} - \mathbf{c}\|_2 + \eta \leq d_{\mathscr{O}}(\mathbf{c}) - \underline{d} + \eta$ holds. Inserting this gives $d_{\mathscr{O}}(\mathbf{c}) + \eta - \underline{d} > d_{\mathscr{O}}(\mathbf{c}^*) - \underline{d}$ and thus contradicts the condition $d_{\mathscr{O}}(\mathbf{c}^*) = \eta + d_{\mathscr{O}}(\mathbf{c})$. ∎

To solve the line search problem (6) we propose to use the distance function's normalized gradient $\mathbf{g} = \frac{\nabla d_{\mathscr{O}}(\mathbf{c})}{\|\nabla d_{\mathscr{O}}(\mathbf{c})\|_2}$ as search direction. Starting from $\eta = \underline{\eta} > 0$ the step size is exponentially increased until a step size $\overline{\eta} > \underline{\eta}$ is found for which the constraint is violated. SOLVENLP solves (5), e.g. by using an ad-hoc solver like Ipopt [26], and computes a new trajectory $\mathbf{w}^*$ (Line 4).

Lemma 4: For a feasible initial guess $\mathbf{w} \in \mathscr{F}_{(3)}$ Alg. 1 finds a feasible point $\mathbf{w}^* \in \mathscr{F}_{(3)}$ with $J(\mathbf{w}^*) \leq J(\mathbf{w})$.

Proof: We prove this in two steps: first we assume that SOLVENLP uses a suitable, working NLP-solver, then we show the feasibility. From $\mathscr{A}_\mathbf{c} \subseteq \mathscr{A}_{\mathbf{c}^*}$ as shown in Lem. 3 follows $\mathscr{F}_{(5)}(\mathbf{C}) \subseteq \mathscr{F}_{(5)}(\mathbf{C}^*)$. Further Lem. 2 yields $\mathscr{F}_{(5)}(\mathbf{C}) \subseteq \mathscr{F}_{(5)}(\mathbf{C}^*) \subseteq \mathscr{F}_{(3)}$. ∎

C. Continuous Time Collision Avoidance for Systems with Bounded Acceleration

The constraints formulated in (4) can be extended to the continuous time case. Using Lem. 1, we can write the continuous time collision avoidance constraint as $\|\mathbf{c}_k - \mathbf{p}(t)\|_2 \leq d_{\mathscr{O}}(\mathbf{c}_k) - \underline{d}, \quad \forall t \in [t_k, t_{k+1}], k = 0, \ldots, N$. Assuming a double integrator model of the form $\mathbf{p}(t) = \mathbf{p}_k + \dot{\mathbf{p}}_k \cdot (t - t_k) + \int_{t_k}^{t} \int_{t_k}^{\tau} \ddot{\mathbf{p}}(s)\, \mathrm{d}s\, \mathrm{d}\tau$ with the shorthand $\mathbf{p}_k = \mathbf{p}(t_k)$ for all $k = 0, \ldots, N$ and $t \in [t_k,\ t_{k+1}]$ it can be written as $\left\| \mathbf{p}_k + \dot{\mathbf{p}}_k \cdot (t - t_k) + \int_{t_k}^{t} \int_{t_k}^{\tau} \ddot{\mathbf{p}}(s)\, \mathrm{d}s\, \mathrm{d}\tau - \mathbf{c}_k \right\|_2 \leq d_{\mathscr{O}}(\mathbf{c}_k) - \underline{d}$. Using the triangle inequality we get $\|\mathbf{p}_k - \mathbf{c}_k\|_2 \leq d_{\mathscr{O}}(\mathbf{c}_k) - \underline{d} - \|\dot{\mathbf{p}}_k\|_2 \cdot (t - t_k) - \left\| \int_{t_k}^{t} \int_{t_k}^{\tau} \ddot{\mathbf{p}}(s)\, \mathrm{d}s\, \mathrm{d}\tau \right\|_2$. With $\|\int \ddot{\mathbf{p}}(\tau) \mathrm{d}\tau\|_2 \leq \int \|\ddot{\mathbf{p}}(\tau)\|_2 \mathrm{d}\tau$ and assuming that system's total acceleration is bounded $\|\ddot{\mathbf{p}}(t)\|_2 \leq \overline{a}\ \ \forall t \in \mathbb{R}$, which is a reasonable assumption for most physical systems, yields $\|\mathbf{p}_k - \mathbf{c}_k\|_2 \leq d_{\mathscr{O}}(\mathbf{c}_k) - \underline{d} - \|\dot{\mathbf{p}}_k\|_2 \cdot (t - t_k) - \frac{\overline{a}}{2} \cdot (t - t_k)^2$. We assume that velocities are bounded in all discretization points, i.e., $\|\dot{\mathbf{p}}_k\|_2 \leq \overline{v}$ for $k = 0, \ldots, N$. Considering that $t_0 - \frac{\Delta t}{2}$ and $t_N + \frac{\Delta t}{2}$ lie outside of the prediction horizon and that $\|\dot{\mathbf{p}}_{k \pm 1}\|_2 \leq \overline{v}$ for $k = 1, \ldots, N-1$, it is sufficient to consider the interval $t_k \pm \frac{\Delta t}{2}$ in each time step t_k and get $\|\mathbf{p}_k - \mathbf{c}_k\|_2 \leq d_{\mathscr{O}}(\mathbf{c}_k) - \underline{d} - \overline{v} \cdot \frac{\Delta t}{2} - \overline{a} \cdot \frac{\Delta t^2}{8}$ for $k = 0, \ldots, N$. For the sake of simplicity we assume

$$\underline{d}_k \geq \overline{v} \cdot \frac{\Delta t}{2} + \overline{a} \cdot \frac{\Delta t^2}{8} + \underline{d} \quad k = 0, 1, \ldots, N \tag{7}$$

from now on, such that (5f) guarantees continuous-time collision freedom is under the assumptions made above.

Note that (7) implies that $\mathscr{A}_{\mathbf{c}_k}$ and $\mathscr{A}_{\mathbf{c}_{k+1}}$ overlap or touch in the point where the trajectory transits from one into the next. Therefor we consider the robot's action radius.

D. Choosing the Objective Function

We use a quadratic cost function for reference tracking with regularization: $J(\mathbf{w}, \mathbf{r}) = \sum_{k=0}^{N-1} \alpha^k \|\mathbf{q}(\mathbf{x}_k) - \mathbf{q}(\hat{\mathbf{x}}_k)\|_Q^2 + \|\mathbf{u}_k - \hat{\mathbf{u}}_k\|_R^2 + \alpha^N \|\mathbf{q}(\mathbf{x}_N) - \mathbf{q}(\hat{\mathbf{x}}_N)\|_{Q_N}^2$, where $\alpha > 1$ leads to exponentially increasing stage cost and reduces oscillating behavior around the goal, $\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k$ denote reference state and controls at stage k, $\mathbf{q} : \mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}^{n_\mathbf{q}}$ augments the state $\mathbf{x}$ by applying proper transformations where applicable, $Q, Q_N \in \mathbb{R}^{n_\mathbf{q} \times n_\mathbf{q}}$, and $R \in \mathbb{R}^{n_\mathbf{u} \times n_\mathbf{u}}$ are positive definite matrices.

V. CIAO-BASED MOTION PLANNING

In this section we propose and detail two algorithms, one for pure trajectory optimization (Alg. 2) and the other for simultaneous trajectory optimization and tracking (Alg. 3).

A. CIAO for Trajectory Optimization

The proposed trajectory optimization algorithm (see Alg. 2) starts by computing a feasible initial guess and a reference trajectory (Lines 1–2). In the general case of nonconvex scenarios, such as cluttered environments, feasible initializations can be obtained through a sampling-based motion planner [27], [28]. To monitor the progress the initial guess is copied (Line 4), before using it as initial guess for the CIAO-ITERATION (Line 5). Lines 4–5 are repeated as long as the COST-function shows an improvement that

Algorithm 2 CIAO for offline trajectory optimization

Require: $\mathbf{x}_\mathrm{S}$, $\mathbf{x}_\mathrm{G}$, Δt, ε ▷ start and goal state
1: $\mathbf{w}^* \leftarrow$ INITIALGUESS($\mathbf{x}_\mathrm{S}$, $\mathbf{x}_\mathrm{G}$, Δt) ▷ feasible initialization
2: $\mathbf{r} \leftarrow$ REFERENCETRAJECTORY($\mathbf{x}_\mathrm{S}$, $\mathbf{x}_\mathrm{G}$, Δt)
3: **do**
4: $\quad\mathbf{w} \leftarrow \mathbf{w}^*$ ▷ set last solution as initial guess
5: $\quad\mathbf{w}^* \leftarrow$ CIAO-ITERATION($\mathbf{w}$; $\mathbf{r}$, $\bar{\mathbf{x}}_0$, Δt) ▷ $\bar{\mathbf{x}}_0 = \mathbf{x}_\mathrm{S}$
6: **while** COST($\mathbf{w}^*$) − COST($\mathbf{w}$) $> \varepsilon$
7: **return** $\mathbf{w}^*$

exceeds a given threshold ε (Line 6). Finally the best known solution $\mathbf{w}^*$ is returned (Line 7). For trajectory optimization the terminal constraint in (5) becomes an equality constraint, which enforces that the goal state $\mathbf{x}_\mathrm{G}$ is reached at the end of the horizon, i.e. $\mathbb{X}_\mathrm{T} = \{\mathbf{x}_\mathrm{G}\}$.

B. CIAO-NMPC

Alg. 3 uses a shorter horizon than Alg. 2 and runs only one CIAO-iteration before shifting it one step forward (Line 7). Therefore the initial guess $\mathbf{w}$ (Line 1) is not required to reach the goal state $\mathbf{x}_\mathrm{G} \in \mathbb{X}_\mathrm{G}$.

Algorithm 3 CIAO-NMPC

Require: $\bar{\mathbf{x}}_0$, $\mathbf{x}_\mathrm{G}$, Δt, $\mathbb{X}_\mathrm{G}$ ▷ current and goal state
1: $\mathbf{w} \leftarrow$ INITIALGUESS($\bar{\mathbf{x}}_0$, $\mathbf{x}_\mathrm{G}$, Δt) ▷ feasible initialization
2: **while** $\bar{\mathbf{x}}_0 \notin \mathbb{X}_\mathrm{G}$ **do**
3: $\quad\bar{\mathbf{x}}_0 \leftarrow$ GETCURRENTSTATE()
4: $\quad\mathbf{r} \leftarrow$ REFERENCETRAJECTORY($\bar{\mathbf{x}}_0$, $\mathbf{x}_\mathrm{G}$, Δt)
5: $\quad\mathbf{w}^* \leftarrow$ CIAO-ITERATION($\mathbf{w}$; $\mathbf{r}$, $\bar{\mathbf{x}}_0$, Δt) ▷ Alg. 1
6: $\quad$APPLYFIRSTCONTROL($\mathbf{w}^*$) ▷ recall $\mathbf{u}_0 \in \mathbf{w}^*$
7: $\quad\mathbf{w} \leftarrow$ SHIFTTRAJECTORY($\mathbf{w}^*$) ▷ recede horizon
8: **end while**

While the robot has not reached the goal region $\mathbb{X}_\mathrm{G}$ (Line 2), it is iteratively steered to it (Lines 3–8). Each iteration starts by updating the robot's current state $\mathbf{x}_0$. Based on the complexity of the scenario REFERENCETRAJECTORY may return a guiding trajectory to the goal or just the goal state itself (Line 4). We run Alg. 1 to compute a new trajectory (Line 5), before sending the first control to the robot (Line 6).

To ensure recursive feasibility, which implies collision avoidance, the terminal constraint (5c) is commonly chosen such that the robot comes to a full stop at the end of the horizon, i.e. $\mathbb{X}_\mathrm{T} = \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} : S_\mathrm{v} \cdot \mathbf{x} = 0\}$, where $S_\mathrm{v} \in \mathbb{R}^{n_\mathrm{v} \times n_\mathbf{x}}$ is the matrix that selects the velocities from the state vector.

VI. EXPERIMENTS AND DISCUSSION

To evaluate CIAO in terms of planning efficiency and final trajectory quality, we compare it against a set of baselines. We challenge CIAO by using nonlinear dynamics and a nonconvex cost function. Further we use a sampling based motion planner to initialize it with a collision free path that does not satisfy the robot's dynamics. In this case a NLP-solver is required to solve the CIAO-NLP (5). We use the primal-dual interior point solver Ipopt [29] with the linear solver MA-27 [30] called through CasADi [31].

A. Comparison of constraint formulations

In a first set of experiments, the numerical performance of the free ball constraint formulation, as derived in Sec. IV-A, is compared to common alternatives: the actual constraint as defined in Eq. (1) (actual), a linearization of the actual constraint (linear), and a log-barrier formulation (log-barrier). They differ only in the way the obstacle avoidance constraint (3f) is formulated. Our findings are reported in Tab. I.

	actual	*linear*	*CIAO*	*log-barrier*
ms / iteration	2.00	0.72	**0.70**	2.23
ms / step	40.78	**13.13**	17.26	50.34
iterations / step	20.35	**18.23**	24.64	22.57
time to goal [s]	**14.35**	14.39	18.67	(13.46)*
path length [m]	**10.22**	10.33	10.58	(9.25)*
max ms / step	448.94	230.07	**179.26**	> 1000
% timeouts	**0**	**0**	**0**	11.3

TABLE I: Comparison of constraint formulations.[1]

The average computation time taken per MPC-step and per Ipopt iteration are given as 'ms / step' and 'ms / iteration' respectively, 'iters / step' are the average Ipopt iterations per MPC-step. The path quality is evaluated in terms of 'time to goal' and 'path length'. Averages in the first five rows are taken over 62 scenarios, for which the maximum CPU time of 1.0s was not exceeded. The percentage of runs that exceed the CPU time is given by '% timeouts'. The maximum CPU time taken for a single MPC-step is given by 'max ms / step'.

We observe that the actual constraint is producing both fastest and shortest paths. This path quality comes at comparatively high computational cost. Linearizing the actual constraint reduces the computational effort, while maintaining a high path quality. In contrast to CIAO, linearization is not an inner approximation and can lead to constraint violations that necessitate computationally expensive recovery iterations. This increases the overall computation time significantly and leads to a higher maximal computation time. At the cost of lower path quality, but a similar average computation time, CIAO overcomes this problem by preserving feasibility. This leads to a lower variance in the computation time, and allows for continuous time collision avoidance guarantees. A further advantage, which is relevant in practice, is that CIAO generalizes to not continuously differentiable distance function implementations, e.g. distance fields.

Including the collision avoidance constraint as a barrier term in the objective, i.e. by adding $-\log(d_\mathscr{O}(p_k) - \underline{d})$ to the stage cost l_k, is an alternative approach to enforce collision freedom. Our results suggest, however, that for our application it is least favorable among the considered options.

B. Trajectory Optimization Benchmark

In a second set of experiments CIAO is compared to GuSTO [7] using the implementation publicly provided by the authors. In these experiments we consider a free-flying Astrobee Robot with 12 states and 6 controls, that has to move from a start position on the bottom front left corner of a $10 \times 10 \times 10$ m cube to a goal in the opposite corner. The room between start and goal point is cluttered with 25 randomly placed static obstacles of varying sizes (between 1 and 2 meters). Fig. 1 shows some examples.

[1]These experiments were conducted in simulation considering a robot with differential drive dynamics (5 states, 2 controls) and a prediction horizon of 50 steps, resulting in a total of 405 optimization variables.
* in Table I: not representative because complex scenarios with long transitions failed.

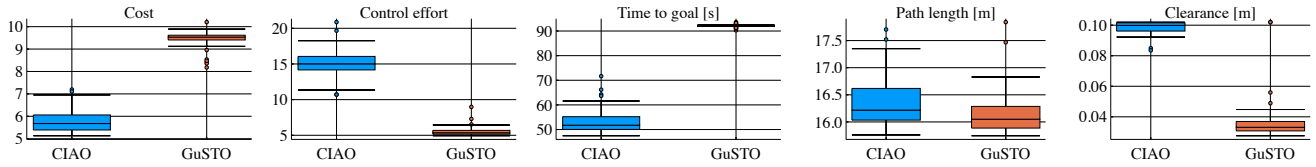


Fig. 3: Trajectory Optimization Benchmark Results. CIAO finds faster trajectories with higher clearance than GuSTO.

measure	CIAO	GuSTO
Compute^2 [s]	14.792±11.966	131.367±130.743
Iterations	30.660±15.886	4.520±1.282
Compute / Iteration [s]	0.475±0.230	27.729±22.096
Linearization Error	4.66e-14±3.67e-15	4.10e-06±1.28e-06

TABLE II: Numerical Performance: Average ± std values.

The results reported in Tab. II and Fig. 3 were performed using JuMP [32] on an Intel Core i7-8559U (2.7GHz) running MacOS. The SCPs formulated by GuSTO [7] are solved with Gurobi [33]. Both algorithms are provided with the same initial guess, which is computed with RRT [34]. We use a horizon of 250 steps and a sampling time of 0.4 s. Since both GuSTO and CIAO use tailored cost functions we evaluate the computed trajectories using a common cost function J_ρ, which is based on the state distance metric $\rho : R^{n_\mathbf{x}} \times R^{n_\mathbf{x}} \to \mathbb{R}$ proposed by [35]: $J_\rho(\mathbf{w}; \mathbf{x}_\mathrm{G}) = \sum_{k=0}^{N} \rho(\mathbf{x}_k, \mathbf{x}_\mathrm{G})$, with goal state $\mathbf{x}_\mathrm{G}$ and all weights of the distance metric chosen equal. The controls are evaluated separately and reported as control effort given by $J_\mathbf{u}(\mathbf{w}) = \sum_{k=0}^{N-1} \Delta t \cdot \|\mathbf{u}_k\|_1$. The path quality is evaluated in terms of time to goal, path length, and clearance (minimum distance to the closest obstacle along the trajectory). The first two measures take the time and path length until the state distance metric falls below a threshold of 0.5, while the latter is evaluated on the entire trajectory. These three metrics are evaluated on an oversampled trajectory using a sampling time $\Delta t = 0.01$ s.

The results in Fig. 3 show that CIAO finds faster trajectories than GuSTO and thereby also achieves significantly lower cost. As depicted in Fig. 1 it maintains a larger distance to obstacles for higher speeds. This behavior allows for a higher average speed, at the cost of a higher control activation and slightly longer paths in comparison to GuSTO.

As reported in Tab. II, CIAO (Alg. 2) requires more iterations to converge, but the individual iterations are cheaper. Moreover CIAO obtains a feasible trajectory after the first iteration and therefore could be terminated early, while GuSTO does not have this property and takes several iterations to find a feasible trajectory. Even though the dynamics are mostly linear we observe linearization errors for GuSTO, originating from the linear model they use.

In summary CIAO finds trajectories of higher quality than GuSTO at lower computational effort.

C. Real-World Experiments - Differential Drive Robot

To qualitatively assess the behavior of CIAO-NMPC (Alg. 3), it was tested in dynamic real-world scenarios with freely moving humans. A representative example is depicted in Fig. 4. Note that CIAO has no knowledge of the humans' future movements. It is instead considering all humans as static obstacles in their current position.

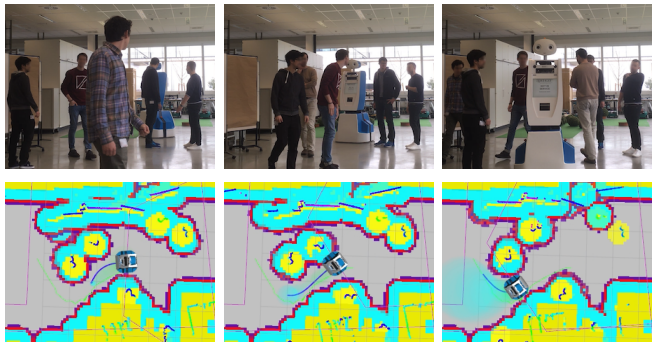


Fig. 4: CIAO steers a wheeled mobile robot through a group of people. Real-world (top) and RViz (bottom): Planned trajectory as blue line, free balls as transparent circles, obstacles in yellow, safety margin in light blue.

As in Sec. VI-A a differential drive robot is used, this time with a horizon of 5 s and a control frequency of 10 Hz resulting in a total of 405 optimization variables (including slacks). For these experiments CIAO was implemented as a C++ ROS-module, the distance function was realized as distance field based on the code by [36]. Initial guesses and reference paths were computed using an A* algorithm [37].

Since GuSTO is not suitable for receding horizon control (RHC), we used an extended version of the elastic-band (EB) method [1]. To obtain comparable results, we used the same A* planner and localization method with both algorithms.

In summary CIAO and the elastic band (EB) approach show similar behavior. In contrast to EB, CIAO computes kinodynamically feasible and guaranteed continuous time collision free trajectories. Further it has a notion of time for the planned motion, such that predictions for dynamic environments can be incorporated in future work.

VII. CONCLUSIONS AND FUTURE WORK

This work proposes CIAO, a new framework for trajectory optimization, that is based on a novel constraint formulation, that allows for NMPC based collision avoidance in real-time. We show that it reaches or exceeds state of the art performance in trajectory optimization at significantly lower computational effort, scales to high dimensional systems, and that it can be used for RHC style MPC of mobile robots in dynamic environments.

Future research will focus on extending CIAO to full body collision checking, guaranteed obstacle avoidance in dynamic environments, time optimal motion planning, and multi body robots. A second focus will lie on efficient numerical methods that exploit the structure and properties of the CIAO-NLP stated above.

REFERENCES

[1] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, vol. 2, 1993, pp. 802–807.

[2] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int. J. Rob. Res.*, vol. 21, no. 12, pp. 1031–1052, 2002.

[3] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian Optimization for Motion Planning," *Int. J. Rob. Res.*, vol. 32, no. 9–10, pp. 1164–1193, 2013.

[4] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Rob. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.

[5] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "FaSTrack: a modular framework for fast and guaranteed safe motion planning," in *IEEE Conf. Decis. Control (CDC)*, 2017, pp. 1517–1522.

[6] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *arXiv preprint arXiv:1711.03449*, 2017.

[7] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "GuSTO: Guaranteed Sequential Trajectory Optimization via sequential convex programming," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2019.

[8] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: a convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, pp. 2318–2327, 2009.

[9] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *IEEE Conf. Decis. Control (CDC)*, 2015, pp. 835–842.

[10] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[11] J. V. Frasch, A. J. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Eur. Control Conf. (ECC)*, 2013, pp. 4136–4141.

[12] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[13] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 1026–1039, 2016.

[14] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2016, pp. 1398–1404.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[16] L. Palmieri and K. O. Arras, "A novel RRT extend function for efficient and smooth mobile robot motion planning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 205–211.

[17] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2775–2781.

[18] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent advances in quadratic programming algorithms for nonlinear model predictive control," *Vietnam J. of Math.*, vol. 46, no. 4, pp. 863–882, 2018.

[19] J. Borenstein and Y. Koren, "The vector field histogram – fast obstacle avoidance for mobile robots," *IEEE Trans. Rob. Autom.*, vol. 7, no. 3, pp. 278 – 288, 1991.

[20] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Rob. Autom. Mag.*, vol. 4, no. 1, pp. 23 – 33, 1997.

[21] N. Y. Ko and R. G. Simmons, "The lane-curvature method for local obstacle avoidance," in *IEEE/RSJ Int. Conf. Intell. Rob. Syst. (IROS)*, vol. 3, 1998, pp. 1615 –1621.

[22] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Rob. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[23] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, February 2004.

[24] Y. xiang Yuan, "Recent advances in trust region algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 249–281, June 2015.

[25] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *Proceedings of the IFAC World Congress*. Pergamon Press, 1984, pp. 242–247.

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

[27] L. Palmieri and K. O. Arras, "Distance metric learning for RRT-based motion planning with constant-time inference," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 637–643.

[28] L. Palmieri, T. P. Kucner, M. Magnusson, A. J. Lilienthal, and K. O. Arras, "Kinodynamic motion planning on gaussian mixture fields," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 6176–6181.

[29] A. Wächter and L. Biegler, "IPOPT - an Interior Point OPTimizer," https://projects.coin-or.org/Ipopt, 2009.

[30] HSL, "A collection of Fortran codes for large scale scientific computation." http://www.hsl.rl.ac.uk, 2011.

[31] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[32] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[33] Gurobi Optimization, LLC, "Gurobi optimizer reference manual."

[34] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

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

[36] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1116–1130, 2013.

[37] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.