Path Planning using Positive Invariant Sets

Danielson, C.; Weiss, A.; Berntorp, K.; Di Cairano, S.

TR2016-160 December 2016

Abstract

We present an algorithm for steering the output of a linear system from a feasible initial condition to a desired target position, while satisfying input constraints and nonconvex output constraints. The system input is generated by a collection of local linear state-feedback controllers. The pathplanning algorithm selects the appropriate local controller using a graph search, where the nodes of the graph are the local controllers and the edges of the graph indicate when it is possible to transition from one local controller to another without violating input or output constraints. We present two methods for computing the local controllers. The first uses a fixed-gain controller and scales its positive invariant set to satisfy the input and output constraints. We provide a linear program for determining the scale-factor and a condition for when the linear program has a closed-form solution. The second method designs the local controllers using a semi-definite program that maximizes the volume of the positive invariant set that satisfies state and input constraints. We demonstrate our path-planning algorithm on docking of a spacecraft. The semi-definite programming based control design has better performance but requires more computation.

IEEE Annual Conference on Decision and Control (CDC)

Path Planning using Positive Invariant Sets

Claus Danielson, Avishai Weiss, Karl Berntorp, and Stefano Di Cairano

Abstract— **We present an algorithm for steering the output of a linear system from a feasible initial condition to a desired target position, while satisfying input constraints and non-convex output constraints. The system input is generated by a collection of local linear state-feedback controllers. The path-planning algorithm selects the appropriate local controller using a graph search, where the nodes of the graph are the local controllers and the edges of the graph indicate when it is possible to transition from one local controller to another without violating input or output constraints. We present two methods for computing the local controllers. The first uses a fixed-gain controller and scales its positive invariant set to satisfy the input and output constraints. We provide a linear program for determining the scale-factor and a condition for when the linear program has a closed-form solution. The second method designs the local controllers using a semi-definite program that maximizes the volume of the positive invariant set that satisfies state and input constraints. We demonstrate our path-planning algorithm on docking of a spacecraft. The semi-definite programming based control design has better performance but requires more computation.**

I. Introduction

In path-planning, the goal is to generate a trajectory between an initial state and a target state while avoiding obstacle collision. This is a computationally difficult problem [1]. Recently, there has been much work on sampling-based motion planning, where the search space of possible trajectories is reduced to a graph search amongst randomly selected samples [2], [3], [4]. Sampling-based planners have been applied to humanoid robots [5], autonomous driving [6], robotic manipulators [7], and spacecraft motion problems [8], [9].

There are still many practical challenges in applying these methods to systems with complex dynamics in high dimensional spaces, since the algorithms must account for the dynamics and constraints of the system [4]. Instead, traditional sampling-based methods solve a two-point boundary-value problem, for each edge of the search graph, to find a feasible input and state trajectory. This *a posteriori* approach to satisfying the constraints and dynamics is computationally challenging [10]. Furthermore, for systems subject to unmeasured disturbances, model uncertainty, and unmodeled dynamics, there is no guarantee that the resulting open-loop trajectories will satisfy the system output constraints.

In [11] a path planner, based on the use of positively invariant sets, was developed for the spacecraft obstacle avoidance problem. Unlike the aforementioned planners, the focus is not on optimal trajectory planning or precise reference tracking. Instead, the planner in [11] implicitly finds a state trajectory from the initial state to the target state that explicitly satisfies the system dynamics and constraints. The algorithm uses a graph to switch between a collection of local feedback controllers. Positive invariant sets are used to determine when it is possible to transition from one controller to another, without violating input or output constraints. A set is positive invariant if any closed-loop state trajectory that starts inside the set remains in the set for all future times. By choosing positive invariant subsets of the input and output constraint sets, it is possible to guarantee that the closed-loop trajectories satisfy the constraints. Using a graph and simple feedback controllers to generate the control input ensures that the algorithm has low computational complexity. This concept can be extended to systems with set-bounded disturbances and differential inclusion model uncertainty, by using robust positive invariant sets [12], [13]. By sampling feedback controllers, as opposed to points in the output space, the planner inherently accounts for the dynamics of the system and produces constraint feasible trajectories. A similar idea is used in [14], [15], [16].

In this paper, we extend the path-planning algorithm presented in [11]. Our analysis provides a sufficient condition for the existence of a path from the initial output to the target output that satisfies the system dynamics and constraints. Furthermore we provide conditions, under which, our path-planning algorithm solves the path-planning problem. We introduce two methods for computing local controllers and their associated positive invariant sets. The first method follows the idea from [11]. It uses a fixed-gain controller and scales its positive invariant set to guarantee input and output constraint satisfaction. In this paper, however, we consider the output constraints as a union of convex sets that represents the free space, and provide a linear program (LP) for determining the scale-factor. We also provide a condition for when this linear program has a closed-form solution. In the second method, we design the local controllers using a semi-definite program (SDP) that maximizes the volume of the positive invariant set that satisfies state and input constraints. This approach increases the number of edges in the controller graph and can potential provide better performance, albeit at the expense of solving a computationally burdensome SDP in place of a simple LP.

The paper begins by describing the path-planning problem in Section II-A along with a condition of the existence of a solution. Section II-B details our path-planning algorithm for solving the path-planning problem. Two new methods for designing the local controllers and associated positive invariant sets are presented in Section III. Finally, we demonstrate

C. Danielson, A. Weiss, K. Berntorp, and S. Di Cairano are with Mitsubishi Electric Research Laboratories, Cambridge MA, USA
e-mail: `danielson@merl.com`

our path-planning algorithm on the docking of a spacecraft in Section IV.

A. *Notation and Definitions*

A ball $\mathcal{B}(c,r) = \{x \in \mathbb{R}^n : \|x-c\|_2 \leq r\} \subseteq \mathbb{R}^n$ is the set of all points $x \in \mathbb{R}^n$ whose Euclidean distance from the center $c \in \mathbb{R}^n$ is less than the radius $r \in \mathbb{R}_+$. A polytope $\mathcal{X} = \{x \in \mathbb{R}^n : H^j x \leq K^j \text{ for } j = 1,\dots,m\} \subseteq \mathbb{R}^n$ is the intersection of a finite number of half-spaces. A polytope $\mathcal{X}$ is called full-dimensional if it contains a ball $\mathcal{B}(c,r) \subseteq \mathcal{X}$ with positive radius $r > 0$. The Chebyshev radius of a full-dimensional polytope $\mathcal{X}$ is the radius $r > 0$ of the largest ball $\mathcal{B}(c,r) \subseteq \mathcal{X}$ contained in $\mathcal{X}$. A point $x \in \mathcal{X}$ is in the interior of $\mathcal{X}$ if there exists a radius $r > 0$ such that the ball $\mathcal{B}(x,r) \subseteq \mathcal{X}$ is contained in the set $\mathcal{X}$. The set of all points $x \in \mathcal{X}$ in the interior of $\mathcal{X}$ is denoted by $\mathrm{int}(\mathcal{X})$. The image of the set $\mathcal{X} \subseteq \mathbb{R}^n$ through the matrix $A \in \mathbb{R}^{n\times n}$ is $A\mathcal{X} = \{Ax : x \in \mathcal{X}\}$. The Pontryagin difference of $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^n$ is the set $\mathcal{X} \ominus \mathcal{Y} = \{x : x + y \in \mathcal{X}, \forall y \in \mathcal{Y}\}$.

A set $\mathcal{O}$ is positive invariant (PI) for the autonomous system $x(t+1) = f(x(t))$ if for every state $x(t) \in \mathcal{O}$ we have $x(t+1) = f(x(t)) \in \mathcal{O}$. If $V(x)$ is a Lyapunov function for the stable autonomous system $x(t+1) = f(x(t))$, then any level-set $\mathcal{O} = \{x \in \mathbb{R}^n : V(x) \leq l\}$ is a PI set since $V(f(x)) \leq V(x)$.

For a system $x(t+1) = f(x(t), u(t))$ with input $u \in \mathcal{U}$ and state $x \in \mathcal{X}$ constraints, the set of states $x \in \mathcal{R}_N(\bar{x})$ reachable from some initial state $\bar{x} \in \mathcal{X}$ in $N \in \mathbb{N}$ steps is defined recursively as $\mathcal{R}_0(\bar{x}) = \{\bar{x}\}$ and

$$\mathcal{R}_{k+1} = \big\{x \in \mathcal{X} : x = f(z,u), z \in \mathcal{R}_k, u \in \mathcal{U}\big\} \cup \mathcal{R}_k$$

for $k = 0,\dots,N-1$, where $\mathcal{R}_k = \mathcal{R}_k(\bar{x})$. The infinite-horizon reachable set is the limit $\mathcal{R}_\infty(\bar{x}) = \lim_{N\to\infty} \bigcup_{k=0}^{N} \mathcal{R}_k(\bar{x})$. If the system $x(t+1) = f(x(t), u(t))$ is locally controllable, f is continuous, and the input set $\mathcal{U}$ and state set $\mathcal{X}$ are full-dimensional and contain a neighborhood of the origin, then there exists an time $N \in \mathbb{N}$ such that the reachable set $\mathcal{R}_N(\bar{x})$ is full-dimensional.

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a set of vertices $\mathcal{V}$ together with a set of ordered pairs $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ called edges. Vertices $i, j \in \mathcal{V}$ are called adjacent if $(i,j) \in \mathcal{E}$ is an edge. A path is a sequence of adjacent vertices. A graph search is an algorithm for finding a path through a graph.

II. PATH-PLANNING ALGORITHM

In this section we define the path-planning problem and our algorithm for solving it. Our algorithm extends the method presented in [11].

A. *Path-Planning Problem*

Consider the following discrete-time linear system

$$x(t+1) = Ax(t) + Bu(t) \tag{1a}$$

$$y(t) = Cx(t) \tag{1b}$$

where $x(t) \in \mathbb{R}^{n_x}$ is the state, $u(t) \in \mathbb{R}^{n_u}$ is the control input, and $y(t) \in \mathbb{R}^{n_y}$ is the output. The pair (A,B) is assumed to be controllable and $\mathrm{rank}(C) = n_y$. The input $u(t)$ and output $y(t)$ are subject to constraints

$$u(t) \in \mathcal{U} \text{ and } y(t) \in \mathcal{Y}$$

where the input set $\mathcal{U} \subset \mathbb{R}^{n_u}$ is a full-dimensional compact polytope

$$\mathcal{U} = \big\{u : H_u^j u \leq K_u^j \text{ for } j = 1,\dots,m_u\big\}. \tag{2a}$$

The output set $\mathcal{Y} \subseteq \mathbb{R}^{n_y}$ is generally non-convex, but can be described as the union of convex sets

$$\mathcal{Y} = \bigcup_{k\in\mathcal{I}_\mathcal{Y}} \mathcal{Y}_k \tag{2b}$$

where the index set $\mathcal{I}_\mathcal{Y}$ is finite $|\mathcal{I}_\mathcal{Y}| < \infty$ and each component set $\mathcal{Y}_k \subseteq \mathbb{R}^{n_y}$ is a full-dimensional compact polytope

$$\mathcal{Y}_k = \big\{y : H_{y_k}^j y \leq K_{y_k}^j \text{ for } j = 1,\dots,m_{y_k}\big\}. \tag{2c}$$

We assume that each output $\bar{y} \in \mathcal{Y}$ corresponds to a feasible equilibrium of the system (1) i.e. for each output $\bar{y} \in \mathcal{Y}$ there exists a state $\bar{x} \in \mathbb{R}^{n_x}$ and feasible input $\bar{u} \in \mathrm{int}(\mathcal{U})$ such that

$$\begin{bmatrix} A-I & B \\ C & 0 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{u} \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{y} \end{bmatrix}. \tag{3}$$

The equilibrium state $\bar{x}$ and input $\bar{u}$ may not be unique.

The objective of the path-planning problem is to drive the system output $y(t)$ from the feasible initial equilibrium output $y(0) = y_0 \in \mathrm{int}(\mathcal{Y})$ to a desired target output $y(t) \to y_f \in \mathrm{int}(\mathcal{Y})$. The path-planning problem is formally stated below.

Problem 1: Find a feasible input trajectory $u(t) \in \mathcal{U}$ for $t \in \mathbb{N}$ that produces a feasible output trajectory $y(t) \in \mathcal{Y}$ that converges to the target output $\lim_{t\to\infty} y(t) = y_f$. $\square$

Before detailing our algorithm for solving Problem 1, we study the conditions for when a solution exists. Consider the following graph $\mathcal{G}_\mathcal{Y} = (\mathcal{I}_\mathcal{Y}, \mathcal{E}_\mathcal{Y})$ where the nodes of the graph are the indices $\mathcal{I}_\mathcal{Y}$ of the component sets $\mathcal{Y}_k$ that comprise $\mathcal{Y}$ in (2b). An edge $(i,j) \in \mathcal{E}_\mathcal{Y}$ connects nodes $i, j \in \mathcal{I}_\mathcal{Y}$ if the intersection of the sets $\mathcal{Y}_i$ and $\mathcal{Y}_j$ has a non-empty interior $\mathrm{int}(\mathcal{Y}_i \cap \mathcal{Y}_j) \neq \varnothing$. The following theorem provides a sufficient condition for the existence of a solution to Problem 1.

Theorem 1: If the graph $\mathcal{G}_\mathcal{Y}$ has a path from a node $\mathcal{Y}_0$ containing the initial equilibrium output y_0 to a node $\mathcal{Y}_f$ containing the target output y_f then Problem 1 can be solved.

The proof of Theorem 1 can be found in [17]. The proof does not rely on the linearity of the dynamics (1) nor the polytopic union structure of the constraints (2). Rather the proof uses the fact that the dynamics $f(x,u) = Ax + Bu$ are continuous and locally controllable about each equilibrium (3), and that the constraints are full-dimensional. Thus Theorem 1 can be extended to non-linear systems with more complicated constraints. However in this paper we consider linear dynamics with polytopic union constraints, since our path-planning algorithm exploits these properties.

B. Path-Planning Algorithm

In this section we present our algorithm for solving Problem 1. This algorithm extends the concept first presented in [11].

The control input $u(t)$ is selected from a collection of local linear state-feedback controllers of the form

$$u_i = F_i(x - \bar{x}_i) + \bar{u}_i \tag{4}$$

for $i \in \mathcal{I}$, where $\mathcal{I}$ is the index set of the local controllers, and $\bar{x}_i$ and $\bar{u}_i \in \text{int}(\mathcal{U})$ are an equilibrium (3) state and input pair corresponding to the output $\bar{y}_i \in \text{int}(\mathcal{Y})$. We assume that the local controller (4) asymptotically stabilizes its equilibrium point $\bar{x}_i$ i.e. the matrix $A + BF_i$ is Schur. Thus each controller (4) has an associated ellipsoidal positive invariant (PI) set

$$\mathcal{O}_i = \left\{x \in \mathbb{R}^{n_x} : (x - \bar{x}_i)^T P_i (x - \bar{x}_i) \leq 1\right\} \tag{5}$$

where $V(x) = (x - \bar{x}_i)^T P_i (x - \bar{x}_i)$ is a quadratic Lyapunov function for the controller (4). We assume that the Lyapunov matrix $P_i \in \mathbb{R}^{n_x \times n_x}$ is scaled such that for every state $x \in \mathcal{O}_i$ in the PI set $\mathcal{O}_i \subset \mathbb{R}^{n_x}$, the output $y = Cx \in \mathcal{Y}$ and input $u = F_i(x - \bar{x}_i) + \bar{u}_i \in \mathcal{U}$ are feasible. Such a scaling is possible since $\bar{u}_i \in \text{int}(\mathcal{U})$ and $\bar{y}_i \in \text{int}(\mathcal{Y})$, and the set $\{x : Cx \in \mathcal{Y}\}$ is full-dimensional, since $\mathcal{Y}$ is full-dimensional and $\text{rank}(C) = n_y$.

The path-planning algorithm selects which local controller (4) to use based on a graph search. The vertices $\mathcal{I}$ of the graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ are the indices of the local feedback controllers (4). Two controllers $i, j \in \mathcal{I}$ are connected by an edge $(i, j) \in \mathcal{E}$ if the equilibrium state $\bar{x}_i$ of controller $i \in \mathcal{I}$ is inside $\bar{x}_i \in \text{int}(\mathcal{O}_j)$ the PI set $\mathcal{O}_j$ of controller $j \in \mathcal{I}$. The presence of an edge $(i, j) \in \mathcal{E}$ means that it is possible to safely transition from controller $i \in \mathcal{I}$ to controller $j \in \mathcal{I}$ once the state $x(t)$ reaches a neighborhood of the equilibrium $\bar{x}_i$.

We make the following assumptions about the controller graph $\mathcal{G}$:

A1. The set of controllers $\mathcal{I}$ contains at least one controller $u_f = F_f(x - \bar{x}_f) + \bar{u}_f$ corresponding to the target output $y_f = C\bar{x}_f \in \mathcal{Y}$.

A2. The initial state $x(0) = x_0$ of the system (1) is contained in the PI set $\mathcal{O}_i$ of at least one controller $i \in \mathcal{I}$.

A3. There exists a path through the graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ from a node containing the initial state $x(0) = x_0$ to a node corresponding to the target output y_f.

Our path-planning algorithm is summarized in Algorithm 1. Offline, the path-planning algorithm searches the controller graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ for a sequence $\sigma_0, \dots, \sigma_N \in \mathcal{I}$ of local controllers (4) from a node σ_0, whose PI set $\mathcal{O}_{\sigma_0}$ contains the inital state $x(0) \in \mathcal{O}_{\sigma_0}$ to a node σ_f, whose PI set $\mathcal{O}_{\sigma_f}$ contains an equilibrium state $\bar{x}_f$ corresponding to the target output y_f. At each time instance $t \in \mathbb{N}$, the path planner uses the control input $u(t) = F_{\sigma(t)}(x(t) - \bar{x}_{\sigma(t)}) + \bar{u}_{\sigma(t)}$ where $\sigma(t)$ is the current node. The controller node $\sigma(t)$ is updated $\sigma(t) = \sigma_{i+1}$ when the state $x(t)$ reaches the PI set $\mathcal{O}_{\sigma_{i+1}}$ for the next local controller σ_{i+1}. The initial node is $\sigma(t) = \sigma_0$.

Algorithm 1 Path-Planning Algorithm

```
1: initial local controller $\sigma(t) = \sigma_0$
2: for each time $t \in \mathbb{N}$ do
3:     if $x(t) \in \mathcal{O}_{\sigma_{i+1}}$ then
4:         update local controller (4) node $\sigma(t) = \sigma_{i+1}$
5:     else
6:         use same local controller $\sigma(t) = \sigma_i$
7:     end if
8:     $u(t) = F_{\sigma(t)}(x(t) - \bar{x}_{\sigma(t)}) + \bar{u}_{\sigma(t)}$
9: end for
```

The following theorem shows that the path-planning algorithm satisfies the constraints and drives the system to the target output $y(t) \to y_f \in \mathcal{Y}$.

Theorem 2: Algorithm 1 solves Problem 1

Proof: The proof can be found in [17]. ■

Algorithm 1 is a general path-planning algorithm. There are three questions that must be addressed to implement this algorithm:

1. How do we design the local controllers (4) such that their PI sets (5) satisfy the input and output constraints?
2. How do we sample the output set $\mathcal{Y}$ such that the controller graph $G = (\mathcal{I}, \mathcal{E})$ contains a path from a node $i \in \mathcal{I}$ whose PI $\mathcal{O}_i$ contains the initial state $x_0 \in \mathcal{O}_i$ to a node $f \in \mathcal{I}$ whose PI set $\mathcal{O}_f$ contains an equilibrium state $x_f \in \mathcal{O}_f$ corresponding to the target output $y_f \in \mathcal{Y}$.
3. How do we weight the graph $G = (\mathcal{I}, \mathcal{E})$ to provide good performance?

This paper focuses on answering the first question.

III. Control Design

In this section we present two methods for designing the local controllers (4) with PI sets (5) that satisfy the input $F_i(\mathcal{O}_i - \bar{x}_i) + \bar{u}_i \subseteq \mathcal{U}$ and output constraints $C\mathcal{O}_i \subseteq \mathcal{Y}$.

A. Fixed-Gain Controller with Scaled Invariant Set

Our first method uses a single feedback gain matrix $F_i = F$ for each $i \in \mathcal{I}$ of the local controller (4). Each of the local controllers (4) has a quadratic Lyapunov function of the form $V(x) = (x - \bar{x}_i)^T P (x - \bar{x}_i)$ that share a common Lyapunov matrix $P_i = P \succ 0$. The PI sets of the local controller $i \in \mathcal{I}$ is the ρ_i^2 level-set of the local Lyapunov function

$$\mathcal{O}_i(\rho_i) = \left\{x \in \mathbb{R}^{n_x} : (x - \bar{x}_i)^T P (x - \bar{x}_i) \leq \rho_i^2\right\}. \tag{6}$$

To maximize the number of edges $(i, j) \in \mathcal{E}$ between the local controllers, we would like to maximize the volume of the PI set (6) by choosing the maximum level ρ_i^2 for which we can still satisfy the input $F(\mathcal{O}_i(\rho_i) - \bar{x}_i) + \bar{u}_i \subseteq \mathcal{U}$ and output $C\mathcal{O}_i(\rho_i) \subseteq \mathcal{Y}$ constraints. Since the output set $\mathcal{Y}$ is generally non-convex, this is a non-convex problem. However the output $y_i \in \mathcal{Y} = \bigcup \mathcal{Y}_k$ must be contained in at least one component set $\mathcal{Y}_k \subseteq \mathcal{Y}$ of $\mathcal{Y}$. Then we can solve the relaxed convex problem

$$\begin{aligned} \text{maximize} \quad & \rho_i && \text{(7a)} \\ \text{subject to} \quad & F(\mathcal{O}_i(\rho_i) - \bar{x}_i) + \bar{u}_i \subseteq \mathcal{U} && \text{(7b)} \\ & C\mathcal{O}_i(\rho_i) \subseteq \mathcal{Y}_k && \text{(7c)} \end{aligned}$$

which finds the maximum level ρ_i such that $C\mathcal{O}_i(\rho_i) \subseteq \mathcal{Y}_k$ is contained in the convex subset $\mathcal{Y}_k \subseteq \mathcal{Y}$ of $\mathcal{Y}$.

If the system (1) has multiple equilibria for the output sample $\bar{y}_i \in \mathcal{Y}$ then the decision variables of problem (7) are the level $\rho_i \in \mathbb{R}$, the equilibrium state $\bar{x}_i \in \mathbb{R}^{n_x}$, and input $\bar{u}_i \in \mathcal{U}$. In this case, problem (7) can be recast as a linear program.

Proposition 1: Problem (7) is equivalent to the linear program

$$\begin{aligned} \max_{\rho_i, \bar{x}_i, \bar{u}_i} \quad & \rho_i && \text{(8a)} \\ \text{s.t.} \quad & H_u^j \bar{u}_i + \rho_i \|H_u^j F P^{-1/2}\|_2 \leq K_u^j && \text{(8b)} \\ & H_{y_k}^j \bar{y}_i + \rho_i \|H_{y_k}^j C P^{-1/2}\|_2 \leq K_{y_k}^j && \text{(8c)} \\ & A\bar{x}_i + B\bar{u}_i = 0,\ C\bar{x}_i = \bar{y}_i, \bar{u}_i \in \mathcal{U} && \text{(8d)} \end{aligned}$$

where (H_u^j, K_u^j) for $j = 1, \ldots, m_u$ are the half-spaces of the input set (2a) and $(H_{y_k}^j, K_{y_k}^j)$ are the half-spaces of the k-th output set (2c).

Proof: The proof can be found in [17]. ■

If the system (1) has a unique equilibrium state $\bar{x}_i$ and input $\bar{u}_i$ for the output sample $\bar{y}_i \in \mathcal{Y}$ then problem (7) has a closed-form solution as shown in Corollary 1 below.

Corollary 1: Let $\bar{x}_i$ and $\bar{u}_i$ be the unique equilibrium state and input (3) respectively for the output $\bar{y}_i \in \mathcal{Y}_k \subseteq \mathcal{Y}$. Then the optimal solution to problem (7) is given by

$$\rho_i^\star = \min \left\{ \frac{K_u^j - H_u^j \bar{u}_i}{\|H_u^j F_i P^{-1/2}\|}, \frac{K_{y_k}^j - H_{y_k}^j \bar{y}_i}{\|H_{y_k}^j C P^{-1/2}\|} \right\} \quad (9)$$

where (H_u^j, K_u^j) for $j = 1, \ldots, m_u$ are the half-spaces of the input set (2a) and $(H_{y_k}^j, K_{y_k}^j)$ are the half-spaces of the k-th output set (2c).

Since the Lyapunov matrix P is shared by all the local controllers, the half-space parameters (H_u^j, K_u^j) and $(H_{y_k}^j, K_{y_k}^j)$ can be normalized offline such that $\|H_u^j F_i P^{-1/2}\| = 1$ for all $j = 1, \ldots, m_u$ and $\|H_{y_k}^j C P^{-1/2}\| = 1$ for all $j = 1, \ldots, m_{y_k}$. Thus evaluating (9) has computational complexity $O(n_u m_u + n_y m_{y_k})$ where n_u and n_y are the number of system inputs and outputs respectively, and m_u and m_{y_k} are the respective number of constraints that define the input $\mathcal{U}$ and output $\mathcal{Y}_k$ sets. Evaluating (9) has the same computational complexity as testing the set memberships $\bar{u}_i \in \mathcal{U}$ and $\bar{y}_i \in \mathcal{Y}_k$. In fact the computations used to test set membership can be reused to evaluate (9). Thus the PI sets (5) for the local controllers (4) and hence the controller graph $\mathcal{G}$ can be constructed efficiently in real-time.

B. Design of Controllers by Semi-Definite Programming

In this section we present a method for obtaining the local controllers (4) by solving a semi-definite program. We assume that the system (1) has a unique equilibrium state $\bar{x}_i$ and input $\bar{u}_i$ for each sample output $\bar{y}_i \in \mathcal{Y}$.

The feedback gain F_i and Lyapunov matrix P_i for the i-th controller are obtained by solving the problem

$$\begin{aligned} \text{maximize} \quad & \log\det P_i^{-1} && \text{(10a)} \\ \text{subject to} \quad & \begin{bmatrix} P_i^{-1} & \cdot \\ AP_i^{-1} + BF_iP_i^{-1} & P_i^{-1} \end{bmatrix} \succ 0 && \text{(10b)} \\ & \begin{bmatrix} P_i^{-1} & \cdot \\ H_{\mathcal{U}}^j F_i P_i^{-1} & \left(K_u^j - H_u^j \bar{u}_i\right)^2 \end{bmatrix} \succeq 0 && \text{(10c)} \\ & \begin{bmatrix} P_i^{-1} & \cdot \\ H_{y_k}^j C P_i^{-1} & \left(K_{y_k}^j - H_{y_k}^j \bar{y}_i\right)^2 \end{bmatrix} \succeq 0. && \text{(10d)} \end{aligned}$$

The constraints of problem (10) are linear and the cost function (10a) is concave in the decision variables $X = P^{-1}$ and $Y = FP^{-1}$. Problem (10) is therefore a semi-definite program, which can be efficiently solved using standard software packages [18], [19].

Proposition 2 shows that problem (10) finds a controller with the largest constraint satisfying PI set.

Proposition 2: Problem (10) finds the local controller (4) with the largest ellipsoidal PI set (5) that satisfies the input (2a) and output (2b) constraints.

Proof: The proof can be found in [17]. ■

IV. Example: Spacecraft Maneuver Planning

In this section we apply our path-planning algorithm to the problem of planning spacecraft docking maneuvers. The relative dynamics of a pair of spacecraft in the orbital-plane are modeled by the Hill-Clohessy-Wiltshire equations [20]

$$\begin{aligned} \ddot{y}_1 &= 2n\dot{y}_2 + 3n^2 y_1 + u_1 && \text{(11a)} \\ \ddot{y}_2 &= -2n\dot{y}_1 + u_2 && \text{(11b)} \end{aligned}$$

where y_1 is the difference in radial position of the spacecraft and y_2 is the difference in position along the orbital velocity direction. The state vector $x = [y_1, y_2, \dot{y}_1, \dot{y}_2]^T$ contains the relative radial and orbital positions and velocities. The inputs u_1 and u_2 are the thrusts normalized by the spacecraft mass along the radial and orbital velocity directions. The dynamics are linearized about a circular orbit of 415 kilometers which gives $n = 1.1 \times 10^{-3}$ inverse-seconds in (11). The dynamics (11) are discretized with a sample period of 30 seconds. The normalized thrusts must satisfy the input constraints

$$-10^{-2} \leq u_1, u_2 \leq 10^{-2} \quad (12)$$

newtons per kilogram.

We consider the scenario of planning a maneuver around a piece of debris shown in Fig. 1. The debris is a square with a 100 meter side length located at $[300, 400]^T$ meters. The output set $\mathcal{Y}$ is the set difference of the bounding box $[-400, 1000] \times [-400, 1100]$ meters and the debris set. The component sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_4$ that comprise the output set $\mathcal{Y} = \mathcal{Y}_1 \cup \cdots \cup \mathcal{Y}_4$ were obtained by flipping each of the 4 constraints that define the debris set and intersecting with the bounding box. The component sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_4$ are shown in Fig. 2.

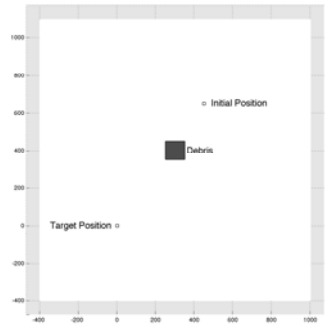
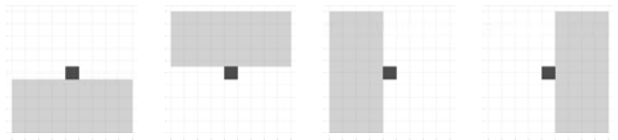


Fig. 1. Initial position and target position of the spacecraft with debris between. The white area is the output set $\mathcal{Y} \subset \mathbb{R}^2$ and the red square is the debris.



(a) $\mathcal{Y}_1$ (b) $\mathcal{Y}_2$ (c) $\mathcal{Y}_3$ (d) $\mathcal{Y}_4$

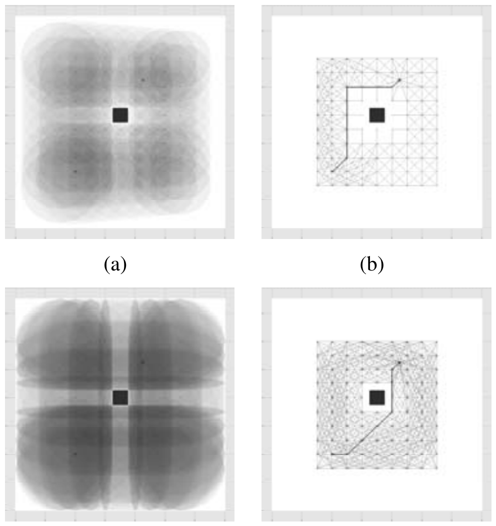
Fig. 2. The four component sets $\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3, \mathcal{Y}_4$ that comprise the non-convex output set $\mathcal{Y} = \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \mathcal{Y}_3 \cup \mathcal{Y}_4$.

The spacecraft is initially positioned at $y = [450, 650]^T$ meters and the target position is the origin $y_f = [0, 0]^T$. The controller graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ was generated by gridding the output set $\text{conv}(\mathcal{Y})$ and computing a local controller (4) at each grid point $\bar{y}_i \in \mathcal{Y}$ using the methods presented in Section III-A and Section III-B. For the first controller graph, the feedback gain $F = F_i$ is the linear quadratic regulator (LQR) with penalty matrices $Q = \text{diag}(10^2, 10^2, 10^7, 10^7)$ and $R = (2 \times 10^7)I_2$ where $\text{diag}(v)$ is a diagonal matrix with elements v on the diagonal and $I_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix. The Lyapunov matrix $P \in \mathbb{R}^{n_x \times n_x}$ is the corresponding solution to the discrete-time algebraic Riccati equation. For the second controller graph, the feedback gains F_i and Lyapunov matrices P_i of the local controllers (4) were obtained by solving the semi-definite program (10) at each grid point.

The projection $C\mathcal{O}_i$ of the PI sets $\mathcal{O}_i$ and the controller graph $\mathcal{G}$ for the fixed-gain controller are shown in Fig. 3(a) and Fig. 3(b) respectively. The projected PI sets $C\mathcal{O}_i$ for the local controllers designed by the semi-definite program (10) are shown in Fig. 3(c). The controller graph is shown in Fig. 3(d).

From Fig. 3(a) and Fig. 3(c) it is evident that the PI sets for the controllers designed using the semi-definite program (10) are larger than the scaled PI sets for the fixed-gain controller. As a result the controller graph shown in Fig. 3(d) has more edges than the controller graph for the fixed-gain controller shown in Fig. 3(b). Thus the path-planning algorithm has a larger number of potential sequences of local controllers that can maneuver the spacecraft to the origin. Hence we expect the path planner using the SDP controllers to perform better than the fixed-gain controller.

The edges $(i, j) \in \mathcal{E}$ of the controller graphs $\mathcal{G} = (\mathcal{I}, \mathcal{E})$



(a) (b) (c) (d)

Fig. 3. (a) and (c) Projected PI sets $C\mathcal{O}_i$ for the local controllers designed using a fixed-gain controller and SDP controller respectively. (b) and (d) Controller graphs $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ for the fixed-gain and SDP controllers.

are weighted W_{ij} using the infinite-horizon cost-to-go from initial state $\bar{x}_i$ to the equilibrium state $\bar{x}_j$ under the local controller (4) given by

$$W_{ij} = (\bar{x}_i - \bar{x}_j)^T S_j (\bar{x}_i - \bar{x}_j)$$

where $S_j = S$ is the solution to the discrete-time algebraic Riccati equation for the fixed-gain controller and S_j is the unique positive definite solution to the Lyapunov equation

$$(A + BF_j)^T S_j (A + BF_j) - S_j = -Q - F_j^T R F_j.$$

for the controller F_j designed using the semi-definite program (10).

The optimal sequence of local controller nodes are shown in Fig. 3(b) and Fig. 3(d) respectively. These controller sequences are used by Algorithm 1 to maneuver the spacecraft to the origin. The resulting output trajectories are shown in Fig. 4. The output trajectories are compared with the output trajectory produced by using a single LQR. Under the single LQR, the spacecraft passed through the debris field before converging to the target position. On the other hand, the output trajectories produced by Algorithm 1 using the fixed-gain and SDP local controllers avoided the debris set while converging to the target position. However the output trajectories took very different paths to reach the origin. We evaluated these trajectories using the cost function cost-to-go to a neighborhood of the origin

$$J = \sum_{t=0}^{N} x(t)^T Q x(t) + u(t)^T R u(t)$$

where $x(t)$ and $u(t)$ were the state and input trajectories, respectively, produced by Algorithm 1. The fixed-gain local controllers had a cost $J_{base} = 1.14 \times 10^{10}$ and the SDP designed local controllers had a cost $J_{sdp} = 2.15 \times 10^9$. Thus for this example problem the SDP designed local controller produced a more efficient path to the origin than the fixed-gain local controllers. The disadvantage of the SDP-based control design is computation time. The PI sets and controller graph for the fixed-gain controller required 0.36 seconds to compute. While the PI sets and controller graph for the SDP based controllers required 48 seconds to compute. Thus the SDP based control design required more than 2 orders of magnitude more time to compute while providing approximately one order of magnitude improvement in performance.

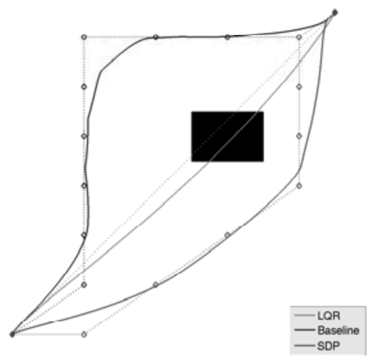


Fig. 4. Trajectories produced by Algorithm 1 by searching the controller graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ and using a sequence of local controllers. Shown are the sequences of equilibrium outputs $\bar{y}_{\sigma_1}, \ldots, \bar{y}_{\sigma_f}$ from the controller graph $\mathcal{G}$ and the resulting output trajectory $y(t)$ produced by switching between these equilibrium outputs.

The normalized thrusts for both controller graphs and the LQR controller are shown in Fig. 5. The LQR violated the thrust constraints while the inputs $u(t)$ produced by Algorithm 1 did not violate constraints. Overall the normalized thrusts produced by the path-planning algorithm are small, hence requiring little propellant.

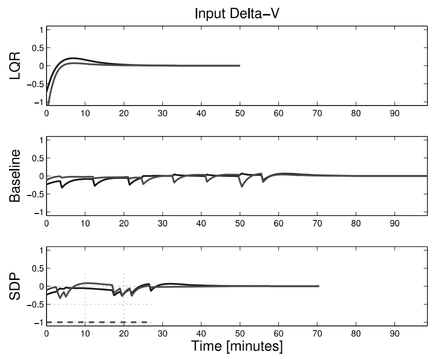


Fig. 5. Input trajectories $u(t)$ for $t \in \mathbb{N}$ produced by the linear quadratic regulator, path planner with the fixed-gain local controller, and path planner with the SDP local controllers. The LQR controller violates input constraints. The fixed-gain and SDP controllers satisfy input constraints.

V. Conclusions and Future Work

This paper presented a path-planning algorithm for linear systems whose output is restricted to a union of polytopic sets. The path-planning algorithm uses a graph to switch between a collection of local linear feedback controllers. We presented two methods for computing the local controllers and their positive invariant sets. The first used a fixed feedback gain and scaled the positive invariant set to satisfy state and input constraints. The second used a semi-definite program to design both the controller gain and positive invariant set. The path-planning algorithm was applied to the spacecraft docking problem.

Future work will address path-planning for systems with disturbances and modeling errors. In addition we will study the problem of sampling the output space $\mathcal{Y}$ in a manner that guarantees that the controller graph $\mathcal{G}$ contains a path from the current output to the target output when one exist. Finally we will study different weighting heuristics for the controller graph edges and how they effect closed-loop performance.

References

[1] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Conference on Foundations of Computer Science*, 1979.
[2] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
[3] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, 2001.
[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, 2011.
[5] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, 2002.
[6] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, 2008.
[7] A. Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. Teller, and M. Walter, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms," in *Intelligent Robots and Systems (IROS)*, 2011.
[8] E. Frazzoli, "Quasi-random algorithms for real-time spacecraft motion planning and coordination," *Acta Astronautica*, 2003.
[9] J. Starek, G. Maher, B. Barbee, and M. Pavone, "Real-time, fuel-optimal spacecraft motion planning under Clohessy-Wiltshire-Hill dynamics," in *IEEE Aerospace Conference*, 2016.
[10] R. Vinter, *Optimal control*. Springer Science, 2010.
[11] A. Weiss, C. Petersen, M. Baldwin, R. Erwin, and I. Kolmanovsky, "Safe positively invariant sets for spacecraft obstacle avoidance," *Journal of Guidance, Control, and Dynamics*, 2015.
[12] I. Kolmanovsky and E. G. Gilbert, "Theory and computation of disturbance invariant sets for discrete-time linear systems," *Mathematical problems in engineering*, 1998.
[13] E. Kerrigan, "Robust constraints satisfaction: Invariant sets and predictive control," Ph.D. dissertation, Dep. of Engineering, University of Cambridge, 2000.
[14] O. Arslan, K. Berntorp, and P. Tsiotras, "Sampling-based algorithms for optimal motion planning using closed-loop prediction," *arXiv preprint arXiv:1601.06326*, 2016.
[15] W. McConley, B. Appleby, M. Dahleh, and E. Feron, "A computationally efficient lyapunov-based scheduling procedure for control of nonlinear systems with stability guarantees," *Transactions on Automatic Control*, 2000.
[16] F. Blanchini, F. Pellegrino, and L. Visentini, "Control of manipulators in a constrained workspace by means of linked invariant sets," *Journal of Robust and Nonlinear Control*, 2004.
[17] C. Danielson, A. Weiss, K. Berntorp, and S. Di Cairano, "Path planning using positive invariant sets," *arXiv preprint arXiv:TBD*, 2016.
[18] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, 1999.
[19] K. Toh, M. Todd, and R. Tutuncu, "SDPT3 — a MATLAB software package for semidefinite programming," *Optimization Methods and Software*, 1999.
[20] B. Wie, *Spacecraft Dynamics and Control*. AIAA, 2010.