Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

CIAO*: MPC-based Safe Motion Planning in Predictable Dynamic Environments

Tobias Schoels [*] **Per Rutquist** [*] **Luigi Palmieri** [**]
Andrea Zanelli [*] **Kai O. Arras** [**] **Moritz Diehl** [*,***]

[*] *Department of Microsystems Engineering, University of Freiburg (e-mail: {tobias.schoels, per.rutquist, andrea.zanelli, moritz.diehl}@imtek.uni-freiburg.de)*
[**] *Robert Bosch GmbH, Corporate Research, Stuttgart, Germany (e-mail: {luigi.palmieri, kaioliver.arras}@de.bosch.com)*
[***] *Department of Mathematics, University of Freiburg*

Abstract: Robots have been operating in dynamic environments and shared workspaces for decades. Most optimization based motion planning methods, however, do not consider the movement of other agents, e.g. humans or other robots, and therefore do not guarantee collision avoidance in such scenarios. This paper builds upon the Convex Inner ApprOximation (CIAO) method and proposes a motion planning algorithm that guarantees collision avoidance in predictable dynamic environments. Furthermore, it generalizes CIAO's free region concept to arbitrary norms and proposes a cost function to approximate time optimal motion planning. The proposed method, CIAO*, finds kinodynamically feasible and collision free trajectories for constrained single body robots using model predictive control (MPC). It optimizes the motion of one agent and accounts for the predicted movement of surrounding agents and obstacles. The experimental evaluation shows that CIAO* reaches close to time optimal behavior.

Keywords: time optimal control, safety, convex optimization, predictive control, trajectory and path planning, motion control, autonomous mobile robots, dynamic environments

1. INTRODUCTION

Safe and smooth robot navigation is still an open challenge particularly for autonomous systems navigating in shared spaces with humans (e.g. intra–logistic and service robotics) and in densely crowded environments (Triebel et al., 2016). In these scenarios, the reactive avoidance of dynamic obstacles is an important requirement. Combined with the objective of reaching time optimal robot behavior, this poses a major challenge for motion planning and control and remains subject of active research.

Recent approaches tackle collision avoidance by formulating and solving optimization problems (Schulman et al., 2014; Bonalli et al., 2019; Schoels et al., 2020). These approaches offer good performance for finding locally optimal solutions but offer no guarantee to find the global optimum. Sampling-based planners, cf. Karaman and Frazzoli (2011), on the other hand, are asymptotically optimal and have been extended to dynamic environments (Otte and Frazzoli, 2015). They are, however, slow to converge and are therefore commonly terminated early. The suboptimal result is then passed to a trajectory optimization algorithm, like CIAO*, the one proposed in this paper.

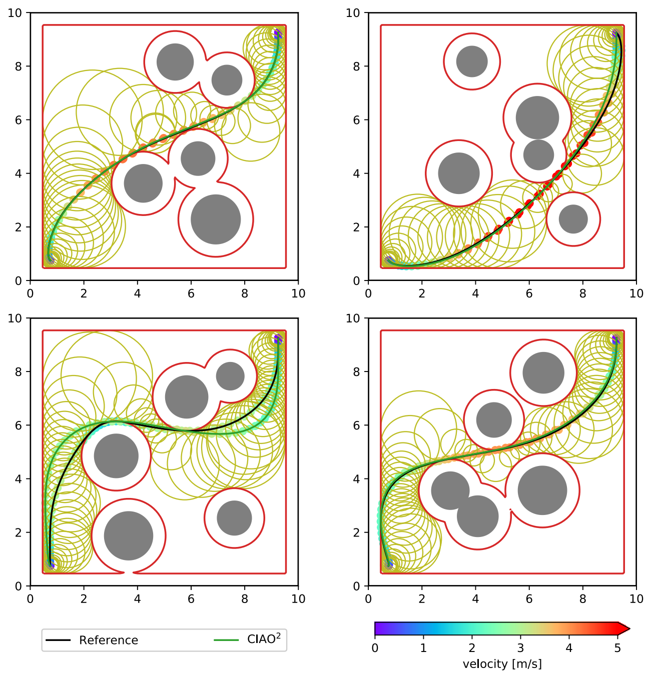

Fig. 1. Example trajectories found by CIAO^2 in green and a time optimal reference in black. The olive colored circles mark the convex free regions (CFRs), the red lines the safety margin to obstacles ρ.

* pronounced *'ciao-star'*, where $\star$ is a wildcard that specifies the norm used for the convex free region.

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).

Related Work: A shortcoming of most common trajectory optimization methods is their incapability to respect dynamic obstacles and kinodynamic constraints, e.g. bounds on the acceleration, and a lack of timing in their predictions (Quinlan and Khatib, 1993; Zucker et al., 2013; Schulman et al., 2014). These approaches are typically limited to the optimization of paths rather than trajectories and impose constraints by introducing penalties.

Classical approaches to obstacle avoidance include Borenstein and Koren (1991); Fox et al. (1997); Ko and Simmons (1998); Fiorini and Shiller (1998); Minguez and Montano (2004); Quinlan and Khatib (1993). In contrast to our approach, they do neither produce optimal trajectories, nor account for the robot's dynamics and constraints, nor handle the obstacles with their full shape (i.e. with convex hulls) and predicted future movements.

Popular recent trajectory optimization methods include CHOMP (Zucker et al., 2013), TrajOpt (Schulman et al., 2014), and GuSTO (Bonalli et al., 2019), which find smooth trajectories in static environments efficiently. An increasing number of these approaches use model predictive control (MPC) based formulations to obtain kinodynamically feasible trajectories, e.g. (Bonalli et al., 2019; Herbert et al., 2017; Zhang et al., 2017). In this framework an optimal control problem (OCP) is solved in every iteration. The OCPs formulated by CIAO* are convex, which makes it a sequential convex programming (SCP) method like TrajOpt and GuSTO. In some cases the formulated problems are linear, such that we obtain a sequential linear programming (SLP) method.

Similarly to TrajOpt (Schulman et al., 2014) and GuSTO (Bonalli et al., 2019), CIAO* uses a signed distance function (SDF) to model the environment. While they linearize the SDF, we find a convex-inner approximation (as depicted in Fig. 1) and propose a continuous time collision avoidance constraint, instead of a penalty term in the cost function. CIAO* generalizes the SDF (and thereby also the collision avoidance constraint) to arbitrary norms, similarly to OBCA (Zhang et al., 2017) and in Hyun et al. (2017). Moreover, as Rösmann et al. (2017), it approximates time optimal behavior.

MPC has been used to combine trajectory tracking and collision avoidance, e.g. Lim et al. (2008). CIAO (Schoels et al., 2020) goes one step further and also allows for trajectory optimization. Additionally it preserves feasibility across iterations using a convex collision avoidance constraint that is based on the Euclidean distance to the closest obstacle. It has been shown to work well in dynamic environments, even for robots with nonlinear dynamics. This paper presents a generalization of CIAO.

Contribution: In contrast to all other methods listed above, CIAO* guarantees collision avoidance in predictable dynamic environments. Further it differs from the original CIAO (Schoels et al., 2020) in four regards:

- CIAO* is norm agnostic, such that the original CIAO is a special case where $\star = 2$, i.e. CIAO $\equiv$ CIAO^2.
- The (predicted) movement of dynamic obstacles is considered explicitly during trajectory optimization.
- We motivate the use of a different cost function to approximate time optimal behavior.
- The collision avoidance constraint is generalized to robots of shapes that can be approximated by a convex, bounding polytope.

To the best of the author's knowledge this makes CIAO* the first MPC approach that approximates time optimal behavior in predictable dynamic environments and guarantees collision avoidance for linear systems. Like the original CIAO we present formulations for both offline trajectory optimization and online MPC based obstacle avoidance and control. In coherence with the theory, the experiments in this paper consider a linear system. CIAO*'s applicability to constrained, nonlinear systems has been demonstrated by Schoels et al. (2020).

Structure: Sec. 2 formalizes the trajectory optimization problem in dynamic environments we want to solve. The proposed approach, CIAO*, is introduced in Sec. 3 including some theoretical considerations. Sec. 4 details two algorithms that use CIAO* for motion planning. The experiments and results are discussed in Sec. 5. A summary and an outlook is given in Sec. 6. In App. A we introduce the *Taylor upper bound*, an approach to continuous time constraint satisfaction.

2. TIME OPTIMAL MOTION PLANNING

We formalize time optimal motion planning as a continuous time optimal control problem (OCP):

$$\begin{aligned}
\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), T} \quad & T && && \text{(1a)}\\
\text{s.t.} \quad & 0 \leq T, && && \text{(1b)}\\
& \boldsymbol{x}(0) = \boldsymbol{x}_\text{s}, && && \text{(1c)}\\
& \boldsymbol{x}(T) = \boldsymbol{x}_\text{g}, && && \text{(1d)}\\
& \dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t), && t \in [0, T], && \text{(1e)}\\
& (\boldsymbol{x}(t), \boldsymbol{u}(t)) \in \mathcal{H}, && t \in [0, T], && \text{(1f)}\\
& \emptyset = \text{int}(\mathcal{R}(\boldsymbol{x}(t))) \cap \mathcal{O}(t), && t \in [0, T], && \text{(1g)}
\end{aligned}$$

where $\boldsymbol{x}(\cdot) : \mathbb{R} \to \mathbb{R}^{n_{\boldsymbol{x}}}$ denotes the robot's state, $\boldsymbol{u}(\cdot) : \mathbb{R} \to \mathbb{R}^{n_{\boldsymbol{u}}}$ is the vector of controls, T is the length of the trajectory in seconds, which is minimized. The fixed vector $\boldsymbol{x}_\text{s}$ is the robot's current state, and $\boldsymbol{x}_\text{g}$ is the goal state. We use the common shorthand $\dot{\boldsymbol{x}}$ to denote the derivative with respect to time, i.e. $\dot{\boldsymbol{x}} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$. The expression $A\boldsymbol{x} + B\boldsymbol{u}$ denotes the system's linear dynamical model, the convex polytopic set $\mathcal{H}$ implements path constraints, e.g. physical limitations of the system. The open set $\text{int}(\mathcal{R}(\boldsymbol{x}(t))) \subset \mathbb{R}^n$ is the interior of the set of points occupied by the robot at time t and $\mathcal{O}(t) \subset \mathbb{R}^n$ is the set of points occupied by obstacles at that time. Finally, n is the dimension of the robot's workspace $\mathbb{R}^n$. Note that for fixed T, problem (1) becomes convex if (1g) is removed.

3. CIAO*: CONVEX INNER APPROXIMATION

In the following we detail the reformulations and parameterizations of (1) used in this paper. First, we introduce our implementation of the collision avoidance constraint. It is based on the concept of convex free regions (CFRs) that utilizes the signed distance function (SDF). Then we discretize (1) followed by a discussion of safety in continuous time.

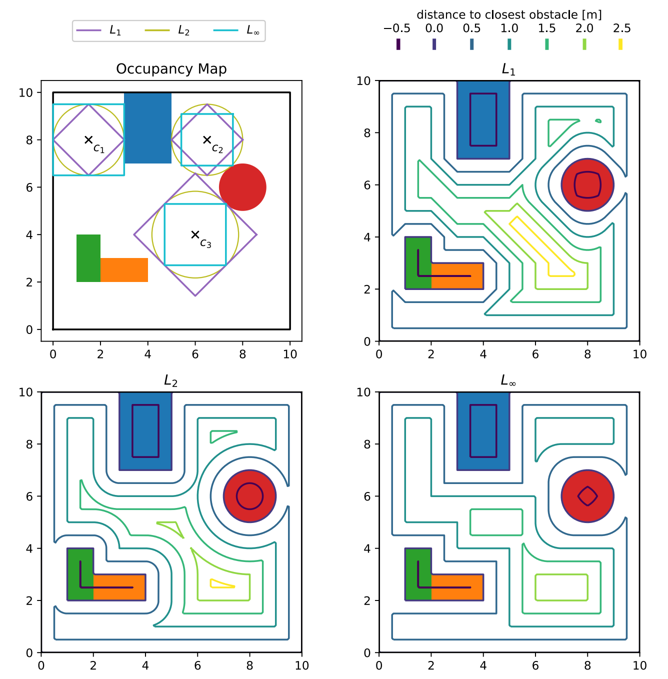

Fig. 2. Contour lines of the s igned distance function for L_1, L_2, and L_∞ norm in an example environment. Top left shows the resulting convex free regions for hand picked locations.

3.1 Collision Avoidance Constraint

The collision avoidance constraint (1g) can be formulated using the signed distance function (SDF) $\mathrm{sd}^\star_{\mathcal{O}}(\cdot)$ for an arbitrary occupied set $\mathcal{O}$. The distance of a given point $\boldsymbol{p} \in \mathbb{R}^n$ to the occupied set $\mathcal{O}$ is defined as

$$d^\star_{\mathcal{O}}(\boldsymbol{p}) = \min_{\mathbf{o}\in\mathcal{O}} \|\boldsymbol{p} - \mathbf{o}\|_\star . \tag{2}$$

Note that $d^\star_{\mathcal{O}}(\boldsymbol{p}) = 0$ for $\boldsymbol{p} \in \mathcal{O}$ and that $\star$ is used as a wildcard, not a dual norm notation. Further, we define the penetration depth as the distance of $\boldsymbol{p} \in \mathbb{R}^n$ to the unoccupied set $\mathbb{R}^n \setminus \mathcal{O}$

$$\mathrm{pen}^\star_{\mathcal{O}}(\boldsymbol{p}) = \min_{\mathbf{o}\in\mathbb{R}^n\setminus\mathcal{O}} \|\boldsymbol{p} - \mathbf{o}\|_\star . \tag{3}$$

Note that $\mathrm{pen}^\star_{\mathcal{O}}(\boldsymbol{p}) = 0$ for $\boldsymbol{p} \notin \mathcal{O}$. We combine (2) and (3) to obtain the SDF

$$\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{p}) = d^\star_{\mathcal{O}}(\boldsymbol{p}) - \mathrm{pen}^\star_{\mathcal{O}}(\boldsymbol{p}). \tag{4}$$

Note that the SDF is in general non-linear, non-convex, and non-differentiable, but continuous. An illustration of the SDF for $\star = \{1, 2, \infty\}$ is shown in Fig. 2.

The *free set* $\mathcal{F}^\star$ contains all points that are unoccupied:

$$\mathcal{F}^\star = \{\boldsymbol{c} \in \mathbb{R}^n : \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}) \geq 0\}. \tag{5}$$

The full-body collision avoidance can now be formulated via the SDF as $\mathrm{int}(\mathcal{R}) \subseteq \mathcal{F}^\star$.

Lemma 1. Let $\mathcal{O}$ and $\mathcal{R}$ be the set of points occupied by obstacles and the robot respectively, then

$$\mathcal{O} \cap \mathrm{int}(\mathcal{R}) = \emptyset \Leftrightarrow \mathrm{int}(\mathcal{R}) \subseteq \mathcal{F}^\star \Leftrightarrow \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{p}) \geq 0,\ \forall \boldsymbol{p} \in \mathcal{R}.$$

Proof: This follows directly from the definitions of the distance function (4) and the free set (5). □

Convex Free Regions (CFRs): The collision avoidance constraint is in general non-convex, non-linear, and non-differentiable, cf. Schulman et al. (2014). This poses a problem for derivative based optimization methods, that are used to solve the OCP in (1). To overcome this problem we use a convex inner approximation of this constraint, called convex free region (CFR).

For any point $\boldsymbol{c} \in \mathcal{F}^\star$ the signed distance function $\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$ yields a *convex free region* $\mathcal{C}^\star_{\boldsymbol{c}}$, which we define as

$$\mathcal{C}^\star_{\boldsymbol{c}} = \{\boldsymbol{p} \in \mathbb{R}^n : \|\boldsymbol{p} - \boldsymbol{c}\|_\star \leq \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})\}. \tag{6}$$

Note that CFR $\mathcal{C}^\star_{\boldsymbol{c}}$ is fully described by its center point $\boldsymbol{c}$ and the used norm $\star$. Fig. 2 shows CFRs obtained for different, hand-picked points in an example environment. Now we show that CFRs are convex subsets of the free set.

Lemma 2. For any free point $\boldsymbol{c} \in \mathcal{F}^\star$ the convex free region $\mathcal{C}^\star_{\boldsymbol{c}}$ is a convex subset of $\mathcal{F}^\star$, i.e. $\mathcal{C}^\star_{\boldsymbol{c}} \subseteq \mathcal{F}^\star$.

Proof: We prove this lemma in two steps. First, we observe that convex free regions are norm balls and therefore convex. Second, we show that $\boldsymbol{c} \in \mathcal{F}^\star \Rightarrow \mathcal{C}^\star_{\boldsymbol{c}} \in \mathcal{F}^\star$ by construction.
Take any $\boldsymbol{p} \in \mathcal{C}^\star_{\boldsymbol{c}}$ and any $\boldsymbol{o} \in \mathcal{O}$, then the reverse triangle inequality yields $\|\boldsymbol{p} - \boldsymbol{o}\|_\star \geq \|\boldsymbol{o} - \boldsymbol{c}\|_\star - \|\boldsymbol{p} - \boldsymbol{c}\|_\star$. Since $\|\boldsymbol{c} - \boldsymbol{o}\|_\star \geq \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$ due to (4) and $-\|\boldsymbol{p} - \boldsymbol{c}\|_\star \geq -\mathrm{sd}^\star_{\mathcal{O}}(c)$ due to (6), we get $\|\boldsymbol{p} - \boldsymbol{o}\|_\star \geq \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}) - \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}) = 0$. □

Remark 1. This lemma is a generalization of Lem. 2 by Schoels et al. (2020) to arbitrary norms $\|\cdot\|_\star$.

Full-body collision avoidance: To implement full-body collision avoidance efficiently, we approximate the robot's shape by a convex polytope, like Schulman et al. (2014).

Assumption 1. Assume that a finite set of points $\overline{\mathcal{R}} = \{\nu_1, \ldots, \nu_{n_{\mathcal{R}}}\}$ exists, such that $\mathcal{R} \subseteq \mathrm{convhull}(\overline{\mathcal{R}})$.

Remark 2. Collision avoidance can be enforced by constraining the spanning vertices to a free region, i.e. $\mathrm{convhull}(\overline{\mathcal{R}}) \subseteq \mathcal{C}^\star_{\boldsymbol{c}} \Leftrightarrow \nu_1, \ldots, \nu_{n_{\mathcal{R}}} \in \mathcal{C}^\star_{\boldsymbol{c}}$. It is easy to show that this is an inner approximation of the actual constraint, i.e. $\mathrm{int}(\overline{\mathcal{R}}) \subseteq \mathcal{C}^\star_{\boldsymbol{c}} \Rightarrow \mathrm{int}(\mathcal{R}) \subseteq \mathcal{C}^\star_{\boldsymbol{c}} \Rightarrow \mathrm{int}(\mathcal{R}) \subseteq \mathcal{F}^\star$.

3.2 Enlarging Convex Free Regions (CFRs)

A CFR $\mathcal{C}^\star_{\boldsymbol{c}}$ is formed around a center point $\boldsymbol{c} \in \mathcal{F}^\star$ with radius $r = \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$. If $\boldsymbol{c}$ approaches an obstacle r shrinks and in the limit ($r = \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}) = 0$) the CFR collapses to a point. Such situations result in very restrictive constraints ($\|\boldsymbol{p} - \boldsymbol{c}\|_\star \leq r$). To avoid this problem, we grow CFRs as proposed by Schoels et al. (2020).

Assumption 2. Assume that $\boldsymbol{c} \in \mathcal{F}^\star$ and that $\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$ is defined $\forall \boldsymbol{c} \in \mathbb{R}^n$.[1]

We then find a larger CFR $\mathcal{C}^\star_{\boldsymbol{c}^*}$ via line search along a search direction $\mathbf{g} = \frac{\nabla_{\boldsymbol{c}}\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})}{\|\nabla_{\boldsymbol{c}}\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})\|_\star}$, i.e., the SDF's normalized gradient[2]. It is defined almost everywhere, except for points where $\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$ is undefined,e.g. ridges. In the latter case we stop the search immediately. For a given initial point $\boldsymbol{c} \in \mathcal{F}^\star$, we compute the optimal step size η^* by solving

$$\eta^* = \arg\max_{\eta \geq 0}\ \eta \quad \text{s.t.} \quad \mathrm{sd}^\star_{\mathcal{O}}(\underbrace{\eta \cdot \mathbf{g} + \boldsymbol{c}}_{=\boldsymbol{c}^*}) = \eta + \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}), \tag{7}$$

We obtain an enlarged CFR $\mathcal{C}^\star_{\boldsymbol{c}^*}$ with center $\boldsymbol{c}^* = \eta^* \cdot \mathbf{g} + \boldsymbol{c}$ and radius $r^* = \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}^*)$. Note that the line search

[1] If $\boldsymbol{c} \notin \mathcal{F}^\star$ we follow the gradient to find a $\boldsymbol{c}' \in \mathcal{F}^\star$.
[2] It is sufficient to implement $\nabla_{\boldsymbol{c}}\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$ using finite differences.

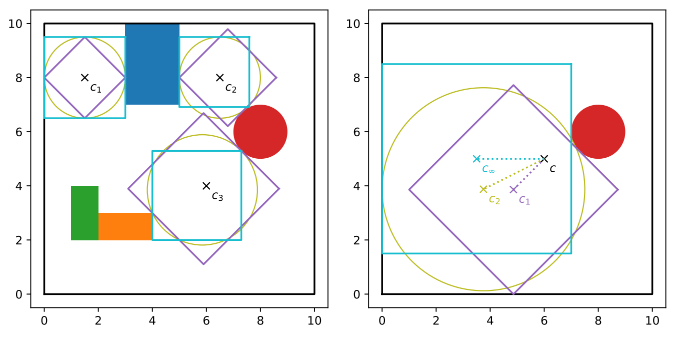

Fig. 3. Enlarged convex free regions (CFRs). Left: enlarged regions for L_1 (purple), L_2 (olive), and L_∞ (cyan) for the starting points and environment from Fig. 2. Right: the same for a simplified environment, including corresponding center points and search directions.

terminates on a ridge. Figure 3 shows free regions produced by this line search approach for different norms.

Solving (7) yields a new free region $\mathcal{C}^\star_{\boldsymbol{c}^*}$ that includes the original one $\mathcal{C}^\star_{\boldsymbol{c}}$. When solving this problem numerically we introduce a small tolerance to allow for numerical errors.

Lemma 3. If $\boldsymbol{c} \in \mathcal{F}^\star$, $\mathbf{g} \in \{g \in \mathbb{R}^n : \|g\|_\star = 1\}$, and $\eta \geq 0$, with $\boldsymbol{c}^* = \eta \cdot \mathbf{g} + \boldsymbol{c}$ and $\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}^*) = \eta + \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$ then $\mathcal{C}^\star_{\boldsymbol{c}} \subseteq \mathcal{C}^\star_{\boldsymbol{c}^*}$.

Proof: We prove this by contradiction, assuming $\exists\, \boldsymbol{p} \in \mathcal{C}^\star_{\boldsymbol{c}}$ such that $\boldsymbol{p} \notin \mathcal{C}^\star_{\boldsymbol{c}^*}$. Using (6) we rewrite our assumption to $\|(\eta \cdot \mathbf{g} + \boldsymbol{c}) - \boldsymbol{p}\|_\star > \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}^*)$. Applying the triangle inequality on the left side yields $\|\boldsymbol{p} - (\boldsymbol{c} + \eta \cdot \mathbf{g})\|_\star \leq \|\boldsymbol{p} - \boldsymbol{c}\|_\star + \|\eta \cdot \mathbf{g}\|_\star = \|\boldsymbol{p} - \boldsymbol{c}\|_\star + \eta$ and based on our assumption $\|\boldsymbol{p} - \boldsymbol{c}\|_\star + \eta \leq \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}) + \eta$ holds. Inserting this gives $\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}) + \eta > \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}^*)$ and thus contradicts the condition $\mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}^*) = \eta + \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$. □

Remark 3. This lemma is a generalization of Lem. 6 by Schoels et al. (2020) to arbitrary norms $\|\cdot\|_\star$.

3.3 Discrete Time OCP

We discretize the OCP (1) by splitting the horizon into $N+1$ time steps $t_k = k \cdot \Delta t$ for $k = 0, \ldots, N$ and a chosen sampling time Δt. For more compact notation, we use the shorthand $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$ to denote discrete time quantities.

Assumption 3. We use piece-wise constant controls, i.e., $\boldsymbol{u}(t) = \boldsymbol{u}_k\, \forall t \in [t_k, t_{k+1})$, and assume that $\boldsymbol{x}_\mathrm{g}$ is a steady state at $\boldsymbol{u} = 0$, i.e., $0 = A\boldsymbol{x}_\mathrm{g}$.

Objective function: We approximate time optimal behavior without time scaling, i.e., for a fixed Δt, using the stabilizing scheme proposed by Verschueren et al. (2017). They show that for the case of point-to-point motion, i.e., if the robot shall move from $\boldsymbol{x}_\mathrm{s}$ to $\boldsymbol{x}_\mathrm{g}$ in minimal time, the time optimal objective function can be approximated by

$$\min_{\substack{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N, \\ \boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}}} \sum_{k=0}^{N-1} \alpha^k \|\boldsymbol{x}_k - \boldsymbol{x}_\mathrm{g}\|$$

with initial condition $\boldsymbol{x}_0 = \boldsymbol{x}_\mathrm{s}$ and terminal constraint $\boldsymbol{x}_N = \boldsymbol{x}_\mathrm{g}$ for both N and $\alpha > 1$ large enough, such that time optimality is recovered. Note that this transformation is norm-agnostic.

Robot Model: The piece-wise constant controls allow us to discretize the dynamical model using the matrix exponential. This yields $A_\mathrm{D} = e^{A\Delta t}$ and $B_\mathrm{D} = \left(\int_0^{\Delta t} e^{At} \mathrm{d}t\right) B$ and the discrete dynamics $\boldsymbol{x}_{k+1} = A_\mathrm{D}\boldsymbol{x}_k + B_\mathrm{D}\boldsymbol{u}_k$.

Occupied Set: In Sec. 3.1 describes a convex inner approximation of the actual collision constraint for an arbitrary occupied set $\mathcal{O}$. We use this formulation to derive a collision avoidance constraint that can accommodate arbitrary time dependent occupied sets $\mathcal{O}(t)$ and can thus account for predictable dynamic obstacles.

The discrete time occupied set $\mathcal{O}_k$ is defined as the union of all occupied sets $\mathcal{O}(t)$ in the time interval $[t_k, t_{k+1}]$:

$$\mathcal{O}_k = \bigcup_{t=t_k}^{t_{k+1}} \mathcal{O}(t). \tag{8}$$

Obviously the continuous time occupied set $\mathcal{O}(t)$ is contained in the discrete time occupied set $\mathcal{O}_k\, \forall t \in [t_k, t_{k+1}]$.

Assumption 4. Assume that the discrete time occupied set $\mathcal{O}_k$ is known for all $k = 0, \ldots, N$. This assumption is realistic as mobile robots typically possess systems to estimate motions and states of surrounding agents.

Collision Avoidance Constraint: To guarantee collision avoidance in continuous time, the robot's movement needs to be accounted for. For more compact notation, we define the set of all points that are occupied by the robot in the time interval $[t_k, t_{k+1}]$ as $\Omega_k = \bigcup_{t=t_k}^{t_{k+1}} \mathcal{R}(\boldsymbol{x}(t))$ and introduce the shorthand $\mathcal{R}_k = \mathcal{R}(x_k)$. We define the robot's action radius for all $k = 0, \ldots, N$ as

$$\rho_k = \max_{\boldsymbol{p} \in \Omega_k} d^\star_{\mathcal{R}_k}(\boldsymbol{p}) = \max_{\boldsymbol{p} \in \Omega_k} \min_{\boldsymbol{p}' \in \mathcal{R}_k} \|\boldsymbol{p} - \boldsymbol{p}'\|_\star. \tag{9}$$

The *Taylor upper bound* introduced in Appendix A, can be used to compute an upper bound for the robot's action radius. The robot's position $\boldsymbol{p}(t)$ can be rewritten as $\boldsymbol{p}(t) = \boldsymbol{p}(t_k) + (\boldsymbol{p}(t) - \boldsymbol{p}(t_k)) = \boldsymbol{p}_k + \Delta\boldsymbol{p}(t, t_k)$. If the first m derivatives of $\boldsymbol{p}$ with respect to time are known and that the mth derivative is globally bounded by $\left\|\boldsymbol{p}^{(m)}(t)\right\|_\star \leq \overline{p}^{(m)}, \forall t \in \mathbb{R}$, then the *Taylor upper bound* yields

$$\|\Delta\boldsymbol{p}(t, t_k)\|_\star \leq \sum_{i=1}^{m-1} \left\|\boldsymbol{p}_k^{(i)}\right\|_\star \frac{\Delta t^i}{i!} + \overline{p}^{(m)} \frac{\Delta t^m}{m!}. \tag{10}$$

Assumption 5. Assume that for all $k = 0, \ldots, N$ $\boldsymbol{p}$'s derivatives $\boldsymbol{p}_k^{(1)}, \ldots, \boldsymbol{p}_k^{(m-1)}$ are known and point wise bounded, such that $\|\boldsymbol{p}_k^{(i)}\|_\star \leq \overline{p}^{(i)}$ for $i = 1, \ldots, m-1$.

This yields a global upper bound ρ of the action radius ρ_k:

$$\|\Delta\boldsymbol{p}(t, t_k)\|_\star \leq \rho := \sum_{i=1}^{m} \overline{p}^{(i)} \frac{\Delta t^i}{i!}, \tag{11}$$

for all $k = 0, \ldots, N$. Note that ρ is independent of the time index k, but requires that all $\boldsymbol{p}_k^{(i)}$ are bounded.

Using Assumption 1, we approximate $\mathcal{R}_k$ by a convex bounding polytope with vertices $\overline{\mathcal{R}}_k = \{\nu_{1,k}, \ldots, \nu_{n_\mathcal{R},k}\}$.

Assumption 6. For simplicity, assume that the robot is constrained to translational movement, such that the vertices are given by $\nu_{i,k} = S_{\boldsymbol{p}} \cdot \boldsymbol{x}_k + l_i$, where $S_{\boldsymbol{p}}$ is a selector matrix, such that $\boldsymbol{p} = S_{\boldsymbol{p}} \cdot \boldsymbol{x}$ is the robot's position.

Assumption 6 yields a convex full-body collision avoidance constraint for $i = 1, \ldots, n_\mathcal{R}$ and $k = 0, \ldots, N$

$$\left\|\nu_{i,k} - \boldsymbol{c}_k\right\|_\star \le \mathrm{sd}^\star_{\mathcal{O}_k}(\boldsymbol{c}_k) - \rho. \tag{12}$$

Note that for a fixed $\boldsymbol{c}_k$ (12) is a conic constraint in $\nu_{i,k}$ and thereby a set of conic constraints in $\boldsymbol{x}_k$. Lem. 4 implies that (12) guarantees continuous time collision avoidance.

Lemma 4. Given the occupied set $\mathcal{O}_k$, the robot action radius $\rho \ge 0$, and a convex bounding polytope with vertices $\overline{\mathcal{R}}_k = \{\nu_{1,k}, \dots, \nu_{n_\mathcal{R},k}\}$ such that $\mathcal{R}_k \subseteq \mathrm{convhull}(\overline{\mathcal{R}}_k)$, then

$$\left\|\nu_{i,k} - \boldsymbol{c}_k\right\|_\star \le \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}_k) - \rho, \quad i = 1, \dots, n_\mathcal{R}$$

implies $\mathrm{int}(\mathcal{R}(\boldsymbol{x}(t))) \cap \mathcal{O}(t) = \emptyset$ for all $t \in [t_k, t_{k+1}]$.

Proof: We prove this lemma by showing that all vertices $\nu_i(t)$ for $t \in [t_k, t_{k+1}]$ and $i = 1, \dots, n_\mathcal{R}$ are inside the convex free region $\mathcal{C}^\star_{\boldsymbol{c}_k}$. First, we note $\left\|\nu_{i,k} - \boldsymbol{c}_k\right\|_\star \le \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}_k) - \rho \Leftrightarrow \left\|\nu_{i,k} - \boldsymbol{c}_k\right\|_\star + \rho \le \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c}_k)$. Further, the distance between $\nu_i(t)$ and $\boldsymbol{c}_k$ is given by $\left\|\nu_i(t) - \boldsymbol{c}_k\right\|_\star = \left\|S_{\boldsymbol{p}}\boldsymbol{x}_k + l_i + \Delta\boldsymbol{p}(t, t_k) - \boldsymbol{c}_k\right\|_\star$. Applying the triangle inequality yields $\left\|\nu_i(t) - \boldsymbol{c}_k\right\|_\star \le \left\|S_{\boldsymbol{p}}\boldsymbol{x}_k + l_i - \boldsymbol{c}_k\right\|_\star + \left\|\Delta\boldsymbol{p}(t, t_k)\right\|_\star$. From (11) we know $\left\|\Delta\boldsymbol{p}(t, t_k)\right\|_\star \le \rho$, which results in $\left\|\nu_i(t) - \boldsymbol{c}_k\right\|_\star \le \left\|\nu_{i,k} - \boldsymbol{c}_k\right\|_\star + \rho \le \mathrm{sd}^\star_{\mathcal{O}}(\boldsymbol{c})$ □

Remark 4. This guarantee can be extended to arbitrary motion, by including an upper bound for the robot's displacement due to rotation, c.f. Schulman et al. (2014).

Path Constraints: To obtain continuous time constraint satisfaction for the convex polyhedral set $\mathcal{H}$ we utilize the *Taylor upper bound* (see App. A). This results in a smaller, convex polyhedral set. In addition, we impose the constraints resulting from Ass. 5. The resulting discrete time path constraints form a convex polytope $\mathcal{H}_\mathrm{D}$.

3.4 The CIAO*-NLP

Applying the reformulations detailed in the Sec. 3.3 to (1), we obtain a nonlinear program (NLP), the CIAO*-NLP. It is a convex conic problem that depends on goal state $\boldsymbol{x}_\mathrm{g}$, the initial state $\boldsymbol{x}_\mathrm{s}$, the sampling time Δt, the tuple of CFR center points $C = (\boldsymbol{c}_0, \dots, \boldsymbol{c}_N)$, and the corresponding radii $r_k = \mathrm{sd}^\star_k(\boldsymbol{c}_k) - \rho$, where ρ is the robot's action radius (11). The CIAO*-NLP is given by

$$\min_{\boldsymbol{w}} \quad \sum_{k=0}^{N-1} \alpha^k \left\|\boldsymbol{x}_k - \boldsymbol{x}_\mathrm{g}\right\|_{Q_\mathrm{x}} \tag{13a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_\mathrm{s}, \tag{13b}$$

$$\boldsymbol{x}_N = \boldsymbol{x}_\mathrm{g}, \tag{13c}$$

$$\boldsymbol{x}_{k+1} = A_\mathrm{D}\boldsymbol{x}_k + B_\mathrm{D}\boldsymbol{u}_k, \qquad k = 0, \dots, N-1, \tag{13d}$$

$$(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{H}_\mathrm{D}, \qquad k = 0, \dots, N, \tag{13e}$$

$$\left\|S_{\boldsymbol{p}}\boldsymbol{x}_k + l_i - \boldsymbol{c}_k\right\|_\star \le r_k,\ i = 1, \dots, n_\mathcal{R}, \quad k = 0, \dots, N, \tag{13f}$$

where $\alpha > 1$ and N are both large enough such that time optimal behavior is recovered (Verschueren et al., 2017). Here $\boldsymbol{w} = \left[\boldsymbol{x}_0^\top, \boldsymbol{u}_0^\top, \dots, \boldsymbol{u}_{N-1}^\top, \boldsymbol{x}_N^\top\right]^\top$ is the vector of optimization variables. CIAO* is norm agnostic, but for the sake of clear notation we introduce Q_x as a norm specifier. The matrices $A_\mathrm{D}, B_\mathrm{D}$ result from the discretization of the model. The convex set $\mathcal{H}_\mathrm{D}$ guarantees continuous time constraint satisfaction of the original path constraints, we choose $\boldsymbol{u}_N = 0$ in accordance with Assumption 3. The collision avoidance constraint (13f) is a reformulation of (12) that uses Assumption 6.

Note that (13) is a linear program (LP) if $Q_\mathrm{x}, \star \in \{1, \infty\}$ and can be solved efficiently, as demonstrated in Sec. 5.

3.5 The CIAO*-Iteration

The *CIAO**-*iteration* computes a new trajectory $\boldsymbol{w}^*$ for a provided initial guess $\boldsymbol{w}$ as described in Alg. 1.

Algorithm 1 the CIAO*-iteration

```
1: function CIAO*-ITERATION($\boldsymbol{w}$ ; $\boldsymbol{x}_\mathrm{g}$, $\boldsymbol{x}_\mathrm{s}$, $\Delta t$)
2:     $C \leftarrow (\boldsymbol{c}_k = S_{\boldsymbol{p}} \cdot \boldsymbol{x}_k$ for $k = 0, \dots, N)$
3:     $C^* \leftarrow (\boldsymbol{c}^* =$ GROWCFR$(\boldsymbol{c})$ for all $\boldsymbol{c} \in C)$          ▷ solve (7)
4:     $\boldsymbol{w}^* \leftarrow$ SOLVECIAO*-NLP$(\boldsymbol{w}; C^*, \boldsymbol{x}_\mathrm{g}, \boldsymbol{x}_\mathrm{s}, \Delta t)$   ▷ solve (13)
5: end function return $\boldsymbol{w}^*$          ▷ return newly found trajectory
```

First, Alg. 1 obtains a tuple of center points C using the initial guess $\boldsymbol{w}$ (Line 2). Recall that the robot's position is given by $\boldsymbol{p}_k = S_{\boldsymbol{p}}\boldsymbol{x}_k$. The center points C are then optimized as described in Sec. 3.2 (Line 3). The CIAO*-NLP is then solved using a suitable solver (Line 4).

Note that the CIAO*-iteration preserves feasibility, i.e. if the guess w is feasible, w^* is feasible. In the case of robot motion planning that means: for a kinodynamically feasible and collision free initial guess $\boldsymbol{w}$ Alg. 1 finds a kinodynamically feasible and collision free trajectory $\boldsymbol{w}^*$ that is faster or equally fast.

4. CIAO* FOR MOTION PLANNING

This section describes the application of the CIAO*-iteration (see Sec. 3.5) for motion planning. We propose two algorithms: one for offline trajectory optimization and a second for online motion planning and control.

4.1 CIAO* for Trajectory Optimization

Alg. 2 iteratively optimizes trajectories and approximates the time optimal solution. Starting with an initial guess $\boldsymbol{w}$, it uses the CIAO*-iteration to improve the initial guess (Line 1). Further CIAO*-iterations follow (Line 3–4), until

Algorithm 2 CIAO* for offline trajectory optimization

```
Require: $\boldsymbol{w}, \boldsymbol{x}_\mathrm{s}, \boldsymbol{x}_\mathrm{g}, \Delta t, \varepsilon$          ▷ initial guess, start and goal state
1: $\boldsymbol{w}^* \leftarrow$ CIAO*-ITERATION($\boldsymbol{w}$ ; $\boldsymbol{x}_\mathrm{g}$, $\boldsymbol{x}_\mathrm{s}$, $\Delta t$)          ▷ see Alg. 1
2: while COST($\mathbf{w}^*$) − COST($\boldsymbol{w}$) $> \varepsilon$ do
3:     $\mathbf{w} \leftarrow \mathbf{w}^*$          ▷ set last solution as initial guess
4:     $\boldsymbol{w}^* \leftarrow$ CIAO*-ITERATION($\boldsymbol{w}$ ; $\boldsymbol{x}_\mathrm{g}$, $\boldsymbol{x}_\mathrm{s}$, $\Delta t$)          ▷ see Alg. 1
5: end while
6: return $\mathbf{w}^*$
```

the improvement of the trajectory w.r.t. some cost function (e.g. CIAO*'s objective function) falls below a chosen threshold ε (Line 2).

Note that Alg. 2 preserves feasibility, because the CIAO*-iteration does. Once it converges to a feasible trajectory, all further iterations yield feasible (and better) trajectories.

4.2 CIAO*-MPC: Online Motion Planning

Additionally, we propose CIAO*-MPC. This algorithm unifies trajectory optimization and tracking, also referred to as online motion planning. To meet the time constraints of the robot's control loop, we use a horizon of fixed length N, which is typically shorter than the one used for trajectory optimization (as described before). As a consequence the goal might not be reachable during the now receding horizon. Therefore we replace the terminal constraint by a terminal cost and obtain an approximation of the CIAO*-NLP (13), the CIAO*-MPC-NLP:

$$\begin{aligned}
\min_{\boldsymbol{w}} \quad & \alpha_{\mathrm{N}} \left\|\boldsymbol{x}_N - \boldsymbol{x}_{\mathrm{g}}\right\|_{Q_{\mathrm{x}}} + \sum_{k=0}^{N-1} \alpha^k \left\|\boldsymbol{x}_k - \boldsymbol{x}_{\mathrm{g}}\right\|_{Q_{\mathrm{x}}} \\
\text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}_{\mathrm{s}}, \\
& A\boldsymbol{x}_N = 0, \\
& \boldsymbol{x}_{k+1} = A_{\mathrm{D}}\boldsymbol{x}_k + B_{\mathrm{D}}\boldsymbol{u}_k, && k = 0,\dots,N-1, \\
& (\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{H}_{\mathrm{D}}, && k = 0,\dots,N, \\
& \left\|S_{\boldsymbol{p}}\boldsymbol{x}_k + l_i - \boldsymbol{c}_k\right\|_\star \le r_k,\ i = 1,\dots,n_{\mathcal{R}}, && k = 0,\dots,N,
\end{aligned} \tag{14}$$

where $\alpha_{\mathrm{N}} \gg \alpha^N$ is the terminal cost's scaling factor. The terminal constraint $A\boldsymbol{x}_N = 0$ ensures that the robot comes to a full stop at the end of the horizon, i.e. it reaches a steady state at $\boldsymbol{u}_N = 0$. This constraint prevents collisions that could occur when shifting the horizon forward in time. The NLP above replaces (13) in Line 4 of Alg. 1.

Algorithm 3 CIAO*-MPC

```
Require: $\boldsymbol{w}$, $\boldsymbol{x}_{\mathrm{g}}$, $\Delta t$   ▷ initial guess, goal state, and sampling time
1: while $\boldsymbol{x}_{\mathrm{s}} \neq \boldsymbol{x}_{\mathrm{g}}$ do   ▷ goal reached?
2:     $\boldsymbol{x}_{\mathrm{s}} \leftarrow$ getCurrentState()
3:     $\boldsymbol{w}^* \leftarrow$ CIAO$^\star$-iteration($\boldsymbol{w}$ ; $\boldsymbol{x}_{\mathrm{g}}$, $\boldsymbol{x}_{\mathrm{s}}$, $\Delta t$)   ▷ Alg. 1 with (14)
4:     controlRobot($\boldsymbol{w}^*$)   ▷ apply control $\boldsymbol{u}_0^*$
5:     $\boldsymbol{w} \leftarrow$ shiftTrajectory($\boldsymbol{w}^*$)   ▷ recede horizon
6: end while
```

Alg. 3 formally introduces the CIAO*-MPC method. While the robot is not at the goal $\boldsymbol{x}_{\mathrm{g}}$ (Line 1), we obtain the robot's current state $\boldsymbol{x}_{\mathrm{s}}$ (Line 2). Next we formulate and solve (14) as described in Alg. 1 (Line 3). We conclude each iteration by controlling the robot (Line 4) and shifting the trajectory $\boldsymbol{w}$ (Line 5).

Under mild assumptions CIAO*-MPC has recursive feasibility, refer to Appendix B due to space constraints. [3]

5. EXPERIMENTS

The performance and resulting behavior of CIAO* was investigated experimentally. A first set of experiments compares trajectories found by CIAO* to a time optimal reference. In a second set of experiments we compare the behavior of CIAO* for $\star \in \{1, 2, \infty\}$. A final set of experiments evaluates CIAO*-MPC's behavior. In all experiments we consider a circular robot with puck dynamics that is controlled through its jerks ($n_{\boldsymbol{x}} = 6, n_{\boldsymbol{u}} = 2$) with

$$A_{\mathrm{D}} = \begin{bmatrix} \mathbb{I}_2 & \Delta t \cdot \mathbb{I}_2, & \Delta t^2/2 \cdot \mathbb{I}_2 \\ & \mathbb{I}_2 & \Delta t \cdot \mathbb{I}_2 \\ & & \mathbb{I}_2 \end{bmatrix}, \quad B_{\mathrm{D}} = \begin{bmatrix} \Delta t^3/6 \cdot \mathbb{I}_2 \\ \Delta t^2/2 \cdot \mathbb{I}_2 \\ \Delta t \cdot \mathbb{I}_2 \end{bmatrix}. \tag{15}$$

The reasons for this choice are twofold: (1) the lemmata above consider linear dynamics, the applicability of CIAO^2 to nonlinear robots and dynamic environments has been demonstrated by Schoels et al. (2020), and (2) nonlinear dynamics would blur the comparison of different norms. Note that a more complex robot shape would not affect the number of distance function evaluations, but only increase the number of constraints in the NLPs.

The initial trajectories were computed from a path found by RRT. All NLPs were formulated as direct multiple-shooting in JuMP (Dunning et al., 2017) and solved by Gurobi (Gurobi Optimization, 2020) on an Intel Core i7-8559U clocked at 2.7 GHz running macOS Mojave.

[3] The Appendix is available at `https://arxiv.org/abs/2001.05449`.

	min	mean	median	max
time to goal [ratio]	1.015	1.024	1.020	1.038
path length [ratio]	0.984	0.999	1.000	1.074
control effort [ratio]	0.925	0.984	0.998	1.011
clearance [ratio]	0.999	0.999	0.999	0.999

Table 1. Evaluation of time optimal behavior approximation. We compute the normalized performance measures, obtained by taking the ratio between CIAO^2's solution and the time optimal reference (see examples in Fig. 1).

5.1 Evaluation of time optimality

To evaluate the quality of CIAO*'s approximation of time optimal behavior, it has been tested in 50 scenarios filled with 5 circular, randomly placed obstacles with radii between 1 and 2 m. The robot has to move from a point in the lower left of the environment through the obstacles to a point in the upper right corner. The values reported in Table 1 were obtained for CIAO^2. We note that CIAO^2 finds close to time optimal trajectories. In the considered scenarios it is always less than 4% slower. One reason, why CIAO^2 does not fully converge to the time optimal solution can be seen in the bottom left plot of Fig. 1. The circular boundaries of the CFRs prevent motion on the safety margin (denoted by the red lines) like done by the reference in that case. We note that the average path length and control effort is even lower than the time optimal solution, meaning that CIAO* finds shorter trajectories that require less control activation, but is slightly slower than the time optimal solution. At the same time it maintains the same or similar minimum distance to all obstacles, reported as clearance.

5.2 Comparison of CIAO^1, CIAO^2, and CIAO$^\infty$

For these experiments we use a similar setup as for the time optimal one. The only difference is that the obstacles are now a mixture of circles and rectangles (both with the same probability). The experimental results reported in Tables 2 & 3 were obtained on 50 simulated scenarios.

Looking at the trajectory quality comparison in Table 2, we observe that CIAO^1, CIAO^2, and CIAO$^\infty$ obtain similar results. CIAO^2 finds the fastest trajectories, while the shortest ones are found by CIAO^1. The reasons for these findings are visible in Fig. 4. In comparison CIAO^1 takes a more direct route at a lower average speed. It also maintains the highest clearance (measured by the Euclidean distance to the closest obstacle) due the diamond shape of the convex free region, which is tied to the L_1 norm. The diamonds are quite restrictive for diagonal movement, but become comparatively large in proximity of corners allowing for smooth maneuvering around corners (see Fig. 2 & 3, c_3). The L_∞ norm on the other hand finds large regions in tunnels and corridors (see Fig. 2, c_1) and is preferred for diagonal movement, but it is restrictive in proximity of corners (see Fig. 2 & 3, c_2 & c_3) which can result in detours (see Fig. 4). The failures of CIAO^1 and CIAO$^\infty$ result from passages that were to narrow to accommodate their CFRs.

In terms of computational efficiency CIAO$^\infty$ reaches the best performance. This has two reasons: First, the CIAO^1-

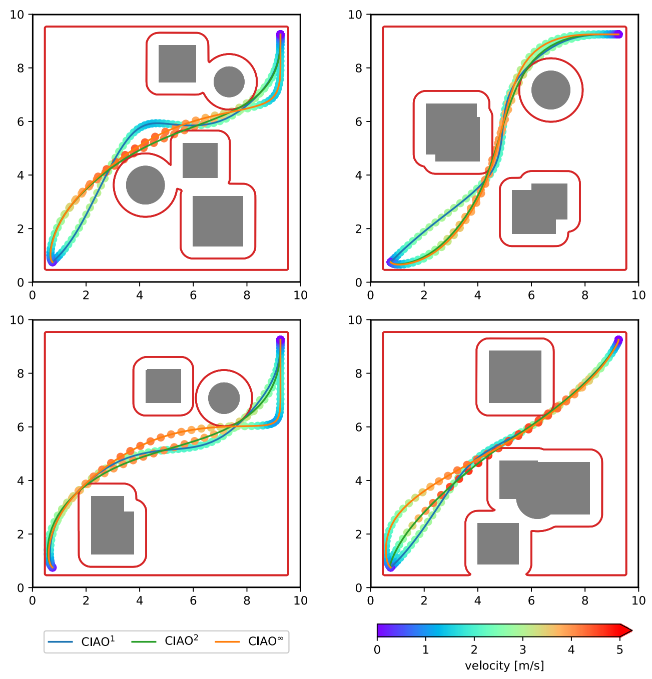

Fig. 4. CIAO* trajectories for different norms (values of $\star$): CIAO^1 (blue), CIAO^2 (green), and CIAO$^\infty$ (orange). For CIAO^2 the safety margin to obstacles ρ is marked in red. Note that the safety margins for CIAO^1 & CIAO$^\infty$ take different shapes, as depicted in Fig 2.

	CIAO^1	CIAO^2	CIAO$^\infty$
success rate	96%	**100%**	92%
time to goal [s]	5.80 (7.40)	**5.70** (**6.30**)	5.80 (7.20)
path length [m]	**13.569**	13.582	13.948
	(15.338)	(**15.249**)	(15.429)
clearance [m]	0.294	0.262	0.262

Table 2. Trajectory quality comparison for CIAO^1, CIAO^2, and CIAO$^\infty$. We report the median (and maximum) seconds and meters in simulation (see examples in Fig. 4) of the successful scenarios.

	CIAO^1	CIAO^2	CIAO$^\infty$
processing time [s]	1.709	1.079	**0.656**
	(23.033)	(5.598)	(**3.182**)
iterations	26 (398)	**8** (**45**)	10.5 (51)
time per iteration [s]	0.070	0.127	**0.064**
	(0.112)	(0.193)	(**0.098**)
iterations to feasible	2 (5)	2 (4)	2 (6)

Table 3. Computational effort of CIAO^1, CIAO^2, and CIAO$^\infty$ for the same experiments as in Table 2. We report the median (and maximum) computation time and iterations of the successful cases.

and CIAO$^\infty$-NLPs are linear programs, while the CIAO^2-NLP is a second order cone program (SOCP). The latter requires more computation time to solve resulting in a higher time per iteration. Second, CIAO$^\infty$ needs fewer iterations than CIAO^1. We note that all algorithms typically find a feasible solution within the first 2 iterations and CIAO^2 takes at most 4 iterations, which indicates fast convergence.

	CIAO^1	CIAO^2	CIAO$^\infty$
success rate	100%	100%	100%
time to goal [ratio]	1.193	**1.037**	1.089
	(2.246)	(**1.150**)	(1.463)
path length [ratio]	1.014	**0.993**	1.014
	(1.334)	(**1.084**)	(1.163)
control effort [ratio]	1.012	**0.992**	1.016
	(1.428)	(**1.076**)	(1.267)
clearance [ratio]	**1.208**	1.004	1.205
	(**1.955**)	(1.231)	(1.917)

Table 4. Trajectory quality evaluation. We report the median (and maximum) of normalized ratios between the solutions found by CIAO*-MPC and the time optimal reference.

	CIAO^1	CIAO^2	CIAO$^\infty$
processing time [s]	0.058	0.073	**0.043**
	(0.124)	(0.095)	(**0.050**)
solver time [s]	4.514e-04	0.017	**4.306e-04**
	(**7.024e-04**)	(0.026)	(9.505e-04)

Table 5. Computational effort of CIAO*-MPC for the same experiments as in Table 4. We report the median (and maximum) computation time per MPC step.

In summary CIAO^2 finds the fastest trajectories and is better suited for cluttered environments. CIAO^1 and CIAO$^\infty$ reach lower computation times per iteration, but are sensitive to the orientation and shape of obstacles. In our experiments they fail to steer the robot through some narrow passages due to the shape and size of their CFRs.

5.3 CIAO*-MPC Evaluation

To evaluate the trajectory quality lost due to the approximation introduced by CIAO*-MPC, we performed experiments on the same 50 scenarios considered in Sec. 5.1. The obtained results are reported in Tables 4 & 5. We use a horizon of 50 steps resulting in a total of 406 optimization variables (plus slacks).

We observe that the trajectories found by CIAO^2-MPC get closest to the time optimal reference and that they are at most 15% slower. CIAO^1 and CIAO$^\infty$ on the other hand find slower trajectories, due to some detours induced by the shapes of their CFRs. This behavior shows effect in all the path length, the time to goal and the clearance.

All, CIAO$^{\{1,2,\infty\}}$, reach processing times shorter than 125 ms per MPC step (see Table 5). Only a small fraction of this time is required for solving the CIAO*-NLP, most of it is consumed by an inefficient implementation of the distance function used for solving (7). For CIAO$^{\{1,\infty\}}$ the CIAO*-MPC-NLP (14) is a LP, which is solved in < 1 ms.

6. CONCLUSION

This paper presents CIAO*, a generalization of CIAO by Schoels et al. (2020) to predictable dynamic environments and arbitrary norms that approximates time optimal behavior. Evaluations in simulation show that CIAO* finds close to time optimal trajectories.

Future research will investigate a theoretical guarantees for nonlinear systems and unpredictable environments. It will also include a comparison to competing approaches and further study the convergence properties of CIAO*.

REFERENCES

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

Borenstein, J. and Koren, Y. (1991). The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Trans. Rob. Autom.*, 7(3), 278 – 288.

Dunning, I., Huchette, J., and Lubin, M. (2017). JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2), 295–320.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *Int. J. Rob. Res.*, 17(7), 760–772.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Rob. Autom. Mag.*, 4(1), 23 – 33. doi:10.1109/100.580977.

Gurobi Optimization, L. (2020). Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

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

Hyun, N.s.P., Vela, P.A., and Verriest, E.I. (2017). A new framework for optimal path planning of rectangular robots using a weighted l_p norm. *IEEE Robotics and Automation Letters*, 2(3), 1460–1465.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *Int. J. Rob. Res.*, 30(7), 846–894. doi:10.1177/0278364911406761.

Ko, N.Y. and Simmons, R.G. (1998). The lane-curvature method for local obstacle avoidance. In *IEEE/RSJ Int. Conf. Intell. Rob. Syst. (IROS)*, volume 3, 1615 –1621.

Lim, H., Kang, Y., Kim, C., Kim, J., and You, B.J. (2008). Nonlinear model predictive controller design with obstacle avoidance for a mobile robot. In *2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*. IEEE.

Minguez, J. and Montano, L. (2004). Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1), 45–59.

Otte, M. and Frazzoli, E. (2015). RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, 35(7), 797–822.

Quinlan, S. and Khatib, O. (1993). Elastic bands: Connecting path planning and control. In *IEEE Int. Conf. Rob. Autom. (ICRA)*, volume 2, 802–807.

Rösmann, C., Hoffmann, F., and Bertram, T. (2017). Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88, 142–153.

Schoels, T., Palmieri, L., Arras, K.O., and Diehl, M. (2020). An NMPC approach using convex inner approximations for online motion planning with guaranteed collision avoidance. In *IEEE Int. Conf. Rob. Autom. (ICRA)*.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *Int. J. Rob. Res.*, 33(9), 1251–1270.

Triebel, R., Arras, K., Alami, R., Beyer, L., Breuers, S., Chatila, R., Chetouani, M., Cremers, D., Evers, V., Fiore, M., et al. (2016). Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and service robotics*, 607–622. Springer.

Verschueren, R., Ferreau, H.J., Zanarini, A., Mercangöz, M., and Diehl, M. (2017). A stabilizing nonlinear model predictive control scheme for time-optimal point-to-point motions. In *IEEE Conf. Decis. Control (CDC)*.

Zhang, X., Liniger, A., and Borrelli, F. (2017). Optimization-based collision avoidance. *arXiv preprint arXiv:1711.03449*.

Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., and Srinivasa, S.S. (2013). CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *Int. J. Rob. Res.*, 32(9–10), 1164–1193.

Appendix A. TAYLOR UPPER BOUND

We want to guarantee continuous time constraint satisfaction for constraints that take the form $\|p(t) - c\| \leq r$, where $p(t) \in \mathbb{R}^n$ is an m-times differentiable function w.r.t. $t \in \mathbb{R}$ and $r \in \mathbb{R}$, $c \in \mathbb{R}^n$ are constant. This problem can be approached by deriving an upper bound for the expression $\|p(t) - c\|$. In a first step we take the Taylor expansion of p around the point $\bar{t}$:

$$p(t) = p(\bar{t}) + \underbrace{\sum_{i=1}^{m-1} p^{(i)}(\bar{t})\frac{(t-\bar{t})^i}{i!} + p^{(m)}(\tilde{t})\frac{(t-\bar{t})^m}{m!}}_{=\Delta p(t;\bar{t}) \text{ with } \tilde{t}\in[\bar{t},t]}, \quad \text{(A.1)}$$

where $p^{(i)}(\bar{t}) = \frac{\partial^i p}{\partial t^i}(\bar{t})$ for more compact notation. Inserting (A.1) into the initial expression and applying the triangle inequality we get

$$\begin{aligned} \|p(t) - c\| &= \|p(\bar{t}) - c + \Delta p(t,\bar{t})\| && \text{(A.2)} \\ &\leq \|p(\bar{t}) - c\| + \|\Delta p(t,\bar{t})\|\,. && \text{(A.3)} \end{aligned}$$

Under the assumption that the global upper bound of the m^{th} derivative of p is known and given by $\bar{p}^{(m)} = \max_t \left\|p^{(m)}(t)\right\|$, we obtain an upper bound for $\|\Delta p(t;\bar{t})\|$

$$\|\Delta p(t;\bar{t})\| \leq \left\|\sum_{i=1}^{m-1} p^{(i)}(\bar{t})\frac{(t-\bar{t})^i}{i!}\right\| + \bar{p}^{(m)}\frac{\|(t-\bar{t})^m\|}{m!}.$$

Note that $\bar{p}^{(m)}$ can be interpreted as a Lipschitz-constant. Assuming that $\|t - \bar{t}\|$ is bounded by $\overline{\Delta t}$ and applying the triangle inequality simplifies this expression to

$$\|\Delta p(t;\bar{t})\| \leq \sum_{i=1}^{m-1} \left\|p^{(i)}(\bar{t})\right\| \frac{\overline{\Delta t}^i}{i!} + \bar{p}^{(m)}\frac{\overline{\Delta t}^m}{m!}. \quad \text{(A.4)}$$

Finally this yields

$$\|p(t) - c\| \leq \|p(\bar{t}) - c\| + \underbrace{\sum_{i=1}^{m-1} \left\|p^{(i)}(\bar{t})\right\| \frac{\overline{\Delta t}^i}{i!} + \bar{p}^{(m)}\frac{\overline{\Delta t}^m}{m!}}_{=\overline{\Delta p}(\bar{t};p^{(1)},\ldots,p^{(m-1)},\overline{\Delta t})}. \quad \text{(A.5)}$$

Note that $\overline{\Delta p}(\bar{t};p^{(1)},\ldots,p^{(m-1)},\overline{\Delta t})$ is an upper bound on $\|\Delta p(t,\bar{t})\|$ that does not depend on t. It can thus be applied to reach continuous constraint satisfaction.