Robust Tracking with Model Mismatch for Fast and Safe Planning: an SOS Optimization Approach

Sumeet Singh, Mo Chen, Sylvia Herbert, Claire Tomlin and Marco Pavone

Robust Tracking with Model Mismatch for Fast and Safe Planning: an SOS Optimization Approach

Sumeet Singh[1], Mo Chen[1], Sylvia L. Herbert[2], Claire J. Tomlin[2], and Marco Pavone[1]*

[1] Dept. of Aeronautics and Astronautics, Stanford University
{ssingh19, mochen72, pavone}@stanford.edu
[2] Dept. of Electrical Engineering and Computer Science, University of California, Berkeley
{sylvia.herbert, tomlin}@berkeley.edu

Abstract. In the pursuit of real-time motion planning, a commonly adopted practice is to compute trajectories by running a planning algorithm on a simplified, low-dimensional dynamical model, and then employ a feedback tracking controller that tracks such a trajectory by accounting for the full, high-dimensional system dynamics. While this strategy of *planning with model mismatch* generally yields fast computation times, there are no guarantees of dynamic feasibility, which hampers application to safety-critical systems. Building upon recent work that addressed this problem through the lens of Hamilton-Jacobi (HJ) reachability, we devise an algorithmic framework whereby one computes, offline, for a pair of "planner" (i.e., low-dimensional) and "tracking" (i.e., high-dimensional) models, a feedback tracking controller and associated tracking bound. This bound is then used as a safety margin when generating motion plans via the low-dimensional model. Specifically, we harness the computational tool of sum-of-squares (SOS) programming to design a bilinear optimization algorithm for the computation of the feedback tracking controller and associated tracking bound. The algorithm is demonstrated via numerical experiments, with an emphasis on investigating the trade-off between the increased computational scalability afforded by SOS and its intrinsic conservativeness. Collectively, our results enable scaling the appealing strategy of planning with model mismatch to systems that are beyond the reach of HJ analysis, while maintaining safety guarantees.

1 Introduction

As robotic systems become more pervasive, real-time motion planning becomes increasingly important. In general, motion planning algorithms must find a trade-off among three key challenges: (1) achieving fast computational speed to enable online re-planning, (2) accounting for complex system dynamics, and (3) ensuring formal safety and robustness guarantees. In particular, to enable high-frequency re-planning, a commonly adopted practice is to compute a trajectory by running a planning algorithm on a simplified, low-dimensional dynamical model, oftentimes just a single-integrator model. A feedback tracking controller that accounts for the full, high-dimensional dynamics of the system is then used to track such a trajectory (we refer to this approach as *planning with model mismatch*, see Figure 1). This approach usually guarantees fast planning speeds, but guaranteeing safety becomes difficult, due to the mismatch between the model used for planning and the actual dynamics of the robotic system. This approximation can introduce tracking error between the planned path and the actual trajectory of the system, potentially resulting in safety violations. Conversely, one could
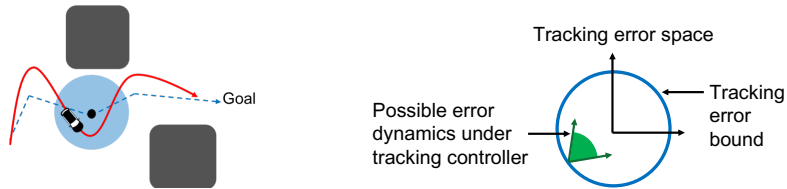
Fig. 1: *Left: Planning with model mismatch.* The tracking system (car, red trajectory) follows a motion plan computed via a low-fidelity dynamical model (blue trajectory). Tracking with model mismatch while maintaining safety requires keeping the maximum tracking error bound (TEB) below a certifiable value. *Right: TEB certification.* The TEB is characterized by the property that on its boundary, all possible error dynamics resulting from the nominal trajectory point inwards under some tracking controller, so that the TEB is never exceeded.

compute a collision-free motion plan by directly accounting for the full set of differential constraints of a robotic system (e.g., by using a general-purpose kinodynamic motion planning algorithm [16]). However, despite significant recent progress [13, 14, 17, 18], kinodynamic motion planning is still computationally intensive [16]. For robots characterized by "slow" or differentially flat dynamics, gradient-based trajectory optimization techniques (e.g., [27, 31]) may provide quick convergence to a feasible solution, but extending such methods to systems with complex dynamics is challenging.

In general, providing formal guarantees on safety and robustness typically requires reasoning on the full dynamical model of a robotic system. One of the most powerful tools in this regard is the concept of "funnels" [5]. The key idea is to ensure that the trajectory of a system remains within precomputed funnels [19, 20, 36]; this is often achieved via Lyapunov analysis techniques. However, while able to provide formal guarantees around nominal trajectories, these methods are less suitable for *a priori* unknown environments in which safe trajectories must be found online. The work in [21] tackles this challenge by considering a library of funnels, from which maneuvers are composed in sequence to form guaranteed safe trajectories. To obtain trajectory-independent tracking error bounds, contraction theory and nonlinear optimization techniques can be used to construct tubes around *dynamically-feasible* trajectories obtained from any planning algorithm [33]. However, as mentioned above, rapidly computing dynamically-feasible nominal trajectories poses a significant computational challenge.

To tackle real-time motion planning problems, recent works such as [6, 11, 15] combine low- and high-fidelity models to strike a balance among the three aforementioned challenges of computational speed, model accuracy, and robustness. In [15], a forward reachable set for the high-fidelity model is computed offline and then used to prune trajectories generated online using the low-fidelity model. The approach relies on an *assumed* model mismatch bound, expressed as the maximum distance between the low- and high-fidelity models under a certain metric. FaSTrack [6, 11] considers a slightly different definition of model mismatch, and casts a motion planning problem as a differential game wherein a low-fidelity "planning" system is being pursued (i.e., tracked) by a high-fidelity "tracking" system. Hamilton-Jacobi (HJ) reachability analysis is used to precompute a worst-case tracking error bound (TEB) as well as the optimal feedback tracking controller. Online, FaSTrack plans in real time using the low-fidelity planning system, while simultaneously tracking the plan and remaining within *guaranteed* tracking error bounds – despite the model mismatch. FaSTrack's ability to ensure real-time planning of dynamic systems with safety guarantees makes it a desirable framework, but it requires a significant precomputation step in computing the TEB and controller. Executing this precomputation using HJ reachability is reliable and straightforward for non-

linear, highly-coupled dynamical systems with up to five states, or higher-dimensional systems of a specific form [8]. However, the computations required for HJ reachability analysis scale exponentially with the number of states, thus making FaSTrack difficult to apply to systems with a large number of states and/or highly-coupled dynamics.

Building upon FaSTrack, we present an approach to designing a feedback tracking controller along with guaranteed tracking error bounds via sum-of-squares (SOS) programming. SOS programs can be cast as semidefinite programs (SDPs), which may be solved using standard convex optimization toolboxes. SOS programming has been used extensively to provide convex relaxations to fundamentally non-convex or even NP-hard problems arising in robust control [19–21, 24, 36]. In the context of this work, we leverage SOS to derive *sufficient* conditions for bounding tracking control error. The price paid for such a convex relaxation is in the tightness of the error bound guarantee (which corresponds to the size/computational tractability of the SOS SDP), but critically not in the existence of the guarantee itself provided the SOS program is feasible. SOS programming thus represents an attractive alternative to exact HJ reachability for those robotic systems where HJ is not directly applicable due computational intractability. In addition, this paper, as well as FaSTrack [6,11], has potential to complement works such as [15] that assume a model-mismatch error, by providing the TEB as well as a feedback tracking controller to achieve such a TEB.

Statement of Contributions: The contribution of this paper is threefold. First, we provide an SOS-based formulation for tracking a nominal motion plan computed on a simplified low-dimensional model. Second, leveraging the SOS formulation, we devise a bilinear optimization algorithm to jointly compute, offline, a *trajectory-independent* feedback tracking controller and the corresponding TEB. Finally, we demonstrate our approach on a 5D example for comparison with FaSTrack [11], and on a 8D example which is beyond the reach of HJ analysis. Collectively, our three contributions enable scaling the appealing strategy of planning with model mismatch to systems that are beyond the reach of HJ analysis, while maintaining safety guarantees.

Organization: The remainder of the paper is organized as follows. In Section 2 we provide the problem statement and give necessary background on SOS programming. Section 3 formalizes the problem as a bilinear SOS program, while Section 4 develops an algorithmic framework for its solution. In Section 5 we present numerical experiments. Finally, in Section 6, we draw our conclusions and suggest directions for future work.

2 Preliminaries

In this section, we present the problem formulation, and provide the necessary background material related to sum-of-squares programming.

2.1 Problem Formulation

Consider a relatively high-fidelity model of a robotic system, referred to as the "tracking model," and a lower-fidelity model, referred to as the "planning model," described, respectively, by the ordinary differential equations (ODEs)

$$\dot{s} = f^s(s, u^s), \quad u^s \in \mathcal{U}^s, \qquad \dot{p} = f^p(p, u^p), \quad u^p \in \mathcal{U}^p, \tag{1}$$

where $s \in \mathbb{R}^{n_s}$ is the tracking model's state (or "tracking state"), and $u^s \in \mathbb{R}^{m_s}$ is its control input (or "tracking control"), constrained to lie within the compact set $\mathcal{U}^s$. Similarly, $p \in \mathbb{R}^{n_p}$ is the planning state and $u^p \in \mathbb{R}^{m_p}$ is the planning control input, constrained to lie within the compact set $\mathcal{U}^p$. The planning model can be thought of as a coarser or lower-fidelity model for which motion planning can be done effectively in real time. For both ODEs in equation (1), we assume that the dynamics are Lipschitz

continuous in the state for fixed controls, so that given any measurable control function, a unique trajectory exists for each system [10]. Furthermore, we assume that *the planning state p is a strict subset of the tracking state s*, in particular $n_p < n_s$ (this assumption allows one to clearly relate the tracking and planning state, as discussed in Section 3.1), and that the dynamics for the tracking model are *control affine* (a rather mild condition that significantly simplifies the SOS formulation, as discussed in Section 3.3).

Crucially, trajectories generated by using the planning model may not be dynamically feasible with respect to the tracking model. Thus, a tracking controller that accounts for the high-fidelity dynamics of the robotic system, as represented by the tracking model, may not be able to track these trajectories exactly and some (possibly large) tracking error may be incurred. Figure 1 illustrates this behavior, for the case of a 4D car model used to track a 2D single integrator model. Specifically, a planning algorithm is used to "quickly" find a nominal motion plan for the planning model, depicted by the dashed blue line. The robot, represented more accurately by the tracking model, strives to track the nominal motion plan as closely as possible, giving rise to the red trajectory. The presence of a minimum turning radius constraint in the tracking model prevents the robot from tracking the sharp corners exactly, thereby yielding a tracking error.

Such a decoupling approach is quite common in the field of motion planning [16], yet very few results exist on how to account for the resulting tracking error. In particular, we advocate that the computation of the nominal motion plan via the planning model should account for the tracking error in order to ensure safe execution (i.e., tracking) by the real system, described by the tracking model. Accordingly, the objective of this paper is to devise an efficient and scalable algorithm to compute (1) a *trajectory-independent* TEB (represented by the light blue circle in Fig. 1, left), which the planner can use as a "safety buffer" to ensure that the actual trajectory is collision free, and (2) a feedback tracking controller to track such a nominal motion plan while respecting the TEB. Intuitively, the tracking controller should have the property that on the boundary of the TEB, the error dynamics are driven "inwards," *regardless of the nominal motion plan* (depicted in Fig. 1, right).

2.2   Sum-of-Squares (SOS) Programming Background

Our approach is rooted in SOS programming, for which we provide a brief review here (for a more detailed review of SOS programming and its applications, please refer to [2, 21, 23]). We begin by discussing semidefinite programs (SDPs), a class of convex optimization problems formulated over the space of symmetric positive semidefinite (psd) matrices. Denote the set of $n \times n$ symmetric matrices as S^n. A symmetric matrix $X \in S^n$ is positive semidefinite (psd) if $x^\top X x \geq 0$ for all $x \neq 0$, and is denoted as $X \succeq 0$. An SDP in standard form is written as:

$$\begin{aligned} \underset{X \in S^n}{\text{minimize}} \quad & \text{Tr}(C^\top X) \\ \text{subject to} \quad & A_i^\top X = b_i, \quad i = 1, 2, ..., m, \\ & X \succeq 0 \end{aligned} \qquad (2)$$

where C and $\{A_i\}_i$ are elements of S^n, and "Tr" denotes the trace operator. SOS programs provide a means of certifying nonnegativity of polynomials either globally or over basic semialgebraic sets. A basic semialgebraic set $\mathcal{S}$ is a subset of the Euclidean space characterized by a finite set of polynomial inequalities and equalities, that is,

$$\mathcal{S} := \{x \in \mathbb{R}^n : \phi_i(x) \leq 0, \psi_j(x) = 0\}, \qquad (3)$$

where $\{\phi_i\}, \{\psi_j\}$ are multivariate polynomials in x. The simplest task within this class of problems is verifying the nonnegativity of a given polynomial $p(x)$ over $\mathcal{S}$, which

is already NP-hard [23]. SOS programming provides a convex relaxation approach to accomplish this task. Specifically, a polynomial p is SOS if it can be written in the form $\sum_k z_k^2$ for some other polynomials $z_k(x)$. While such a decomposition is not necessary, it is sufficient to guarantee (global) nonnegativity of $p(x)$. If one can find a set of SOS polynomials $L_i(x)$ and ordinary polynomials $q_j(x)$ such that

$$p + \sum_i L_i \phi_i + \sum_j q_j \psi_j \text{ is SOS,} \tag{4}$$

then one obtains a *certificate* of nonnegativity of $p(x)$ over $\mathcal{S}$. Indeed, in the above equation, when $\phi_i(x) \leq 0$ and $\psi_j(x) = 0$, one has that $p(x) \geq -\sum_i L_i(x)\phi_i(x) \geq 0$, as required. Such a certificate is the extension of the generalized S-procedure [12] to the setting of real-valued polynomials [23], and additionally constitutes a necessary condition for a subclass of semialgebraic sets [25].

The computational advantage of SOS programming stems from its intrinsic link to SDPs. Specifically, a polynomial p of degree $2d$ is SOS if and only if $p(x) = z(x)^T Q z(x)$, where $Q \succeq 0$ and $z(x)$ is a vector of monomials up to order d. Thus, certifying that a polynomial is SOS reduces to the task of finding a psd matrix Q subject to a finite set of linear equalities, thus taking the form in (2). Certificates of the form in (4) will form the building block for our approach.

3 SOS-Based Robust Tracking

3.1 Relative System

Given the tracking and planning models as in equation (1), we define the *relative system state* $r \in \mathbb{R}^{n_r}$ as $r := \phi(s,p)(s - Qp)$, where $\phi(\cdot,\cdot) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_r \times n_s}$, with $n_r \leq n_s$, is a matrix-valued map with bounded derivatives, and $Q = \left[\mathbf{I}_{n_p}, \mathbf{0}_{n_p \times (n_s - n_p)} \right]^\top$ is a projection matrix from $\mathbb{R}^{n_p}$ to $\mathbb{R}^{n_s}$ ($\mathbf{I}$ and $\mathbf{0}$ denote the identity and zero matrices, respectively). In the definition of Q, we leveraged the assumption that the planning state p is a strict subset of the tracking state s. We make the following assumption on the dynamics of the relative system state, which is central for our approach:

Assumption 1 (Relative System Dynamics) *The relative system dynamics can be written as a Lipschitz continuous function f^r of (r, u^s, u^p), that is, $\dot{r} = f^r(r, u^s, u^p)$.*

Lipschitz continuity is necessary for the existence and uniqueness of solutions to the ODEs. The structural property in Assumption 1 is satisfied for a number of dynamical models of mobile robotic systems. For example, for ubiquitous cases where the planning model has integrator dynamics, Assumption 1 is satisfied by simply selecting $\phi(\cdot,\cdot)$ as the identity map. However, in general, it is difficult to characterize the conditions on the planning/tracking models such that Assumption 1 is satisfied. Therefore, in the online version of this work [32], we provide a "catalog" of tracking and planning models fulfilling Assumption 1 for a representative set of robotic systems, along with the map ϕ that guarantees fulfillment of Assumption 1 and the resulting relative system dynamics. Specifically, the relative system states in the 5th column are obtained by combining the tracking state in the 2nd column, planning state in the 4th column, and the transformation in the 5th column via $r = \phi(s,p)(s-Qp)$. Henceforth, we will consider planning/tracking models that fulfill Assumption 1, and refer to the system $\dot{r} = f^r(r, u^s, u^p)$ as the *relative system*.

Finally, we define the *error state e* as the relative system state *excluding* the absolute states of the tracking model, and the *auxiliary state η* as the relative system state *excluding* the error state. Hence, $r = (e, \eta)$. For instance, for the case of a 5D car model used to track a 3D Dubins car model (second example in Table 1), $e = (x_r, y_r, \theta_r)$ and $\eta = (v, \omega)$. Roughly speaking, considering relative system dynamics allows one to reformulate the tracking problem as a stabilization problem to the origin.

3.2 Optimization Problem

In order to define a notion of *tracking error bound* (TEB), let $c(r) : \mathbb{R}^{n_r} \mapsto \mathbb{R}^{n_{\tilde{r}}}$ be a smooth mapping representing a vector of quantities we wish to remain bounded, with $n_{\tilde{r}} \geq n_r$; henceforth, we will refer to $c(r)$ as the *bounded state*. The simplest example for $c(\cdot)$ is the identity map. However, for more complex planning/tracking models, the mapping $c(\cdot)$ can be more sophisticated to better capture the structure of the planner/tracker dynamics pair – some examples are provided in Table 1. The goal then is to find a closed and bounded set $\mathcal{B}$ in the bounded state such that $c(r(t)) \in \mathcal{B}$ for all $t \geq 0$ (as an illustration, see the light blue ball in Figure 1). Such an invariance condition is captured by the implication

$$c(r(0)) \in \mathcal{B} \Rightarrow c(r(t)) \in \mathcal{B}, \quad \forall t \geq 0. \tag{5}$$

To parameterize this set, let ρ be a positive constant and $\tilde{V}(\tilde{r}) : \mathbb{R}^{n_{\tilde{r}}} \mapsto \mathbb{R}$ be a smooth function with a bounded ρ−sublevel set. Also, define the function $V(r) : \mathbb{R}^{n_r} \mapsto \mathbb{R}$ as simply the composition $\tilde{V}(c(\cdot))$. Suppose that the following implication is true:

$$V(r) = \rho \quad \Rightarrow \quad \dot{V} = \left.\frac{\partial \tilde{V}(\tilde{r})}{\partial \tilde{r}}\right|^T_{\tilde{r}=c(r)} \frac{\partial c(r)}{\partial r} f^r(r, u^s, u^p) < 0.$$

That is, there exists some tracking control u^s such that the time-derivative of V on the boundary $V(r) = \rho$ is negative. It follows that the set $\{r : V(r) \leq \rho\}$ is invariant. Then, the set $\{c(r) : \tilde{V}(c(r)) \leq \rho\}$ is a valid error bound $\mathcal{B}$ in the sense of equation (5). An illustration of this idea is provided in Fig. 1 (right).

Remark 1. Note that the set $\{c(r) : \tilde{V}(c(r)) \leq \rho\}$ is equivalent, up to a translation and scaling factor, to the set $\{c(r) : \alpha\left(\tilde{V}(c(r))) + \beta\right) \leq \alpha(\rho + \beta)\}$, where $\beta \in \mathbb{R}$ and $\alpha > 0$. To eliminate this redundancy, we impose, without loss of generality, the conditions $\tilde{V}(0) = 0$ and $\rho = 1$. The choice $\tilde{V}(0) = 0$ essentially corresponds to a translational adjustment that "centers" the set $\mathcal{B}$ on a zero bounded state.

In the context of this paper, set $\mathcal{B}$ is used as a "buffer" that a planner should consider along each candidate nominal motion plan, to ensure that the *realized* tracking trajectory is safe, that is, collision free. Clearly, the search for set $\mathcal{B}$ is intertwined with the search for a tracking controller that can keep the realized tracking trajectory within it. We parameterize the tracking controller u^s as a function of the relative system state and planning control signal, that is $u^s = K(r, u^p)$. Denote the closed-loop dynamics as $f^r_K(r, u^p) := f^r(r, K(r, u^p), u^p)$ and define

$$\dot{V}_K(r, u^p) := \left.\frac{\partial \tilde{V}(\tilde{r})}{\partial \tilde{r}}\right|^T_{\tilde{r}=c(r)} \frac{\partial c(r)}{\partial r} f^r(r, K(r, u^p), u^p).$$

The search for a suitable function V and controller $K(\cdot, \cdot)$ is then formalized by the optimization problem:

$$\underset{K(\cdot,\cdot), \mathcal{B}}{\text{minimize}} \quad \text{volume}(\mathcal{B}) \tag{6a}$$

$$\text{subject to} \quad \left.\begin{array}{l} V(r) = 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow \dot{V}_K(r, u^p) < 0 \tag{6b}$$

$$\left.\begin{array}{l} V(r) \leq 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow K(r, u^p) \in \mathcal{U}^s \tag{6c}$$

We will next encode the constraints and objective of problem (6) as SOS certificates.

3.3 Reformulating the Constraints as SOS Certificates

We consider the constraints first. Under the assumption that the dynamics for the tracking model are control affine (common for most robotic systems), the function f^r is control affine in u^s. That is, f^r takes the form $f^r(r, u^p, u^s) = h(r, u^p) + B(r)u^s$. Thus,

$$\dot{V} = \frac{\partial V(r)}{\partial r}^T h(r, u^p) + \frac{\partial V(r)}{\partial r}^T B(r)u^s.$$

Hence, the invariance condition requires the existence of a tracking controller $K(r, u^p)$ such that

$$\left.\begin{array}{r} V(r) = 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow \frac{\partial V(r)}{\partial r}^T h(r, u^p) + \frac{\partial V(r)}{\partial r}^T B(r)K(r, u^p) < 0. \qquad (7)$$

We can *equivalently* state inequality (7) as:

$$\left.\begin{array}{r} V(r) = 1 \\ u^p \in \mathcal{U}^p \end{array}\right\} \Rightarrow \frac{\partial V(r)}{\partial r}^T h(r, u^p) + \min_{K(r, u^p) \in \mathcal{U}^s} \left(B(r)^T \frac{\partial V(r)}{\partial r}\right)^T K(r, u^p) < 0. \qquad (8)$$

The equivalence between inequalities (7) and (8) follows from the observation that inequality (7) holds for *some* $u^s \in \mathcal{U}^s$ if and only if it holds for a $u^s \in \mathcal{U}^s$ minimizing the left hand side in inequality (8). Importantly, the minimization in inequality (8) is *independent* of u^p, as the constraint set $\mathcal{U}^s$ and objective coefficient vector $B^T \partial V/\partial r$ do not depend on u^p. Thus, we can simplify K by making it a function of r only (with a slight abuse of notation, we still refer to such a simplified tracking controller with K).

We can now define SOS certificates for the constraints in problem (6). Suppose set $\mathcal{U}^p$ can be written as the semialgebraic set $\{u^p \in \mathbb{R}^{m_p} : g_i^p(u^p) \leq 0, \ i = 1, \ldots, N_p\}$ and let the controller K be a *polynomial* function in r. Expanding $\dot{V}_K$, one can encode constraint (6b) using the multiplier polynomials $L^{\text{Lya}}(r, u^p), \{L_i^p(r, u^p)\}_{i=1}^{N_p}$ such that

$$-\dot{V}_K + L^{\text{Lya}} \cdot (V - 1) + \sum_{i=1}^{N_p} L_i^p \cdot g_i^p \text{ is SOS}, \qquad \{L_i^p\} \text{ are SOS}, \qquad i = 1, \ldots, N_p. \qquad (9)$$

(In the following we drop the summation notation and write the collective sum as a "dot-product:" $L^p \cdot g^p$). To capture the control constraint in (6c), one may leverage two different techniques. The first one is to compose the tracking controller K with a saturation function and write $\dot{V}$ in case form corresponding to the unsaturated and saturated regimes, respectively. However, this approach scales exponentially in the dimension of the control input, as one needs to capture all combinations of saturation regimes [21]. Instead, we assume that the tracking control set $\mathcal{U}^s$ is polytopic, i.e., $\mathcal{U}^s = \{u^s \in \mathbb{R}^{m_s} : g^s(u^s) \leq 0\}$, where $g^s(u^s) \leq 0$ is a set of linear inequalities in u^s of the form $g_i^s = a_{s_i}^T u^s - b_{s_i} \leq 0, \ i = 1, \ldots, N_s$. Note that this is not an overly restrictive assumption as the control constraints are often taken to be box constraints. Then, we encode constraint (6c) using the multipliers $\{L_i^s(r, u^p)\}$:

$$-g_i^s(K) + L_i^s \cdot (V - 1) \text{ is SOS}, \qquad L_i^s \text{ are SOS}, \qquad i = 1, \ldots, N_s. \qquad (10)$$

Crucially, this approach scales linearly with respect to the number of inputs.

3.4 Reformulating the Objective as SOS Certificates

Minimizing the volume of $\mathcal{B}$, which itself is a non-linear mapping of the $1-$sublevel set of V, is a difficult task if one reasons in terms of arbitrary functions V. One approach

is to approximate the volume by the integral of V over an Euclidean ball of radius R such that the ball is contained within the $1-$sublevel set of V – an approach taken, for example, in [24]. Alternatively, one can minimize the volume of an encapsulating ellipsoid, easily captured using an additional constraint and a convex pseudo-objective.

Toward this end, we define the ellipsoid $\mathcal{E} := \{r : c(r)^\top E c(r) \le 1\}$ for some positive definite matrix $E \ge \delta_E I$, where $\delta_E > 0$ is a small tolerance parameter. Then, the inclusion constraint $\mathcal{B} \subseteq \mathcal{E}$ is captured using the SOS multiplier $L^{\mathcal{E}}(r)$:

$$1 - c^\top E c + L^{\mathcal{E}}(V-1) \text{ is SOS}, \qquad L^{\mathcal{E}} \text{ is SOS}, \qquad E \ge \delta_E I. \qquad (11)$$

As the volume of $\mathcal{E}$ is inversely proportional to the square root of its determinant, the minimum volume objective reduces to maximizing the determinant of E, and can be written in the form of (2) [4, Chapter 4]. For non-identity mappings $c(\cdot)$, the first SOS constraint in equation (11) can quickly become computationally challenging due to the quadratic terms in $c(r)$ and the complexity of the relation $V(\cdot) = \tilde{V}(c(\cdot))$. In this case, one may consider the following simplification. Since

$$\mathcal{B} = \{c(r) \in \mathbb{R}^{n_{\tilde{r}}} : \tilde{V}(c(r)) \le 1\} \subseteq \{\tilde{r} \in \mathbb{R}^{n_{\tilde{r}}} : \tilde{V}(\tilde{r}) \le 1\}, \qquad (12)$$

the simplification is to outer bound the set on the right by using an ellipsoid. Specifically, let the ellipsoid $\mathcal{E}$ be given by $\mathcal{E} = \{\tilde{r} : \tilde{r}^\top E \tilde{r} \le 1\}$ for some positive definite matrix $E \ge \delta_E I$. The inclusion condition $\tilde{V}(\tilde{r}) \le 1 \Rightarrow \tilde{r}^\top E \tilde{r} \le 1$, is captured using the SOS multiplier $L^{\mathcal{E}}(\tilde{r})$:

$$1 - \tilde{r}^\top E \tilde{r} + L^{\mathcal{E}}(\tilde{V}-1) \text{ is SOS}, \qquad L^{\mathcal{E}} \text{ is SOS}, \qquad E \ge \delta_E I. \qquad (13)$$

Crucially, the constraints above are deliberately written with respect to the variable $\tilde{r}$, which is treated as an *independent indeterminate* from r. This is beneficial when using a mapping $c(\cdot)$ other than the trivial identity map, as it allows one to approximate the otherwise complex set $\mathcal{B}$ with potentially simpler functions in $\tilde{r}$.

3.5 SOS Formulation of Optimization Problem

Collecting all the results so far, our approach entails conservatively (as we rely on sufficient SOS certificates) solving the optimization problem (6) as a SOS program:

$$\underset{K,\tilde{V},E,\mathbf{L}}{\text{maximize}} \quad \log \det(E) \qquad (14a)$$

$$\text{subject to} \quad \text{eqs. (9), (10), (13)} \qquad (14b)$$

$$E > \delta_E I, \rho \ge 0 \qquad (14c)$$

$$\tilde{V}(0) = 0 \qquad (14d)$$

where $\mathbf{L} := \{L^{\text{Lya}}, L^p, L^s, L^{\mathcal{E}}\}$, and $\mathbf{1}$ is the vector of 1s. Reinforcing the ideas in Section 3.4, we iterate that $\tilde{V}$ is considered as the optimization variable in the independent indeterminate $\tilde{r}$, with constraints (9), (10) encoded using the indeterminate r.

Constraints in (14b) are bilinear in the decision variables. Consequently, similar to the bilinear solution algorithms in [21] and [24], one must alternate between the decision variable sets $\{E, K, \mathbf{L}\}$ and $\{\tilde{V}, E, L^p\}$, each time holding the other variable set fixed. Specifically, we refer to the sub-problem where $K(\cdot)$ is part of the decision variable set as the K sub-problem, and the sub-problem where $\tilde{V}$ is part of the decision variable set as the V sub-problem. Direct implementation of alternations is hindered by two challenges. First, one requires a feasible initial guess for $\tilde{V}$ to begin the alternations. In [21] and [24], the authors leverage locally linearized models and the solution to the Riccati equation to generate such a guess. In this paper we address a fundamentally more

challenging problem as we consider the controllability of a relative dynamical system between two different dynamical models. Consequently, the relative system is often not linearly controllable at $r = 0$ (even if the individual models are) and thus one requires an additional procedure to generate a feasible function $\tilde{V}$ for problem (14) from an initially infeasible guess. Second, alternating optimization is prone to numerical instabilities as the solutions of the individual convex problems frequently lie on boundaries of the respective feasible sets. We address these challenges next.

4 Solving the Bilinear Optimization Problem

The general idea to tackle both of the aforementioned challenges entails using iterative slack minimization. We first discuss the solutions to the K and V sub-problems (Sections 4.1 and 4.2, respectively) and then present the general solution algorithm in Section 4.3.

4.1 The K Sub-Problem

A naïve implementation of the K sub-problem would entail solving problem (14) with respect to $\{E, K, \mathbf{L}\}$ for a given $\tilde{V}$. However, constraint (9) as written may generate controllers that are numerically unstable when held fixed in the V sub-problem. Given the polytopic constraint set $\mathcal{U}^s$ and a fixed $\tilde{V}$, one can exactly characterize the "most stabilizing controller" as the function K that minimizes the left hand side in inequality (8) for all r such that $V(r) = 1$. Thus, we propose the following two-step method for solving the K sub-problem, *for a given function $\tilde{V}$*:

1. Find the "most stabilizing controller" $K(\cdot)$ as the solution to:

$$\underset{K, L^{\text{Lya}}, L^P, L^s, \gamma}{\text{minimize}} \quad \gamma \tag{15a}$$

$$\text{subject to} \quad -\dot{V}_K + \gamma + L^{\text{Lya}} \cdot (V - 1) + L^P \cdot g^P \quad \text{is SOS} \tag{15b}$$

$$-g^s(K) + L^s \cdot (V - 1) \text{ is SOS}, \qquad \{L_i^P\}, \{L_i^s\} \quad \text{are SOS}, \tag{15c}$$

where γ is a slack variable for the invariance constraint. Notice that when $V(r) = 1$ and $g^P(u^P) \leq 0$, then $\dot{V}_K(r, u^P) \leq \gamma$. Denote the optimal slack as γ_c^*.

2. Compute the tightest bounding ellipsoid $\mathcal{E}$:

$$\underset{L^{\mathcal{E}}, E}{\text{maximize}} \quad \log \det(E) \tag{16a}$$

$$\text{subject to} \quad 1 - \tilde{r}^\top E \tilde{r} + L^{\mathcal{E}}(\tilde{V} - 1) \quad \text{is SOS} \tag{16b}$$

$$E \succeq \delta_E I, \qquad L^{\mathcal{E}} \text{ is SOS.} \tag{16c}$$

The two steps above comprise the K sub-problem. The benefit of decomposing the K sub-problem in this fashion is that we independently search for the most stabilizing controller while simultaneously relaxing the invariance constraint by using the slack variable γ; in particular, γ accounts for the suboptimality of the computed controller with respect to the left hand side of inequality (9).

4.2 The V Sub-Problem

Given a controller $K(\cdot)$, and multiplier polynomials $\{L_i^s\}$, $L^{\mathcal{E}}$, and L^{Lya} from the K sub-problem, the V sub-problem is defined as

$$\underset{\tilde{V},E,L^p,\gamma,\epsilon}{\text{maximize}} \quad \lambda \log \det(E) - \gamma - \|\epsilon\|_1 \tag{17a}$$

$$\text{subject to} \quad -\dot{V}_K + \gamma + L^{\text{Lya}} \cdot (V-1) + L^p \cdot g^p \quad \text{is SOS} \tag{17b}$$

$$-g^s(K) + \epsilon + L^s \cdot (V-1) \text{ is SOS} \tag{17c}$$

$$1 - \tilde{r}^\top E \tilde{r} + L^{\mathcal{E}}(\tilde{V}-1) \quad \text{is SOS} \tag{17d}$$

$$E \succeq \delta_E I, \quad \tilde{V}(0) = 0, \tag{17e}$$

$$\{L_i^p\} \quad \text{are SOS} \tag{17f}$$

where $\epsilon \in \mathbb{R}_{\geq 0}^{N_s}$ is a slack vector for the control constraints and λ is a Pareto trade-off parameter. Notice that the control slack ϵ is only necessary in the V sub-problem since the controller is held fixed within this problem. In contrast, in problem (15), we directly optimize over the set of strictly *feasible* controllers. Given these slack-based relaxed sub-problems, we now provide a solution algorithm for problem (14).

4.3 Solution Algorithm

Before providing the full solution algorithm, we describe an initialization procedure that generates a numerically stable initial guess for $\tilde{V}$. The procedure is detailed in Algorithm 1. In particular, the algorithm may be initialized by using any polynomial guess for $\tilde{V}$ such that $\tilde{V}(0) = 0$ (e.g., $\tilde{r}^T \tilde{r}$). Notice that in line 6, when problem (17) is

---

Algorithm 1 Generating Feasible Guess

 1: **Input:** Initial guess $\tilde{V}$ satisfying $\tilde{V}(0) = 0$; slack tolerance $\delta \in [0,1)$; max # of iterations N.
 2: $i \leftarrow 0$.
 3: **while** $i < N$ **do**
 4: $\quad \{K, L^{\text{Lya}}, L^s, L^{\mathcal{E}}, E, \gamma_c^*\} \leftarrow$ Solve (15)–(16).
 5: $\quad$ **if** $\gamma_c^* \leq \delta$ **then return** $(\tilde{V}, E, K)$
 6: $\quad \{\tilde{V}, E, \gamma^*, \epsilon^*\} \leftarrow$ Solve (17) with $\lambda = 0$ and subject to additional constraint $\gamma \leq \gamma_c^*$.
 7: $\quad$ **if** $\max\{\gamma^*, \|\epsilon^*\|_\infty\} \leq \delta$ **then return** $(\tilde{V}, E, K)$
 8: $\quad i \leftarrow i + 1$.
 9: **end while**
10: **return** Failure.

---

solved, the Pareto parameter λ is set to 0 since the objective for the initialization algorithm is to simply generate a feasible initial guess for problem (14). Furthermore, we impose the additional constraint $\gamma \leq \gamma_c^*$ to ensure monotonic improvement in solution quality between sub-problems. As the original problem is still non-convex, the convergence of the slack variables below the tolerance threshold is not guaranteed for every initial guess of $\tilde{V}$. However, typical guesses such as $\tilde{r}^T Q \tilde{r}$ for a positive diagonal matrix Q appear to work quite well, as the experiment section will illustrate.

Given an initial guess generated by Algorithm 1, we are now ready to solve problem (14). To do so, we will make use of a slightly modified version of the V sub-problem, given below:

$$\underset{\tilde{V},\gamma,E,L^p}{\text{maximize}} \quad \lambda \log \det(E) - \gamma \tag{18a}$$

$$\text{subject to} \quad \text{eqs. (17b)} - \text{(17f)} \tag{18b}$$

$$\underline{\alpha} E^* \preceq E \preceq (1+\alpha)E^* \tag{18c}$$

where E^* is the previous numerically stable solution (i.e., with optimal slack values $\gamma^*, \|\epsilon^*\|_\infty \leq \delta$), $\underline{\alpha} \in (0,1)$ is a fixed parameter, and $\alpha \in (0,1)$ is a backtracking search parameter that is adjusted in an iterative fashion. Specifically, we iteratively solve problem (18), *each time* checking the slack tolerances to ensure numerical stability, while using constraint (18c) to enforce a *trust region* around the current numerically stable solution. The full solution algorithm is summarized in Algorithm 2.

Algorithm 2 Solving Problem (14)

1: **Input:** Output tuple $(\tilde{V}, E, K)$ from Algorithm 1; slack tolerance δ; termination tolerance $\theta_1 \in (0,1)$.
2: $i \leftarrow 1$, $\mathtt{converged} \leftarrow \mathtt{false}$.
3: **Initialize**: $(\tilde{V}^*, E^*, K^*, c_0^*) \leftarrow (\tilde{V}, E, K, \log\det(E))$.
4: **while** $i < N \wedge \neg\, \mathtt{converged}$ **do**
5:     $\{K, L^{\mathrm{Lya}}, L^s, L^{\mathcal{E}}, E\} \leftarrow$ SOLVE (15)– (16) subject to additional constraint: $\gamma \leq 0$.
6:     **if** Line 5 successfully solved **then**
7:         $(E^*, K^*) \leftarrow (E, K)$.
8:         $(\tilde{V}^*, E^*, c_i^*) \leftarrow \mathtt{Backtrack}\big(\{K^*, L^{\mathrm{Lya}}, L^s, L^{\mathcal{E}}\}, E^*, \tilde{V}^*\big)$.
9:         **if** $|c_i^* - c_{i-1}^*| \leq \theta_1 |c_{i-1}^*|$ **then** $\mathtt{converged} \leftarrow \mathtt{true}$. **else** $i \leftarrow i+1$.
10:     **else**
11:         $\mathtt{converged} \leftarrow \mathtt{true}$.
12:     **end if**
13: **end while**
14: **return** $(\tilde{V}^*, \rho^*, E^*, K^*)$

Algorithm 3 Backtrack

1: **Input:** Solution $\{K^*, L^{\mathrm{Lya}}, L^s, L^{\mathcal{E}}\}$ and E^* from line 5 in Algorithm 2 and current best solution $\tilde{V}^*$; slack tolerance δ; backtrack parameters $\overline{\alpha}$, β, $\theta_2 \in (0,1)$.
2: **Initialize**: $\alpha \leftarrow \overline{\alpha}$, $\mathtt{bt} \leftarrow \mathtt{false}$, $c^* \leftarrow \log\det(E^*)$.
3: **while** $\neg\mathtt{bt}$ **do**
4:     $\{\tilde{V}, E, \gamma^*, \epsilon^*\} \leftarrow$ SOLVE (18).
5:     **if** $\max\{\gamma^*, \|\epsilon^*\|_\infty\} \leq \delta$ **then** $(\tilde{V}^*, E^*, c^*) \leftarrow (\tilde{V}, E, \log\det(E))$. **else** $\mathtt{bt} \leftarrow \mathtt{true}$.
6: **end while**
7: **while** $\mathtt{bt} \wedge \alpha > \theta_2\overline{\alpha}$ **do**
8:     $\alpha \leftarrow \beta\alpha$.
9:     $\{\tilde{V}, E, \gamma^*, \epsilon^*\} \leftarrow$ SOLVE (18).
10:     **if** $\max\{\gamma^*, \|\epsilon^*\|_\infty\} \leq \delta$ **then** $(\tilde{V}^*, E^*, c^*) \leftarrow (\tilde{V}, E, \log\det(E))$. **else** $\mathtt{bt} \leftarrow \mathtt{false}$.
11: **end while**
12: **return** $(\tilde{V}^*, E^*, c^*)$

The backtrack search procedure in line 8 is summarized in Algorithm 3. Within this procedure, we first iteratively maximize $\log\det(E)$ within the trust region (18c) centered on the previous stable numerical solution, using the slack values as a check on solution quality (lines 3—6). Once solution quality degrades, we backtrack (shrink) the trust region in lines 7—11 until we again fall below the slack tolerances. Note that Algorithm 3 will either return an updated $\{\tilde{V}, E\}$ that is numerically stable (with respect to slack tolerances), or it will simply return the numerically stable solution from line 5 in Algorithm 2 if unable to make progress. Thus, Algorithm 2 terminates if either (1) improvement in $\log\det(E)$ stalls, or (2) the function $\tilde{V}^*$ returned by the backtrack procedure is not *strictly* feasible (i.e., $\gamma \leq 0$) with respect to line 5 in Algorithm 2, but is still acceptable according to the slack tolerances.

The key difference between the alternating method described here and a similar procedure in [24] is that the authors in [24] minimize the slack variable in both sub-

problems and use binary search to iteratively refine an upper bound on the cost function within the V sub-problem. On the other hand, our algorithm maintains numerical stability by using the slack variables *solely as a check* on the allowable change in the solution, while taking advantage of minimizing the true objective (i.e., $\log\det(E)$) within *both* sub-problems. This is especially useful in situations where the second phase of the algorithm struggles to make notable improvements on the objective (e.g., due to a smaller set of decision variables), thereby allowing the K sub-problem to take over.

5 Numerical examples

In this section we numerically validate our proposed approach. Specifically, in Section 5.1, we compare our approach with the HJ-based approach in [11], while in Section 5.2, we study a high-dimensional system which is beyond the reach of HJ analysis. Additional numerical examples are provided in [32]. The code is available at https://github.com/StanfordASL/Model-Mismatch.

5.1 Comparison with the HJ Method

In this section, we compare our method with the HJ-based approach in [11] for the case of a 5D car model used to track a 3D Dubins car model (that is, the second example in Table 1 in [32]). This example was chosen because five dimensions is the current limit for standard grid-based HJ reachability methods when techniques such as decomposition cannot be applied. Specifically, the system dynamics and bounded state definition are provided in the second row of Table 1. The model parameters are chosen as $|a| \leq 1$ m/s^2, $|\alpha| \leq 3$ rad/s^2, $\hat{v} = 1.0$ m/s, $|\hat{\omega}| \leq 0.1$ rad/s. For the SOS method, we parameterize K and $\tilde{V}$ as 2nd order polynomials in r and $\tilde{r}$, respectively. The trigonometric terms $\cos\theta_r$ and $\sin\theta_r$ are approximated with Chebyshev polynomials up to degree 2, over the range $\theta_r \in [-\pi/6, \pi/6]$ rad. To ensure the validity of these approximations, an additional constraint is appended to problems (16) and (17), namely:

$$-g + L_g(\tilde{V} - 1) \text{ is SOS,} \qquad\qquad L_g \text{ is SOS,} \qquad\qquad (19)$$

where $g = \theta_r^2 - (\pi/6)^2$ and L_g is a SOS polynomial in $\tilde{r}$. The initial guess for $\tilde{V}$ is simply $\tilde{r}^T Q \tilde{r}$, where Q is a diagonal matrix with randomly sampled positive entries. In order to ensure a fair comparison, the cost function in the min-max game for the HJ method is $x_r^2 + y_r^2 + \theta_r^2 + (v\cos\theta_r - \hat{v})^2 + v^2\sin^2\theta_r + \omega^2$, which is the closest analogue to the SOS objective of minimizing the entire volume of $\mathcal{B}$ and not simply its projection onto the position coordinates. Figure 2 plots the projections of the boundary of set $\mathcal{B}$ onto the (x_r, y_r, θ_r) components (i.e., the dimensions most relevant for collision checking) for the HJ and SOS solutions, respectively. The right-most panel in this figure provides a top-down view onto the (x_r, y_r) plane. As expected, the HJ solution provides a better position error bound, specifically, 42% of the SOS position error bound, or about 0.2m in absolute terms. The main reason behind this is that by using a grid-based approach to solve the min-max HJ game, one *exactly* computes the *non-smooth* "optimal controller" in (8), whereas the SOS approach attempts to find the best polynomial approximation. On the other hand, the computation time difference for the two solutions is *substantial* – approximately 25 hours for the HJ solution versus 5 minutes for the SOS solution. Specifically, the HJ solution was obtained by solving a corresponding HJ PDE [11] using a C++ implementation of the Lax Friedrichs numerical scheme [35]. The SOS programs were solved using the Spotless polynomial optimization toolbox [37] and MOSEK SDP solver [3]. Computations were done on a desktop computer with an Intel Core i7-2600K CPU and 16 GB of RAM. At least for this example, the SOS approach appears to provide a high-quality approximation to the optimal (HJ-based) solution, in a fraction of the time required by the HJ-based approach.
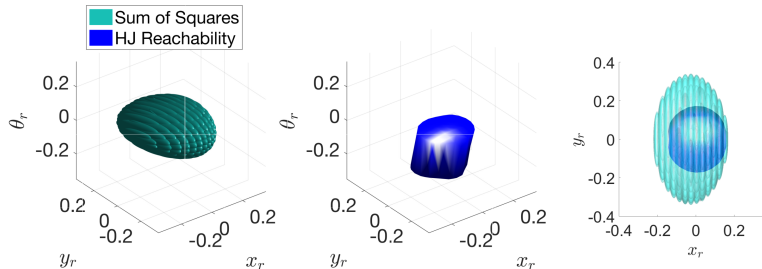
Fig. 2: Projection of the boundary of $\mathcal{B}$ onto (x_r, y_r, θ_r). *Left*: SOS. *Middle*: HJ. *Right*: Top-down view (i.e., onto (x_r, y_r) for both solutions). The HJ positional error bound is smaller (42% of SOS bound), but the SOS solution requires only 0.3% of the computation time required by HJ analysis.

In [32] we provide an additional comparison between the HJ and SOS methods for a 4D dynamically extended Dubins car tracking a 2D single integrator planning model (row 1 in Table 1). This example highlights a case in which the HJ method is efficient enough to warrant its use instead of the SOS method (i.e. when the dynamics are less than five dimensions and/or can be decomposed into subsystems that are each less than five dimensions). In this example, an optimal bound of 0.1m between the two models was computed using HJ while SOS yielded a bound of 0.3m. The computation time comparison was on the order of 2 minutes for SOS vs. 5 minutes for HJ. The projection of the optimal $\mathcal{B}$ onto the translational plane from the HJ method as shown in the appendix in [32] clearly illustrates the limitation of SOS in its ability to approximate non-smooth quantities. Indeed, herein lies the primary weakness of the SOS method – tight approximations require high degree polynomials which can rapidly increase the size of the SOS problems. Despite this limitation inherent in the SOS method, the approach allows us to establish a conservative over-approximation of the tracking error bound (when one exists) for a given planner/tracker pair and gauge the sensitivity of the expected tracking error to changing model parameters such as inertial properties and control authorities. Furthermore, as the next section illustrates, the SOS method allows us to perform this analysis for higher dimensional systems that are severely challenging (albeit, not impossible) for HJ. As such, the HJ and SOS methods should really be viewed as complementary to one another.

5.2 High-Dimensional Example
In this section, we present numerical results for a high-dimensional system for which the HJ approach is intractable. Specifically, we consider the case where an 8D plane model is used to track a 4D decoupled Dubins plane model (that is, the fourth example in Table 1 in [32]). To ensure that this example cannot be solved using standard grid-based HJ reachability, we must first determine that decomposition techniques would not be applicable. The 8D relative system dynamics are highly coupled, which means that approximate decomposition techniques would lead to results that are overly conservative [8]. The dynamics also do not form self-contained subsystems, making exact decomposition impossible [7]. This example well illustrates the usefulness of our approach. The planning model (i.e., decoupled Dubins plane) is a benchmark planning model with well characterized optimal trajectories [9]. The full 8D dynamics of the plane, however, are considerably more difficult to plan trajectories for online. Specifically, the system dynamics and bounded state definition are provided in the fourth row of Table 1. The planner model parameters are as follows. The constant speed $\hat{v}$ is set to be

the nominal speed for the plane in straight level flight conditions (lift equals gravitational force) at a nominal angle of attack $\alpha_0 = 5^o$, which corresponds to $\hat{v} = 6.5$m/s (see [30] for the relevant constants used to compute this quantity). The maximum magnitude of the turning rate for the planner, $|\hat{\omega}|$, is set to be 20% of the plane's maximum turning rate in a horizontal coordinated turn at nominal speed $\hat{v}$, and is computed to be 0.21 rad/s (equivalently, a minimum turning radius of 30m for the horizontal Dubins model). Finally, the planner's vertical velocity is limited to the range $[-0.1, 0.1]\hat{v}$.

Taking advantage of the structure of the dynamics, the normalized acceleration control u_a is chosen to exactly cancel drag, plus an additional 2nd degree polynomial in r as determined by the SOS program. The rest of the controller components and $\tilde{V}$ are also parameterized as 2nd order polynomials in r and $\tilde{r}$, respectively. The plane's (tracking) control limits are chosen to be $u_a \in [-8, 8]$ m/s^2 (≈ 10 times the acceleration needed to cancel drag at level trim conditions), $u_{\dot{\phi}} \in [-120, 120]^o$/s, and $u_{\dot{\alpha}} \in [-60, 60]^o$/s. To enforce the validity of the Chebyshev approximation for the trigonometric and $1/v$ terms in the dynamics, we enforce the additional constraints $\phi \in [-\pi/4, \pi/4]$ rad, $\gamma \in [-\pi/6, \pi/6]$ rad, $\psi_r \in [-\pi/6, \pi/6]$ rad, and $v \in [3, 10]$ m/s. The overall (initialization plus main optimization) computation time was under 2 hours. The projection of the set $\mathcal{B}$ onto (x_r, y_r, z_r) (the position errors *in the frame of the Dubins car*) had span $[-5.6, 5.6] \times [-3.8, 3.8] \times [-4.5, 4.5]$m. The bound is naturally more loose in the (x_r, z_r) dimensions since the planning model's horizontal velocity is equal to the plane's trim conditions at level flight, limiting the plane's ability to use its remaining velocity range for both tracking and ascending/descending. Thus, for such a choice of planning and tracking models, the bound appears reasonable. To obtain tighter bounds, one could for instance use as planning model the coupled kinematic plane model proposed in [22].

For an application of this bound to online motion planning, we set up a cluttered obstacle environment with trees and buildings (see Figure 3a). The goal region is the blue shaded box in the corner, and the plane starts at position $(1, 1, 5)$. The nominal motion plan is computed by using kinodynamic FMT* [29] (using the locally optimal trajectories in [9] as steering connections). Having computed all steering connections offline (< 2 min computation time for 5000 samples), the online computation time for the trajectory is on the order of 100ms. Collision checking was performed using obstacles inflated by the size of the plane (roughly a $2 \times 2 \times 0.5$m box envelope) and the projection of the ellipsoidal bound $\mathcal{B}$ onto (x_r, y_r, z_r) (rotated and translated by the orientation and position of the Dubins plane). In Figure 3a, the nominal motion plan (i.e., the plan for the decoupled Dubins model) is shown in red, while the actual trajectory followed by the plane is shown in black. We overlay the projection of set $\mathcal{B}$ onto position space, and its sweep along the nominal Dubins airplane motion plan. The snapshot confirms that this "tube" remains obstacle free, and highlights the necessity of using a $\mathcal{B}$-based buffer as the plane negotiates difficult turns (see Figure 3b). A video of this simulation can be found at https://www.youtube.com/watch?v=UsfaWsjectQ. Additional plots and simulations are provided in [32].

6   Conclusions

In this work we have presented a principled and scalable approach for planning with model mismatch. Specifically, by harnessing the tool of SOS programming, we designed an algorithmic framework to compute, offline, for a given pair of planning and tracking models, a feedback tracking controller and associated tracking bound; this bound can then used by the planning algorithm as a safety-margin when generating motion plans. The efficacy of the approach was verified via illustrative examples, which include a comparison with a state-of-the-art method based on reachability analysis.
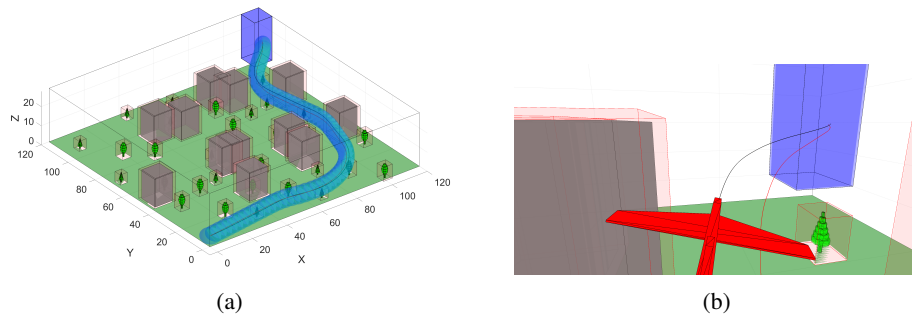
(a) (b)

Fig. 3: Left: snapshot of the nominal Dubins motion plan (red), along with the bounding ellipsoidal tube. The actual trajectory (black) stays within the ellipsoidal tube at all times. Right: Close-up of a tight turn at the end towards the goal.

There are several key avenues for future work: First, we would like to investigate structural conditions that guarantee fulfillment of Assumption 1, along with augmenting the catalog of planning/tracking models in Table 1. Second, we plan to investigate how the conservativeness of our approach may be reduced by using more complex certificates such as the Stengle Positivstellensatz. Third, to fully explore the scalability advantages of our approach, we will consider more tractable relaxations of SOS programming, such as SDSOS and DSOS optimization [1]. Fourth, it is of interest to study how to minimize the *projected* volume of the TEB onto the error states of interest. Finally, we plan on implementing our approach on agile robotic systems such as unmanned aerial vehicles.

References

1. Ahmadi, A.A., Majumdar, A.: DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization. In: IEEE Annual Conf. on Information Sciences and Systems (2014)
2. Ahmadi, A.A., Majumdar, A.: Some applications of polynomial optimization in operations research and real-time decision making. Optimization Letters 20(4), 709–729 (2016)
3. ApS, M.: MOSEK optimization software (2017), Available at https://mosek.com/
4. Ben-Tal, A., Nemirovski, A.: Lectures on modern convex optimization: analysis, algorithms, and engineering applications. SIAM (2001)
5. Burridge, R.R., Rizzi, A.A., Koditschek, D.E.: Sequential composition of dynamically dexterous robot behaviors. Int. Journal of Robotics Research 18(6), 534–555 (1999)
6. Chen, M., Bansal, S., Fisac, J.F., Tomlin, C.J.: Robust sequential path planning under disturbances and adversarial intruder. IEEE Transactions on Control Systems Technology (2018), in press
7. Chen, M., Herbert, S., Tomlin, C.J.: Fast reachable set approximations via state decoupling disturbances. In: Proc. IEEE Conf. on Decision and Control (2016)
8. Chen, M., Herbert, S.L., Vashishtha, M.S., Bansal, S., Tomlin, C.J.: Decomposition of reachable sets and tubes for a class of nonlinear systems. IEEE Transactions on Automatic Control (2018), in press
9. Chitsaz, H., LaValle, S.M.: Time-optimal paths for a dubins airplane. In: Proc. IEEE Conf. on Decision and Control (2007)
10. Coddington, E.A., Levinson, N.: Theory of Ordinary Differential Equations. McGraw-Hill (1955)
11. Herbert, S.L., Chen, M., Han, S., Bansal, S., Fisac, J.F., Tomlin, C.J.: FaSTrack: a modular framework for fast and guaranteed safe motion planning. In: Proc. IEEE Conf. on Decision and Control (2017)
12. Iwasaki, T., Hara, S.: Generalized KYP lemma: unified frequency domain inequalities with design applications. IEEE Transactions on Automatic Control 50(1), 41–59 (2005)

13. Janson, L., Schmerling, E., Clark, A., Pavone, M.: Fast Marching Tree: a fast marching sampling-based method for optimal motion planning in many dimensions. Int. Journal of Robotics Research 34(7), 883–921 (2015)
14. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. Journal of Robotics Research 30(7), 846–894 (2011)
15. Kousik, S., Vaskov, S., Johnson-Roberson, M., Vasudevan, R.: Safe trajectory synthesis for autonomous driving in unforeseen environments. In: Proc. ASME Dynamic Systems and Control Conf. (2017)
16. LaValle, S.M.: Motion planning: Wild frontiers. IEEE Robotics and Automation Magazine 18(2), 108–118 (2011)
17. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. Journal of Robotics Research 20(5), 378–400 (2001)
18. Li, Y., Littlefield, Z., Bekris, K.E.: Asymptotically optimal sampling-based kinodynamic planning. Int. Journal of Robotics Research 35(5), 528âĂŞ564 (2016)
19. Majumdar, A., Ahmadi, A.A., Tedrake, R.: Control design along trajectories with sums of squares programming. In: Proc. IEEE Conf. on Robotics and Automation (2013)
20. Majumdar, A., Tedrake, R.: Robust online motion planning with regions of finite time invariance. In: Algorithmic Foundations of Robotics X. Springer (2013)
21. Majumdar, A., Tedrake, R.: Funnel libraries for real-time robust feedback motion planning. Int. Journal of Robotics Research 36(8), 947–982 (2017)
22. Owen, M., Beard, R.W., McLain, T.W.: Implementing dubins airplane paths on fixed-wing UAVs. In: Handbook of Unmanned Aerial Vehicles. Springer Dordrecht (2015)
23. Parrilo, P.A.: Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization. Ph.D. thesis, Massachusetts Inst. of Technology (2000)
24. Posa, M., Koolen, T., Tedrake, R.: Balancing and step recovery capturability via sums-of-squares optimization. In: Robotics: Science and Systems (2017)
25. Putinar, M.: Positive polynomials on compact semi-algebraic sets. Indiana Univ. Mathematics Journal 42(3), 969–984 (1993)
26. Rajamani, R.: Vehicle Dynamics and Control. Springer, Boston, MA, 2 edn. (2012)
27. Ratliff, N., Zucker, M., Bagnell, J.A., Srinivasa, S.: CHOMP: gradient optimization techniques for efficient motion planning. In: Proc. IEEE Conf. on Robotics and Automation (2009)
28. Schmerling, E., Janson, L., Pavone, M.: Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics. In: Proc. IEEE Conf. on Decision and Control (2015)
29. Schmerling, E., Janson, L., Pavone, M.: Optimal sampling-based motion planning under differential constraints: the driftless case. In: Proc. IEEE Conf. on Robotics and Automation (2015), Extended version available at http://arxiv.org/abs/1403.2483/
30. Schmerling, E., Pavone, M.: Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning. In: Robotics: Science and Systems (2017)
31. Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., Abbeel, P.: Finding locally optimal, collision-free trajectories with sequential convex optimization. In: Robotics: Science and Systems (2013)
32. Singh, S., Chen, M., Herbert, S.L., Tomlin, C.J., Pavone, M.: Robust tracking with model mismatch for fast and safe planning: an SOS optimization approach. In: Workshop on Algorithmic Foundations of Robotics (2018), In Press, Available at: https://arxiv.org/abs/1808.00649
33. Singh, S., Majumdar, A., Slotine, J.J.E., Pavone, M.: Robust online motion planning via contraction theory and convex optimization. In: Proc. IEEE Conf. on Robotics and Automation (2017), Extended Version, Available at http://asl.stanford.edu/wp-content/papercite-data/pdf/Singh.Majumdar.Slotine.Pavone.ICRA17.pdf
34. Steinhardt, J., Tedrake, R.: Finite-time regional verification of stochastic non-linear systems. Int. Journal of Robotics Research 31(7), 901–923 (2012)
35. Tanabe, K., Chen, M.: beacls Library, Available at https://github.com/HJReachability/beacls
36. Tedrake, R., Manchester, I.R., Tobenkin, M., Roberts, J.W.: LQR-trees: Feedback motion planning via sums-of-squares verification. Int. Journal of Robotics Research 29(8), 1038–1052 (2010)
37. Tobenkin, M.M., Permenter, F., Megretski, A.: SPOTLESS polynomial and conic optimization (2013), software available from https://github.com/spottoolbox/spotless