Robust Online Motion Planning with Regions of Finite Time Invariance

Citation: Majumdar, Anirudha and Russ Tedrake. "Robust Online Motion Planning with Regions of Finite Time Invariance." Algorithmic Foundations of Robotics X, edited by E. Frazzoli et al, Springer, 2013: 543–558.

As Published: http://dx.doi.org/10.1007/978-3-642-36279-8_33

Publisher: Springer Berlin Heidelberg

Persistent URL: https://hdl.handle.net/1721.1/124358

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Robust Online Motion Planning with Regions of Finite Time Invariance

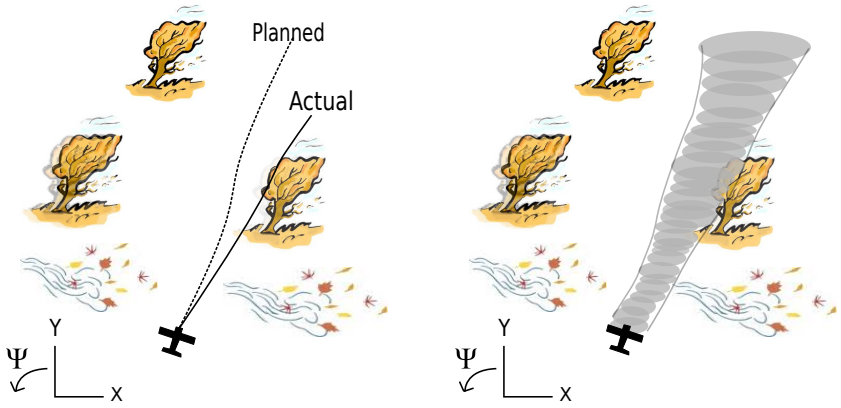
Anirudha Majumdar and Russ Tedrake

Abstract In this paper we consider the problem of generating motion plans for a nonlinear dynamical system that are guaranteed to succeed despite uncertainty in the environment, parametric model uncertainty, disturbances, and/or errors in state estimation. Furthermore, we consider the case where these plans must be generated online, because constraints such as obstacles in the environment may not be known until they are perceived (with a noisy sensor) at runtime. Previous work on feedback motion planning for nonlinear systems was limited to offline planning due to the computational cost of safety verification. Here we take a *trajectory library* approach by designing controllers that stabilize the nominal trajectories in the library and pre-computing regions of finite time invariance ("funnels") for the resulting closed loop system. We leverage sums-of-squares programming in order to efficiently compute funnels which take into account bounded disturbances and uncertainty. The resulting *funnel library* is then used to *sequentially compose* motion plans at runtime while ensuring the safety of the robot. A major advantage of the work presented here is that by explicitly taking into account the effect of uncertainty, the robot can evaluate motion plans based on how vulnerable they are to disturbances. We demonstrate our method on a simulation of a plane flying through a two dimensional forest of polygonal trees with parametric uncertainty and disturbances in the form of a bounded "cross-wind".

1 Introduction

The ability to plan and execute dynamic motions under uncertainty is a critical skill with which we must endow our robots in order for them to perform useful tasks in the real world. Whether it is an unmanned aerial vehicle (UAV) flying at high

Anirudha Majumdar and Russ Tedrake
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge MA - 02139, e-mail: {anirudha@mit.edu, russt@mit.edu}



(a) A plane deviating from its nominal planned trajectory due to a heavy cross-wind.

(b) The "funnel" of possible trajectories.

Fig. 1 Not accounting for uncertainty while planning motions can lead to disastrous consequences.

speeds through a cluttered environment in the presence of wind gusts, a legged robot traversing rough slippery terrain, or a micro-air vehicle with noisy on-board sensing, the inability to take into account disturbances, model uncertainty and state uncertainty can have disastrous consequences.

Motion planning has been the subject of significant research in the last few decades and has enjoyed a large degree of success in recent years. Planning algorithms like the Rapidly-exploring Randomized Tree (RRT) [12], RRT* [11], and related trajectory library approaches [13] [5] can handle large state space dimensions and complex differential constraints, and have been successfully demonstrated on a wide variety of hardware platforms [22] [21]. However, a significant failing is their inability to explicitly reason about uncertainty and feedback. Modeling errors, state uncertainty and disturbances can lead to failure if the system deviates from the planned nominal trajectories. A cartoon of the issue is sketched in Figure 1(a), where a UAV attempting to fly through a forest with a heavy cross-wind gets blown off its planned nominal trajectory and crashes into a tree.

More recently, planning algorithms which explicitly take into account feedback control have been proposed. LQR-Trees [24] and the minimum snap trajectory generation approach [16] operate by generating locally optimal trajectories through state space and stabilizing them locally. However, the former approach lacks the ability to handle scenarios in which the task and environment are unknown till runtime since it is too computationally intensive for online implementation. The latter technique has no mechanism for reasoning about uncertainty in the form of unmodeled disturbances, state errors and model uncertainty.

In this paper, we present a partial solution to these issues by combining trajectory libraries, feedback control, and sums-of-squares programming [18] in order to perform robust motion planning in the face of uncertainty. In particular, in the offline computation stage, we design a finite library of motion primitives and augment them with feedback controllers that locally stabilize them. Then, using sums-of-squares programming, we compute robust regions of finite time invariance ("funnels") around these trajectories that guarantee stability of the closed loop system in the presence of sensor noise, parametric model uncertainty, unmodeled bounded disturbances and changes in initial conditions. A cartoon of these regions of finite time invariance, or "funnels", is shown in Figure 1(b). Finally, we provide a way of composing these robust motion plans online in order to operate in a provably safe manner.

One of the most important advantages that our approach affords us is the ability to choose between the motion primitives in our library in a way that takes into account the dynamic effects of uncertainty. Imagine a UAV flying through a forest that has to choose between two motion primitives: a highly dynamic roll maneuver that avoids the trees in front of the UAV by a large margin or a maneuver involves flying straight while avoiding the trees only by a small distance. A traditional approach that neglects the effects of disturbances and uncertainty may prefer the former maneuver since it avoid the trees by a large margin and is therefore "safer". However, a more careful consideration of the two maneuvers leads to a different conclusion: the dynamic roll maneuver is far more susceptible to wind gusts and state uncertainty than the second one. Thus, it may be much more advantageous to execute the second motion primitive. Further, it may be possible that neither maneuver is guaranteed to succeed and it is safer to abort the mission and simply transition to a hover mode. Our approach allows robots to make these critical decisions, which are essential if robots are to move out of labs and operate in real-world environments.

2 Related Work

The motion planning aspect of our approach draws inspiration from the vast body of work that is focused on computing motion primitives in the form of trajectory libraries. For example, trajectory libraries have been used in diverse applications such as humanoid balance control [13], autonomous ground vehicle navigation [21] and grasping [4]. The Maneuver Automaton [5] attempts to capture the formal properties of trajectory libraries as a hybrid automaton, thus providing a nice unifying theoretical framework. Further theoretical investigations have focused on the offline generation of diverse but sparse trajectories that ensure the robot's ability to perform the necessary tasks online in an efficient manner [6]. More recently, tools from sub-modular sequence optimization have been leveraged in the optimization of the sequence and content of trajectories evaluated online [4].

Robust motion planning has also been a very active area of research in the robotics community. Early work focused on the purely kinematic problem of plan-

ning paths through configuration space with "tubes" of specified radii around them such that all paths in the tube remained collision-free [9] [8]. Recent work has focused on reasoning more explicitly about the manner in which uncertainty/disturbances influence the dynamics of the robot, and is closer in spirit to the work presented here. In particular, [20] approaches the problem through dynamic programming on a model with disturbances by making use of the Maneuver Automaton framework mentioned earlier. However, the work does not take into account obstacles in the environment and does not provide or make use of any explicit guarantees on allowed deviations from the planned trajectories in the Maneuver Automaton. The authors of [27] attempt to address the robust motion planning problem through H^∞ control techniques. However, the results focused on a planar vehicle and the basic motion planning framework does not easily extend to three dimensional environments. Also, while regions of invariance are approximated analytically and numerically, they are used only to evaluate the performance of the controller and not while constructing motion plans. Another approach that is closely related to ours is Model Predictive Control with Tubes [15]. The idea is to solve the optimal control problem online with guaranteed "tubes" that the trajectories stay in. However, the method is limited to linear systems and convex constraints.

A critical component of the work presented here is the computation of regions of invariance for nonlinear systems via Lyapunov functions. This idea, along with the metaphor of a "funnel", was introduced to the robotics community in [2], where funnels were *sequentially composed* in order to produce dynamic behaviors in a robot. In recent years, sums-of-squares programming has emerged as a way of checking the Lyapunov function conditions [18]. The technique relies on the ability to check positivity of multivariate polynomials by expressing them as a sum-of-squares. This can be written as a semi-definite optimization program and is amenable to efficient computational algorithms such as interior point methods [17]. Assuming polynomial dynamics of our system, one can check that a polynomial Lyapunov candidate, $V(x)$, satisfies $V(x) > 0$ and $\dot{V}(x) < 0$ in some region B_r. Importantly, the same idea can be used in computing regions of finite time invariance ("funnels") around time-indexed trajectories of the system [25]. Further, robust regions of attraction can be computed using an approach that verifies stability/invariance of sets around fixed points under parametric uncertainty [26]. In this paper, we show how to combine these ideas to compute robust regions of finite-time invariance around *trajectories* that guarantee that the if the system starts off in the set of given initial conditions, it will remain in the computed "funnel" even if the model of the dynamics is uncertain and the system is subjected to bounded disturbances and state uncertainty.

3 Contributions

This paper makes two main contributions. First, we provide a way of generating regions of finite time invariance ("funnels") for time-varying polynomial systems subjected to a general class of uncertainty (bounded uncertainty in parameters en-

tering polynomially in the dynamics). This is an extension of the results presented in [25], which presents a method for computing "funnels" for systems with time-varying polynomial dynamics assuming no model uncertainty/disturbances. Second, we show how a library of such funnels can be precomputed offline and composed together at runtime in a receding horizon manner while ensuring that the resulting closed loop system is "safe" (i.e. avoids obstacles and switches between the planned sequence of funnels). This can be viewed as an extension of the LQR-Trees algorithm [24] for feedback motion planning, which was limited to offline planning due to the relatively large computational cost of computing the funnels; our algorithm is suitable for real-time, online planning. We expect this framework to be useful in robotic tasks where the dynamics and perceptual system of the robot are difficult to model perfectly and for which the robot does not have access to the geometry of the environment till runtime.

4 Approach

A considerable amount of research effort in the motion planning community has focused on the design of trajectory libraries (Section 2). Further, a substantial but largely separate literature in control theory [19] [7] exists for the design of controllers that stabilize uncertain non-linear systems along nominal trajectories. Hence, in this section, we do not focus on the particulars of the generation of trajectory libraries and controllers. Rather, we concentrate on how these powerful methods from the fields of motion planning and control theory can be combined with sums-of-squares programming in order to perform robust online motion planning in the face of dynamic and state uncertainty. To this end, we assume that we are provided with a trajectory library consisting of a finite set of nominal feasible trajectories for the robot and a corresponding set of controllers that stabilize these trajectories. We discuss one particular method for obtaining trajectory libraries, controllers and Lyapunov functions in Section 5.

4.1 Robust Regions of Finite Time Invariance

The techniques for the computation of regions of finite time invariance presented in [25] can be extended to handle scenarios in which there is uncertainty in state and dynamics. This computation is similar to the one presented in [26], but instead of computing regions of attraction for asymptotically stable equilibrium points, we ask for certificates of finite time invariance around trajectories ("funnels"). Let $x_0(t) : [0,T] \mapsto \mathbb{R}^n$ be a trajectory in our trajectory library and $u(x,t)$ be the controller that stabilizes $x_0(t)$. Then, the closed loop dynamics of the system can be written as:

$$\dot{x} = f(x,t,w(t,x)) \tag{1}$$

where $w(t,x) \in W \subset \mathbb{R}^d$ is a free (but bounded) uncertainty term that can be used to model instantaneous disturbances, parametric model uncertainty, and state uncertainty.

We further assume that we have a time varying Lyapunov function, $V(x,t)$, that locally guarantees stability around the trajectory. An example of how such a candidate Lyapunov function may be obtained in practice is provided in Section 5. Given a set of initial conditions $X_0 = \{x|V(x,t) \leq \rho(0)\}$, our goal is to compute a tight outer estimate of the set of states the system may evolve to under the closed loop dynamics and bounded uncertainty. In particular, we are concerned with finding a set of states $X_t = \{x|V(x,t) \leq \rho(t)\}$ for each time $t \in [0,T]$ such that:

$$x(0) \in X_0 \implies x(t) \in X_t, \forall t \in [0,T].$$

As described in [25], under mild technical conditions it is sufficient to find $\rho(t)$ such that:

$$V(x,t) = \rho(t) \implies \dot{V}(x,t,w) \leq \dot{\rho}(t), \forall w \in W. \tag{2}$$

As noted in [25], checking condition (2) on a finite set of time samples, t_i, results in large computational gains while still maintaining accuracy. Thus, assuming that the closed loop dynamics are polynomial in x and w for a given t, and that w belongs to a *bounded* semi-algebraic set $W = \{w|W_j(w) \geq 0\}$, we can write a sums-of-squares optimization program to compute $\rho(t_i)$ (and thus X_t):

$$\begin{aligned} \underset{\rho, L_i, M_j}{\text{minimize}} \quad & \sum_{t_i} \rho(t_i) \\ \text{subject to} \quad & \dot{\rho}(t_i) - \dot{V}(x,w) - L_i(x)[V(x) - \rho(t_i)] - \sum_j M_j(w) W_j(w) \geq 0, \forall t_i \\ & M_j > 0, \quad \rho(t_i) \geq 0 \end{aligned} \tag{3}$$

It is easy to see that (3) is a sufficient condition for (2) since when $W_j(w) \geq 0$ and $V(x) = \rho(t_i)$, we have that $\dot{\rho}(t_i) - \dot{V}(x,w) \geq 0$. This optimization program is bilinear in the decision variables and is amenable to an alternating search over ρ and Lagrange multipliers (as in [25]). Note that the objective in our sums-of-squares program, $\sum_{t_i} \rho(t_i)$, helps us find a *tight* conservative estimate of the set of states the closed loop system may evolve to under the given uncertain dynamics. Further, one can very easily augment this optimization program to handle actuator saturations in a manner similar to the one described in [24].

4.2 Funnel Libraries

The tools from Section 4.1 can be used to create libraries of funnels offline. Given a trajectory library, $\mathscr{T}$, consisting of finitely many trajectories $x_i(t)$, locally stabilizing controllers $u_i(x,t)$, and associated candidate Lyapunov functions $V_i(x,t)$, we can compute a robust funnel for each trajectory in $\mathscr{T}$. However, there is an important

issue that needs to be addressed when designing libraries of funnels and has an analogy in the traditional trajectory library approach. In particular, trajectories in a traditional trajectory library need to be designed in a way that allows them to be sequenced together. More formally, let $\mathscr{P}$ denote the projection operator that projects a state, x, onto the subspace formed by the non-cyclic dimensions of the system (i.e. the dimensions with respect to which the Lagrangian of the system is *not* invariant). Then, for two trajectories $x_i(t)$ and $x_j(t)$ to be executed one after another, we must have

$$\mathscr{P}(x_i(T_i)) = \mathscr{P}(x_j(0)).$$

Note that the cyclic coordinates do not pose a problem since one can simply "shift" trajectories around in these dimensions. This issue is discussed thoroughly in [5] and is addressed by having a *trim trajectory* of the system that other trajectories (*maneuvers*) start from and end at (of course, one may also have more than one *trim trajectory*).

In the case of *funnel* libraries, however, it is neither necessary nor sufficient for the nominal trajectories to line up in the non-cyclic coordinates. It is the *interface* between funnels that is important. Let $x_i(t)$ and $x_j(t)$ be two nominal trajectories in our library and $F_i(t) = \{x|V_i(x,t) \leq \rho_i(t)\}$ and $F_j(t) = \{x|V_j(x,t) \leq \rho_j(t)\}$ be the corresponding funnels. Further, we write $x = [x_c, x_{nc}]$, where x_c represent the cyclic dimensions and x_{nc} the non-cyclic ones. Following Burridge and Koditschek [2], we say that two funnels are *sequentially composable* if:

$$\begin{aligned} &\mathscr{P}(F_i(T_i)) \subset \mathscr{P}(F_j(0)) \\ &\iff \forall x = [x_c, x_{nc}] \in F_i(T_i), \quad \exists x_{0,c} \quad s.t. \quad [x_{0,c}, x_{nc}] \in F_j(0) \end{aligned} \tag{4}$$

While (4) is a necessary and sufficient condition for two funnels to be executed one after another, the dependence of $x_{0,c}$ on x makes searching for $x_{0,c}$ a non-convex problem in general. Thus, we set $x_{0,c}$ to be the cyclic coordinates of $x_j(0)$, resulting in a stronger sufficient condition that can be checked via a sums-of-squares program:

$$\forall x = [x_c, x_{nc}] \in F_i(T_i), \quad [x_{0,c}, x_{nc}] \in F_j(0) \tag{5}$$

Intuitively, (5) corresponds to "shifting" the inlet of funnel F_j along the cyclic dimensions so it lines up with x_c. Assuming $F_i(T_i)$ and $F_j(0)$ are semi-algebraic sets, a simple sums-of-squares feasibility program can be used to check this condition:

$$\begin{aligned} &\rho_j(0) - V_j(\mathscr{P}([x_j(0), x_{nc}], 0)) + L(x)[V_i([x_c, x_{nc}], T_i) - \rho_i(T_i)] \geq 0 \\ &L(x) \geq 0 \end{aligned} \tag{6}$$

where $L(x)$ is a non-negative Lagrange multiplier. Note that not all pairs of funnels in the library will be *sequentially composable* in general. Thus, as we discuss in Section 4.3, we must be careful to ensure *sequential composability* when planning sequences of funnels online.

4.3 Online Planning with Funnels

Having computed libraries of funnels in the offline pre-computation stage, we can proceed to use these primitives to perform robust motion planning online. The robot's task specification may be in terms of a goal region that must be reached (as in the case of a manipulator arm grasping an object), or in terms of a nominal direction the robot should move in while avoiding obstacles (as in the case of a UAV flying through a forest or a legged robot walking over rough terrain). For the sake of concreteness, we adopt the latter task specification although one can easily adapt the contents of this section to the former specification. We further assume that the robot is provided with polytopic regions in configuration space that obstacles are guaranteed to lie in and that the robot's sensors only provide this information up to a finite (but receding) spatial horizon. Our task is to sequentially compose funnels from our library in a way that avoids obstacles while moving forwards in the nominal direction. The finite horizon of the robot's sensors along with the computational power at our disposal determines how long the sequence of planned funnels can be at any given time.

The most important computation that needs to be performed at runtime is to check whether a given funnel intersects an obstacle. For the important case in which our Lyapunov functions are quadratic in x, this computation is a Quadratic Program (QP) and can be solved very efficiently (as evidenced by the success of Model Predictive Control [3]). Let the funnel be given by the sets $X_{t_i} = \{x | V(x,t_i) \leq \rho(t_i)\}$ for $i = 1,\ldots,N$, and a particular obstacle defined by half-plane constraints $A_j x \geq 0$ for $j = 1,...,M$. Note that A_j will typically be sparse since it will contain zeros in places corresponding to non-configuration space variables (like velocities). Then, for $i = 1,\ldots,N$, we solve the following QP:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & V(x,t_i) \\ \text{subject to} \quad & A_j x \geq 0, \forall j \end{aligned} \tag{7}$$

Denoting the solution of (7) for a given t_i as $V^\star(x^\star,t_i)$, the funnel does not intersect the obstacle if and only if $V^\star(x^\star,t_i) \geq \rho(t_i), \forall t_i$. Multiple obstacles are handled by simply solving (7) for each obstacle. An important point that should be noted is that we do not require the obstacle regions to be convex. It is only required that they are represented as unions of convex polytopic sets.

For higher order polynomial Lyapunov functions, the following sums-of-squares conditions need to be checked for all t_i:

$$\begin{aligned} & V(x,t_i) - \rho(t_i) - L_i(x) \sum_j A_j x \geq 0 \\ & L_i(x) \geq 0 \end{aligned} \tag{8}$$

However, these provide only sufficient conditions for non-collision. Thus, if the conditions in (8) are met, one is guaranteed that there is no intersection with the obstacle. The converse is not true in general. Further, depending on the state space

dimension of the robot, this optimization problem may be computationally expensive to solve online. Hence, for tasks in which online execution speed is crucial, one may need to restrict oneself to quadratic Lyapunov functions. However, since most popular control design techniques like the Linear Quadratic Regulator, H^∞ and LMI methods ([19] [7]) provide quadratic Lyapunov function candidates, the loss is not substantial.

Algorithm 1 provides a sketch of the online planning loop. At every control cycle, the robot updates its state in the world along with the obstacle positions. It then checks to see if the sequence of funnels it is currently executing may lead to a collision with an obstacle (which should only be the case if the sensors report new obstacles). If so, it replans a sequence of funnels that can be executed from its current state and don't lead to any collisions. The $ReplanFunnels(x, \mathscr{O})$ subroutine assumes that funnel sequences that are *sequentially composable* in the sense of Section 4.2 have been ordered by preference during the pre-computation stage. For example, sequences may be ordered by the amount they make the robot progress in its nominal direction for a navigation task. $ReplanFunnels(x, \mathscr{O})$ goes through funnel sequences and checks two things. First, it checks that its current state is contained in the first funnel in the sequence (after appropriately shifting the funnel in the cyclic dimensions). Second, it checks that the sequence leads to no collisions with obstacles. The algorithm returns the first sequence of funnels that satisfies both criteria. Finally, the online planing loop computes which funnel of the current plan it is in and applies the corresponding control input $u_i(x, t.internal)$.

Of course, several variations on Algorithm 1 are possible. In practice, it may not be necessary to consider re-planning at the frequency of the control loop. Instead, longer sections of the plan may be executed before re-planning. Also, instead of choosing the most "preferred" collision-free sequence of funnels, one could choose the sequence that maximizes the minimum over t_i of $\frac{V^\star(x^\star,t_i)}{\rho(t_i)}$. As before, $V^\star(x^\star,t_i)$ is the solution of the QP (7). Since the 1-sublevel set of $\frac{V(x,t_i)}{\rho(t_i)}$ corresponds to the funnel, maximizing this is a reasonable choice for choosing sequences of funnels.

Algorithm 1 Online Planning

Initialize current planned funnel sequence, $\mathscr{P} = \{F_1, F_2, \ldots, F_n\}$
for $t = 0, \ldots$ **do**
  $\mathscr{O} \Leftarrow$ Obstacles in sensor horizon
  $x \Leftarrow$ Current state of robot
  Collision $\Leftarrow$ Check if $\mathscr{P}$ collides with $\mathscr{O}$ by solving QPs (7)
  if Collision **then**
    $\mathscr{P} \Leftarrow ReplanFunnels(x, \mathscr{O})$
  end if
  $F.current \Leftarrow F_i \in \mathscr{P}$ such that $x \in F_i$
  $t.internal \Leftarrow$ Internal time of $F.current$
  Apply control $u_i(x, t.internal)$
end for

5 Example

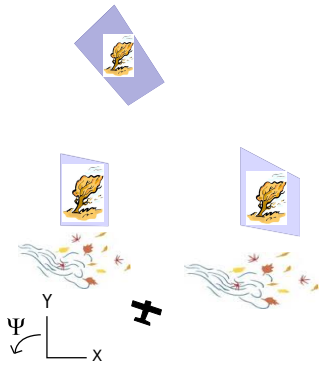


Fig. 2 Visualization of the system showing the coordinate system, polygonal obstacles, and "cross-wind".

We demonstrate our approach on a model of an aircraft flying in two dimensions through a forest of polygonal trees. A pictorial depiction of the model is provided in Figure 2. The aircraft is constrained to move at a fixed forward speed and can control the second derivative of its yaw angle. We introduce uncertainty into the model by assuming that the speed of the plane is uncertain and time-varying and that there is a time-varying "cross-wind" whose magnitude is instantaneously bounded. The full non-linear dynamics of the system are then given by:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \psi \\ \dot{\psi} \end{bmatrix}, \qquad \dot{\mathbf{x}} = \begin{bmatrix} -v(t)\cos\psi \\ v(t)\sin\psi \\ \dot{\psi} \\ u \end{bmatrix} + \begin{bmatrix} w(t) \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9}$$

with the speed of the plane $v(t) \in [9.5, 10.5] m/s$ and cross-wind $w(t) \in [-0.3, 0.3 m/s]$.

The plane's trajectory library, $\mathscr{T}$, consists of 11 trajectories and is shown in Figure 3(a). The trajectories $x_i(t) : [0, T_i] \mapsto \mathbb{R}^4$ and the corresponding nominal open-loop control inputs $u_i(t)$ were obtained via the direct collocation trajectory optimization method [1] by constraining $x_i(0)$ and $x_i(T_i)$ and locally minimizing a cost of the form:

$$J = \int_0^{T_i} [1 + u_0(t)^T R(t) u_0(t)]\, dt.$$

Here, R is a positive-definite cost matrix. For each $x_i(t)$ in $\mathscr{T}$ we design a H^∞ controller that locally stabilizes the system. This is done by first computing a time-varying linearization of the dynamics and disturbances about the trajectory:

$$\dot{\bar{x}} \approx A_i(t)\bar{x}(t) + B_i(t)\bar{u}(t) + D_i(t)w(t)$$

where $\bar{x} = x - x_i(t)$ and $\bar{u} = u - u_i(t)$. The optimal linear control law, $\bar{u}^\star(x,t) = -R^{-1}B_i(t)^T S_i(t)\bar{x}$ is obtained by solving the Generalized Riccati Differential Equation:

$$-\dot{S}_i(t) = Q + S_i(t)A_i(t) + A_i(t)^T S_i(t) - S_i(t)[B_i(t)R^{-1}B_i(t)^T - \frac{1}{\gamma^2}D_i(t)D_i(t)^T]S_i(t)$$

with final value conditions $S_i(T) = S_{T,i}$. The quadratic functional:

$$V_i(x,t) = (x - x_i(t))^T S_i(t)(x - x_i(t))$$

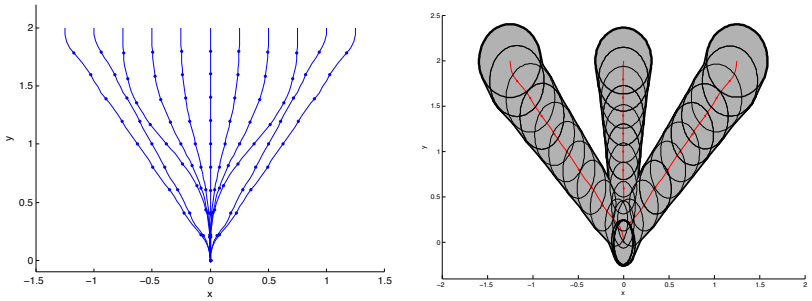
is our locally valid Lyapunov candidate. Then, as described in Section 4.1, we use $V_i(x,t)$ to compute robust regions of finite-time invariance ("funnels"). A few implementation details are worth noting here. Similar to [24], we perform the verification on the time-varying nonlinear system by taking third-order Taylor-approximations of the dynamics about the nominal trajectories. Secondly, the set of possible initial conditions for a given funnel, $X_0 = \{x | V(x,t) \leq \rho(0)\}$ needs to be specified. Since the Riccati equation does not allow us to fix $S(0)$ directly, we can only affect the shape of the initial ellipsoidal region by scaling $\rho(0)$. In order to choose a physically meaningful set of initial conditions we specify a desired ellipsoidal set $X_{0,des} = \{x | V_{des}(x) \leq 1\}$, and then choose the smallest $\rho(0)$ such that $X_{0,des} \subset X_0$. The following sums-of-squares program can be solved to obtain such a $\rho(0)$:

$$\begin{aligned} \underset{\rho(0),\tau}{\text{minimize}} \quad & \rho(0) \\ \text{subject to} \quad & \rho(0) - V(x,0) + \tau(V_{des}(x) - 1) \geq 0 \\ & \tau \geq 0 \end{aligned} \tag{10}$$

Three of the funnels in our library are shown in Figure 3(b). Note that the funnels have been projected down from the original four dimensional state space to the x-y plane for the sake of visualization.

Figure 5 demonstrates the use of the online planning algorithm in Section 4.3. The plane plans two funnels in advance while nominally attempting to fly in the y-direction and avoiding obstacles. The projection of the full sequence of funnels executed by the plane is shown in the figure. Figures 4(a) and 4(b) show the plane flying through the same forest with identical initial conditions. The only difference is that the cross-wind term is biased in different directions. In Figure 4(a), the cross-wind is primarily blowing towards the right, while in Figure 4(b), the cross-wind is biased towards the left. Of course, the plane is not directly aware of this difference, but ends up following different paths around the obstacles as it is buffeted around by the wind.

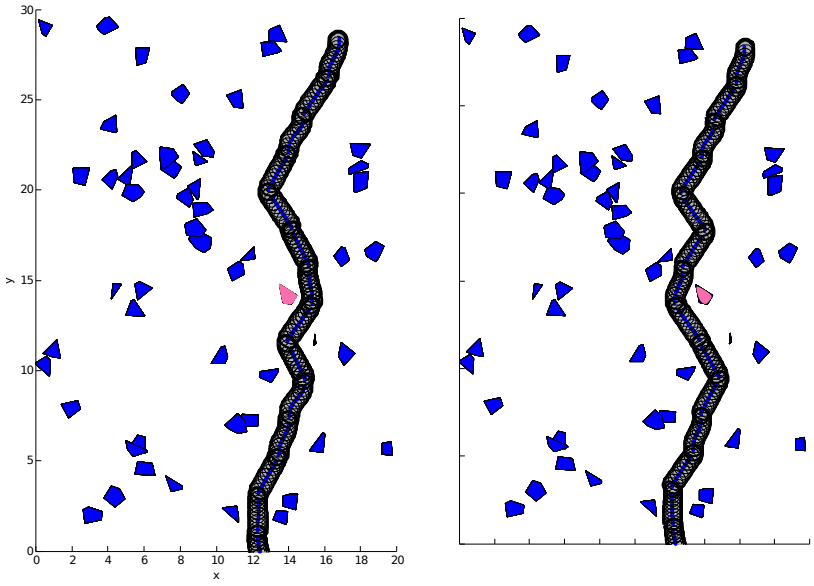
Finally, we demonstrate the utility of explicitly taking into account uncertainty in Figure 5. There are two obstacles in front of the plane. The two options available to the plane are to fly straight in between the obstacles or to bank right and attempt



(a) Trajectory library consisting of 11 locally optimal trajectories.

(b) Three funnels in our funnel library projected onto the x-y plane.

Fig. 3 Trajectory and funnel libraries for the plane.



(a) Cross-wind biased towards the right.

(b) Cross-wind biased to the left.

Fig. 4 Robust online planning though a forest of polygonal obstacles. The two subfigures show the plane flying through the same forest, but with the cross-wind biased in different directions. The eventual path through the forest is different, as the plane goes right around the marked obstacle in (a) and left in (b).

to go around them. If the motion planner didn't take uncertainty into account and chose the one that maximized the average distance to the obstacles, it would choose the trajectory that banks right and goes around the obstacles. However, taking the funnels into account leads to a different decision: going straight in between the obstacles is safer even though the distance to the obstacles is smaller.

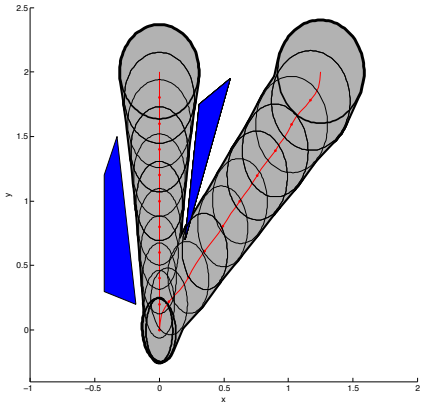


Fig. 5 This figure shows the utility of explicitly taking uncertainty into account while planning. The intuitively more "risky" strategy of flying in between two closely spaced obstacles is guaranteed to be safe, while the path that avoids going in between obstacles is less robust to uncertainty and could lead to a collision.

6 Discussion and Future Work

6.1 Stochastic Verification

Throughout this paper, we have assumed that all disturbances and uncertainty are bounded with probability one. In practice, this assumption may either not be fully valid or could lead to over-conservative performance. In such situations, it is more natural to provide guarantees of finite time invariance of a probabilistic nature. Recently, results from classical super-martingale theory have been combined with sums-of-squares programming in order to compute such probabilistic certificates of finite time invariance [23]. i.e. provide upper bounds on the probability that a stochastic nonlinear system will leave a given region of state space. The results presented in [23] can be directly combined with the approach presented in this work to perform robust online planning on stochastic systems and will be the subject of future work.

6.2 Continuously Parameterized Families of Funnels

As discussed in Section 4.2, we are currently partially exploiting invariances in the dynamics by shifting trajectories (and corresponding funnels) that we want to execute next in the cyclic coordinates so they line up with the cyclic coordinates of the robot's current state. In our example from Section 5, this simply corresponds to translating and rotating funnels so the beginning of the next trajectory lines up

with the current state's x,y and yaw. However, we could further exploit invariances in the dynamics by shifting funnels around locally to ensure that they don't intersect an obstacle while still maintaining the current state inside the funnel. One can then think of the nominal trajectories and funnels being *continuously parameterized* by shifts in the cyclic coordinates. Interestingly, it is also possible to use sums-of-squares programming to compute conservative funnels for cases in which one shifts the nominal trajectory in the *non-cyclic* coordinates [14]. Thus, one could potentially significantly improve the richness of the funnel library by pre-computing continuously parameterized funnel libraries instead of just a finite family. However, choosing the right "shift" to apply at runtime is generally a non-convex problem (since the free-space of the robot's environment is non-convex) and thus one can only hope to find "shifts" that are locally optimal.

6.3 Sequence optimization for Large Funnel Libraries

For extremely large funnel libraries, it may be computationally difficult to search all the funnels while planning online. This is a problem that traditional trajectory libraries face too [4]. Advances in submodular sequence optimization were leveraged in [4] to address this issue. The approach involves limiting the set of trajectories that are evaluated online and optimizing the sequence in which trajectories are evaluated. Guarantees are provided on the sub-optimality of the resulting strategy. This technique could be adapted to work in our framework too and will be addressed in future work.

6.4 Designing and Evaluating Trajectory Libraries

As discussed in Section 2, the design of trajectory libraries has been the subject of a large body of work in motion planning. One of the most exciting recent results is presented in [10] in which percolation theory is used to provide mathematical bounds on the speed at which a robot can safely fly through a forest. Assuming that the probability distribution that gave rise to the forest is ergodic and under a few technical conditions on the dynamics of the robot, it is shown that there exists a critical speed beyond which the robot cannot fly forever without collisions. If the robot flies below this speed, there exists an infinite path through the forest. We believe that the techniques used in [10] can be used to *evaluate and compare* funnel libraries too. In the setting where the distribution of the obstacle generating process is known and is ergodic, we can compute the probability that there exists a sequence of funnels that can be chained together to traverse the forest without colliding with a tree. Intuitively, the idea is to consider all possible funnel sequences through the forest and then computing the probability that there exists a particular sequence with no tree lying inside any funnel in that sequence.

This may provide an important extension to the approach presented in this paper since it allows us to directly deal with *uncertainty in the nature of the environment* too. It also provides us with a meaningful way to evaluate whether a given funnel library is "good enough" or whether more primitives are required in order to have a high chance of success.

7 Conclusion

In this paper, we have presented an approach for motion planning in *a priori* unknown environments with dynamic uncertainty in the form of bounded parametric model uncertainty, disturbances, and state errors. The method augments the traditional trajectory library approach by constructing stabilizing controllers around the nominal trajectories in a library and computing robust regions of finite time invariance ("funnels") for the resulting closed loop controllers via sums-of-squares programming. The pre-computed set of funnels is then used to plan online by *sequentially composing* them together in a manner that ensures obstacles are avoided. By explicitly taking into account uncertainty and disturbances while making motion plans, we can evaluate trajectory sequences based on how susceptible they are to disturbances. We have demonstrated our approach on a simulation of a plane flying in two dimensions through a forest of polygonal obstacles. Future work will focus on extending our results to scenarios in which a stochastic description of uncertainty is more appropriate and using the tools from this paper to evaluate different trajectory libraries in order to determine whether they can be successfully employed to perform the task at hand.

8 Acknowledgements

This work was partially supported by ONR MURI grant N00014-09-1-1051 and the MIT Intelligence Initiative.

References

1. J. T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM Advances in Design and Control. Society for Industrial and Applied Mathematics, 2001.
2. R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555, June 1999.
3. E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer-Verlag, 2nd edition, 2004.
4. Dey, D., Liu, T.Y., Sofman, B., Bagnell, and D. Efficient optimization of control libraries. Technical report, Technical Report (CMU-RI-TR-11-20), 2011.

5. E. Frazzoli, M. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, December 2005.
6. C. Green and A. Kelly. Toward optimal sampling in the space of paths. In *13th International Symposium of Robotics Research*, November 2007.
7. M. Green and D. Limebeer. *Linear Robust Control*. Prentice Hall, 1995.
8. Hu, TC, Kahng, A.B., Robins, and G. Optimal robust path planning in general environments. *Robotics and Automation, IEEE Transactions on*, 9(6):775–784, 1993.
9. Jacobs, P., Canny, and J. Robust motion planning for mobile robots. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 2–7. IEEE, 1990.
10. Karaman and Frazzoli. High-speed flight through an ergodic forest. In *IEEE Conference on Robotics and Automation (submitted)*, 2012.
11. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. Journal of Robotics Research*, 30:846–894, June 2011.
12. J. Kuffner and S. Lavalle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, 2000.
13. Liu, C., Atkeson, and C.G. Standing balance control using a trajectory library. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3031–3036. IEEE, 2009.
14. A. Majumdar, M. Tobenkin, and R. Tedrake. Algebraic verification for parameterized motion planning libraries. In *Proceedings of the 2012 American Control Conference (ACC)*, 2012.
15. Mayne, D.Q., Seron, M.M., Rakovic, and SV. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.
16. Mellinger, D., Kumar, and V. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.
17. Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied Mathematics. Society for Industrial Mathematics, 1995.
18. P. A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, May 18 2000.
19. Petersen, I.R., Ugrinovskii, V.A., Savkin, and A.V. *Robust control design using H-8 methods*. Communications and control engineering series. Springer, 2000.
20. Schouwenaars, T., Mettler, B., Feron, E., How, and J.P. Robust motion planning using a maneuver automation with built-in uncertainties. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 2211–2216. IEEE, 2003.
21. Sermanet, P., Scoffier, M., LeCun, and C.C.U.M.Y. Learning maneuver dictionaries for ground robot planning. In *Proc. 39th International Symposium on Robotics (ISR08)*, 2008.
22. A. Shkolnik. *Sample-Based Motion Planning in High-Dimensional and Differentially-Constrained Systems*. PhD thesis, MIT, February 2010.
23. J. Steinhardt and R. Tedrake. Finite-time regional verification of stochastic nonlinear systems. In *Proceedings of Robotics: Science and Systems (RSS) 2011*, January 17 2011.
24. R. Tedrake, I. R. Manchester, M. M. Tobenkin, and J. W. Roberts. LQR-Trees: Feedback motion planning via sums of squares verification. *International Journal of Robotics Research*, 29:1038–1052, July 2010.
25. M. M. Tobenkin, I. R. Manchester, and R. Tedrake. Invariant funnels around trajectories using sum-of-squares programming. *Proceedings of the 18th IFAC World Congress, extended version available online: arXiv:1010.3013 [math.DS]*, 2011.
26. Topcu, U., Packard, A.K., Seiler, P., Balas, and G.J. Robust region-of-attraction estimation. *IEEE Transactions on Automatic Control*, 55(1):137 –142, Jan 2010.
27. Toussaint and G.J. *Robust control and motion planning for nonlinear underactuated systems using H-infinity Techniques*. PhD thesis, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2000.