Efficient and Robust Feedback Motion Planning Under Uncertainty using the Pontryagin Difference

Dengwei Gao[1,2], Jianjun Luo[1], Weihua Ma[1], Shi Bai[2] and Brendan Englot[2]

Abstract— **This paper proposes a novel application of recent research on sums-of-squares (SOS) optimization to feedback motion planning. We use nonlinear programming (NLP) to provide open-loop control and dynamic trajectories for a vehicle in segments, and then consider the problem of generating global trajectories by using a probabilistic roadmap (PRM) or a rapidly-exploring random tree (RRT). Furthermore, we compute *funnels* (reachable sets) using SOS optimization along the trajectory in which the vehicle's state is guaranteed to remain [1]. Considering the expensive computation of SOS, we adopt a "funnel library" to pre-compute funnels [2]. A vehicle is subjected to disturbances due to model uncertainty and sensor noise, and the funnel library is computed without any knowledge of the severity of noise before motion planning. Therefore, we propose to use the Pontryagin difference method to shrink the funnels to account for noise-corrupted measurements, whose availability varies spatially throughout the state space. Our major contribution is to take into account the effect of measurement and model uncertainty in funnel computation, and we propose two efficient algorithms, feedback belief roadmap (FBRM) motion planning and feedback rapidly-exploring random belief trees (FRRBT) motion planning, to generate safe trajectories. Our algorithms are demonstrated in simulated experiments showing their advantages over others.**

I. Introduction

Vehicles maneuvering in complex environments lack guarantees on their safety when in close proximity to obstacles without an accurate dynamic model and accurate measurements. Autonomous maneuvering is also challenging when a vehicle is governed by nonlinearities, underactuated dynamics, and input constraints. Meanwhile, disturbances may drive a vehicle out of safe and stable operating regimes [2]. Relevant applications include flying through a cluttered area [3], control design for stabilization of robots [4], and planning under challenging task constraints [5]. These applications share the challenges of improving robot performance under fast motion, operation in close proximity to obstacles, and difficult state estimation. In this paper, we focus on obtaining a safe path to a goal under such conditions.

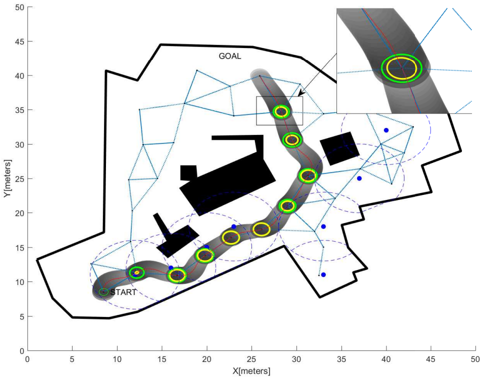


Fig. 1: A safe and stable trajectory, with a high initial uncertainty, found by FBRM. A longer trajectory leading to the lowest uncertainty at the goal was demonstrated in Fig. 1(d) in [11].

In particular, a stable feedback control strategy was developed for vehicles around a nominal trajectory, using a sums-of-squares (SOS) algorithm to certify a vehicle's stability and safety in [6]. The proposed algorithm was designed for nonlinear feedback motion planning in which each of a sequence of open loop trajectories is locally stabilized to lead to a specified goal point. An SOS algorithm is a reliable way to certify the stability of local regions employing specific feedback policies, and to approximate these regions throughout the duration of a trajectory. These sets of stability regions constrained by local linear feedback control are called *funnels*, as developed in [7]. Following these initial works, SOS was used in invariant funnels around trajectories, which computed regions of stability of limit cycles for time-varying Lyapunov functions [1]. However, their time complexity prohibits their effective use in practical, real-time planning and maneuvering applications.

In [2], Majumdar et al. presented a funnel library which is capable of being composed at runtime. Moreover, they considered noise in SOS programming subject to an uncertainty term that models external disturbances or parametric model uncertainties, where the uncertainty must be described as a semi-algebraic set, and then converted to SOS constraints. The robust funnel library is computed offline, and consequently when the uncertainties have changed we need to re-compute the whole funnel library, which is costly. In our proposed algorithms, we estimate disturbances using the extended set-membership filter (ESMF) and use the Pontryagin difference to shrink the funnel. If there is any additive disturbance, the one-step set can be computed by the Pontryagin difference [8]. We apply the Pontryagin difference to compute the set difference between the nominal stable region and the noise-corrupted probabilistic region under which changing uncertainties are considered, which is real-time viable.

In order to compute funnels, we need to obtain a collision-free open-loop trajectory. Path planning algorithms like the

[1] School of Astronautics, Northwestern Polytechnical University, Xi'an, Shaanxi, China, gaodengwei123@gmail.com, jjluo@nwpu.edu.cn, whma_npu@nwpu.edu.cn

[2] Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, New Jersey, USA, sbai1@stevens.edu, benglot@stevens.edu

rapidly-exploring random tree (RRT) [9] and probabilistic roadmap (PRM) [10] are suitable for quickly identifying collision-free paths. If measurement uncertainty is taken into account in path planning, the belief roadmap (BRM) [11] and rapidly-exploring random belief trees (RRBT) [12] may be used instead to search for feasible paths in belief space. However, a vehicle's safety may be further improved if its behavior under uncertainty can be stabilized and also certified. Thus, we propose two algorithms, feedback belief roadmap (FBRM) motion planning and feedback rapidly-exploring random belief trees (FRRBT) motion planning, based on BRM and RRBT respectively, for motion planning using funnel libraries, which allow us to compute safe and low-cost paths efficiently for a nonlinear system.

II. Background

A. Problem Statement

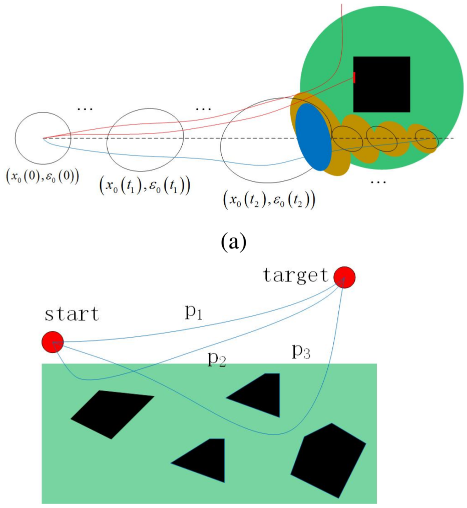
BRM and RRBT produce paths that achieve low state estimate uncertainty en route to the goal. RRBT extends to nonlinear systems, linearizing about a nominal trajectory which employs local linear feedback control. However, RRBT has two limitations. First, for nonlinear systems, local linear controllers will not work all the time. If the state is outside of the local region of attraction (RoA), the feedback controller cannot drive the system back to an equilibrium point [13]. Hence we need to predict the deviations in advance in order to take corrective actions [14]. This can also be explained by the local separation principle: in practice, an optimal feedback controller is employed by designing an optimal observer for estimating the state of the system, and the optimal estimate is also required to be inside funnels to ensure convergence, since linearization and stability are only guaranteed inside funnels. The limitation is illustrated in Fig. 2a, where a vehicle's uncertainty decreases significantly after entering a measurement zone, but two possible paths in red don't pass through the blue region and may collide.

Second, an optimal path with respect to a covariance matrix's trace, as produced by the BRM algorithm, would traverse regions where the vehicle is able to acquire more observations. For the purpose of reaching the goal position with low uncertainty, more energy-conservative paths (e.g. shorter ones) that may nonetheless be capable of safely reaching the goal are omitted. Also, a vehicle may elevate its risk of colliding with obstacles in order to obtain obstacle-relative measurements. One such example is illustrated in Fig. 2b. Path P_3 leads to the most certain goal state, whereas paths P_1, P_2 are potentially safe to follow with lower cost.

The systems we are interested in are generally nonlinear and partially observable. The robot is described by its dynamic model and sensor model:

$$\begin{aligned} \dot{x} &= f(x(t), u(t), w(t)) \\ z &= h(x(t), v(t)) \end{aligned} \tag{1}$$

where state vector $x(t) \in \mathcal{X} \subset \mathbb{R}^n$, $u(t)$ is the control input $u(t) \in \mathcal{U} \subset \mathbb{R}^m$, $w(t)$ is the model disturbance $w(t) \in \mathcal{W} \subset \mathbb{R}^d$, and $v(t)$ is the measurement disturbance $v(t) \in \mathcal{V} \subset \mathbb{R}^{d'}$. For any time $t \in [0, T]$, the piecewise-continuous open-loop control $u_0(t) : [0, T] \to \mathcal{U}$, can be integrated to compute a nominal trajectory $x_0(t) : [0, T] \to \mathcal{X}$. The state space $\mathcal{X}$ can be decomposed into $\mathcal{X}^{free}$ and $\mathcal{X}^{obs}$ which represent whether states are in collision. Our approach to finding feasible trajectories in $\mathcal{X}^{free}$ requires the computation of funnels representing regions in which the closed-loop system is guaranteed to remain.


$(x_0(0), \varepsilon_0(0))$
$(x_0(t_1), \varepsilon_0(t_1))$
$(x_0(t_2), \varepsilon_0(t_2))$
...
start
target
p1
p2
p3

(a)

(b)

Fig. 2: (a) Predicted error ellipses (black) and closed-loop ellipses (orange), a slice of funnel (blue), measurement zones (green regions), and the nominal trajectory (dashed line). If the actual trajectory is out of the blue region, the vehicle may collide with obstacles even though its closed-loop ellipses and error ellipses are collision-free, because the linear feedback control law is invalid outside of this region. (b) Three different trajectories are shown. P_1 is the optimal trajectory if the measurement is neglected, and P_3 is an optimal trajectory which can be found by BRM and RRBT with the lowest goal uncertainty. In fact, if we can guarantee the final state following P_1 or P_2 is within a set of funnels, they would be safer and less costly than P_3.

Finding open-loop controls and a nominal trajectory could be described as an nonlinear programming (NLP) problem. There are a variety of algorithms capable of solving this problem, and in our paper, GPOPS [16] is implemented to generate optimal trajectory primitives. The local optimal trajectory is produced by minimizing a cost function, for which we employ the LQR cost-to-go function,

$$\begin{aligned} J &= \int_0^T (x_0^T Q x_0 + u_0^T R u_0) \mathrm{d}t \\ Q &= Q^T \geqslant 0, R = R^T > 0, \end{aligned} \tag{2}$$

where Q and R are positive-definite matrices. This cost-to-go function is the objective function of each trajectory primitive used to construct a motion planning solution; similar functions are used as the cost metric in feedback motion planning algorithms such as [6].

B. Funnel Libraries

In this section we describe the process of computing the outer approximations of reachable sets around trajectories for a nonlinear system. The procedure is used to verify motion primitives, which are later connected to obtain a fully verified vehicle trajectory, and to parameterize motion primitives in order to cover continuous sets of behaviors, which are achieved in [2] with SOS techniques.

Given a set of initial conditions $\chi_0 \subset \mathbb{R}^n$ with $x_0(0) \in \chi_0$, our aim is to find a tight outer approximation of the subset of states the system may evolve to at time $t \in [0, T]$. In particular, we are concerned with sets $F(t) \subset \mathbb{R}^n$ such that:

$$x(0) - x_0(0) = \bar{x}(0) \in \chi_0 \Rightarrow \bar{x}(t) \in F(t), \forall t \in [0, T]. \quad (3)$$

Definition 1[2]. *A funnel $F : [0, T]$ maps from the time-interval $[0, T]$ to the power set of $\mathbb{R}^n$ such that the sets $F(t)$ satisfy the condition (3) above.*

The sets $F(t)$ are parameterized as sub-level sets on value ρ of non-negative time-varying functions V ([1]):

$$\begin{aligned} F(t) &= \{\bar{x} \in \mathbb{R}^n \mid V(t, \bar{x}(t)) \leqslant \rho(t)\} \\ &= \{\bar{x} \in \mathbb{R}^n \mid \bar{x}(t)^T S(t) \bar{x}(t) \leqslant 1, S(t) \geqslant 0\}, \end{aligned} \quad (4)$$

where $S(t)$ is a symmetric positive definite matrix. In [1], Tobenkin first presented an SOS algorithm to compute the regions of finite-time invariance (funnels) around solutions of polynomial differential equations. Moore [17] presented S-parameterization to improve the volume of funnel. Majumdar [15] presented feedback control to find a larger-volume funnel. In motion planning, sequential composability of funnels is analogous to the compatibility condition required for sequencing trajectories in the library of a Maneuver Automaton [18]. A Maneuver Automaton for the solution of a class of motion-planning problems is described as the concatenation of well-defined motion primitives selected from a finite library. Specifically for our problem, funnel libraries can be pre-computed using trajectory libraries which are derived from the graph edges of a PRM or the edges of an RRT, to which sequential composition will be applied.

III. Extension to Uncertainty Funnels

This section presents a novel solution to the problem of calculating funnels with uncertainty. In [2], uncertainty is described as a bounded semi-algebraic set $\mathcal{W} = \{w \in \mathbb{R}^d \mid g_{w,j}(w) \geqslant 0, \forall j = 1, ..., N_w\}$ which can be easily described as a SOS constraint and solved by the S-procedure. This is a conservative computation for uncertainty with a funnel library since the library is computed based on initial conditions and uncertainty in the dynamics without considering the distribution of uncertainty throughout the environment. Robustness is considered in our paper by taking the Pontryagin difference between a funnel and bounded uncertainty. This step will ensure the stability of the disturbance invariant set for the controlled system with bounded disturbances. Bounded uncertainty can be captured using the extended set-membership filter (ESMF) [19] or extended Kalman filler (EKF)[12], which produce error ellipsoids that account for measurement noise.

A. Pontryagin Difference

Definition 2. *The Pontryagin difference [20] of two sets of the same dimension $U \subset \mathbb{R}^n$, $V \subset \mathbb{R}^n$ is as follows:*

$$U \sim V = \{x_1 \in U : x_1 + x_2 \in U, \forall x_2 \in V\}.$$

The properties of the Pontryagin difference are given in [8], Theorem 2.1.

So if U represents the initial funnel at a particular time without disturbances, it is a convex set which can be described as an ellipsoid. The Pontryagin difference does not restrict a vehicle's uncertainty to be an ellipsoid, or even convex. The Pontryagin difference shows that if the state is inside the robust funnel, the state with added disturbances is also inside the initial funnel.

Definition 3. *Ellipsoid $\varepsilon(q, Q)$ in $\mathbb{R}^n$ with center q and shape matrix Q is the set,*

$$\varepsilon(q, Q) = \{x \in \mathbb{R}^n \mid \langle (x - q), Q^{-1}(x - q) \rangle \leq 1\},$$

wherein Q is positive definite $Q = Q^T$ and $\langle x, Qx \rangle > 0$ for all nonzero $x \in \mathbb{R}^n$. Here $\langle \cdot, \cdot \rangle$ denotes the inner product. An ellipsoid set is a bounded and convex set.

A funnel can be expressed using ellipsoids at a particular time t, with $q = x_0(t)$ and $Q = S(t)^{-1}$, such that:

$$\begin{aligned} \varepsilon(t) &= \{x \in \mathbb{R}^n \mid \varepsilon(x_0, S(t)^{-1})\} \\ &= \{\bar{x} \in \mathbb{R}^n \mid \varepsilon(0, S(t)^{-1})\}. \end{aligned} \quad (5)$$

The Pontryagin difference between the initial set and the disturbance set means if the state is inside the resulting set, after adding any disturbances the state will remain within the resulting set.

Assumption 1. *The disturbances we take into account are bounded and convex. We suppose the disturbances are expressed as an ellipsoid at time t: $\varepsilon_t(0, \delta) \subset \mathcal{W}$, where δ is the shape matrix representing the extent of uncertainty.*

Theorem 1. *Robustness funnel set $F(t)$ at time t is the set of $\varepsilon(t) \sim \varepsilon_t(0, \delta)$.*

Proof: Assume the two sets $U = \varepsilon(t)$ and $V = \varepsilon_t(0, \delta)$ always represent two different ellipsoids in this paper respectively, which are symmetric, convex and bounded, so that $U \sim V$ is symmetric, convex and bounded [8]. Suppose U is compact and convex and $U \sim V \neq \emptyset$, then the convex region U can be expressed as $U = \{u \mid \eta^T u \leqslant h_U(\eta), \eta \in \mathbb{R}^n\}$, where $h_U(\eta) = \sup_{u \in U} \eta^T u$, similar to set V. Accordingly, the robust funnel at t may be described as $\varepsilon = U \sim V = \{z \in \mathbb{R}^n \mid \eta^T z = h_{U \sim V}(\eta) \leqslant h_U(\eta) - h_V(\eta), \forall \eta \in \mathbb{R}^n\}$. Thus upper bound $U \sim V$ is a solution of max $\eta^T z$ subject to $\eta^T z \leqslant h_U(\eta) - h_V(\eta)$; the robust funnel is a compact subset of this region. This is approximately equivalent to the procedure for constructing the maximal volume of ellipsoids max $vol(\varepsilon)$ to find the set boundary. SOS programming can be also used to find the robust convex set in [2]. Fortunately, if U and V are ellipsoidal sets and supposing $U = \{u \mid s_i^T u \leqslant r_{Ui}, i = 1, ..., N\}$ (subscript i refers to different directions - s_i is the direction vector and r_{Ui} is the radius in different directions), then so does V, and they can be approximated by polytopes. However the Pontryagin difference can be much easier and faster to compute (see details in [21]). The Pontryagin difference will provide points on the boundary in different directions, and can be easily computed by optimization methods like [22].

B. Uncertainty Prediction

In this section, a singular value decomposition based algorithm is developed and applied to nonlinear nominal trajectory time-varying state estimation. In the previous section, it is more practical to assume that noise is unknown but bounded (UBB), especially when the bounds of noise can be obtained. The key strategy of the formulation is to find a feasible set such that the bounded error specification is met for any member of this set. As a result, set-membership filtering (SMF) is aimed at estimating the feasible set itself [19], which computes a compact feasible set in which the true states or parameters lie only under the UBB noise assumption. The approach can also be extended to nonlinear systems, e.g. [24] and [25]. Linearization error is also taken into account and convergence is proved. However, extended set-membership filtering (ESMF) leads to a large computation time. So if the problems of state or parameter estimation can be solved via stochastic approaches based on Bayesian theory, with assumptions such as white noise and known mean and covariance, the ESMF can also be replaced by a Gaussian filter, e.g. an EKF [12], to predict error ellipsoids with a high probability of being inside.

Considering the disturbances acting on the system, the robot dynamics of Eq. 1 are transformed to discrete time:

$$\begin{aligned} x_{k+1} &= g(x_k, u_k, w_k) \\ y_{k+1} &= h(x_k, v_k), \end{aligned} \tag{6}$$

where $x_k \in \mathcal{X}$, $u_k \in \mathcal{U}$, w_k, y_k is the measurement vector, and $v_k \in \mathcal{V}$. The noise follows the conditions:

$$w_k \in \varepsilon(0, Q_k), \quad v_{k+1} \in \varepsilon(0, R_{k+1}). \tag{7}$$

We can take appropriate partial derivatives for any sequence of state vectors x_k, and obtain the following time varying linear system:

$$\begin{aligned} \bar{x}_{k+1} &= A_k \bar{x}_k + B_k \bar{u}_k + w_k \\ \bar{y}_{k+1} &= H_k \bar{x}_k + v_k, \end{aligned} \tag{8}$$

where A_k, B_k, and H_k are the linearization coefficients near a nominal trajectory (x_{0k}, u_{0k}) and $\bar{x}_k = x_k - x_{0k}$, $\bar{u}_k = u_k - u_{0k}$, $\bar{y}_k = y_k - y_{0k}$, represent the deviation from the nominal path. This is the most restrictive assumption. It states that our system must be perfectly locally linear. Nonetheless, this is a reasonable approximation, and it is justified since we are using a feedback control law to stay close to the nominal trajectory. A convergence analysis is shown in [12].

The local feedback control gain K_k for a nominal trajectory can be obtained from LQR:

$$u_k = u_{0k} + K_k(\hat{x}_k - x_{0k}), \tag{9}$$

where $\hat{x}_k$ is the optimal estimate. Assume $e^e = x - \hat{x}$ is the estimation error, where x contains the real states, and thus:

$$e^e_k \in \varepsilon(0, P_k), \tag{10}$$

where P_k is the posterior covariance, which can be calculated by an ESMF [25]. Assume $e^c = \hat{x} - x_{0k}$ is the error between estimated states and nominal states, so if we use the control law (9), then we will obtain the standard filter:

$$e^c_{k+1} = A_k e^c_k + L_{k+1}(H_{k+1} e^e_k + v_k). \tag{11}$$

Let H_{k+1} be the observation matrix, $w^c_k = L_{k+1}(H_{k+1} e^e_k + v_k)$, and by fusing the two sets $L_{k+1} H_{k+1} P_k \oplus L_{k+1} R_k$, $L_{k+1} = \frac{P_{k+1|k}}{1-\rho_k} H^T_k W^{-1}_k$, $W_k = H_{k+1} \frac{P_{k+1,k}}{1-\rho_k} H^T_{k+1} + \frac{R_{k+1}}{\rho_k}$, $\rho_k = \frac{r_m}{\sqrt{p_m} + \sqrt{r_m}}$, and p_m and r_m are the maximum singular values of matrices $H_{k+1} P_{k+1} H^T_{k+1}$ and R_{k+1}, respectively. So the above Eq. (11) can be rewritten as:

$$e^c_{k+1} = A_k e^c_k + w^c_k, \tag{12}$$

expressing the error between the real system and the nominal system, and thus we employ the equation:

$$x = x_0 + \bar{x} = x_0 + (e^c + e^e). \tag{13}$$

In every step, the real error (the error between the real system and nominal system) is $\bar{x}_k \in \varepsilon(0, \Sigma_k)$, where Σ_k is the ellipsoid envelope matrix, and control law (9) will drive the system to:

$$\begin{aligned} \bar{x}_{k+1} &= A_k \bar{x}_k + B_k K_k \left(x_k - e^e_k - x_{0k}\right) + w_k \\ &= A_k \bar{x}_k + B_k K_k \bar{x}_k - B_k K_k e^e_k + w_k \\ &= A_G \bar{x}_k + (-B_k K_k e^e_k + w_k) \\ &= A_G \bar{x}_k + w'{}_k, \end{aligned} \tag{14}$$

where,

$$\begin{aligned} & A_G = A_k + B_k K_k \\ & w'_k = w_k - B_k K_k e^e_k \in W'_k = \varepsilon(0, \check{Q}_k) \\ & \check{Q}_k = \frac{-B_k K_k P_k}{1-\beta_1} + \frac{Q_k}{\beta_1} \\ & \beta_1 = \frac{\sqrt{tr(Q_k)}}{\sqrt{tr(-B_k K_k P_k)} + \sqrt{tr(Q_k)}}. \end{aligned} \tag{15}$$

If we have a one-step prediction, $\bar{x}_k \in \varepsilon(0, \Sigma_k)$:

$$\begin{aligned} & \Sigma_{k+1} = A_G \frac{\Sigma_{k|k}}{1-\beta_2} A^T_G + \frac{\check{Q}_k}{\beta_2} \\ & \beta_2 = \frac{\sqrt{tr(\check{Q}_k)}}{\sqrt{tr(A_G \Sigma_{k|k} A^T_G)} + \sqrt{tr(\check{Q}_k)}}. \end{aligned} \tag{16}$$

UD-factorization in [25] ensures all sets above are positive definite and symmetric, which are sufficient conditions to construct the ellipsoid. This derivation is similar to [12] for the EKF, where the real state is estimated in the set $\varepsilon(x_0, \Sigma)$ near the nominal trajectory under feedback control. If disturbances obey a Gaussian distribution, the set could be replaced by a high probability confidence error ellipsoid from the EKF, which is much faster than the ESMF. The

Pontryagin difference between ROA ellipsoids (5) and error boundary ellipsoids $\varepsilon(0, P_k)$ will provide the subset of the ROA that holds under uncertainty. In the path planning algorithms to follow, we will check whether $\varepsilon(x_0, \Sigma)$ is inside the funnel or not.

IV. Feedback Belief Roadmap motion planning

We assume that a PRM with node set V and edge set E is provided. Funnel libraries are computed offline using the SOS algorithm described in Section II.

Assumption 2. *Each edge $e \in E$ is associated with an open-loop trajectory library and funnel library resulting from a different combination of from-edge and to-edge.*

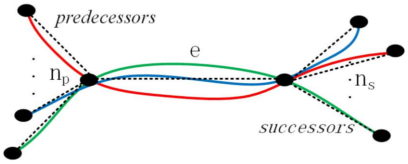


Fig. 3: Three possible trajectories along edge e.

Assumption 2 is a simplification for funnel libraries without shifting. As shown in Fig. 3, different lines represent different dynamic open-loop trajectories which could be sequentially composed along edge e. The number of trajectories associated with edge e is $N = n_p \times n_s$, where n_p and n_s are the numbers of edges which are linked with current edge e. In practice, $N < n_p \times n_s$ due to dynamic constraints.

A. Algorithm Description

For vertex $v \in V$, $v.x$ is the position of vertex v, $v.path$ is the path from the initial vertex, and $v.c$ is the cost from the initial state. $v.\varepsilon$ is the error ellipsoid in (10), $v.\Sigma$ is the closed-loop ellipsoid in (16) for the real state and nominal state, and for each edge $e \in E$, $MT = \{\tau, F, c, u\}$ is a tuple to represent the maneuver along every edge. τ are dynamic trajectories with open-loop control associated with a funnel F with cost c in (2). The selection of $e[MT]$ is based on predecessor and successor vertices, and u is the feedback control. The tuple is built with E before searching the roadmap. The ellipsoid prediction described in Section III.B is implemented by FORECAST($e[MT], v$).

Sequential composition of two funnels is defined by SEQCOMPOSITION(F_1, F_2), which is shown in detail in [2], Chapter 5. In other words, two funnels are sequentially composable if the "outlet" of $F_1(T)$ is contained within the "inlet" of $F_2(0)$. Suppose that the x_{out} in $F_1(T)$ is the state along the nominal trajectory at time T. Closed-loop ellipsoid $\varepsilon(x_{out}, \Sigma)$ defined in (16) is used to express an estimated region for real state $x \in \varepsilon(x_{out}, \Sigma)$, if $\varepsilon(x_{out}, \Sigma) \subseteq F_2(0) \sim \varepsilon$ (see Assumption 3). This check is important to ensure the state will enter the next funnel even under uncertainty. $\varepsilon(x_{out}, \Sigma) \nsubseteq F_2(0) \sim \varepsilon$ may happen when an error ellipsoid grows so fast that feedback control is unable to drive the system back to the nominal trajectory and into the next funnel. If $\varepsilon(x_{out}, \Sigma) \nsubseteq F_2(0) \sim \varepsilon$, this funnel will return 0. It is more reasonable for path planning with environmental uncertainty to check $\varepsilon(x_{out}, \Sigma), F_2(0) \sim \varepsilon$ than checking the sequential composition of funnel F_1 and F_2. Sequential composition between a polytope ($F_2(0) \sim \varepsilon$) and ellipsoid (x_{out}, Σ) is easy to check: transform the ellipsoid to a polytope in the same direction as the first polytope. For two polytopes U, V, if all $r_{Ui} \leqslant r_{Vi}$, then U is inside V, where r_{Ui}, r_{Vi} are the radii for each direction as described in **Theorem 1.**.

Assumption 3. *The initial closed-loop ellipsoid in Eq. (16) at each entrance of funnel F should be updated by the set $F(0) \sim \varepsilon_{max}$, which expresses every possible entrance state.*

Calculating the initial closed-loop ellipsoid is a simple convex optimization problem which can be solved efficiently [22]. This procedure computes an inscribed ellipsoid of polytope $F(0) \sim \varepsilon_{max}$, where ε_{max} is the largest error ellipsoid in funnel F. However, this is a rough estimate. The most accurate way is finding the error ellipsoid which is closest to the funnel at $t \in [0, T]$ (the distance can be computed using the method in [14]; this is a QCQP optimization). We then compute the Pontryagin difference at t and shrink all of the funnels within $[0, t]$ to ensure that all the states enter the most narrow region of the funnel. However, this accurate approach is costly, since the funnel and error ellipsoid do not change much within a short period so that we could use $F(0) \sim \varepsilon_{max}$ to estimate the funnel entrance. The less length of a single funnel we compute, the more accuracy we can get.

Algorithm 1 Feedback Belief Roadmap

```
1:  Build PRM in obstacle Map $\mathcal{O}$. Initialize node set V and
    edge set E. Build $MT$ for every $e \in E$
2:  Initialize $n.\varepsilon = \varepsilon_0$, $v.\Sigma = \Sigma_0$, $v.path = v.x_{init}$
3:  PUSH $v \rightarrow Q$
4:  while $Q \neq \emptyset$ do
5:      POP $n \leftarrow Q$ with the smallest n.c in Q
6:      if $n = v_{goal}$ then
7:          return n.path
8:      end if
9:      for all $v \in E[n]$ and $v \notin n.path$ do
10:         $v \leftarrow$ FORECAST$(n, v)$
11:         Find $F_1 \in n[MT]$, $F_2 \in v[MT]$
12:         if SEQCOMPOSITION$(F_1, F_2)$ then
13:             v.c = n.c+v[MT].c
14:             v.path = n.path $\cup$ v.x
15:             PUSH $v \rightarrow Q$
16:         end if
17:     end for
18: end while
```

The FBRM search process is shown in Algorithm 1. The algorithm builds a graph, whose edges are based on [23]. Algorithm 1 assumes that the initial error ellipsoid is ε_0 and Σ_0 is an estimated value which satisfies $\Sigma_0 <$ all $(F(0) \sim \varepsilon_0)$ to express that the real state is inside the funnel at the initial state. We push the initial vertex into search queue Q. Line 5 in Algorithm 1 finds the optimal cost for each node in Q; n.c could rely on an A* heuristic to find the goal faster.

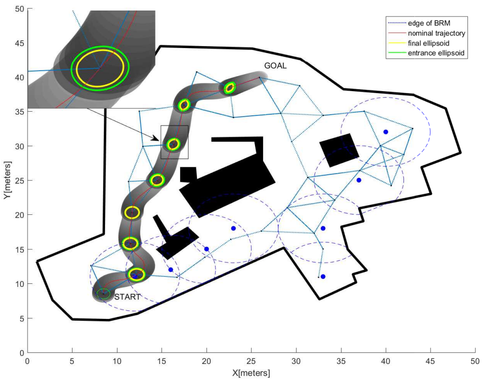


Fig. 4: The best trajectory is shown when a robot is initialized with low uncertainty.

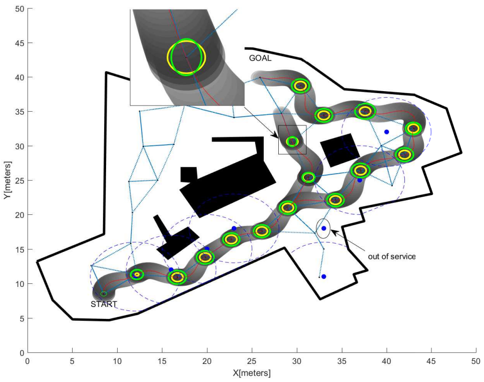


Fig. 5: The best trajectory has been found when a beacon is out of service. Compared to Fig. 4, the vehicle chose a different path with more observations.

Lines 9-18 are the kernel of Algorithm 1. Line 9 searches the near node along an edge which is not in the path of the current node. Line 11 chooses a funnel along the edge. If the condition in line 13 is not satisfied, we continue the loop to find other nodes which may satisfy the funnel constraints. If the funnel is shrunken by an obstacle, the condition will be harder to satisfy. If the state through the funnel can reach the new node, the node is added to queue Q.

B. *Simulation Results*

In this section, we applied the FBRM algorithm proposed in this paper to a simulated ground vehicle built in Matlab. The ground vehicle moves with constant speed; the dynamic model is formulated as follows,

$$x = \begin{bmatrix} x \\ y \\ \psi \\ \dot{\psi} \end{bmatrix}, \dot{x} = \begin{bmatrix} v\left(t\right)\sin\left(\psi\right) \\ v\left(t\right)\cos\left(\psi\right) \\ \dot{\psi} \\ u \end{bmatrix}. \qquad (17)$$

The vehicle has a fixed forward speed of $10m/s$ and the yaw angle ψ can be controlled. The uncertainty in the simulation is assumed to consist of modeling and measurement error. The initial state $x(0)$ was set to $[8.5; 8.5; \pi/2; 0]$, the final state was set to $[26; 40; 0; 0]$, and the control is bounded in the range $[-2\pi, 2\pi]\ rad/s^2$.

As shown in Figs. 1, 4, and 5, we used a similar map to that of the original BRM algorithm [11]. In the figures, geometric obstacles are illustrated in filled regions, and observation beacons are distributed in the map to provide measurements nearby. The edges of the BRM are shown as blue dotted lines, and computed funnels are shown in the gray regions. Green lines represent the projection onto the $x-y$ plane of the entrance ellipsoid of each funnel, whereas yellow lines represent the projection of the final ellipsoid with the closed-loop estimation. Two connected edges are safe for the vehicle to traverse if and only if the final ellipsoid of the first funnel completely falls into the start ellipsoid of the second funnel. If the funnels can be connected from the start to the goal by checking the above condition in the graph, then we obtain a stable and safe trajectory near the nominal one without collision and we remain within the sets of funnels all the time. The path generated by our algorithm is highlighted in red with the funnels overlaid on top of it.

In Figs. 1 and 4, we demonstrate the performance of our algorithm by configuring the vehicle with different initial uncertainty. With a lower initial uncertainty, it's unnecessary for the vehicle to localize itself (Fig. 4). In contrast, higher initial uncertainty makes the localization by beacons essential, which results in a detour through the beacons (Fig. 1). In Fig. 5, we assume one of the beacons is out of service, so the trajectory in Fig. 1 no longer achieves all funnel connections and a better trajectory is found to ensure safety.

V. Feedback Rapidly-exploring Random Belief Trees Motion Planning

A. *Algorithm Description*

In Section IV, we proposed a new algorithm FBRM and compared it with the BRM algorithm. However in Algorithm 1, if there is a large number of edges in the graph, we need more trajectory libraries and funnels along each edge to connect adjacent funnels with sequential composition, which is expensive. Therefore, a more efficient algorithm than FBRM is proposed here. Funnel libraries and trajectory libraries can shift along cyclic coordinates (see [2] and [18] for details on shifting). Hence, we can build general funnel libraries offline, each of which is associated with a nominal trajectory and cost, and after that an RRT may be built with the edges from the funnel libraries [2].

The FRRBT algorithm is shown in Algorithm 2. The major components of the algorithm include the vertex $v \in V$, where V is the node set. FT is a funnel tuple $\{F, c, \tau\}$, $FT.F$ is the funnel, $FT.c$ is the cost, and $FT.\tau$ is the nominal trajectory from the predecessor node. The definition of $v.x$, $v.c$, $v.\Sigma$, $v.\varepsilon$, and $v.path$ are the same as in Section IV. E is the edge set, which is comprised of funnel tuples.

The SAMPLE() and NEAREST(V, x) functions are the same as in RRBT. Since $\mathcal{F}$ could connect two nodes with a specific trajectory, RANDOMGROW(v) returns a node which a random funnel shifted from $F \in \mathcal{F}$ connected with vertex v can be connected to. The new random vertex belongs to a specified distribution based on the constructed $\mathcal{F}$.

Algorithm 2 Feedback Rapidly-exploring Random Belief Trees

```
1: Build Funnel Library $\mathcal{F}$
2: Initialize Map $\mathcal{O}$, $v.\varepsilon = \varepsilon_0$, $v.\Sigma = \Sigma_0$, $v.path = v.x_{init}$,
   $v.c = 0$, $V = v$, $E = \emptyset$
3: while $i < M$ do
4:    $x_{rand}$ = SAMPLE()
5:    $v_{nearest}$ = NEAREST($V, x_{rand}$)
6:    $v_{rand}$ = RANDOMGROW($v_{nearest}$)
7:    $FT$ = SHIFT($\mathcal{F}, v_{nearest}, v_{rand}$)
8:    if $\neg$ CheckFunnelCollision($FT, \mathcal{O}$) then
9:       n = PROPAGATE($v_{nearest}, v_{rand}$)
10:      if n $\neq \emptyset$ then
11:         PUSH $n \rightarrow Q$
12:      end if
13:   end if
14:   $E = E \cup FT$
15:   while $Q \neq \emptyset$ do
16:      POP $n \leftarrow Q$
17:      for all $v \in V$ near $n$ find a feasible
         $FT' \in$SHRIFT($\mathcal{F}, v, n$) do
18:         if $\neg$CheckFunnelCollision($FT', \mathcal{O}$) and $v.c + FT'.c < n.c$ then
19:            $n'$ = PROPAGATE($v, n$)
20:            if n' $\neq \emptyset$ then
21:               PUSH $n' \rightarrow Q$
22:            end if
23:         end if
24:      end for
25:   end while
26:   i = i+1
27:   $V = V \cup n'$
28:   $E = E \setminus FT$, $E = E \cup FT'$
29: end while
```

SHIFT($\mathcal{F}, v, n$) is used to find a funnel in $\mathcal{F}$ and shift in cyclic coordinates to connect two nodes v and n. CheckFunnelCollision($F, \mathcal{O}$) checks if a funnel intersects $\mathcal{X}^{obs}$. If the funnel intersects $\mathcal{X}^{obs}$ partially, the funnel is shrunk based on $\mathcal{X}^{obs}$ ([2]). Meanwhile the function returns a safer modified funnel to replace funnel F.

PROPAGATE(v, n) is a function to propagate the error ellipsoid and closed-loop ellipsoid, and to find a funnel in $\mathcal{F}$ to connect two nodes. SEQCOMPOSITION() finds two funnels with uncertainties that satisfy the condition of sequential composition which is introduced in Section IV. If the funnels can be sequentially composed, then lines 4-5 will update the cost and path, and return the node n.

In Algorithm 2, lines 15-28 are an optimization process that tries to find all candidate feasible parents of node n and add the best node to the tree. At line 18 is an inequality which ensures the minimization of cost to the current node. Lines 19-20 determine whether the new edge to node n' with minimum cost can satisfy the constraints.

Algorithm 3 n=PROPAGATE(v, n)

```
1: $n \leftarrow$ FORECAST($v, n$)
2: $FT$ = SHRIFT($\mathcal{F}, v, n$)
3: if SEQCOMPOSITION($E[v].F, FT.F$) then
4:    $n.c = v.c + FT.c$
5:    $n.path = v.path \cup n.x$
6: else
7:    return NULL
8: end if
```

B. Experimental Results

In this section, we applied the FRRBT algorithm to the model described in section IV.B. Fig. 6 illustrates a trajectory library and its funnel library (one funnel is shown).

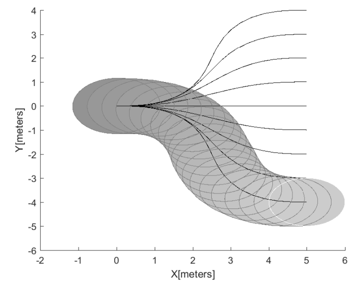


Fig. 6: Funnel libraries and trajectories.

Figs. 7 and 8 show results from Algorithm 2. The start point is located at $[0, 40]m$ and goal at $[80, 40]m$. The gray region cannot be reached due to the distribution of funnel libraries. The black regions are obstacles, and we can obtain position measurements near the obstacles through relative measurement. Similarly to the experiments in the above section, lower initial uncertainty leads to a shorter trajectory, and higher initial uncertainty forces the vehicle to take a longer path to collect observations. In Fig. 7, the tree was built after 300 iterations and the green curve represents the shortest safe trajectory to the goal. The enlarged part highlights that a funnel is shrunk near an obstacle. However, we didn't plot the entire tree after 1000 iterations, whose resulting path is depicted in Fig. 8. FRRBT is faster than FBRM in off-line computation, since we use the shift of funnel libraries so that fewer funnels need to computed.

VI. Conclusions

In this paper we presented an approach for feedback motion planning in uncertain environments to ensure a vehicle safely avoids obstacles and instability. We construct stable feedback controllers around the nominal trajectory provided by optimal control to ensure the state remains inside a funnel. The method extends feedback motion planning by considering a belief space with environment uncertainty, uses the Pontryagin difference to update the current funnel (reachable set), and we can thus guarantee that all real vehicle states will lie within sequential composition funnels. The simulations provided demonstrate that our approach

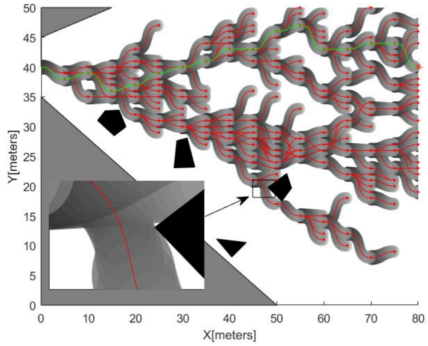


Fig. 7: FRRBT finds the shortest trajectory to the goal.

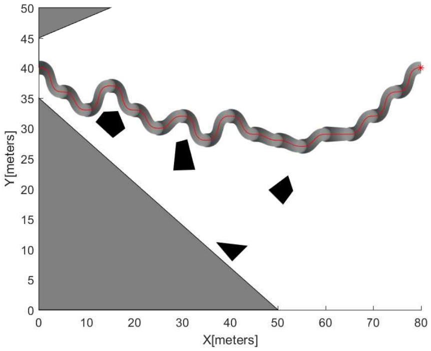


Fig. 8: A longer trajectory returned under higher initial uncertainty.

offers new capabilities for trajectory planning and control in the presence of noise and uncertainty. We have applied our approach in uncertain environments with results that guarantee vehicle stability and collision avoidance.

However, there are several remaining challenges. The extended set-membership filter is slower than an EKF, but it can provide a well-defined boundary for any distribution to ensure safety. We haven't yet tested a real robot to verify our approach, but this will be a subject of future work. We are hopeful that this approach has the potential to assist in the safe operation of autonomous vehicles, for which safe operation under uncertainty is a pressing concern.

ACKNOWLEDGMENTS

This work was supported by the Major Program of the National Natural Science Foundation of China under Grant Numbers 61690210 & 61690211. The authors are grateful to Jinkun Wang, Fanfei Chen, and other members of the Robust Field Autonomy Lab at Stevens Institute of Technology for their helfpul comments on this manuscript. We also thank the Aerospace Flight Dynamics Lab at Northwestern Polytechnical University in China.

REFERENCES

[1] 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*, vol. 44(1), pp. 9218-9223, 2011.

[2] A. Majumdar and R. Tedrake, "Funnel Libraries for Real-Time Robust Feedback Motion Planning," *The International Journal of Robotics Research*, vol. 36(8), pp. 947-982, 2017.

[3] M. Nieuwenhuisen and S. Behnke, "Local Multiresolution Trajectory Optimization for Micro Aerial Vehicles Employing Continuous Curvature Transitions," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3219-3224, 2016.

[4] I.R. Manchester and U. Mettin, "Stable Dynamic Walking over Uneven Terrain," *The International Journal of Robotics Research*, vol. 30(3), pp. 265-279, 2011.

[5] A. Shkolnik, M. Levashov, I.R. Manchester, and R. Tedrake, "Bounding on Rough Terrain with the Little Dog Robot," *The International Journal of Robotics Research*, vol. 30(2), pp. 192-215, 2010.

[6] R. Tedrake, I.R. Manchester, M.M. Tobenkin, and J.W. Roberts, "LQR-Trees: Feedback Motion Planning via Sums of Squares Verification," *The International Journal of Robotics Research*, vol. 29(8), pp. 1038-1052, 2010.

[7] R.R. Burridge, A.A. Rizzi, and D.E. Koditschek, "Sequential Composition of Dynamically Dexterous Robot Behaviors," *The International Journal of Robotics Research*, vol. 18(6), pp. 534-555, 1998.

[8] I. Kolmanovsky and E.G. Gilbert, "Theory and Computation of Disturbance Invariant Sets for Discrete-time Linear Systems," *Mathematical Problems in Engineering*, vol. 4(4), pp. 317-367, 1998.

[9] S. Karaman and E. Frazzoli, "Sampling-Based Algorithms for Optimal Motion Planning," *The International Journal of Robotics Research*, vol. 30(7), pp. 846-894, 2011.

[10] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, vol. 12(4), pp. 566-580, 1996.

[11] S. Prentice and N. Roy, "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance," *The International Journal of Robotics Research*, vol. 28(11-12), pp. 1448-1465, 2009.

[12] A. Bry and N. Roy, "Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 723-730, 2011.

[13] H.K. Khalil, *Nonlinear Systems*, Upper Saddle River, NJ: Prentice Hall, 2001.

[14] C. Vömel and D. Pardo, "On Reachability Sets for Optimal Feedback Controllers: Monitoring the Approach of a Region of Attraction," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1441-1446, 2016.

[15] A. Majumdar, A.A. Ahmadi, and R. Tedrake, "Control Design along Trajectories with Sums of Squares Programming," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4054-4061, 2013.

[16] M.A. Patterson, and A.V. Rao, "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, vol. 41(1), pp. 1-37, 2014.

[17] J. Moore, R. Cory, and R. Tedrake, "Robust Post-Stall Perching with a Simple Fixed-Wing Glider using LQR-Trees," *Bioinspiration and Biomimetics*, vol. 9(2), pp. 13-28, 2014.

[18] E. Frazzoli, M. Dahleh, and E. Feron, "Maneuver-based Motion Planning for Nonlinear Systems with Symmetries," *IEEE Transactions on Robotics*, vol. 21(6), pp. 1077-1091, 2005.

[19] P.S.R. Diniz, *Adaptive Filtering Algorithms and Practical Implementation*, New York: Springer, 2008.

[20] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*, Basel: Birkhauser, 2008.

[21] A. Kurzhanskiy and P. Gagarinov, "Ellipsoidal Toolbox," *Proceedings of the IEEE International Conference on Decision and Control*, pp. 1498-1503, 2006.

[22] A.A. Ahmadi, G. Hall, A. Makadia, and V. Sindhwani, "Geometry of 3D Environments and Sum of Squares Polynomials," *Optimization and Control*, arXiv:1611.07369 [math.OC], 2017.

[23] S.M. LaValle, M.S. Branicky, and S.R. Lindemann, "On the Relationship Between Classical Grid Search and Probabilistic Roadmaps," *The International Journal of Robotics Research*, vol. 23(7-8), 2004.

[24] T. Rassi, N. Ramdani, and Y. Candau, "Set Membership State and Parameter Estimation for Systems Described by Nonlinear Differential Equations," *Automatica*, vol. 40(10), pp. 1771-1777, 2004.

[25] B. Zhou, J. Han and G. Liu, "A UD Factorization-based Nonlinear Adaptive Set-Membership Filter for Ellipsoidal Estimation," *International Journal of Robust Nonlinear Control*, vol. 18, pp. 1513-1531, 2008.