2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Spline Trajectory Tracking and Obstacle Avoidance for Mobile Agents via Convex Optimization

Akua Dickson, Christos G. Cassandras and Roberto Tron

***Abstract*— We propose a motion planning technique based on output feedback that enable agents to converge to a specified polynomial trajectory while avoiding collisions within a polygonal environment. To achieve this, we *1)* decompose the polygonal environment into overlapping cells, *2)* express the polynomial trajectories as the output of a reference dynamical system with given initial conditions, *3)* formulate convergence and safety (collision avoidance) constraints as Linear Matrix Inequalities (LMIs) on our controller using Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs), and *4)* synthesize a controller for each convex cell via a semi-definite programming (SDP) problem that includes the derived constraints. We test our method with simulations. We find that the synthesized controller is robust to changes in initial conditions, and maintains safety relative to the boundaries of the polygonal environment even in the presence of significant amounts of noise.**

I. INTRODUCTION

The primary goal of path planning is to generate a trajectory for a mobile agent between an initial state and a goal state while ensuring collision avoidance at all times; it is a critical cornerstone of modern autonomous systems.

Several path-planning algorithms have been proposed in the robotics literature and have been applied to various problems [8]. We briefly review here the most representative approaches and their characteristics. Potential field algorithms are a traditional method that produces relatively smooth trajectories by attracting the agent toward the goal and repelling it from obstacles; they are computationally cheap but myopic and incur the risk of becoming trapped in local minima [26]. Alternative algorithms that are *complete* (i.e., that find a solution if one exists) are A* and RRT* (optimized Rapidly Exploring Random Trees); the former requires a discretization of the environment, and does not scale well with the problem dimension; the latter is at the basis of many state-of-the-art in high-dimensional planning spaces, and shows good practical performance [17], [26]. While complete, these algorithms typically do not produce smooth trajectories. One way to obtain smooth solutions is via post-processing with trajectory optimization methods such as direct collocation methods [25]. These methods parameterize the trajectory using a discrete number of control points, and then solve the planning problem as a constrained nonlinear optimization problem. This class of algorithms is well-suited for problems with complex system dynamics and complex constraints, but offers only local convergence guarantees, and it typically needs to be initialized from a complete planner such as A* or RRT*.

Cell decomposition is another traditional method where a specified convex polygonal environment is decomposed into a set of cells [19]; A* is used to find a feasible sequence of cells, and other techniques are then used to produce paths in each cell. RRT* algorithms may also be used to build the decomposition of the polygonal environment as seen in [5].

In this context, and in the context of collocation methods, it is advantageous to parameterize smooth trajectories using a polynomial basis, such as in spline curves [12], [16], [22]. In recent years, splines have been applied to applications with both ground [14], [21], and aerial [15], [18], [20] vehicles. A particular type of splines is given by Bézier curves [9], [11], [28], which are divided into segments defined by sets of control points. The main advantage of these curves is their strong convex hull property: by using a Bernstein polynomial basis, every trajectory will be contained in the convex hull of its control points. Therefore, by choosing the control points that lie completely within a convex collision-free set, the entire polynomial trajectory is guaranteed to be collision-free. This useful parametrization needs to be combined with a path planning algorithm (as those above) for a complete solution.

In general, the majority of exiting planning methods produce individual paths that need to be tracked using a separate low-level controller. For the latter, a modern solution for nonlinear input-affine systems is given by Control Lyapunov Functions (CLFs). CLFs are the natural extension of Lyapunov functions to systems with control inputs, but use point-wise optimization to compute a control input u that can asymptotically stabilize the system as desired. Similar concepts are used in Control Barrier Functions (CBFs) [1]–[3], which extend barrier functions to ensure safety (in the form of forward-invariance of safe sets) of control-affine systems.

In the case of unexpected events or disturbances, changes in the goal of the planning problem or infeasible initial conditions, most of the methods reviewed so far would require replanning (i.e., solving the problem almost from scratch). We build instead upon our previous work [4], [5], which combines cell-decomposition-based approaches with control synthesis. Instead of directly parametrizing a single solution, this line of work designs a series of output feedback controllers that take as input the relative displacement of the agent with respect to a set of landmarks, steering the agent through the environment toward a goal location. This approach is robust to discrepancies and reduces the need for re-planning; however it does not offer an easy way to manipulate directly the final path taken by the agent.

[1]Akua Dickson, Christos Cassandras and Roberto Tron are with the Division of Systems Engineering, Boston University, 730 Commonwealth Ave, MA 02215, United States {akuad, cgc, tron}@bu.edu
This work was supported in part by NSF under grants ECCS-1931600, DMS-1664644, CNS-1645681, and by ARPA-E under grant DEAR0001282.

Paper contributions. In this paper we combine elements of many of the techniques above, namely, control-based cell-decomposition methods, polynomial splines for representing trajectories, and CBFs and CLFs to guarantee convergence and safety over all for all points in the environment. These concepts are combined to generate constraints for Semi-Definite Program (SDP) optimization problems [7], which are solved to synthesize a sequence of controllers that the agent can use to converge toward a path while avoiding collisions with the boundaries of the environment.

Our proposed method has advantages with respect to all the existing works reviewed above. With respect to potential-based methods, we offer completeness (and thus avoid local minima) while still, intuitively, pulling the agent toward a desired path. With respect to A*, RRT*-based solutions, direct collocation methods, and basic spline parametrizations, we synthesize low-level, output-feedback controllers directly instead of single reference paths that need separate controllers. With respect to cell decomposition methods, we synthesize an output-feedback controller for each cell in the convex polygonal environment that consequently ensures convergence and safety for the entire polygonal environment. With respect to CLF-CBF methods, we ensure that the problem is always feasible, and our approach is computationally more efficient since we do not need to solve any optimization problems online. Moreover, with respect to our own work, we offer convergence to a reference trajectory that can be pre-optimized to improve performance[1].

Our work consists of two main parts. Firstly, we generate the reference polynomial trajectories by writing these trajectories as the output of a reference dynamical system with given initial conditions. The second part involves the design of an output feedback controller in order to track the reference path from an initial state toward a goal state while avoiding collisions and remaining within the given polygonal environment. We use off-the-shelf SDP packages to solve a sequence of robust convex optimization problems, where CLFs [3] and CBFs [27] constraints for the given system over all valid states. The CLF and CBF constraints lead to, respectively, LMIs and linear inequalities (the latter after using duality theory) on the control coefficients [7], [24]. Note that, while we limit our discussion to linear systems, the results can be extended to nonlinear systems that admit linearization via local controllers.

In summary, our work provides a novel way to express a reference trajectory from given control points as a linear system, and provides a convex optimization approach to synthesize tracking controllers that remaining safe relative to the boundaries of the environment, leading to robust and efficient implementations.

II. PRELIMINARIES

In this section, we review the key elements that set the foundation of our approach. We define the dynamics of the agent and the decomposition of the polygonal environment, and we discuss how the reference polynomial trajectories are generated from given Bézier control points. We then provide a formal problem statement.

A. Dynamical system

We model the agent as a Linear Time Invariant system:

$$\dot{x} = Ax + Bu, \quad y = Cx, \tag{1}$$

where $x \in \mathbb{R}^d$ is the state of the system, $y \in \mathbb{R}^{d_y}$ is the output, $u \in \mathbb{R}^{d_u}$ is the vector of control inputs, and A, B, C are matrices of appropriate dimensions.

Special multidimensional case. We will prove additional results for the common case where y can be decomposed as a collection of d_y independent system, each one with output dimension $d_y = 1$. In this case we have that A, B, C are block diagonal ($A = \text{blkdiag}(\{A_k\}_{k=1}^{d_y}), B = \text{blkdiag}(\{B_k\}_{k=1}^{d_y}), C = \text{blkdiag}(\{C_k\}_{k=1}^{d_y})$), and x, y can be partitioned as $x = \text{stack}(\{x_k\}_{k=1}^{d_y}), y = \text{stack}(\{y_k\}_{k=1}^{d_y})$. Each $y_k \in \mathbb{R}$ is one-dimensional.

B. Polynomial trajectories

We assume trajectories in polynomial form $p(t) : [0, 1] \to \mathbb{R}^d$ where

$$p(t) = \sum_{i=0}^{n_p} a_i t^i = \sum_{i=0}^{n_p} P_i b_{i,n_p}(t), \tag{2}$$

for $t \in [0, 1]$, where $P = \begin{bmatrix} P_0 & P_1 & P_2 & ..... & P_{n-1} \end{bmatrix} \in \mathbb{R}^{d \times (n_p+1)}$ is a set of n *control points*, and $b_{i,n}$ represent the Bernstein basis polynomials $b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \in \mathbb{R}$, $i \in \{0, 1, 2, \ldots, n\}$, and $A_{\text{coeff}} = \begin{bmatrix} a_0 & a_1 & \ldots & a_{n-1} \end{bmatrix}$ is a matrix of polynomial coefficients. (More complex trajectories can be constructed by joining multiple polynomial segments linked by continuity constraints, see Sec. III-F.) Equation (2) gives two equivalent representations for the reference polynomial trajectories adopted in this paper. This equivalence is established by the following lemma.

Lemma 1 ([13]): Given the standard polynomial basis coefficients A_{coeff}, there exist a unique invertible transformation matrix D such that the Bernstein polynomial basis coefficient P can be computed as $A_{\text{coeff}} = PD$. Moreover, the matrix D is defined by the relation $\text{stack}(\{b_{i,n}(t)\}) = D\,\text{stack}(\{t^i\})$ for arbitrary t.

For instance, a cubic spline can be written as

$$\begin{aligned} &p(t) = a_3t^3 + a_2t^2 + a_1t + a_0 = \\ &P_3t^3 + 3P_2(t^2 - t^3) + 3P_1(t - 2t^2 + t^3) + P_0(1 - 3t + 3t^2 - t^3) \end{aligned} \tag{3}$$

where

$$\begin{aligned} &a_3 = P_3 - 3P_2 + 3P_1 - P_0, \quad &&a_2 = 3P_2 - 6P_1 + 3P_0, \\ &a_1 = 3P_1 - 3P_0, &&a_0 = P_0. \end{aligned} \tag{4}$$

The polynomial coefficient representation A_{coeff} is vital for the definition of the reference trajectory controller (Sec. III-E), while the Bernstein coefficient representation P is key for the convex hull property (Sec. II-C).

[1]The approach of this paper does not preclude the joint optimization of the trajectory and controllers, although this is left as future work

C. Bounding Bernstein polynomials and their derivatives

We include two lemmata that allow us to set polytopic bounds on polynomials and their derivatives given their Bernstein coefficients.

Lemma 2 (Matrix representation of derivatives): Let $b_n \in \mathbb{R}^{n+1}(t)$ be the vector of all Bernstein polynomials of order n. Let the matrix $H_n \in \mathbb{R}^{(n+1)\times n}$ be defined as $H_n = n\left(\begin{bmatrix} -I \\ 0^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} 0^{\mathrm{T}} \\ I \end{bmatrix}\right)$, where 0^{T} denotes a row of zeros; define also $H_{n,q} = \prod_{m=n}^{n-q+1} H_m$. Then, we have $\dot{b}_n(t) = H_n b_{n-1}(t)$ and, for higher-order derivatives of order q, we have $b_n^{(q)}(t) = H_{n,q} b_{n-q}(t)$.

Proof: From the definition of Bernstein polynomial and expanding, one can verify $\dot{b}_{\nu,n}(x) = n\left(b_{\nu-1,n-1}(x) - b_{\nu,n-1}(x)\right)$. The claim follows by writing the relation in matrix form, and applying the same multiple times for higher-order derivatives. ∎

We define the n-dimensional probability simplex as: $\Delta_n = \{\rho \in \mathbb{R}^{n+1} : 0 \le \rho \le 1, \mathbf{1}^{\mathrm{T}}\rho = 1\}$.

Lemma 3 (Convex hull property and derivative bounds): The polynomial $p(t)$ can be bounded as $p(t) \in \{P\rho : \rho \in \Delta_{n_p}\} = \mathrm{co}(\{P_i\})$ for all $t \in [0,1]$ (where $\mathrm{co}(\cdot)$ denotes the convex hull operation). The derivatives of $p(t)$ can be bounded as $p^{(q)}(t) \in \{PH_{n_p,q}\rho : \rho \in \Delta_{n_p-q}\}$ for all $t \in [0,1]$.

Proof: The first part of the proof is well known, and is a consequence of the partition of unity property of Bernstein polynomials [23]. The second part follows from the first, $p = Pb_n$ and Lemma 2. ∎

Lemma 3 allows us to synthesize controllers for each polytope while ensuring that the both the polynomial trajectories and the controlled trajectories remain within the polytope and avoid collisions with the walls of the environment at all times; this result requires the representation of p with Bernstein coefficient.

D. Polygonal Environment Decomposition

We assume a polygonal environment as our free configuration space for both trajectory generation and controller synthesis. The convex polygonal environment $\mathcal{E} \in \mathbb{R}^d$ is decomposed into a finite number of convex cells each given as $\mathcal{X}$. These convex cells may overlap, but must collectively cover the entire environment $\mathcal{E}$, i.e., $\bigcup_e \mathcal{X}_e = \mathcal{E}$. Each cell $\mathcal{X}$ is a polytope represented as $A_h x \le b_h$, where x is the state of the agent. We assume that the control points of each segment of the overall reference trajectory are contained in a single cell (see Lemma 3), and that each segment starts and ends at the intersection of two cells (this will be used for switching controllers in Section III-F).

E. Problem statement

The goal of our work is to design an output feedback controller for each convex cell, and ensure that within each cell the agent converges to the specified reference polynomial trajectory while remaining safe with respect to the walls of the polygonal configuration environment. Specifically, we want to design an output feedback control of the form

$$u = K\begin{bmatrix} y \\ p \end{bmatrix}, \quad K = \begin{bmatrix} K_y & K_p \end{bmatrix} \tag{5}$$

where K_y is the component of the matrix K that corresponds to the system output y and K_p is the component of K that corresponds to the reference trajectory p. The tracking control objective is $\lim_{t\to\infty}\big(p(t) - x(t)\big) = 0$

III. PROPOSED SOLUTION

Our proposed solution comprises five parts. We begin by writing the reference polynomial trajectories as the output of a reference dynamical system with given initial conditions. We then consider both the agent system dynamics and the reference dynamical system as joint system. We derive stability constraints in order to ensure that the synthesized controllers steer the agent onto the reference polynomial trajectories. We then derive safety constraints that guarantee collision avoidance with the boundaries of the environment. We conclude by combining both safety and stability constraints in a controller synthesis problem which produces a controller that tracks the given reference polynomial trajectory without collisions with walls.

A. Polynomial trajectories

In this section, we show that any polynomial trajectory of the form (2) can be written as the output of an autonomous linear dynamical system where the system matrices are fixed by the order of the polynomial, and the initial conditions determine the overall shape. We formally define the reference system as

$$\dot{x}_p = A_p x_p, \quad y_p = C_p x_p, \tag{6}$$

where A_p, C_p are given below. All proofs are omitted for space reasons, but can be found in [10].

Lemma 4: Assume $d_y = 1$. Given a polynomial $p(t)$ of the form (2), let

$$A_p = \begin{bmatrix} 0_{n_p} & I_{n_p} \\ 0 & 0_{n_p^{\mathrm{T}}} \end{bmatrix}, \tag{7a}$$

$$C_p = \begin{bmatrix} I_d & 0_{d\times n_p d} \end{bmatrix}. \tag{7b}$$

Then $y_p(t) = p(t) = [x_p]_0(t)$ (the first entry of the state vector) for all t if the initial conditions of the system satisfy

$$[x_p(0)]_i = i!a_i \quad \forall i, \tag{8}$$

where $i!$ denotes the factorial of the index.

Corollary 1: For arbitrary d_y, we have

$$A_p = \mathrm{blkdiag}(\{A_{p,k}\}_{k=1}^{d_y}), \quad C_p = \mathrm{blkdiag}(\{C_{p,k}\}) \tag{9}$$

where each $A_{p,k}, C_{p,k}$ is given by Lemma 1.

B. Combined Dynamics

Define the aggregate state $z = \begin{bmatrix} x \\ x_p \end{bmatrix} \in \mathbb{R}^{d_n}$, as the stack of the states x of the agent and the states x_p of the polynomial trajectory, ordered by dimension and then derivative order:

$$z = \text{stack}\big(x, \{x_{p_k}, \dot{x}_{p_k}, \dots, x_{p_k}^{(n_p-1)}\}_{k=1}^{n_p}\big). \tag{10}$$

Combining (1), (5), and (6), the closed-loop dynamics can be written as:

$$\dot{z} = \mathcal{A}z + \mathcal{B}u_z, \quad y = \mathcal{C}z \tag{11}$$

where $K = \begin{bmatrix} K_y & K_p \end{bmatrix}$, $\mathcal{A} = \text{blkdiag}(A, A_p)$, $\mathcal{B} = \text{stack}(B, 0_{n_p d \times d})$, and $\mathcal{C} = \text{blkdiag}(C, I_{n_p})$, and $u_z = \begin{bmatrix} K_y & K_p \end{bmatrix} \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} z$.

C. Convergence Conditions via Linear Matrix Inequalities

Considering the system dynamics of the form (11), we define a Lyapunov function as follows:

$$V(z) = V\left(\begin{bmatrix} x \\ x_p \end{bmatrix}\right) = \|x - C_p x_p\|^2. \tag{12}$$

Using the combined state z, the function V can then be rewritten in the form $V = z^T M z$, where M is given by

$$M = \begin{bmatrix} I & -C_p \\ -C_p^T & C_p^T C_p \end{bmatrix}. \tag{13}$$

The derivative of the Lyapunov function V is

$$\dot{V} = \nabla V^T \dot{z} \quad = z^T \left((M + M^T)^T \mathcal{A} + \mathcal{B}K\mathcal{C}\right) z. \tag{14}$$

Definition 1: A function V has *relative degree one* with respect to the dynamics (1) if $\mathcal{L}_B V \doteq \nabla V^{\mathrm{T}} B \neq 0$ (where $\mathcal{L}_B$ denotes the vector of Lie derivatives with respect to the fields given by the columns of B).

Assumption 1: The Lyapunov function V has relative degree equal to one, i.e., $M\mathcal{B} \neq 0$.

Intuitively, Assumption 1 implies that $\dot{V}$ explicitly depends on the control u (and hence on the choice of K).

Proposition 1: A sufficient and necessary condition that ensures $\dot{V} \leq 0$ for all z is that there exists $\mu \geq 0$ such that

$$S \preceq -\mu I, \tag{15}$$

$$S = \text{sym}\left((M + M^T)^T (\mathcal{A} + \mathcal{B}K\mathcal{C})\right). \tag{16}$$

In practice, μ is a bound on the convergence rate of the Lyapunov function V toward zero. For systems that can decompose along different dimensions (i.e., multi-dimensional systems), the lemma below shows that separating the problem across each dimension (but with a common convergence rate), still ensures convergence constraints for the whole system.

Lemma 5: Let K_{yi}, K_{pi} be matrices that satisfy the constraints (15) considering only the i-th system (i.e., using only $A_{pi}, C_{pi}, A_i, B_i, C_i$) with a common μ. Let K be the matrix formed as: $K = [\text{blkdiag}(\{K_{yi}\}), \text{blkdiag}(\{K_{pi}\})]$ Then, K satisfies the constraints for the joint system (i.e., using A_p, C_p, A, B, C).

See [10] for an example.

D. Safety Constraints by Control Barrier Functions

Let $A_{h,i} \in \mathbb{R}^{1 \times d_n}, b_{h,i} \in \mathbb{R}$ be the i-th row of A_h, b_h which define the polytope of a convex cell under consideration (see Section II-D). We define the following candidate Control Barrier Function:

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

where $i \in \{1, \cdots, s_h\}$ such that s_h is the number of walls of the convex cell.

1) Safety Constraints on K: Considering the agent's dynamics (1) and a continuously differentiable function $h(x)$ defining a forward invariant safe set $\mathcal{X}$, the function $h(x)$ is a Control Barrier Function (CBF) if there exists $\alpha \in \mathbb{R}$ and control input u such that:

$$\mathcal{L}_{Ax}h(x) + \mathcal{L}_B h(x)u(x, x_p) + \alpha^T h(x) \geq 0, \forall x \in \mathcal{X}, x_p \in \mathcal{P}, \tag{18}$$

$\mathcal{P}$ is the polytope bounding the Bernstein polynomial and its derivatives derived in II-C. Rewriting the CBF constraint to change the inequality direction,[2] we have:

$$\begin{aligned} &-(\mathcal{L}_{Ax}h(x) + \mathcal{L}_B h(x)u(x, x_p) + \alpha^T h(x)) \leq 0, \\ &\forall x \in \mathcal{X}, x_p \in \mathcal{P} \end{aligned} \tag{19}$$

Note that the constraint (19) must be satisfied for all x in the cell $\mathcal{X}$. In other words, the control input should satisfy the CBF and CLF constraints at every single point in the cell. For the implementation, we rewrite the constraint *for all* x as a maximization problem:

$$\begin{bmatrix} \max\limits_{x} -(\mathcal{L}_{Ax}h_i(x) + \mathcal{L}_B h_i(x)KCx + \alpha^T h_i(x)) \\ \text{s.t. } x \in \mathcal{X}, x_p \in \mathcal{P}. \end{bmatrix} \leq 0 \tag{20}$$

In practice, we aim to find a controller that satisfies this CBF constraint with some margin δ (which can be interpreted as a distance from walls). Using a CBF $h_i(x)$ of the form in (17), we can write:

$$\begin{bmatrix} \max\limits_{x, x_p, \delta} -A_{h,i}[(A + BK_yC + \alpha)x + BK_p x_p] \\ \text{s.t. } A_h x \leq b_h,\ x_p \in \mathcal{P} \end{bmatrix} \leq \delta + \alpha b_{h,i} \tag{21}$$

where the matrix A_z and the vector b_z describe each polygonal convex cell as stated in Section II-D. The above constraint is convex in K (because the point-wise maximum over x ensures that the left-hand side of the inequality is a convex function [7]. However, this form of the constraint cannot be implemented in an off-the-shelf solver because the controller K (which we would like to design) appears bi-linearly with the variable x. Instead, we can take the dual of the maximization problem, thereby obtaining:

$$\begin{bmatrix} \min\limits_{\{\lambda_{b,i}\}, \{\gamma_{b,i}\}, \delta} \lambda_{b,i}^T b_z + \gamma_{b,i}^T \mathcal{P} \\ \text{s.t. } A_z^T \lambda_{b,i} = [-A_{h,i}\mathcal{W}]^T \\ \gamma_{b,i} = [-A_{h,i}(BK_p)]^T \\ \lambda_{b,i} \geq 0, \end{bmatrix} \leq \delta + \alpha b_{h,i} \tag{22}$$

[2]This change becomes advantageous for the formulation of the overall control synthesis problem in Section III-C.

where $\mathcal{W} = A + BK_yC + \alpha$ and for every $i = \{1, \cdots, s_h\}$. Note that K appears linearly in the constraint and s_h denotes the number of faces of $\mathcal{X}$.

Remark 1: By strong duality, once a linear programming problem has an optimal solution, its dual is also guaranteed to have an optimal solution such that both their respective optimal costs are equal [6]. Therefore the constraint problems (21) and (22) are equivalent.

E. Control Synthesis

In this section, we synthesize the controller by solving a convex optimization problem subject to both stability and safety constraints that were formulated in Sections III-C and III-D. Specifically, we combine the safety (22) and stability (15) constraints in a single optimization problem (note that the min operator can be dropped from (22) without changing the meaning of the constraint).

$$\begin{aligned} \min_{\mu,S,K,\delta,\{\lambda_{b,i}\},\{\gamma_{b,i}\}} \quad & \mu \\ s.t.: \quad & \mu \leq 0, \quad S \preceq \mu I \\ & S = \text{sym}\left((M + M^T)^T (\mathcal{A} + \mathcal{B}K\mathcal{C})\right) \\ & \lambda_{b,i}^T b_z + \gamma_{b,i}^T \mathcal{P} \leq \delta + \alpha b_{h,i} \\ & A_z^T \lambda_{b,i} = [-A_{h,i}(A + BK_yC + \alpha)]^T \\ & \gamma_{b,i} = [-A_{h,i}(BK_p)]^T \\ & \lambda_{b,i} \geq 0, \quad \delta \geq 0, \quad i = \{1, \cdots, s_h\} \end{aligned} \tag{23}$$

where $K = \begin{bmatrix} K_y & K_p \end{bmatrix}$. The objective function and all the constraints in this optimization problem are linear, and $\mathcal{X}$ is a convex set.

Remark 2: Equation (23) is an SDP problem that can be handled with off-the-shelf solvers

F. Trajectory segments and switching controllers

As mentioned in the introduction, the overall reference trajectory is split in polynomial segments, and the environment is partitioned in overlapping convex cells, where each cell contains a segment of the trajectory. Using the method illustrated above, we can use (23) to synthesize a controller for each cell. There are two problems that remain to be addressed. First, the state of the reference system x_p needs to be initialized. Lemma 4 suggests an initialization corresponding to the initial point of the segment; however, this might make the agent travel back during the transitory convergence phase; instead we found that it is possible to initialize the system with the closest point on the trajectory (and the corresponding derivatives). We will prove this fact in a future version of the paper. Second, At the last control point for each segment, which we refer to as the *switching point*, one segment ends while another begins, and it is therefore necessary to switch between the output feedback controllers as well; while this might, in general, create discontinuities in the control signal, we believe that the convergence conditions of Section III-C and the continuity between segments of the reference trajectory imply smooth tracking, at least in a neighborhood of the reference; again, this will be investigated in future work.

IV. SIMULATIONS

A. Simulation Setup and Results

In order to assess the effectiveness of the proposed path planning algorithm, we run a set of MATLAB simulations. In our simulations, we generate a two-dimensional polygonal environment composed of ten different convex overlapping cells, as shown by the polygonal cells of different colors in Figure 1. All the reference polynomial trajectories have four control points (i.e., they are are cubic splines). The polygonal environment is 8-shaped with one self-intersection. There is one polynomial segment within each cell that is generated using the given control points and the agents have two-dimensional single-integrator dynamics ($A = B = C = I \in \mathbb{R}^{2\times 2}$). Given the control points corresponding to each reference trajectory, we synthesize a controller for each cell by solving the optimization problem we formulated in (23). We present the result of multiple, randomly-generated initial conditions (marked as $*$). As mentioned in Section III-F, the state x_p is initialized to the closest point on the reference trajectory. We observe from Figure 1 that the multiple random initializations of the agent within each convex cell all converge (relatively quickly) to the reference polynomial trajectory while avoiding the walls of the cell. We also observe that the switch between the various controllers takes place within the overlapping region as intended. Therefore, each simulated trajectory successfully completes its maneuver within the polygonal environment.

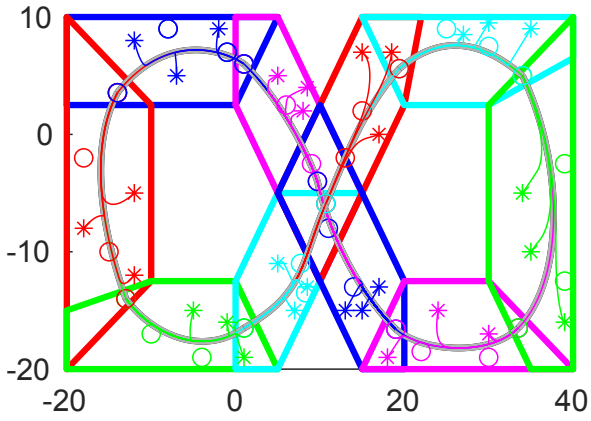

Fig. 1: Simulation results without noise. This figure presents multiple initializations of an agent that tracks the predefined polynomial trajectory. Convex cells may overlap, as in the middle portion of the environment. The reference polynomial trajectory is shown as a gray colored line. The control points defining each segment have the same colors as the corresponding cell (except for the final point, which has the color of the following cell). The agent trajectory colors correspond to the colored walls of the region in which they are initialized.

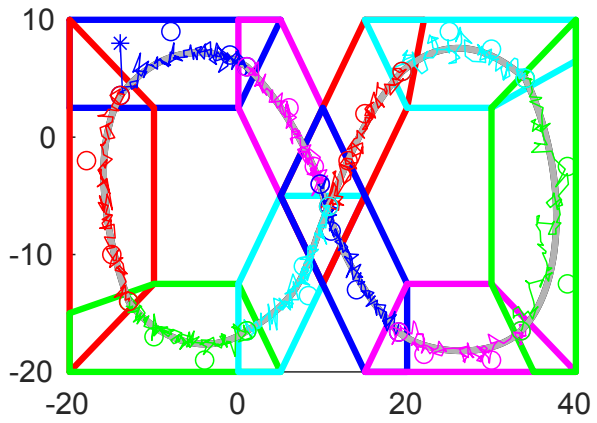

Fig. 2: Simulation results with noise. We present results for one random initialization of an agent in the presence of Gaussian noise.

B. Simulation Setup With Noise

In this section, we test the robustness of our path planning algorithm to noise. We introduce Gaussian noise into the dynamics of our agent as seen in Fig. 2. In particular, at every time step of the simulation, we add a random value to the control input with variance 0.25. The controllers implemented in this simulation are exactly the same as before. We observe that our controllers are robust to the presence of Gaussian noise, and the agent converges without collisions with the walls of the environment.

V. CONCLUSION

We proposed a novel technique for synthesizing linear output feedback controllers that successfully steer agents onto predefined polynomial trajectories while ensuring safety relative to the walls of the polygonal environment. The resulting controllers are robust to changes in agent initial conditions and noise. In the future, we will consider the joint optimization problem of finding optimal reference trajectories (control points) and synthesizing controllers to track said reference trajectories.

REFERENCES

[1] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431, 2019.

[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, 2014.

[3] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. 62(8):3861–3876, 2017.

[4] M. Bahreinian, E. Aasi, and R. Tron. Robust path planning and control for polygonal environments via linear programming. pages 5035–5042, 05 2021.

[5] M. Bahreinian, M. Mitjans, and R. Tron. Robust sample-based output-feedback path planning. pages 5780–5787, 09 2021.

[6] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. 01 1998.

[7] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[8] S. Campbell, N. O'Mahony, A. Carvalho, L. Krpalkova, D. Riordan, and J. Walsh. Path planning techniques for mobile robots a review. In *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pages 12–16, 2020.

[9] J.-w. Choi, R. Curry, and G. Elkaim. Path planning based on bézier curve for autonomous ground vehicles. In *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, pages 158–166, 2008.

[10] A. Dickson, C. G. Cassandras, and R. Tron. Spline Trajectory Tracking and Obstacle Avoidance for Mobile Agents via Convex Optimization. *arXiv e-prints*, page arXiv:2403.16900, Mar. 2024.

[11] Z. Duraklı and V. Nabiyev. A new approach based on bezier curves to solve path planning problems for mobile robots. *Journal of Computational Science*, 58:101540, 2022.

[12] M. Egerstedt and C. Martin. Control theoretic splines. optimal control, statistics, and path planning. *Control Theoretic Splines: Optimal Control, Statistics, and Path Planning*, 12 2009.

[13] N. Ezhov, F. Neitzel, and S. Petrovic. Spline approximation, part 2: From polynomials in the monomial basis to b-splines—a derivation. *Mathematics*, 9(18), 2021.

[14] C. Götte, M. Keller, T. Nattermann, C. Haß, K.-H. Glander, and T. Bertram. Spline-based motion planning for automated driving. *IFAC-PapersOnLine*, 50(1):9114–9119, 2017. 20th IFAC World Congress.

[15] K. Judd and T. McLain. Spline based path planning for unmanned air vehicles. 08 2001.

[16] H. Kano and H. Fujioka. Spline trajectory planning for path with piecewise linear boundaries. pages 439–445, 12 2018.

[17] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. pages 1478–1483, 2011.

[18] S. Lai, M. Lan, and B. Chen. Optimal constrained trajectory generation for quadrotors through smoothing splines. pages 4743–4750, 10 2018.

[19] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters*, 2(3):1688–1695, 2017.

[20] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[21] T. Mercy, R. Van Parys, and G. Pipeleers. Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*, 26(6):2182–2189, 2018.

[22] N. T. Nguyen, L. Schilling, M. S. Angern, H. Hamann, F. Ernst, and G. Schildbach. B-spline path planner for safe navigation of mobile robots. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 339–345, 2021.

[23] L. Piegl and W. Tiller. *The NURBS book*. Springer-Verlag, Berlin, Heidelberg, 1995.

[24] F. Rahmani, R. Rahmani, S. Mobayen, and A. Fekih. Lmi-based state feedback design for quadcopter optimal path control and tracking. In *2021 American Control Conference (ACC)*, pages 4655–4659, 2021.

[25] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9), 2014.

[26] P. Victerpaul, D. Saravanan, S. Janakiraman, and J. Pradeep. Path planning of autonomous mobile robots: A survey and comparison. *Journal of Advanced Research in Dynamical and Control Systems*, 9(12):1535–1565, 2017.

[27] W. Xiao, C. G. Cassandras, and C. Belta. *Control Barrier Functions*, pages 7–18. Springer International Publishing, Cham, 2023.

[28] L. Zheng, P. Zeng, W. Yang, Y. Li, and Z. Zhan. Bézier curve-based trajectory planning for autonomous vehicles with collision avoidance. *IET Intelligent Transport Systems*, 14(13):1882–1891, 2020.