A Duality-based Approach for Real-time Obstacle Avoidance between Polytopes with Control Barrier Functions

Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath

Abstract—Developing controllers for obstacle avoidance between polytopes is a challenging and necessary problem for navigation in a tight space. Traditional approaches can only formulate the obstacle avoidance problem as an offline optimization problem. To address these challenges, we propose a duality-based safety-critical optimal control using control barrier functions for obstacle avoidance between polytopes, which can be solved in real-time with a QP-based optimization problem. The dual variables are introduced to represent polytopes and the Lagrangian function for the dual form is applied to construct a control barrier function. We demonstrate the proposed controller on a moving sofa problem where non-conservative maneuvers can be achieved in a tight space.

I. INTRODUCTION

A. Motivation

Achieving safety-critical navigation for autonomous robots in an environment with obstacles is a vital problem in the robotics research. Recently, control barrier functions (CBFs) together with QP-based optimizations have become a popular method to design safety-critical controllers. Among existing approaches of implementation, the robots and other surrounding obstacles are usually approximated as points, ellipses or hyper-spheres, and the distance between these shapes is an explicit analytic expression that is chosen as CBF. The obstacle avoidance performance under this approximation is usually conservative, as the shapes of the controlled robots or obstacles are usually over approximated. When robots or obstacles are approximated as polytopes, the obstacle avoidance maneuver becomes less conservative, but the signed distance function is implicit and cannot be used directly as a CBF. In this paper, we propose a novel duality-based approach to formulate the obstacle avoidance problem between polytopes into QP-based programs in the continuous domain using control barrier functions, which could then be deployed in real-time.

B. Related Work

1) Control Barrier Functions: One approach to provide safety guarantees for obstacle avoidance in control problems is to draw inspiration from control barrier functions [1]. CBF-QPs [2] permit us to find the minimum perturbation for a given feedback controller to guarantee safety. The method of CBFs can also be generalized for high-order systems [3], [4] and discrete-time systems [5], [6]. Specifically, the control barrier functions are widely used for obstacle avoidance with a variety of applications for autonomous robots, including autonomous cars [7], aerial vehicles [8] and legged robots [9]. The shapes of robots and obstacles are usually approximated as points [7], ellipses [10] or hyper-spheres [11], where the distance function can be calculated explicitly as an analytic expression from their geometric configuration. The distance functions for these shapes are differentiable and can be used as control barrier functions to construct safety-critical optimal control. However, these approximations usually over-estimate the dimensions of the robot and obstacles, e.g., a rectangle shape is approximated as circle containing it. When a tight fitting obstacle avoidance is expected, shown in Fig. 1, robots and obstacles are usually required to be approximated as polytopes. This makes maneuvers less conservative for obstacle avoidance, however, the distance between two polytopes requires additional effort of calculation [12], and since it is not in an explicit form, it cannot be used directly as a CBF.

2) Obstacle Avoidance between Polytopes: We narrow our discussions about obstacle avoidance between polytopes into optimization-based approaches. In [13], obstacle avoidance between rectangle-shape objects in an offline planning problem is studied, where collision avoidance is ensured by keeping all vertices of the controlled object outside the obstacle. Generally, when controlled objects are polyhedral,
the collision avoidance constraints can be reformulated with integer variables [14]. This method applies well for linear systems using mixed-integer programming but cannot be deployed as real-time controllers for general nonlinear systems due to the complexity from integer variables. The obstacle avoidance problem between convex regions could also be solved by using sequential programming [15], where penalizes collisions with a hinge loss is considered an offline optimization problem. Recently, a duality-based approach [16], [17], [18] was introduced to non-conservatively reformulate obstacle avoidance constraints as a set of smooth non-convex ones, which is validated on navigation problems in tight environments. This idea does optimize the computational time compared with others, but nonlinear non-convex programming is still involved and this approach can only be used for offline planning for nonlinear systems. To sum up, real-time obstacle avoidance between polytopes with convex programming for general nonlinear systems is still a challenging problem.

C. Contributions

The contributions of this paper are as follows:

- We propose a novel approach to reformulate a minimization problem for obstacle avoidance between polytopes into duality-based quadratic program with CBFs.
- The dual variables are introduced to represent polytopes and the Lagrangian function for the dual form is applied to construct a CBF. This CBF is used in a QP-based program for real-time safety-critical obstacle avoidance.
- The proposed approach is validated numerically for obstacle avoidance in the moving sofa problem [19] with nonlinear dynamics, where an L-shaped controlled object can maneuver safely in a tight L-shaped corridor, whose width is less than the diagonal length of the controlled object.

D. Paper Structure

The paper is structured as follows. Sec. II briefly introduces the background of control barrier functions. Sec. III describes the obstacle avoidance problem between two convex polytopes. In Sec. IV, the duality-based formulation for obstacle avoidance using CBFs is presented. In Sec. V, the proposed approach is validated with numerical results on the moving sofa problem and concluding remarks are briefly discussed in Sec. VI.

II. BACKGROUND

Consider a nonlinear, control affine system with state \( s \in \mathbb{R}^n \), input \( u \in \mathcal{U} \subset \mathbb{R}^m \) with dynamics:

\[
\dot{s} = f(s) + g(s)u
\]

where, \( f : \mathbb{R}^n \to \mathbb{R}^n \) and \( g : \mathbb{R}^n \to \mathbb{R}^{n \times m} \) are locally Lipschitz.

We define the safe set \( \mathcal{C} \) as the zero-superlevel set of a continuously differentiable function \( h : D \subset \mathbb{R}^n \to \mathbb{R} [1], \)

\[
\mathcal{C} := \{ s \in D : h(s) \geq 0 \}. \tag{2}
\]

\( h \) is a control barrier function if there exists an extended class \( K_\infty \) function \( \alpha \) such that for all \( s \in D, \)

\[
\sup_{u \in \mathcal{U}} [L_f h(s) + L_g h(s)u] \geq -\alpha(h(s)), \tag{3}
\]

where \( L_f h(s) \) and \( L_g h(s) \) are Lie derivatives of \( h(s) \) along \( f(s) \) and \( g(s) \) respectively, and

\[
h(s, u) = L_f h(s) + L_g h(s)u. \tag{4}
\]

Further, if \( \frac{\partial h}{\partial s} \neq 0 \) \( \forall s \in \partial \mathcal{C} \), any Lipschitz controller \( k(s) \) satisfying (3) renders \( \mathcal{C} \) forward invariant. If \( \mathcal{C} \) lies within our allowable set of states, then \( s(t) \) remains in the allowable set of states for the entire trajectory, and thus the closed loop system is safe.

Let \( u_{\text{nom}}(s) \) be any nominal controller, which could be a tracking controller, a stabilizing controller, or a controller obtained using a control Lyapunov function (CLF). In particular, \( u_{\text{nom}}(s) \) need not guarantee safety. Then, imposing the CBF constraint

\[
h(s, u) \geq -\alpha(h(s)) \tag{5}
\]

is a way to guarantee safety by minimally restricting the nominal controller [2]. This new controller can be written as a QP as follows:

\[
u(s) = \arg \min \| u - u_{\text{nom}}(s) \|^2_2 \quad \text{s.t.} \quad L_f h(s) + L_g h(s)u \geq -\alpha(h(s)), \quad u \in \mathcal{U}. \tag{6}
\]

A feasible optimal solution from the above CFB-QP generates a safe trajectory by guaranteeing invariance of \( \mathcal{C} \) as in (2).

III. PROBLEM STATEMENT

We consider \( N \) nonlinear, control affine systems, with states \( s_i \in \mathbb{R}^n \) and dynamics:

\[
\dot{s}_i = f_i(s_i) + g_i(s_i)u_i, \quad i \in \{1, \ldots, N\}, \tag{7}
\]

where, \( u_i \in \mathbb{R}^m \) and \( f_i : \mathbb{R}^n \to \mathbb{R}^n \) and \( g_i : \mathbb{R}^n \to \mathbb{R}^{n \times m} \) are locally Lipschitz. While, the dimensions of the states and inputs for each system can be different, however we assume them to be the same across for simplicity.

For each state \( s_i \in \mathbb{R}^n \) of the \( i \)-th system, we define the polytope \( \mathcal{P}_i(s_i) \) as the physical domain associated to the system at state \( s_i \), with

\[
\mathcal{P}_i(s_i) := \{ x \in \mathbb{R}^n : A_i(s_i)x \leq b_i(s_i) \}, \tag{8}
\]

where \( A_i : \mathbb{R}^n \to \mathbb{R}^{m \times n} \) and \( b_i : \mathbb{R}^n \to \mathbb{R}^{m \times 1} \) are continuously differentiable. \( A_i(s_i) \) and \( b_i(s_i) \) are such that \( \mathcal{P}_i(s_i) \) is always non empty, and the number of rows of \( A_i(s_i) \) is constant for all \( s_i \).

We want to design a safety-critical controller such that none of the polytopes intersect with each other over their trajectories, i.e. \( \mathcal{P}_i(s_i(t)) \cap \mathcal{P}_j(s_j(t)) = \emptyset \) for \( \forall t \geq 0, \forall i \neq j \). Such a controller can be used for obstacle avoidance, when \( \mathcal{P}_i(s_i) \) would represent the physical domain of the \( i \)-th robot.

In other words, the minimum distance between any two points from a pair of polytopes should be strictly greater than
We assume that $A_i(s_i)$ and $P_j(s_j)$ is defined as the control barrier function $h_{ij}(s_i, s_j)$, which can be computed using the following QP:

$$h_{ij}(s_i, s_j) := \min_{\{x, y\}} \|x - y\|_2^2$$

s.t. $A_i(s_i)x \leq b_i(s_i)$,
$$A_j(s_j)y \leq b_j(s_j),$$
$$x, y \in \mathbb{R}^n.$$  \hfill (9)

Note that compared to the prior work, the distance $h_{ij}$ is implicit and is a solution of the minimization problem. Variables $x$ and $y$ denote points inside the polytopes $P_i(s_i)$ and $P_j(s_j)$ respectively. Also, the cost is not strictly convex in $x$ and $y$. We define $C_{ij}$ as the zero-superlevel set of $h_{ij}$, i.e.,

$$C_{ij}(s_i, s_j) := \{(s_i, s_j) : h_{ij}(s_i, s_j) \geq 0\}$$  \hfill (10)

Remark 1. For the simplicity of discussion, the later discussions will be firstly illustrated for 2 systems $i$ and $j$, and results will be generalized where necessary. Further, for simplicity of notation, we denote $(s_i, s_j) := s, h_{ij}(s_i, s_j) := h(s), C_{ij}(s_i, s_j) := C(s), \text{ and } (u_i, u_j) := u.$

Remark 2. Since $h(s)$ is quadratic in nature, its gradient (if it exists) can be zero at $\partial C$, which can affect forward invariance of $C$ as mentioned in Sec. II. So, we consider an $\epsilon > 0$ and the $c^2$-level set of $h(s)$ as:

$$L_{c^2} = \{s \in \mathbb{R}^{2n} : h(s) = c^2\}$$  \hfill (11)

We assume that $A_i, b_i \forall i$ are such that $\frac{\partial h}{\partial s_i} \neq 0 \forall s \in L_{c^2}$. We can then redefine $h(s)$ as $(h(s) - c^2)$. If the new $h$ satisfies (3), then the $c^2$-superlevel set is safe. Thus, we can directly replace $h(s)$ with $(h(s) - c^2)$ in the controller derived in the later proposed formulation.

IV. DUAL FORMULATION

In this section, we will illustrate a general duality-based approach of imposing CBF constraints.

To impose the CBF constraint, we need to differentiate $h(s)$, which is the optimal value of (9). Then, we can write a CBF-QP similar to (6) to implement a controller.

Remark 3. Although $h(s)$ need not be differentiable or Lipschitz continuous in $s$, directional derivatives of $h(s)$ exist. For a given differentiable trajectory $s(t)$, the $\dot{h}(t)$ is the directional derivative of $h(s)$ along $\dot{s}(t)$, and so $h(s(t))$ is right differentiable [20]. However, $h(s(t))$ need not be continuous.

A. Dual of the Minimum Distance QP

Loosely speaking, since $h(s)$ in (9) is a minimization problem in the space variables $x$ and $y$, $\hat{h}(s, u)$ can be written as another minimization problem in $\hat{x}$ and $\hat{y}$. Since $\hat{h}(s, u)$ now is a solution of an optimization problem we cannot directly use it in a constraint to the enforce the CBF constraint (5). We could indirectly enforce the CBF constraint with some upper bound $\hat{h}$ of $\hat{h}(s, u)$. If we can find a lower bound of $\hat{h}(s, u)$, enforcing the CBF constraint on the lower bound would also impose the CBF constraint (5) on $\hat{h}(s, u)$.

The dual program of a minimization problem is a maximization problem in terms of the corresponding dual variables. For the QP-based program in (9), the dual program has the same optimal solution as that of (9). So, differentiating the dual program will give $\hat{h}(s, u)$ as a maximization problem. This then leads us to use the dual program of (9) to get a lower bound of $\hat{h}(s, u)$.

The Lagrangian function for (9) is as follows [21]:

$$\Lambda(x, y, \lambda_1, \lambda_2) = \|x - y\|^2 + \lambda_1(A_i(s_i)x - b_i(s_i)) + \lambda_2(A_j(s_j)y - b_j(s_j)),$$  \hfill (12)

where $\lambda_1, \lambda_2 \in \mathbb{R}^{1 \times m_i}$ are the dual variables. The Lagrangian dual function can be written in (13):

$$L(\lambda_1, \lambda_2) = \inf_{x, y \in \mathbb{R}^n} (\Lambda(x, y, \lambda_1, \lambda_2)),$$  \hfill (13)

together with the Weak Duality Theorem [21], we have,

$$L(\lambda_1, \lambda_2) \leq h(s), \ \forall \lambda_1, \lambda_2 \geq 0.$$  \hfill (14)

As the constraints in (9) are affine in $x$ and $y$ at any given time, constraint qualification holds and the Strong Duality Theorem can be applied, resulting in

$$\max_{\lambda_1, \lambda_2 \geq 0} L(\lambda_1, \lambda_2) = h(s).$$  \hfill (15)

$L(\lambda_1, \lambda_2)$ can be explicitly computed. Let the minimizer for (13) be $x, y$. Then,

$$\frac{\partial \Lambda}{\partial x} = 0 \Rightarrow 2(x - y)^T + \lambda_1A_i(s_i) = 0,$$  \hfill (16)

$$\frac{\partial \Lambda}{\partial y} = 0 \Rightarrow 2(y - x)^T + \lambda_2A_j(s_j) = 0.$$  \hfill (17)

Hence, for the minimum of (13) to exist, from (16) the following must hold:

$$\lambda_1A_i(s_i) + \lambda_2A_j(s_j) = 0,$$  \hfill (18)

Substituting the value of $(x - y)$ from (17) in (12), we have

$$L(\lambda_1, \lambda_2) = -\frac{1}{2} \lambda_1A_i(s_i)^T\lambda_1 - \lambda_1b_i(s_i) - \lambda_2b_j(s_j).$$  \hfill (19)

Remark 4. If $P_i(s_i) \cap P_j(s_j) = \emptyset$, then $\exists (\lambda_1, \lambda_2)$ such that $\lambda_1b_i(s_i) + \lambda_2b_j(s_j) < 0$ and $\lambda_1A_i(s_i) + \lambda_2A_j(s_j) = 0$. Here $c = \lambda_1A_i(s_i)$ is the normal vector to a corresponding separating plane. This property is analyzed in [22].

Therefore, the following QP-based dual program can be used to compute $h(s)$:
\[ h(s) = \max_{(\lambda_1, \lambda_2)} \frac{1}{4} \lambda_1 A_i(s_i) A_i(s_i)^T \lambda_1^T - \lambda_1 b_i(s_i) - \lambda_2 b_j(s_j) \]
\[ \text{s.t.} \quad \lambda_1 A_i(s_i) + \lambda_2 A_j(s_j) = 0, \quad \lambda_1, \lambda_2 \geq 0. \] (19)

We note that since an optimal solution to (9) always exists, an optimal solution to (19) also always exists.

**B. Differentiability of Feasible Dual Solutions**

In order to enforce the CBF constraints, we want to differentiate \( h(s) \) obtained from (19). However, since \( h(s) \) is the solution of a maximization problem over \( \lambda_i \), we also expect \( h(s, u) \) to be the solution of a maximization over \( \lambda_i \).

Let \( s(\tau) \) be a differentiable trajectory in \( C \), as defined in (10), for \( \tau \in [t, t + \Delta t) \) and for some \( u(\tau) \). As a shorthand, \( h(s(\tau)) \) will be abbreviated as \( h(t) \) when appropriate, and similarly for others.

Let \( (\lambda_1^*(t), \lambda_2^*(t)) \) be some set of optimal solution of (19) for time \( t \). In order to differentiate the cost and constraints of (19) about the optimal dual variables, we want the optimal dual solutions to be differentiable.

Therefore, we make the following two assumptions on the differentiability of optimal and feasible dual solutions:

**Assumption 1:** There exists at least one set of optimal dual solutions \( (\lambda_1(\tau), \lambda_2(\tau)) \) of (19) such that \((\lambda_1(t), \lambda_2(t)) = (\lambda_1^*(t), \lambda_2^*(t))\), and \((\lambda_1(\tau), \lambda_2(\tau)) \) is continuous in \([t, t + \Delta t)\) and right differentiable at \( \tau = t \), for some \( \Delta t > 0 \).

**Assumption 2:** For each \((\lambda_1, \lambda_2)\) satisfying,
\[ \dot{\lambda}_1 A_i(t) + \lambda_1^*(t) \dot{A}_i(t) + \dot{\lambda}_2 A_j(t) + \lambda_2^*(t) \dot{A}_j(t) = 0, \] (20)
\[ [\dot{\lambda}_1]_k \geq 0 \quad \text{if} \quad [\lambda_1^*(t)]_k = 0, \]
\[ [\dot{\lambda}_2]_k \geq 0 \quad \text{if} \quad [\lambda_2^*(t)]_k = 0, \]
there exists at least one set of feasible dual solutions \( (\lambda_1^*(\tau), \lambda_2^*(\tau)) \) of (19) that is continuous in \([t, t + \Delta t)\) and \((\lambda_1, \lambda_2) = (\lambda_1^*(t), \lambda_2^*(t))\). Here, \([\cdot]_k\) represents the \(k\)-th row of a matrix.

Note that (20) is the derivative of the constraints of (19) about \((\lambda_1^*(t), \lambda_2^*(t))\).

**C. Optimal Control for Obstacle Avoidance**

We can now write \( \dot{h}(t) \) as the optimal value of a linear program as follows:

**Lemma 1.** Under the assumptions IV-B and IV-B, \( \dot{h}(t) \) can be computed as the optimal solution of the following LP:
\[ \dot{h}(t) = \max_{(\lambda_1, \lambda_2)} \frac{1}{2} \dot{L}(t, \lambda_1, \lambda_2, \dot{\lambda}_1, \dot{\lambda}_2) \]
\[ \text{s.t.} \quad \dot{\lambda}_1 A_i(t) + \lambda_1^*(t) \dot{A}_i(t) + \dot{\lambda}_2 A_j(t) + \lambda_2^*(t) \dot{A}_j(t) = 0, \] (21)
\[ [\dot{\lambda}_1]_k \geq 0 \quad \text{if} \quad [\lambda_1^*(t)]_k = 0, \]
\[ [\dot{\lambda}_2]_k \geq 0 \quad \text{if} \quad [\lambda_2^*(t)]_k = 0. \]

where,
\[ \dot{L}(t, \lambda_1, \lambda_2, \dot{\lambda}_1, \dot{\lambda}_2) = -\frac{1}{2} \dot{\lambda}_1 A_i(t) A_i(t)^T \lambda_1^T - \dot{\lambda}_1 b_i(t) \]
\[ -\frac{1}{2} \lambda_1 A_i(t) \dot{A}_i(t)^T \lambda_1^T - \dot{\lambda}_2 b_j(t) - \lambda_2 A_j(t)^T \lambda_2^T - \lambda_2 b_j(t). \] (22)

**Proof.** See Appendix A.

Based on the linear program in (21), we can conservatively implement the CBF constraint by enforcing (for some \((\lambda_1, \lambda_2)\) satisfying (20)):
\[ \dot{L}(t, \lambda_1^*, \lambda_2^*, \dot{\lambda}_1, \dot{\lambda}_2) \geq -\alpha(h(t)). \] (23)

Lem. 1 then guarantees that
\[ \dot{h}(t) \geq -\alpha(h(t)) \] (24)
since \( \dot{h}(t) \) is the maximum among all \( \dot{L} \). The input \( u(t) \) implicitly affects \( \dot{L} \) via the derivatives of the boundary matrices \( A_i, A_j, b_i, b_j \). We note that \( \dot{L} \) is affine in \( \lambda_1, \lambda_2, \) and \( u \). So, (23) is a linear constraint in \( \lambda_1, \lambda_2, \) and \( u \).

So, at each time \( t \), (19) is used to compute \( h(t) \), \( \lambda_1^*(t) \), and \( \lambda_2^*(t) \) and the followingQP is used to compute inputs:

**Polytope-CBF-QP:**
\[ u(t) = \arg\min_{\{u, \lambda_1, \lambda_2\}} \| u - u_{\text{nom}}(t) \|^2 \]
\[ \text{s.t.} \quad \dot{L}(t, \lambda_1^*, \lambda_2^*, \dot{\lambda}_1, \dot{\lambda}_2, u) \geq -\alpha(h(t) - \epsilon^2), \]
\[ \lambda_1 A_i(t) + \lambda_1^* \dot{A}_i(t, u) \]
\[ + \lambda_2 A_j(t) + \lambda_2^* \dot{A}_j(t, u) = 0, \]
\[ [\dot{\lambda}_1]_k \geq 0 \quad \text{if} \quad [\lambda_1^*(t)]_k = 0, \]
\[ [\dot{\lambda}_2]_k \geq 0 \quad \text{if} \quad [\lambda_2^*(t)]_k = 0. \] (25)

This formulation can be extended to more than 2 robots by including the CBF constraints and the constraints in (20), for each pair of robots.

The CBF safety requirement is enforced in the constraints of (25). So, the safety of the system is not affected by the cost function. Further, as discussed in Sec. III, we substitute \( h(t) \) with \( (h(t) - \epsilon^2) \) for a small value of \( \epsilon > 0 \) to avoid gradient of \( h(s) \) becoming zero.

Note that \( u(t) \) is a feedback controller since it implicitly depends on \( s(t) \) and enforces \( h(s(t)) \geq -\alpha(h(s(t))) \).

**Remark 5.** The feedback controller \( u(s(t)) \) in (25) need not necessarily satisfy sufficient conditions for continuity or Lipschitz continuity, such as in [23]. Then, (7) can be seen as a differential inclusion where \( u(s(t)) \) can take any value in a set, and the closed loop system may not have a unique local solution. As long as the elements of this set are feasible solutions of (25), \( \dot{L}(t, \lambda_1^*, \lambda_2^*, \dot{\lambda}_1, \dot{\lambda}_2, u) \geq -\alpha(h(t) - \epsilon^2) \). So, \( h(t) \geq -\alpha(h(t) - \epsilon^2) \) irrespective of the input at \( t \).

If \( h(0) \geq \epsilon^2 \) then \( h(t) \geq \epsilon^2 \), and the system is safe for all possible trajectories.
We note the general problem description in Sec. III allows for such static obstacles by eliminating the dependence of $A_i^W$ and $b_i^W$ on the state.

Similarly, the sofa is represented as:

$$\mathcal{P}_j(s) := \{ x \in \mathbb{R}^2 : A_i^S(s)x \leq b_i^S(s) \} \ j \in \{1, 2\}$$

Again, our description allows for this by choosing the same states and inputs for two polytopes.

The CBF is chosen as the square of the minimum distance as in (9). Notice that the QP-based program will always have a solution, as $v = 0$, $\omega = 0$ is always a feasible solution for the QP-based program. Since we have static obstacles and the controlled object consists of two polytopes, we don’t need a CBF between every pair. We only enforce CBF constraints between $W_i$ and $P_j$ for $i \in \{1, 2, 3\}, j \in \{1, 2\}$. The margin $\epsilon$ as defined in Sec. III is chosen as 1.5 cm.

A control Lyapunov function constraint with a slack variable is introduced in the final QP formulation to substitute for the nominal controller. The slack variable ensures that the feasibility of the QP-based program is not affected by the CLF constraint. The final position of the sofa is chosen to be at the end of the corridor. The initial and final orientations are chosen as $\frac{\pi}{4}$ and $-\frac{\pi}{4}$ respectively.

**B. Results**

The simulations are performed on a MacBook Pro with a 2.8 Ghz Intel Core i7 processor running IPOPT [24] on MATLAB, and the visualization of polytopes are generated by MPT3 [25]. The snapshots of the sofa trajectory is shown in Fig. 1 and the corresponding animation video can be found in [1].

1) **Enforcement of CBF constraint:** Fig. 3 shows the values of $h(t)$, and LHS and RHS terms in the CBF constraints from (25). We can see that by Lem. 1, the LHS term in the CBF constraint is always a lower bound to $\dot{h}(t)$. This verifies the safety property of our duality-based approach.

**TABLE I: Computation time per iteration**

<table>
<thead>
<tr>
<th>mean (ms)</th>
<th>std (ms)</th>
<th>min (ms)</th>
<th>max (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>16</td>
<td>19.5</td>
<td>1130</td>
</tr>
</tbody>
</table>

2) **Computation time:** The optimization problem constructed for the sofa problem has 51 variables and at most 72 constraints. From Table I, we can see that there are large outliers in computation time per iteration. It is possible that these spikes are due to $h(t)$ approaching close to zero. As $h(t)$ approaches zero, both sides of the CBF constraint become zero and it needs more iterations in the solver to find feasible solutions. Nevertheless, from Table I, we can apply our controller at 50Hz. We have also observed that distances obtained as the optimal value of a QP can also become negative due to numerical errors. In this case, the QP might become infeasible.

3) **Continuity of $\hat{h}(t)$:** Fig. 3 also shows that both $\dot{h}(t)$ and the lower bound of $\hat{h}(t)$ can have discontinuities. As discussed in Rem. 3, $\dot{h}(t)$ need not be continuous. For the case of the moving sofa problem, this discontinuity can arise when the sofa is rotating and the point of minimum distance

1<https://youtu.be/-5nkztMv8IY>
on the sofa with any wall jumps from one arm to another. Since the end points of the sofa arm need not have the same velocity when rotating, there can be a discontinuity in \( \dot{h}(t) \).

4) **Deadlocks**: For various initial conditions, the sofa can get stuck in a deadlock in the corridor. This can happen when the arms of the sofa are so large that it cannot turn at the corner. It can also happen when the two arms of the sofa get too close to the wall and the sofa cannot turn because it would cause one of its arms to penetrate the wall. Our controller still ensures safety in this case. A high-level planner could help in deadlock free trajectory at low frequency that serves as input to our control law.

**C. Discussions**

1) **Nonlinearity of the system dynamics**: We note that the duality-based formulation (25) is a convex quadratic program even when the system dynamics is nonlinear, as long as it is control affine.

2) **Optimality of solution vs computation time**: As noted in Sec. IV, the cost function of the QP-based program (25) does not affect the safety of the system. If the optimization solver does not converge to the optimal solution, the current solution can be used if it is feasible. This can be useful in real-time implementations where both input frequency and safety matter. A feasible solution to (25) can be directly found in some cases, such as when the CBF is constructed using a safe backup controller [26].

3) **Trade-off between computation speed and tight maneuvering**: A polytope with \( m_i \) faces would require \( m_i \) dual variables in the dual formulation (25) and additional constraints. Using a hyper-sphere as an over-approximation to the polytope would require fewer dual variables, as in [4]. However, such an approximation can be too conservative and completely ignore the rotational geometry of the polytope. On the other hand, using the full polytope structure can prove beneficial in dense environments, such as the sofa problem, where a spherical approximation would not work. So, there is a trade-off between computation speed and maneuverability. In practice, a hybrid approach should be used: a hyper-sphere approximation when two obstacles are far away, and the polytope structure when closer.

4) **Computation time vs. number of faces of a polytope**: As the number of faces of a polytope increases, so does the number of dual variables. For the duality-based formulation (25), as the number of faces of any polytope increase, both the number of constraints and variables of (25) increase linearly, and thus the computation time also increases.

5) **Robustness with respect to the safe set**: Since we use the minimum distance between two polytopes as the CBF and not the signed distance, the minimum distance is uniformly zero when two polytopes intersect. So, the proposed controller is not robust in the sense that if the state leaves the safe set, it will not converge back to it. If due to numerical errors the systems leaves the safe set, the \( \epsilon \) margin can still make the state converge back to the safe set.

6) **Numerical errors**: If the controller is implemented in a discretized first-order hold form and the time step is too large, the obstacles can intersect each other. Intersection can occur due to large velocities and discontinuous \( \dot{h}(t) \). The maximum velocity of any point on an obstacle can be calculated using the velocity and angular velocity bounds, and the diameter of the corresponding polytopic domain. The maximum velocity should be such that the maximum possible distance any point on an obstacle can cover is less than the \( \epsilon \) margin. \( \epsilon \) should be judiciously chosen so that the system is safe but, not too conservative. This arises due to the discretization of the problem and using a fixed-time integration scheme. This issues can be potentially prevented by considering the discrete-time version of this problem directly.

**VI. CONCLUSION**

In this paper, we present a general framework for obstacle avoidance between polytopes using the dual program of a minimum distance QP problem. We show that the control input using our method can be computed using a QP-based program for systems with control affine dynamics, enabling real-time implementation. We numerically verify the safety performance of our controller for the problem of moving an L-shaped sofa through a tight corridor. In the future, we will generalize our framework and provide proofs for the assumptions. We also have explored properties such as robustness and deadlock avoidance.

**REFERENCES**

A. Proof of Lem. 1

Let $\mathcal{F}$ be a family of functions defined as:

$$
\mathcal{F} := \{(\lambda_1, \lambda_2) \in \mathcal{C}^0[0, t+\Delta t] \}^2 : \lambda_1(t), \lambda_2(t) \geq 0; \\
\lambda_1(\tau)A_1(\tau) + \lambda_2(\tau)A_2(\tau) = 0; \\
(\lambda_1(t), \lambda_2(t)) = (\lambda_1(t), \lambda_2(t)); \\
(\lambda_1(\tau), \lambda_2(\tau)) \text{ is right differentiable at } \tau = t; \\
\forall \tau \in [t, t+\Delta t]
$$

Clearly, every $(\lambda_1, \lambda_2) \in \mathcal{F}$ satisfies (20) for $\tau = t$. Additionally, by Assumption IV-B, every $(\lambda_1, \lambda_2)$ satisfying (20) has a corresponding function in $\mathcal{F}$. So, $(\lambda_1, \lambda_2)$ satisfies (20) if and only if there is a corresponding function in $\mathcal{F}$.

1) $\mathcal{F}$ is non-empty: By assumption IV-B, there exists at least one set of optimal dual solutions $(\lambda_1(\tau), \lambda_2(\tau))$ of (19) such that $(\lambda_1(\tau), \lambda_2(\tau)) = (\lambda_1^*(\tau), \lambda_2^*(\tau))$, and $(\lambda_1(\tau), \lambda_2(\tau))$ continuous in $[t, t+\Delta t]$ and right differentiable at $\tau = t$. Since the optimal dual solution satisfies the constraints of (19) for all times, $(\lambda_1(\tau), \lambda_2(\tau)) \in \mathcal{F}$. So, $\mathcal{F} \neq \emptyset$.

2) Lower bound of $\dot{h}(t)$: Since $(\lambda_1^*(t), \lambda_2^*(t))$ is optimal, $L(t, \lambda_1^*(t), \lambda_2^*(t)) = h(t)$, by the Weak Duality Theorem. For $(\lambda_1, \lambda_2) \in \mathcal{F}$, we get

$$
L(\tau, \lambda_1(\tau), \lambda_2(\tau)) \leq h(\tau) \quad \forall \tau \in [t, t+\Delta t],
$$

and

$$
L(t, \lambda_1(t), \lambda_2(t)) = h(t).
$$

For $(\lambda_1, \lambda_2) \in \mathcal{F}$, by using (30) and (31), we obtain

$$
\frac{L(t, \lambda_1(\tau), \lambda_2(\tau)) - L(t, \lambda_1(t), \lambda_2(t))}{\tau - t} \leq \frac{h(\tau) - h(t)}{\tau - t}.
$$

Taking the limit $\tau \to t$ on both sides and by the Squeeze Theorem,

$$
\dot{L}(t, \lambda_1(t), \lambda_2(t)) \leq \dot{h}(t).
$$

3) Upper bound of $\dot{h}(t)$: Since $(\lambda_1^*(t), \lambda_2^*(t))$ is differentiable at $\tau = t$,

$$
\dot{L}(t, \lambda_1^*(t), \lambda_2^*(t)) = \dot{h}(t).
$$

Additionally, we have $(\lambda_1^*(t), \lambda_2^*(t)) \in \mathcal{F}$. Hence,

$$
\max_{(\lambda_1, \lambda_2) \in \mathcal{F}} \dot{L}(t, \lambda_1(t), \lambda_2(t), \dot{\lambda}_1(t), \dot{\lambda}_2(t)) \geq \dot{h}(t)
$$

4) LP for $\dot{h}(t)$: By (33) and (35),

$$
\max_{(\lambda_1, \lambda_2) \in \mathcal{F}} \dot{L}(t, \lambda_1(t), \lambda_2(t), \dot{\lambda}_1(t), \dot{\lambda}_2(t)) = \dot{h}(t)
$$

$(\lambda_1, \lambda_2) \in \mathcal{F}$ if and only if $(\dot{\lambda}_1(t), \dot{\lambda}_2(t))$ satisfies (20). So, (21) holds.