

Robust Path Planning and Control For Polygonal Environments via Linear Programming

Mahroo Bahreinian¹, Erfan Aasi² and Roberto Tron³

Abstract—We propose a novel approach for navigating in polygonal environments by synthesizing controllers that take as input relative displacement measurements with respect to a set of landmarks. Our algorithm is based on solving a sequence of robust min-max Linear Programming problems on the elements of a cell decomposition of the environment. The optimization problems are formulated using linear Control Lyapunov Function (CLF) and Control Barrier Function (CBF) constraints, to provide stability and safety guarantees, respectively. The inner maximization problem ensures that these constraints are met by all the points in each cell, while the outer minimization problem balances the different constraints in a robust way. We show that the min-max optimization problems can be solved efficiently by transforming it into regular linear programming via the dualization of the inner maximization problem. We test our algorithm to agents with first and second order integrator dynamics, although our approach is in principle applicable to any system with piecewise linear dynamics. Through our theoretical results and simulations, we show that the resulting controllers: are optimal (with respect to the criterion used in the formulation), are applicable to linear systems of any order, are robust to changes to the start location (since they do not rely on a single nominal path), and to significant deformations of the environment.

I. INTRODUCTION

Path planning is a major research area in the context of mobile robots, as it deals with the problem of finding a path from an initial state toward a goal state while considering collision avoidance. Traditional path planning methods focus on finding *single nominal paths* in a given *known map*, and the majority of them makes the implicit assumption that the agent possesses a lower-level *state feedback* controller for following such nominal path in the face of external disturbances and imperfect models. Biological system do not rely on the same restrictive assumptions; for instance, consider a person navigating in an unfamiliar room: despite the fact that the person does not have a precise blueprint of the floor, and does not know its precise location, they can reliably and robustly navigate toward a desired door, from any location in the room. While this ability in biological systems is the result of complex and not fully understood mechanisms, in this paper we aim to narrow the gap between planning algorithms and biological systems by synthesizing *output-feedback controllers* that

This work was supported by ONR MURI N00014-19-1-2571 “Neuro-Autonomy: Neuroscience-Inspired Perception, Navigation, and Spatial Awareness”

¹Mahroo Bahreinian is with Division of Systems Engineering at Boston University, Boston, MA, 02215 USA. Email: mahroobh@bu.edu

²Erfan Aasi is with Department of Mechanical Engineering at Boston University, Boston, MA, 02215 USA. Email: eaasi@bu.edu

³Roberto Tron is with Faculty of Department of Mechanical Engineering at Boston University, Boston, MA, 02215 USA. Email: tron@bu.edu

are robust to *imprecise map knowledge*. By synthesizing controllers instead of specific paths, we more tightly integrate the high-level path planning and the low-level regulation tasks, allowing the agent to cope with disturbances without replanning; moreover, the focus on the controllers allows us to plan by directly using measurements (outputs) available to the agent, instead of assuming full state knowledge; finally, since the controllers depend on the environment indirectly (through measurements that are taken online), we empirically show that such controllers are robust to (often very significant) changes in the map. In order to pursue strong theoretical guarantees, in this paper we assume agents with controllable linear dynamics, and environments that admit a polygonal convex cell decomposition (e.g., via Delaunay triangulations [9] or trapezoidal decompositions [22]). Methods to address these limitations are planned as part of our future work (see also the Conclusions section).

Previous work. Existing works on path planning can be roughly classified into two categories: combinatorial path planning methods, and sample-based path planning methods [11]. Some of the path planning methods consider a continuous model for the environment and therefore provide a continuous path, such as potential fields [18], [20] and navigation functions [27], while the other group solves the planning problem by abstracting the environment to a finite representation and find a discrete path, such as probabilistic roadmaps [16] and cell decomposition methods [25].

One of the well known combinatorial path planning algorithms is cell decomposition where a complex environment is decomposed into a set of cells, avoiding obstacles by planning straight paths in individual cells; for each individual step, traditional methods use midpoints [7], [23], [28], while more recent solutions aim to optimize path length [19]. Our work can be seen as a descendant of previous work that handles the cell decomposition vis-à-vis the continuous dynamic through a hybrid system perspective by synthesizing a state-feedback controller for each cell. Initial work proposed potential-based controllers [8], while others characterize the theoretical conditions [13] and closed-form solutions [3] for linear affine controllers. Although the latter approaches were extended to nonlinear systems in [10] and to uncertain maps [32] (using intelligent re-planning), they all assume that each cell in the decomposition is a *simplex* (a polytope in \mathbb{R}^d with $d + 1$ vertices, e.g., a 2-D triangle). In contrast, our method can handle arbitrary convex polytopes, and design *output-feedback controllers* (instead of state-feedback).

Sampling-based planning algorithms, such as rapidly exploring random tree (RRT), have become popular in last

few years due to their good practical performance, and their probabilistic completeness [15], [23], [24]. For trajectory planning that takes into account non-trivial dynamical systems of the robot, kinodynamic RRT [23], [24] and closed-loop RRT (CL-RRT, [21]) and CL-RRT# grow the tree by sampling control inputs and then propagating forward the nonlinear dynamics (with the optional use of stabilizing controllers and tree rewiring to approach optimality). Further in this line of work, there has been a relatively smaller amount of works on algorithms that focus on producing controllers as opposed to simple reference trajectories.

The `safeRRT` algorithm [6], [30] generates a closed-loop trajectory from initial state to desired goal by expanding a tree of local state-feedback controllers to maximize the volume of corresponding positive invariant sets while satisfy the input and output constraints. Based on the same idea and following the RRT approach, the LQR-tree algorithm [29] creates a tree by sampling over state space and stabilizes the tree with an linear quadratic regulator (LQR) feedback. With respect to the present paper, the common traits among all these works is the use of a full state feedback (as opposed to output feedback), although they do not require prior knowledge of convex cell decomposition of the environment.

Finally, our work builds upon real-time synthesis of point-wise controls that trade off safety and stability for nonlinear input-affine systems through a Quadratic Program (QP) formulation [2], [14]. To the best of our knowledge, our paper is the first to use similar conditions for synthesizing controls over entire convex regions rather than single points.

Proposed approach and contributions. In this work, we propose a novel approach to synthesize a set of output-feedback controllers on a convex cell decomposition of a polygonal environment via Linear Programming (LP). We define constraints in terms of a Control Lyapunov Function (CLF) and Control Barrier Functions (CBF) to ensure, respectively, stability and safety (collision avoidance) throughout all the states in a cell, while automatically balancing the two aspects to maximize robustness. Our formulation results in a linear min-max optimization problem, which is solved by converting it to a LP form.

With respect to previous work: 1) We allow a cell to be any generic convex polytope (instead of a simplex). 2) We consider output feedback based on any affine function of the state (under the natural assumption that the overall dynamics is controllable), although, for the sake of presenting a concrete application, we focus on controls using measurements of the relative position of the agent with respect to landmarks in the environment. 3) We apply the CLF-CBF to the new framework of control synthesis. We believe that our solution can be extended to sample-based methods and non-linear systems, although these are beyond the scope of the current paper (see the Conclusions section for details).

II. NOTATION AND PRELIMINARIES

In this section we review CLF and CBF constraints in the context of our application on agents with linear dynamics and a convex cell decomposition of the environment.

A. System dynamics

We start by considering a control-affine dynamical system¹

$$\dot{x} = Ax + Bu, \quad (1)$$

where $x \in \mathcal{X}$ denotes the state, $u \in \mathcal{U}$ the system input, and $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$ define the linear dynamics, and $\mathcal{X} \subset \mathbb{R}^{n_x}$, $\mathcal{U} \subset \mathbb{R}^{n_u}$ denotes limits on the states, and actuators, respectively. We assume \mathcal{X}_{dyn} and \mathcal{U} are polytopic,

$$\mathcal{X}_{\text{dyn}} = \{x \mid A_{x,\text{dyn}}x \leq b_{x,\text{dyn}}\}, \quad \mathcal{U} = \{u \mid A_u u \leq b_u\}, \quad (2)$$

where $A_{x,\text{dyn}} \in \mathbb{R}^{s_d \times n_x}$, $A_u \in \mathbb{R}^{s_u \times n_u}$, $b_{x,\text{dyn}} \in \mathbb{R}^{n_x}$, $b_u \in \mathbb{R}^{n_u}$, and that $0 \in \mathcal{X}_{\text{dyn}}$. s_d and s_u are the number of dynamic constraints and controller constraints respectively.

Since the system (1) can be higher-order, but the environment constrains only positions, we give the following.

Definition 1: We assume that a subset of the state x in (1) represents the position $x_{\text{pos}} = P_{\text{pos}}x$ of the agent in the world, while $x_{\text{dyn}} = P_{\text{dyn}}x$ represents the rest of the state (e.g., velocities in a second order system), where $P_{\text{pos}} \in \mathbb{R}^{d \times n_x}$ and $P_{\text{dyn}} \in \mathbb{R}^{(n_x-d) \times n_x}$ are orthogonal projection matrices.

Remark 1: In this section, we only define constraints for the dynamic part of the states, x_{dyn} , i.e., $P_{\text{pos}}A_{x_{\text{dyn}}}^T = 0$. The constraints on x_{pos} will be derived from the environment.

B. High relative degree functions and transverse dynamics

Given a function h of the state of the dynamical system (1), the following notions characterize the relation between the derivatives along the system's trajectories and the inputs u of the system. Note that we assume that h is sufficiently smooth so that all the necessary derivatives are well defined.

Definition 2: The Lie derivative of a differentiable function h for the dynamics (1) with respect to the vector field Ax is defined as $\mathcal{L}_{Ax}h(x) = \frac{\partial h(x(t))}{\partial x}^T Ax$. The Lie derivative of order r is denoted as \mathcal{L}_{Ax}^r , and is recursively defined by $\mathcal{L}_{Ax}^r h(x) = \mathcal{L}_{Ax}(\mathcal{L}_{Ax}^{r-1} h(x))$, with $\mathcal{L}_{Ax}^1 h(x) = \mathcal{L}_{Ax}h(x)$ [33].

Definition 3: A function $h(x)$ is said to have relative degree r with respect to the dynamics (1) if $\mathcal{L}_B \mathcal{L}_{Ax}^i h(x) = 0$ for $0 \leq i \leq r-1$ and $\mathcal{L}_B \mathcal{L}_{Ax}^r h(x) \neq 0$; equivalently, it is the minimum order of the time derivative of the system, $h^r(x)$, that explicitly depends on the inputs u . The Lie derivative of $h(x)$ with relative degree r for dynamics (1) is defined as

$$h^r(x) = \mathcal{L}_{Ax}^r h(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} h(x)u \quad (3)$$

Definition 4: Given a function $h(x)$ with relative degree r for the dynamics (1), we define the *transversal state*

$$\xi_h(x) = \begin{bmatrix} h(x) \\ \dot{h}(x) \\ \vdots \\ h^{r-1}(x) \end{bmatrix} = \begin{bmatrix} h(x) \\ \mathcal{L}_{Ax}h(x) \\ \vdots \\ \mathcal{L}_{Ax}^{r-1}h(x) \end{bmatrix}, \quad (4)$$

and the *transversal dynamics*

$$\begin{aligned} \dot{\xi}_h(x) &= F\xi_h(x) + G\mu_h, \\ h(x) &= C\xi_h \end{aligned} \quad (5)$$

¹The CLF-CBF concepts are applicable to input-affine systems, but in this work we assume linear time-invariant systems, and affine barrier functions.

where $F \in \mathbb{R}^{r \times r}$, $G \in \mathbb{R}^r$ and $C \in \mathbb{R}^{1 \times r}$ are defined as

$$F = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad C = [10 \dots 0], \quad (6)$$

and the virtual control input $\mu_h = h^r$ is a function of the actual input u .

We use these concepts below to define higher-order CBFs and CLFs.

Remark 2: When $h(x)$ is an affine function (see Sec. III-A), all Lie derivatives are linear functions of x , and the system (6) can be interpreted as (1) in observable canonical form.

C. Safety Constraints by Control Barrier Function

Suppose we have a sufficiently smooth function $h(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ which defines a safe set \mathcal{C} such that

$$\begin{aligned} \mathcal{C} &= \{x \in \mathbb{R}^n \mid h(x) \geq 0\}, \\ \partial\mathcal{C} &= \{x \in \mathbb{R}^n \mid h(x) = 0\}, \\ \text{Int}(\mathcal{C}) &= \{x \in \mathbb{R}^n \mid h(x) > 0\}. \end{aligned} \quad (7)$$

We say that the set \mathcal{C} is *forward invariant* (also said *positive invariant* [6]) if $x(t_0) \in \mathcal{C}$ implies $x(t) \in \mathcal{C}$, for all $t \geq t_0$ where $x(t)$ is well defined [31].

Definition 5 (ECBF, [26]): Consider the control system (1), and a continuously differentiable function $h(x)$ with relative degree $r \geq 0$ defining a set \mathcal{C} as in (7). The function $h(x)$ is an *Exponential Control Barrier Function* (ECBF) if there exist $c_h \in \mathbb{R}^r$ and control inputs $u \in \mathcal{U}$ such that

$$\mathcal{L}_{Ax}^r h(x) + \mathcal{L}_A \mathcal{L}_{Ax}^{r-1} h(x) u + c_h^T \xi_h(x) \geq 0, \quad \forall x \in \text{Int}(\mathcal{C}). \quad (8)$$

Proposition 1: Given an ECBF $h(x)$ and control inputs u from Definition 5, if c_h stabilizes the transversal dynamics, i.e., the closed-loop matrix $A_h - B_h c_h^T$ is stable, then

- 1) $\mathcal{L}_A^j h(x) \geq -p_1 \mathcal{L}_A^{j-1} h(x)$ for $1 \leq j \leq r$, $p_1 \geq 0$
- 2) the set \mathcal{C} is forward invariant.

Proof: From [26, equation (41) and Theorem 2], for the family of outputs $y_i : \mathbb{R}^n \rightarrow \mathbb{R}$ for $i = 1, \dots, r$ we have

$$y_i = \dot{y}_{i-1} + p_i y_{i-1} \quad (9)$$

where $p_i \in \mathbb{R}^+$ for $i = 1, \dots, r$ is a pole location of $\mu_h = -c_h^T \xi_h$, $C_i = \{x \in \mathbb{R}^n \mid y_i \geq 0\}$, and $y_0 = \mathcal{L}_A^0 h(x)$ so

$$y_1 = \mathcal{L}_A^0 h(x) + p_1 \mathcal{L}_A^0 h(x) \geq 0 \quad (10)$$

which implies $\mathcal{L}_A^j h(x) \geq -p_1 \mathcal{L}_A^{j-1} h(x)$.

For proof of claim 2 see [26, Theorem 1]. ■

Note that if $r = 1$, $h(x)$ is also a special case of a *Zeroing Control Barrier Function* (ZCBF, [31], [33]).

D. Stability Constraints by Control Lyapunov Function

In this section we present an analogous definition extending CLFs [1] to higher-order relative degrees.

Definition 6: Consider the control system (1), and a continuously differentiable function $V(x)$ defined over a set \mathcal{X} with $V(x) \geq 0$ and relative degree $r \geq 0$. The function $V(x)$

is a *Exponential Control Lyapunov Function* (ECLF) if there exists $c_V \in \mathbb{R}^r$ and control inputs $u \in \mathcal{U}$ such that

$$\mathcal{L}_A^r V(x) + \mathcal{L}_B \mathcal{L}_A^{r-1} V(x) u + c_V^T \xi_V(x) \leq 0, \quad \forall x \in \mathcal{X}. \quad (11)$$

For $r = 1$, we recover the definition of Exponentially Stabilizing CLFs (ES-CLFs, [1]). It is possible to use the ECLF to design controllers that exponentially stabilize the original dynamics (1), as shown by the following:

Proposition 2: Given an ECLF $V(x)$ and controls u from Definition 6, if \mathcal{X} is a forward-invariant set, and c_V^T stabilizes the transversal dynamics, i.e., the matrix $F - G c_V^T$ is stable, then:

- 1) $\mathcal{L}_A^j V(x) \leq -q_1 \mathcal{L}_A^{j-1} V(x)$ for $1 \leq j \leq r$, $q_1 \geq 0$
- 2) $\lim_{t \rightarrow \infty} V(x(t)) = 0$ with exponential convergence;
- 3) If $V(x)$ in addition satisfies

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|) \quad (12)$$

where α_1, α_2 are class- \mathcal{K} functions, then $\lim_{t \rightarrow \infty} x = 0$ with exponential convergence.

Proof: The proof mirrors a simplified version of the ideas in [26]. Setting the virtual input μ_V in the transversal dynamics to $\mu_V = -c_V^T \xi_V$, we have that $\lim_{t \rightarrow \infty} \xi_V = 0$ with exponential convergence (since it is an LTI system and c_V contains stabilizing feedback gains). We can apply $\mu_V \leq -c_V^T \xi_V$, then

$$\dot{\xi}_V \leq (F - G c_V^T) \xi_V, \quad (13)$$

in which the last element correspond to the condition in (11). Subclaim 1) can be proved similar to the claim 1 in Proposition 1 where $q_i \in \mathbb{R}^+$ for $i = 1, \dots, r$ is a pole location of $\mu_V = -c_V^T \xi_V$. Applying Gronwall's comparison lemma [12], we then conclude that $\lim_{t \rightarrow \infty} \xi_V = 0$, which, in particular, implies subclaim 2). Finally, subclaim 3) can be shown using 2) in combination with (12) and standard arguments from Lyapunov theory [17, Chapter 4]. ■

Note that this result can be applied to any point other than the origin with a simple change of coordinates.

E. Polygonal environment decomposition

We assume a compact polygonal environment $\mathcal{P} \subset \mathbb{R}^{n_x}$, not necessarily simply-connected, decomposed in a finite number of convex cells $\{\mathcal{X}_{i,\text{pos}}\}$, such that $\bigcup_i \mathcal{X}_{i,\text{pos}} = P$, and set $\mathcal{X}_{i,\text{pos}}$ is a polytope defined by linear inequality constraints of the form $A_{x_{\text{pos}},i}^T x \leq b_{x_{\text{pos}},i}$.

Our goal is to design a different linear feedback controller u for each cell \mathcal{X}_i . The feedback signal used by the controller will be based on linear relative measurements with respect to a set of *landmarks*.

Definition 7: A landmark is defined as a point $\hat{y} \in \mathbb{R}^d$ whose location is known and fixed in the environment. For each convex section \mathcal{X}_i , we have a finite number of landmarks. In this paper, we choose the landmarks as the vertices of the convex section \mathcal{X}_i , although this choice does not make any difference in terms of the actual method.

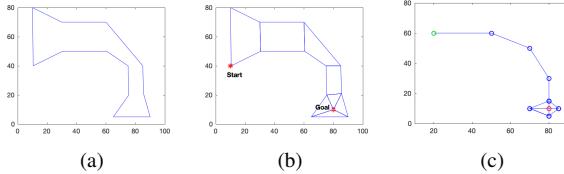


Fig. 1: The polygonal environment in Fig .1a is decomposed to 8 convex sections Fig .1b and the corresponding graph is shown in Fig . 1c

F. High-level planning

We consider two overall objectives for the controller design:

- (O1) Point stabilization: given the stabilization point (where $\dot{x} = 0$) in the environment and starting from any point, we aim to converge to the stabilization point (e.g. Fig. 2a).
- (O2) Patrolling: starting from any point, we aim to patrol the environment by converging to a path, and then traversing the same path (e.g. Fig. 4a).

To specify the convergence objective for each controller u , we first abstract the polygonal environment P into a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex $i \in \mathcal{V}$ represents a cell \mathcal{X}_i in the partition of P , and an edge $(i, j) \in \mathcal{E}$ if and only if cells corresponding to i and j have a face in common.

In the case of the point stabilization objective (O1), the stabilization point is one of the vertices of the graph and if the stabilization point is in the middle of the cell, without loss of generality, we can decompose the cell into new convex cells such that the stabilization point is one of the vertices of the new cells. Then, we add one vertex to the set \mathcal{V} , which will be the stabilization point and also, we add edges between the new vertex and any cell that has a face in common with the cell includes the stabilization point to the set \mathcal{E} .

For each cell, we then select one *exit edge* (a pointer) such that, when considered together, all such edges provide a solution in the abstract graph \mathcal{G} to the high level objective. For instance, in the case of objective (O1), the exit edge of each cell will point in the direction of the shortest path toward the vertex of the stabilization point. In the case of objective (O2), following the exit edges will lead to a cyclic path in the graph.

To give an example, the polygonal environment in Fig. 1a is converted to the connected graph in Fig. 1c based on the cell decomposition of the environment in Fig. 1b. Starting from the first node in Fig. 1c which is shown by the green point, we find the path from the start node to the equilibrium node shown by the red point, through the path planning algorithms (e.g. using Dijkstra's algorithm). Regarding to that path, we define the *exit face* as the face of the convex section the path moves through and based on that we design the controller.

Definition 8: For each cell \mathcal{X}_i in the decomposition of the environment, we define an *exit face* \mathcal{P}_{exit} or, respectively, *stabilization point* $\mathcal{P}_{exit} = \{x_g\}$ to be the face or, respectively, vertex corresponding to the *exit edge* in the abstract graph \mathcal{G} . The *exit direction* z is an inward-facing normal or, respectively, direction of \mathcal{P}_{exit} .

In this work we desire to design a controller for each convex section of the environment that drives the system in the exit direction toward the exit face or the stabilization point, while avoiding the boundary of the environment.

Overall, thanks to the high level planning in the abstract graph \mathcal{G} , and the controller design in each cell \mathcal{X}_i (explained in the sections below) the system will traverse a sequence of cells to reach a given equilibrium point, or achieve a periodic steady state behavior (examples in Section V) according to the desired objective.

III. PROBLEM SETUP

The goal of this section is to synthesize a robust controller for a convex cell \mathcal{X} (with respect to previous sections, we dropped the subscript i to simplify the notation) where $\mathcal{X} = \mathcal{X}_{dyn} \cap \mathcal{X}_{pos}$. According to Definition 1 we divide x into two parts, x_{pos} and x_{dyn} . We assume that the agent has direct access to x_{dyn} , but for x_{pos} the agent can only measure the relative displacements between the robot's position x_{pos} and the landmarks in the environment, which corresponds to the output function

$$y = (Y - x_{pos}\mathbf{1}^T)^\vee, \quad (14)$$

where $Y \in \mathbb{R}^{d \times n_l}$ is a matrix of landmark locations and A^\vee represents the vectorized version of a matrix A . Our goal is to find a feedback controller that, given y , provides an input u that drives the system toward an exit face or vertex of \mathcal{X} while avoiding obstacles (non-exit faces of \mathcal{X}). Note that the landmarks do not necessarily need to belong to \mathcal{X} . We assume (1) is controllable and choose a controller of the form

$$u(K_1, K_2) = K_1 y + K_2 x_{dyn}, \quad (15)$$

where $K_1 \in \mathbb{R}^{n_u \times d n_l}$ and $K_2 \in \mathbb{R}^{n_u \times (n_x - d)}$ are feedback gains that need to be designed. Note that $Y - x_{pos}\mathbf{1}^T$ gives a matrix where each column is the relative displacement between each landmark and the current position of the system; as such, we are looking for a controller that feeds back linear combinations of these displacements. From the distributivity property of vectorization we can write u as

$$\begin{aligned} u(K_1, K_2) &= K_1 Y^\vee - K_1 (\mathbf{1}_{nl} \otimes I_d) x_{pos} + K_2 x_{dyn} \\ &= K_1 Y^\vee + K_x x, \end{aligned} \quad (16)$$

where $K_x \in \mathbb{R}^{n_u \times n_x}$ and $K_x = [-K_1 (\mathbf{1}_{nl} \otimes I_d) \quad K_2]$.

Remark 3: In general, our framework can handle general linear output $y = Cx + D$, but we focus here on the path planning application.

A. Control Barrier Function

Let $A_{h,i} \in \mathbb{R}^{1 \times n_x}$ belongs to the union of all rows of $A_{x_{dyn}}$ and $A_{x_{pos}}$ except the one row defining the exit face, and $b_{h,i} \in \mathbb{R}$ we define the following candidate ECBF:

$$h_i(x) = A_{h,i} x + b_{h,i} \quad (17)$$

where $i = \{1, \dots, s_p + s_d\}$ such that s_p denotes the number of faces of \mathcal{X} except the one associated to an exit face (or all of them in the case of a stabilization point) and s_d denotes the number of boundaries to limit x_{dyn} . We define $s_x = s_p + s_d$.

B. Control Lyapunov Function

To stabilize the system, we define the Lyapunov function $V(x)$ for cell \mathcal{X} as,

$$V(x) = z^T x + b_V, \quad (18)$$

where $P_{\text{pos}}z \in \mathbb{R}^d$ is the exit direction for the cell \mathcal{X} (see Definition 8), and $P_{\text{dyn}}z = 0$, and $b_V \in \mathbb{R}$ is chosen such that the function reaches its minimum $V(x) = 0$ when x is in the exit face ($V(x) < 0$ correspond to points outside the cell). Note that this Lyapunov function represents, up to a constant, the distance $d(x_{\text{pos}}, \mathcal{P}_{\text{exit}})$ between the current system position and the exit face. When the exit face reduces to an exit point $\mathcal{P}_{\text{exit}} = x_{\text{exit}}$, the Lyapunov function states the distance $d(x_{\text{pos}}, x_{\text{exit}})$ between the current position and exit point, up to a constant, and the minimum of $V(x) = 0$ reaches when x is identical to the exit point x_{exit} .

Remark 4: The function $V(x)$ can be defined as a function of the vertices of the exit face instead of its normal. For instance, in \mathbb{R}^2 , we have

$$V(x) = \det([v_1 - v_0 \quad x_{\text{pos}}]) \quad (19)$$

where v_0, v_1 are two distinct points (e.g., vertices) in the exit face (with their order determining the correct sign in $V(x)$). Based on the same idea, in \mathbb{R}^3 , $V(x) = \det([v_1 - v_0 \quad v_2 - v_0 \quad x_{\text{pos}}])$ where v_0, v_1, v_2 are three distinct points in the exit face (e.g., three vertices of the exit plane) and respectively. This concept can be generalized to any dimension.

C. Finding the Controller by Robust Optimization

Our goal is to find controllers u (more precisely, control gains K) that maximize the motion of the robot toward the exit face, while avoiding the boundary of the environment. Using the CLF-CBF constraints reviewed in Section II, we encode our goal in the following feasibility problem:

$$\begin{aligned} & \text{find } K \\ & \text{s.t. :} -(\mathcal{L}_{Ax}^r h_i(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} h_i(x) u + c_b^T \xi_{bi}(x)) \leq 0, \\ & \quad \mathcal{L}_{Ax}^r V(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} V(x) u + c_V^T \xi_V(x) \leq 0, \\ & \quad u \in \mathcal{U}, \\ & \quad \forall x \in \mathcal{X}, \quad i = \{1, \dots, s_h\}. \end{aligned} \quad (20)$$

In practice, we aim to find a controller that satisfies the constraints in (20) with some margin, hence we focus on the following robust optimization problem:

$$\begin{aligned} & \min_{K, S_l, S_b} w_b^T S_b + w_l S_l \\ & \text{s.t. :} -(\mathcal{L}_{Ax}^r h_i(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} h_i(x) u + c_b^T \xi_{bi}(x)) \leq S_{bi}, \\ & \quad \mathcal{L}_{Ax}^r V(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} V(x) u + c_V^T \xi_V(x) \leq S_l, \\ & \quad S_l, S_b \leq 0, u \in \mathcal{U}, \\ & \quad \forall x \in \mathcal{X}, \quad i = \{1, \dots, s_h\}. \end{aligned} \quad (21)$$

Note that the constraints in (21) need to be satisfied for all x in the cell \mathcal{X} , i.e., the same control gains should satisfy the

CLF-CBF constraints at every point in the cell. We handle this type of constraint by rewriting (21) using a min-max formulation, where (21) is equivalent to,

$$\begin{aligned} & \min_{K, S_l, S_b} w_b^T S_b + w_l S_l \\ & \text{s.t. :} \\ & \left[\begin{array}{l} \max_x -(\mathcal{L}_{Ax}^r h_i(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} h_i(x) u + c_b^T h_i(x)) \\ \text{s.t. } x \in \mathcal{X}, u \in \mathcal{U} \end{array} \right] \leq S_{bi}, \\ & \left[\begin{array}{l} \max_x \mathcal{L}_{Ax}^r V(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} V(x) u + c_V^T V(x) \\ \text{s.t. } x \in \mathcal{X}, u \in \mathcal{U} \end{array} \right] \leq S_l, \\ & S_l, S_b \leq 0, \quad i = \{1, \dots, s_h\}. \end{aligned} \quad (22)$$

the weights $w_b \in \mathbb{R}^{s_h}$ and $w_l \in \mathbb{R}$ are user-defined constants defining the trade-off between the barrier functions and Lyapunov function constraints: larger w_b implies solutions moving away from the walls, while larger w_l implies solutions moving faster toward the exit face. Now we compute the time r -th order derivative of $h_i(x)$ such that

$$h_i^r(x) = \mathcal{L}_{Ax}^r h_i(x) + \mathcal{L}_B \mathcal{L}_{Ax}^{r-1} h_i(x) u \quad (23)$$

Combining $h(x)$ from (17) and (23) implies

$$h_i^r(x) = A_{h,i} A^{r-1} (Ax + Bu) = A_{h,i} A^{r-1} \dot{x}, \quad (24)$$

where A^r represents the r -th power of A . Similar to $h_i^r(x)$, the time derivative of $V(x)$ is defined as

$$V^r(x) = z A^{r-1} (Ax + Bu) = z A^{r-1} \dot{x}. \quad (25)$$

Substituting the Lie derivatives (24) and (25) in (22) results

$$\begin{aligned} & \min_{K, S_l, S_b} w_b^T S_b + w_l S_l \\ & \text{s.t. :} \\ & \left[\begin{array}{l} \max_x -(A_{h,i} A^r + A_{h,i} A^{r-1} B K_x + A_{h,i} c_b)x \\ \text{s.t. : } A_x x \leq b_x \end{array} \right] \leq \\ & \quad S_{bi} + c_b b_{h,i} + A_{h,i} A^{r-1} B K_1 Y^\vee \\ & \left[\begin{array}{l} \max_x (z^T A^r + z^T A^{r-1} B K_x + z^T c_l)x \\ \text{s.t. : } A_x x \leq b_x \end{array} \right] \leq \\ & \quad S_l - c_l b_V - z^T A^{r-1} B K_1 Y^\vee, \\ & \left[\begin{array}{l} \max_x (A_{uj} K_x)x \\ \text{s.t. : } A_x x \leq b_x \end{array} \right] \leq b_{uj} - A_{uj} K_1 Y^\vee \\ & S_b, S_l \leq 0, \quad i = \{1, \dots, s_h\}, j = \{1, \dots, n_u\} \end{aligned} \quad (26)$$

In (26), we have a bi-level optimization problem with constraints that are given themselves by other optimization problems. As all constraints and objective function are linear and \mathcal{X} is a convex set, (26) and inner maximization problems are linear programming problem so we can change the min-max problem (26) to min-min problem by replacing the inner

maximization problems with their dual forms,

$$\begin{aligned}
& \min_{K, S_l, S_b} w_b^T S_b + w_l S_l \\
& \text{s.t. :} \\
& \left[\begin{array}{c} \min_{\lambda_b} \lambda_b^T b_x \\ \text{s.t. :} \\ A_x^T \lambda_b = -(A_{h,i} A^r + A_{h,i} A^{r-1} B K_x + A_{h,i} c_b)^T \\ \lambda_b \geq 0, \\ S_b + c_b b_{h,i} + A_{h,i} A^{r-1} B K_1 Y^\vee \end{array} \right] \leq \\
& \left[\begin{array}{c} \min_{\lambda_l} \lambda_l^T b_x \\ \text{s.t. :} \\ A_x^T \lambda_l = (z^T A^r + z^T A^{r-1} B K_x + z^T c_l)^T \\ \lambda_l \geq 0 \\ S_l - c_l b_V - z^T A^{r-1} B K_1 Y^\vee \end{array} \right] \leq \\
& \left[\begin{array}{c} \min_{\lambda_u} \lambda_u^T b_x \\ \text{s.t. :} \\ A_x^T \lambda_u = (A_{u,j} K_x)^T \\ \lambda_u \geq 0 \end{array} \right] \leq b_{uj} - A_{uj} K_1 Y^\vee \\
& S_b, S_l \leq 0, i = \{1, \dots, s_h\}, j = \{1, \dots, n_u\}
\end{aligned} \tag{27}$$

where min-min problem (27) is equivalent to the minimization problem,

$$\begin{aligned}
& \min_{K, S_l, S_b, \lambda_l, \lambda_b} w_b^T S_b + w_l S_l \\
& \text{s.t. : } \lambda_b^T b_x \leq S_{bi} + c_b b_{h,i} + A_{h,i} A^{r-1} B K_1 Y^\vee \\
& \quad \lambda_l^T b_x \leq S_l - c_l b_V - z^T A^{r-1} B K_1 Y^\vee \\
& \quad \lambda_u^T b_x \leq b_{uj} - A_{uj} K_1 Y^\vee \\
& \quad A_x^T \lambda_b = -(A_{h,i} A^r + A_{h,i} A^{r-1} B K_x + A_{h,i} c_b)^T \\
& \quad A_x^T \lambda_l = (z^T A^r + z^T A^{r-1} B K_x + z^T c_l)^T \\
& \quad A_x^T \lambda_u = (A_{u,j} K_x)^T \\
& \quad \lambda_l, \lambda_b, \lambda_u \geq 0 \quad S_b, S_l \leq 0 \\
& \quad i = \{1, \dots, s_h\}, j = \{1, \dots, n_u\}
\end{aligned} \tag{28}$$

In the following we prove that the feasible optimal solution for (26) is also the feasible optimal solution for (28).

Remark 5: By strong duality [5, Theorem 4.4], if a linear programming problem has an optimal solution, so does its dual, and the respective optimal costs are equal. This remark allows us to prove the following.

Lemma 1: Optimization problems (26) and (27) have the same feasible optimal solution.

Proof: The optimization problems in (26) and (27) have the same objective functions. Constraints in (26) are in the form of LP optimization problem and the constraints in (27) are the duals. According to the Remark 5, the optimal cost of constraints in (26) and (27) are equal and result the same constraints with the same objective functions which implies (26) and (27) have the same optimal solution. ■

Lemma 2: Optimization problems (27) and (28) have the same feasible optimal solution.

Proof: Assume we have an optimal solution for (28), then the solution is also feasible for (27) and the objective costs are the same. In the same way, if we have an optimal

solution for (27), so there must exist dual variables for inner optimization problem in (27) which are also feasible for (28) and result in the same objective cost [4]. ■

Theorem 1: From Lemma 1 and Lemma 2, the optimization problems (26), (27) and (28) are equivalent.

Proposition 3: Solving (28) and assuming optimal S_l is strictly less than zero, then the trajectory exit from the cell in finite time.

Proof: For the first order system $c_V > 0$ is scalar and $V(x)$ is positive definite, define maximum distance from the exit face as

$$d_{max} = \max\{V(x) | A_x x \leq b_x\}. \tag{29}$$

For the first order system, the CLF constraints in (27) implies

$$\dot{V}(x(t)) + c_V V(x(t)) \leq S_l \tag{30}$$

where $S_l < 0$ and $g_1 = c_V V(x(t)) \geq 0$ and results in

$$\dot{V}(x(t)) \leq S_l - g_1 = g_2, g_2 \leq 0 \tag{31}$$

Solving the above differential equation shows $V(x(t)) \leq V(x_0) + g_2 t$. To pass the exit face we need to have $V(x(t)) = 0$ as the $V(x(t))$ shows the distance from the exit face, so $t_{exit} \leq -\frac{d_{max}}{g_2}$ and the controller reaches the exit face in finite time as $-\frac{d_{max}}{g_2}$ has a finite value. ■

IV. STATIONARY POINT

Consider the stabilization objective (O1) defined in Section II-F, and let x_g be the stabilization point in \mathcal{X} , i.e., the exit vertex in \mathcal{P}_{exit} (see Definition 8). In this section we provide sufficient conditions that show the controllers synthesized with our proposed method indeed introduce an equilibrium point at x_g . Before proceeding, we need the following. We use $\text{stack}()$ to denote the operator that stacks vertically all its matrix arguments.

Fact 1: Let $A_{h,exit}$ be the matrix whose rows are the row vectors in the set $\{A_{h,i} : h_i(x_g) = 0\}$. Then z belongs to the proper cone $\{v : A_{h,exit} v \geq 0\}$.

This fact is intuitive given Definition 8: $A_{h,exit}$ represents the normals of the active constraints at the stabilization point, and z needs to be inward-pointing. Note that the rows of $A_{h,exit}$ are a subset of the rows of $A_{x_{pos}}$. We can now state the main result of this section.

Proposition 4: Assume the pair $(A, \text{stack}(A_{h,exit}, z^T))$ is observable and that all h_i and V have the same relative degree r . Then, any solution to the min-max problem (21) (or, equivalently, the linear program (28)) guarantees that $\dot{x} = 0$ when $x_{pos} = x_g$.

Note that the assumption about having a homogeneous relative degree is reasonable, since z^T and $A_{h,exit}$ all essentially represent generic planes in the environment.

Proof: Any feasible controller must satisfy the CBF and CLF constraints in (21). As discussed above, we have $V(x_g) = 0$ for the Lyapunov function, and $h_i(x_g) = 0$ for the constraints corresponding to $A_{h,exit}$. Recalling that $\mathcal{L}_A^0 V = V$, and using the fact the CLF constraint in (21) implies Proposition 2, claim 1), we have that $\mathcal{L}_A^j V =$

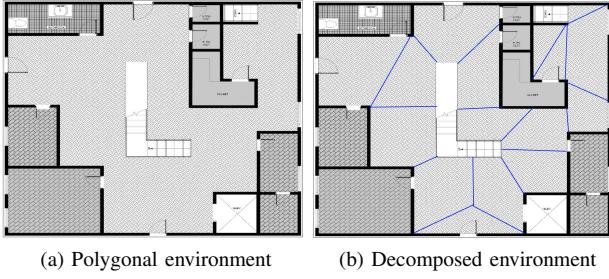


Fig. 2: Non-simply connected environment is decomposed to a set of convex cells.

$z^T A^j \dot{x} \leq -c_v \mathcal{L}_A^{j-1} V = z^T A^{j-1} \dot{x}$ for all $0 \leq j \leq r$. A similar argument with the CBF constraint in (21) and Proposition 1 implies that $A_{h,\text{exit}} A^j \dot{x} \geq -c_h A_{h,\text{exit}} A^{j-1} \dot{x}$. From Fact 1, we have that the sets described by $A_{h,\text{exit}} v \geq 0$ and $z^T v \leq 0$ intersect only at the point $v = 0$; hence, $\text{stack}(A_{h,\text{exit}}, z^T) A^j \dot{x} = 0$ for all $0 \leq j \leq r-1$, which can be compactly described as $\mathcal{O}_A \dot{x} = 0$, where \mathcal{O}_A is the observability matrix from the pair $(A, \text{stack}(A_{h,\text{exit}}, z^T))$. Since the latter is observable, \mathcal{O}_A is full rank, and hence $\dot{x} = 0$ as claimed. ■

Intuitively, the proof shows that the CLF and CBF constraints fix x_{pos} to x_g , which together with the observability implies that also $x_{\text{dyn}} = 0$.

V. NUMERICAL EXAMPLES

In this section, we apply our proposed method to two non-simply connected environments to find a output-feedback controller, then we deform the environment and implement the same controller to represent the robustness of the controller. We apply our method to the fist and second integrator systems to achieve two planning objectives of our algorithm. The first example considers the point stabilization (O1) objective and the second example shows the patrolling (O2) objective (The choice of the fist and second integrator systems is independent of the planning objective and complexity of the environment).

A. First Order Controller

Given the environment in Fig 2, we want to find a set of controllers to move the agent from three different entrances (start points) of a building to the exit door (goal point). To achieve this goal, we decompose the layout of the floor into 16 convex cells. We choose $c_b = c_v = 0.5$. Given the decomposed environment, we find a controller for each cell individually and move the agent from the start points. In this example, we assume the landmarks for each cell are same as the vertices of the cell and we define as z the exit direction of the cell. Solving the optimization problem (28) finds the optimal K_1 for each cell which implies the optimal controller. Entering the building from different doors in Fig. 3a, the first order controller moves the agent from to the exit door without violating any constraints. Assume the layout of the building is changed due to the construction purposes in Fig. 3b and

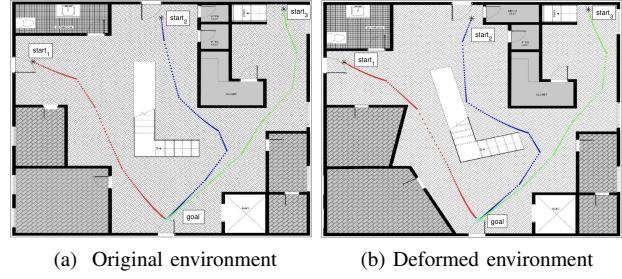
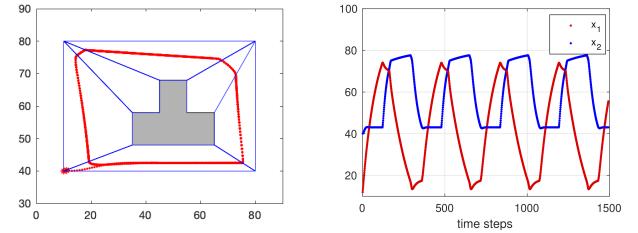


Fig. 3: In this example, we deform the layout and apply the first order controller of the original environment to the deformed layout. Despite the fact that we deform the layout significantly, the exact same original controllers generate successful trajectories.



(a) Non-simply connected environment (b) Changes of the states versus time

Fig. 4: Fig 4a is a non-simply connected environment and the gray polygon is an obstacle. In Fig. 4a an agent starts from position $(10,40)$ and continuously moves through all sections. In Fig. 4b x_1 and x_2 variation versus time is shown.

the agent enters the building aims to proceed to the exit door. When the layout deformed, the convex cells change as the position of landmarks and given the old optimal K_1 from the original layout and new position of landmarks, the agent is able to proceed to the exit door starting from different entrances and meeting all the safety and stability constrains.

B. Second Order Controller

In this section, we find a controller for a second order system and similar to the first order system, we assume the landmarks are equivalent to the vertices of each cell and we denote z the exit direction of the cell. we apply our method to a non-simply connected environment. The controller moves the agent from the start point at $[10, 40]$ in Fig. 4, we choose $c_b = [1, 1]^T$ and $c_v = [1, 1]^T$ for all $i = \{1, \dots, s_h\}$. Then, we enlarge the obstacle and apply the same controller to the agent in Fig. 5a. Our method guarantees that the agent moves through the environment completely without violating safety and stability constraints and when obstacle rotates $\pi/4$ counterclockwise in Fig. 5b, the agent moves through the feasible path to cover all the environment with similar control gain K_1 and K_2 .

VI. CONCLUSIONS

In this paper we proposed a novel approach to design a output-feedback controller with cell decomposition, through

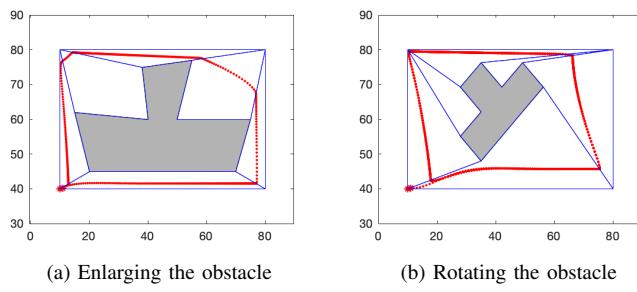


Fig. 5: In this two examples we deform the obstacle and apply the second order controller of the original environment to the new environment.

Linear Programming. We defined a controller such that it depends on the relative displacement measurements with respect to the landmarks of the convex cells and formed the min-max convex problem. Then we changed the min-max optimization problem to min-min optimization problem by forming the dual of the inner maximization problems and we found the controller which is robust to the significant changes of the environment. We validate our approach on different examples for the first and second order dynamic control systems. As presented, our current cell-focused approach has two limitations: First the controllers are discontinuous at the boundaries of the cells; this can be addressed by adding smoothness constraints between cells (at the price of solving a single large linear program instead of multiple ones). Second, we assume the environment is already discretized in convex cells; it is possible to relax this assumption by using sampling and Voronoi partitions. We plan to study these extensions in our future work. In addition, we aim to implement our method to constrained nonlinear systems based on [10]. Moreover, we also plan to formally investigate theoretical guarantees for the robustness of the synthesized controllers that we have empirically demonstrated in this paper.

REFERENCES

- [1] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 59(4):876–891, 2014.
- [2] A. D. Ames, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.
- [3] C. Belta, V. Isler, and G. J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, 2005.
- [4] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.
- [5] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.
- [6] F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.
- [7] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.
- [8] D. C. Conner, A. A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 4, pages 3546–3551. IEEE, 2003.
- [9] S. Fortune. Voronoi diagrams and delaunay triangulations. In *Computing in Euclidean geometry*, pages 193–233. World Scientific, 1992.
- [10] A. Girard and S. Martin. Motion planning for nonlinear systems using hybridizations and robust controllers on simplices. In *2008 47th IEEE Conference on Decision and Control*, pages 239–244. IEEE, 2008.
- [11] R. Gonzalez, M. Kloetzer, and C. Mahulea. Comparative study of trajectories resulted from cell decomposition path planning approaches. In *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, pages 49–54. IEEE, 2017.
- [12] T. H. Gronwall. Note on the derivatives with respect to a parameter of the solutions of a system of differential equations. *Annals of Mathematics*, pages 292–296, 1919.
- [13] L. Habets, P. J. Collins, and J. H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6):938–948, 2006.
- [14] S.-C. Hsu, X. Xu, and A. D. Ames. Control barrier function based quadratic programs with application to bipedal robotic walking. In *2015 American Control Conference (ACC)*, pages 4542–4548. IEEE, 2015.
- [15] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
- [16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.
- [17] H. K. Khalil and J. W. Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.
- [18] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.
- [19] M. Kloetzer, C. Mahulea, and R. Gonzalez. Optimizing cell decomposition path planning for mobile robots using different metrics. In *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 565–570. IEEE, 2015.
- [20] B. Krogh. A generalized potential field approach to obstacle avoidance control. In *Proc. SME Conf. on Robotics Research: The Next Five Years and Beyond, Bethlehem, PA*, 1984, pages 11–22, 1984.
- [21] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How. Motion planning in complex environments using closed-loop prediction. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7166, 2008.
- [22] J.-C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.
- [23] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [24] S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.
- [25] F. Lingelbach. Path planning using probabilistic cell decomposition. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 467–472. IEEE, 2004.
- [26] Q. Nguyen and K. Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*, pages 322–328. IEEE, 2016.
- [27] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *Departmental Papers (ESE)*, page 323, 1992.
- [28] A. Schürmann. Computational geometry of positive definite quadratic forms. *University Lecture Series*, 49, 2009.
- [29] R. Tedrake. *Lqr-trees: Feedback motion planning on sparse randomized trees*. MIT Press, 2009.
- [30] A. Weiss, C. Danielson, K. Berntorp, I. Kolmanovsky, and S. Di Cairano. Motion planning with invariant set trees. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1625–1630. IEEE, 2017.
- [31] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 48(27):54–61, 2015.
- [32] H. Yan, H. Wang, Y. Chen, and G. Dai. Mobile robot navigation in the triangulation of dynamic environment. In *2008 International Conference on Information and Automation*, pages 776–783. IEEE, 2008.
- [33] G. Yang, C. Belta, and R. Tron. Self-triggered control for safety critical systems using control barrier functions. *arXiv preprint arXiv:1903.03692*, 2019.