Robust Trajectory Planning for Autonomous Parafoils under Wind Uncertainty

Brandon Luders[*], Ian Sugel[†], and Jonathan P. How[‡]

Aerospace Controls Laboratory

Massachusetts Institute of Technology, Cambridge, MA

{luders,isugel,jhow}@mit.edu

A key challenge facing modern airborne delivery systems, such as parafoils, is the ability to accurately and consistently deliver supplies into difficult, complex terrain. Robustness is a primary concern, given that environmental wind disturbances are often highly uncertain and time-varying, coupled with under-actuated dynamics and potentially narrow drop zones. This paper presents a new on-line trajectory planning algorithm that enables a large, autonomous parafoil to robustly execute collision avoidance and precision landing on mapped terrain, even with significant wind uncertainties. This algorithm is designed to handle arbitrary initial altitudes, approach geometries, and terrain surfaces, and is robust to wind disturbances which may be highly dynamic throughout the terminal approach. Explicit, real-time wind modeling and classification is used to anticipate future disturbances, while a novel uncertainty-sampling technique ensures that robustness to possible future variation is efficiently maintained. The designed cost-to-go function enables selection of partial paths which intelligently trade off between current and reachable future states. Simulation results demonstrate that the proposed algorithm reduces the worst-case impact of wind disturbances relative to state-of-the-art approaches.

I. Introduction

A key challenge facing modern airborne delivery systems, such as parafoils, is the ability to accurately and consistently deliver supplies into difficult, complex terrain. This terminal guidance problem – guiding the parafoil from a potentially high altitude to land precisely with a desired position and heading – presents significant technical challenges, particularly for the large-scale systems considered in this work. Parafoil dynamics are highly non-linear and under-actuated, with potentially large turning radii and severely limited (if any) vertical control, resulting in a descent rate determined by atmospheric conditions and disturbances. Parafoil drop locations have arbitrary, non-convex terrain maps, which can pose a significant problem for constraint satisfaction even if mapped in advance. Parafoils are subject to uncertain and variable wind environments, which, if uncompensated, often result in unacceptably large errors between predicted and actual trajectories. Finally, many applications often have tight landing restrictions; missing the target location, even by a small distance, can lead to unintended collisions with natural or man-made hazards (including the parafoil itself), or even theft of cargo. Precise delivery is essential to avoid loss of supplies or unacceptably dangerous recovery efforts.

This paper presents a new on-line trajectory planning algorithm that enables a large, autonomous parafoil to robustly execute collision avoidance and precision landing on mapped terrain, even with significant wind uncertainties. This algorithm is designed to handle arbitrary initial altitudes, approach geometries, and terrain surfaces, and is robust to significant wind disturbances which may be highly dynamic throughout terminal approach, including updrafts and downdrafts. The planner is able to quickly identify, refine, and update accurate landing trajectories, even subject to nonlinear dynamics and low controllability.

[*]PhD Candidate, Department of Aeronautics and Astronautics, MIT; Member AIAA

[†]SM Candidate, Department of Aeronautics and Astronautics, MIT

[‡]Richard C. Maclaurin Professor of Aeronautics and Astronautics, MIT; Associate Fellow AIAA

The developments in this paper build upon chance-constrained rapidly-exploring random trees (CC-RRT), an online framework for robust motion planning in cluttered, non-convex environments [1]. CC-RRT leverages the benefits of sampling-based algorithms (*e.g.,* incremental construction, trajectory-wise constraint checking, no state-space discretization) and particularly RRTs [2] (*e.g.,* rapid exploration of high-dimensional configuration spaces, dynamically feasible trajectories by construction), while using chance constraints [3] to ensure probabilistic feasibility with guaranteed, user-specified bounds. By utilizing trajectory-wise constraint checking, CC-RRT can efficiently evaluate the risk of constraint violation online due to multiple sources of both internal and external uncertainty, including dynamic obstacles [4]. Previous work has extended this formulation to nonlinear dynamics and/or non-Gaussian uncertainty [5]. These formulations use full time-series simulation of the future uncertainty distribution (or its particle-based approximation) along each trajectory, approximating probabilistic feasibility guarantees against polyhedral constraints.

This paper considers three contributions to this RRT-based framework which result in superior performance on the parafoil terminal guidance problem (Section III) compared to state-of-the-art algorithms [6] (Section II). First, a novel wind uncertainty model is developed, using real-time observed wind data to classify and anticipate the wind uncertainty environment online (Section IV). The wind model consists of the sum of two components: a simple moving-average filter for the mean wind, and multi-modal, colored-noise dynamics for the transient wind that are trained offline from previous wind observations. A trained SVM classifier then classifies the observed wind profile in real-time, using features such as intensity and invariance. The resulting model is shown to accurately represent true wind behavior, while adjusting the conservatism to reflect prevailing conditions.

Second, we utilize this multi-class wind model to derive the analytic *a priori* uncertainty distribution over future possible trajectories (Section V). This is leveraged in a novel variation of the CC-RRT path planner (Section VI), which uniformly samples the uncertainty distributions for constraint checking, ensuring robust avoidance of undesirable collisions with arbitrary, possibly aggressive terrain maps. Unlike previously-proposed particle-based formulations [5,7], dynamic state propagation is not required, yielding a more efficient robustness formulation.

Finally, the relative value of paths is assessed via a novel terminal cost-to-go function, which utilizes a fixed-horizon discrete approximation of the parafoil reachability set. This enables selection of partial paths from any altitude that intelligently trade off between current and reachable future states (Section VII).

Extensive simulation results (Section VIII) show the effectiveness of each of these components, and that the full parafoil CC-RRT algorithm can achieve superior landing accuracy in both average-case and worst-case performance relative to state-of-the-art algorithms, such as BLG [6]. In particular, we demonstrate that the analytic-sampling approach achieves stronger wind robustness than mean-wind estimation or replanning alone, and that the planner is largely invariant to changes in altitude or terrain, unlike previously-developed planners.

II. Background and Related Work

Autonomous resupply is an active area of military research, which can be largely subdivided into two categories of terminal guidance. The first category, glide-slope-based planning, utilizes the concept of the glide-slope surface or cone: the set of all position/heading states which, assuming constant velocities and disturbances, guide the parafoil to the goal state. Calise and Preston [8] utilize a series of scripted maneuvers to estimate the parameters required to accurately compute the glide-slope, then execute turning maneuvers to drive the parafoil to the glide-slope. Though useful as an approach trajectory, this framework heavily constrains the solution space, and requires tracking from a large initial distance, both vertically and laterally. This makes the approach sensitive to uncertainty in both the vehicle dynamics and environment, especially given that the glide-slope surface shifts as a function of current wind conditions. Slegers et al. [9] track the glide-slope using nonlinear model predictive control (MPC), improving rejection of small-scale disturbances, but still require long-term glide-slope tracking. Bergeron et al. [10] use feedback control, known as glide-slope guidance (GSG), to drive the goal approach based on the estimated glide-slope and wind conditions. This minimizes the effect of coupled system uncertainty and ensures a maximum heading deviation from the estimated wind direction. While this approach takes some measures to account for the effect of wind uncertainty on the parafoil landing position, the approach offers no robustness to interaction with terrain obstacles, nor does it overcome the fundamental constraint of the solution space imposed by the glide-slope approach paradigm.

Trajectory-based approaches, on the other hand, generate arbitrary reference trajectories online to optimize a pre-determined cost function, utilizing various control schemes to track these trajectories. Gimadieva[11] formulates parafoil terminal guidance as an optimal control problem and establishes necessary conditions for optimality, but this approach lacks the computational efficiency needed for real-time implementation (thus making it unable to adjust for varying wind conditions and model uncertainties during flight).

The Band Limited Guidance (BLG) algorithm[6] uses direct optimization via Nelder-Mead simplex search to minimize a cost function of the predicted terminal vehicle state. BLG guarantees that control bandwidth constraints are satisfied to ensure accurate trajectory following, and its computational efficiency enables the use of online replanning, making it effective for many nominal wind and terrain conditions. However, BLG is fundamentally limited in its starting altitude due to optimization scalability and high dimensionality, constraining mission flexibility. BLG incorporates no notion of wind variations in the planner planner, instead relying on reactive replanning to address unexpected wind effects. The direct optimization technique does not consider the possibility off-nominal, adverse terrain interactions caused by changing wind conditions, particularly on complex terrain maps.

The IDVD algorithm developed by Yakimenko and Slegers[12] utilizes inverse dynamics to connect the initial vehicle state to the target terminal state, while guaranteeing the terminal conditions of this nonlinear boundary value problem (BVP) are satisfied. While also computationally efficient, this approach cannot guarantee satisfaction of control bandwidth constraints, requiring iteration in order to ensure the trajectory plan can be tracked by the controller. Moreover, this approach also relies on rapid, reactive replanning to offset uncertainties during execution, assuming a constant wind during planning. Subsequent work has added robustness to wind variations in the formulation, by utilizing GPUs to parallelize a Monte Carlo simulation of possible future winds, and the resulting parafoil trajectories, based on available measurements[13]. However, significant computational effort is required to run these simulations online. Within these simulations, the solution space is restricted to a limited number of candidate solutions of the original BVP (*e.g.,* constant-rate turn), and all simulated wind profiles are assumed constant. While this constant-wind approach effectively incorporates the overall, trajectory-wide wind effect, it is overly optimistic: failing to consider dynamic wind changes may result in missing future possible terrain collisions. Recent work considers the use of Bezier curves to perform optimized path planning (and replanning) for a small parafoil in three-dimensional obstacle fields[14]. However, the proposed optimization is highly sensitive to initial conditions, and scales poorly with its degrees of freedom (*i.e.,* the control points of the Bezier curves), limiting the environmental complexity and starting altitude. Further, robustness is only demonstrated against wind conditions that are either constant or show little variation throughout the descent.

In summary, the general body of parafoil terminal guidance algorithms is subject to some or all of the following limitations: (1) an artificially-constrained solution space, often based on pre-conceived notions of the solution form; (2) reliance on a previous descent phase (see Section III) to bring the parafoil to initial conditions suitable for successful terminal guidance; (3) implicit or explicit constraints on the starting altitude; and/or (4) a reactive approach to handling the effect of wind uncertainty. The algorithm developed in this paper is designed to address these limitations, allowing for real-time parafoil planning with arbitrary environment geometries, initial conditions (including altitude), and dynamic wind uncertainty. Though the proposed algorithm is not optimal, it can quickly identify feasible solutions taking the parafoil near the target, then using remaining computation time online to refine and update the solution with changing wind conditions.

III. Problem Statement

The terminal guidance paradigm typically utilizes a combination of a homing phase, designed to steer the parafoil directly toward the target, and an energy management phase, designed to descend above the target until an appropriate altitude is reached for terminal guidance[6,12,14]. Such algorithms typically assume that terminal guidance will begin in relative proximity to the target location, in both lateral distance and altitude. Though the approach in this work will often operate under similar conditions, such assumptions are not necessary. The parafoil CC-RRT algorithm, introduced below, will guide the parafoil as close to the target as possible from any initial conditions, including altitude.

The vehicle state is represented as $\mathbf{x} = \begin{bmatrix} \mathbf{p}^T & \psi & \mathbf{s}^T \end{bmatrix}^T$, where $\mathbf{p} = (p_x, p_y, p_z)$ is the position, ψ is the heading, and $\mathbf{s}$ represents a vector of any additional states needed to characterize the parafoil's motion.

In the terminal guidance problem, the objective is to guide the parafoil from some initial position $\mathbf{p}_I$ and heading ψ_I (full state $\mathbf{x}_I$) to some target location $\mathbf{p}_G$ (full state $\mathbf{x}_G$). The parafoil dynamics are represented as the nonlinear state-space system

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}), \quad \mathbf{x}(t_I) = \mathbf{x}_I, \tag{1}$$

where t_I is the initial time, $\mathbf{u}$ are the control inputs and $\mathbf{w} = (w_x, w_y, w_z)$ are the wind disturbances. The parafoil model used in this work is detailed in Section III.A.

The wind disturbances are unknown at current and future times; denote the most recent wind observation as $\mathbf{w}_I$ (if none have been taken, a prior value from forecasting data may be applied, or simply assume $\mathbf{w}_I = 0$). In this work, we choose to represent the wind disturbances using the generalized model

$$\dot{\mathbf{w}} = f_w(\mathbf{w}, \overline{\mathbf{w}}, \mathbf{v}), \quad \mathbf{w}(t_I) = \mathbf{w}_I, \tag{2}$$

where $\overline{\mathbf{w}}$ is an estimate of the mean wind, assumed to be available to the planner, and $\mathbf{v}$ is unknown model noise. The wind model chosen for this work is derived in Section IV.

The parafoil terminal guidance problem is a specific case of a more general trajectory planning problem. At each time step, the path planner attempts to solve the optimal control problem

$$\begin{aligned}
\min_{\mathbf{u}} \quad & \phi_f(\widehat{\mathbf{x}}(t_f), \mathbf{x}_G) + \int_{t_I}^{t_F} \phi(\widehat{\mathbf{x}}, \mathbf{x}_G) && (3)\\
\text{s.t.} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}), \quad \mathbf{x}(t_I) = \mathbf{x}_I, && (4)\\
& \dot{\widehat{\mathbf{x}}} = f(\widehat{\mathbf{x}}, \mathbf{u}, \widehat{\mathbf{w}}), \quad \widehat{\mathbf{x}}(t_I) = \mathbf{x}_I, && (5)\\
& \dot{\mathbf{w}} = f_w(\mathbf{w}, \overline{\mathbf{w}}, \mathbf{v}), \quad \mathbf{w}(t_I) = \mathbf{w}_I, && (6)\\
& \dot{\widehat{\mathbf{w}}} = f_w(\widehat{\mathbf{w}}, \overline{\mathbf{w}}, 0), \quad \widehat{\mathbf{w}}(t_I) = \mathbf{w}_I, && (7)\\
& \mathbf{u} \in \mathcal{U} \quad \forall t, && (8)\\
& \mathbb{P}_{\mathbf{v}}(\mathbf{x} \in \mathcal{X}) \geq p_{\text{safe}}. && (9)
\end{aligned}$$

The wind state $\widehat{\mathbf{w}}$ and parafoil state $\widehat{\mathbf{x}}$ are those in which $\mathbf{v} \equiv 0$, representing deterministic, nominal propagation of the dynamics under the assumption of mean wind $\overline{\mathbf{w}}$. This is utilized to ensure that the objective (3) being optimized is deterministic, though stochastic formulations may be applied.

The sets $\mathcal{U}$ and $\mathcal{X}$ represent constraints on the input and state, respectively. The state constraints $\mathcal{X}$ must be *probabilistically* satisfied, *i.e.,* satisfied with probability over all possible $\mathbf{v}$ (represented by $\mathbb{P}_{\mathbf{v}}$) of at least p_{safe}, specified by the user. These constraints include the terrain map $T(p_x, p_y)$, which is assumed to be perfectly known; the terminal time t_F is the time at which $p_z \leq T(p_x, p_y)$. Additional state constraints may be included, such as internal state bounds or environmental conditions, such as no-fly zones, though this is not explored further in this work. Note that the cost objective (3), which utilizes functions ϕ and ϕ_G (Section VII), depends only on $\widehat{\mathbf{x}}$.[a] As such, the stochastic elements of this optimization manifest themselves only in the final chance constraint (9); Section V details the implementation of this chance constraint, yielding a deterministic optimization.

In practice, the optimization (3) is solved repeatedly during the descent, with $\mathbf{x}_I$ and $\mathbf{w}_I$ being set to the most recent state and wind measurements, respectively, at current time t_I. The algorithm developed in this work functions identically regardless of the extent of planning that previously took place. For example, it is not required to be preceded by a guidance phase which places the parafoil in a favorable initial state. As it result, the framework developed in this paper can be incorporated into a variety of planning architectures.

III.A. Parafoil Model

The parafoil is modeled as a Dubins vehicle[15] descending at a rate fixed by atmospheric conditions subject to updrafts/downdrafts, with the input-to-heading-rate mapping governed by complex lag dynamics. The lack of altitude control, coupled with a large minimum turning radius and slow turning rate, necessitates significant advance planning for precision guidance and landing. This is exacerbated by the presence of heavy winds, which can lead to loss of goal reachability or terrain collisions if not properly anticipated.

[a] In practice, the t_f used in this objective is based on which the deterministic state $\widehat{\mathbf{x}}$ is expected to intersect with the terrain, acknowledging that, even though vertical mean wind $\overline{w}_z$ is incorporated, wind variation may affect the true landing time.

The parafoil velocity $v(p_z)$ is assumed to be a function of the vehicle altitude p_z, via[16]

$$v(p_z) = v_0 e^{p_z/2\tau_z}, \tag{10}$$

where $\tau_z = 10^4$ m, and v_0 is the nominal vehicle velocity at sea level. In this work, we adopt the 10,000-pound Dragonfly parafoil used by Carter et al.[17], with $v_0 = 17.8$ m/s and lift-to-drag ratio $L_D = 2.8$. The heading rate of the parafoil is modeled as a second-order approximation of the canopy Dutch roll mode; our specific model selects random values of the parameter ranges as specified by Carter et al.[17]. As in that work, a first-order lag is used to model the differential toggle control input mechanism, while the controller is a PID with feedforward gains tuned to achieve comparable performance[17]. In total, this yields a 5th order state $\mathbf{s}$ and dynamics (A, B, C, D), augmented to the state vector $\mathbf{x}$ and dynamics (1) respectively. The control input is a scalar, $\mathbf{u} \equiv u \equiv \dot{\psi}_d$, representing the desired heading rate, subject to the symmetric bounds $\mathcal{U} = \{u \mid |u| \leq \ omega_{\max}\}$.

The overall parafoil dynamics (1) thus take the form

$$\dot{p}_x = v(p_z)\cos\psi + w_x, \tag{11}$$

$$\dot{p}_y = v(p_z)\sin\psi + w_y, \tag{12}$$

$$\dot{p}_z = \frac{-v(p_z)}{L_D} + w_z, \tag{13}$$

$$\dot{\mathbf{s}} = A\mathbf{s} + Bu, \tag{14}$$

$$\dot{\psi} = \mathrm{sat}(C\mathbf{s} + Du, -\omega_{\max}, \omega_{\max}), \tag{15}$$

where the saturation function $\mathrm{sat}(a, b, c)$ bounds a between b and c and is linear in between, and $\omega_{\max} = \pi/15 = 0.2094$ rad/s[18] (such that the vehicle's minimum turning radius is $R_{\min} = v_0/\omega_{\max} = 85$ m). Note that in this formulation, only the position states $\mathbf{p}$ are affected by the wind disturbance uncertainty $\mathbf{w}$, including possible effects on altitude p_z by updrafts and downdrafts via w_z. Within the RRT planning framework, which operates in discrete time, the optimization (3)) is discretized via the timestep $dt = 0.1$ s.

IV. Real-Time Wind Modeling

The wind model detailed in this section is utilized by the planner to improve prediction accuracy and robustness for the parafoil terminal guidance problem. The development of this wind model is based on satisfying three main objectives. First, the wind model should improve predictability of future wind effect, compared to a zero-wind assumption (*e.g.,* $\mathbf{w} = 0$). Improved predictability, especially in scenarios where there is significant prevailing wind, can mitigate the amount of replanning required and improve the quality of solutions provided by the RRT. Second, the wind model should capture the uncertainty of future wind effects, given the planner knowledge of a distribution over possible outcomes of a planned trajectory. Characterizing and utilizing such an uncertainty distribution in a probabilistic framework (Section V) facilitates planner robustness to terrain obstacles. Finally, the wind model should be kept as simple as possible, to maintain real-time planner operation and discourage data overfitting.

Given the importance of wind modeling in many engineering applications, there has been considerable work on developing wind prediction and estimation models, including the case of online estimation. Delahaye and Puechmorel[19] utilize linear and unscented Kalman filters to estimate wind vectors from radar data, but requires either accurate airspeed measurement or excitation via the vehicle trajectory. In contrast, the approach of Petrich and Subbarao[20] requires only onboard sensing to achieve real-time wind sensing, without trajectory excitation. However, such approaches cannot provide forward prediction for UAV systems, in either space or time. Hunt and Nason[21] use wavelets to generate time series models for wind prediction, but on a timescale of days, rather than the minutes during which a parafoil descends. Conversely, the approach of Jiang et al.[22], which uses adaptive Gaussian processes, requires months of training data for accurate prediction, an assumption which may not be satisfied by typical parafoil drop zones. None of these modeling approaches address the wind prediction problem over the short timescales and limited datasets inherent in parafoil precision guidance. This section fits an uncertainty model to the wind which can be incorporated into the planner to enforce robustness.

This approach includes online learning to determine, in real time, the class of wind scenario being experienced by the parafoil. Such a class determines the parameters of the variational estimate; the variational model associated with each class is tuned to capture the amount of uncertainty typical to wind profiles

within its class. In this manner, the level of conservatism in the planner can be adjusted online to reflect the wind conditions being observed. Draper Laboratories has released 194 altitude-dependent wind profiles from parafoil drops[23], collected using the sensor configuration and estimation procedure outlined in work by Carter et al.[6]. These profiles are used as training data for various components of the wind model, as discussed below.

IV.A. Model Form

The wind model is assumed to take the form (2) (Section III), written in discrete-time form as

$$\mathbf{w}_{t+1} \quad = \quad f_w(\mathbf{w}_t, \overline{\mathbf{w}}, \mathbf{v}_t), \quad \mathbf{w}_0 = \mathbf{w}_I, \tag{16}$$

where timestep 0 occurs at system time t_I (Section III). The 3-D wind estimate at timestep t, $\mathbf{w}_t$, is assumed to take the form

$$\mathbf{w}_t = \overline{\mathbf{w}} + \delta\mathbf{w}_t, \tag{17}$$

comprising the sum of a 3-D persistent estimate $\overline{\mathbf{w}}$ and a 2-D variational estimate $\delta\mathbf{w}_t$.

The persistent estimate $\overline{\mathbf{w}}$ reflects the notion that there typically exists a prevailing wind which acts on the parafoil throughout the entire mission, and must be accounted for during the state prediction. It is represented using a finite impulse response filter,

$$\overline{\mathbf{w}} = \frac{1}{m} \sum_{i=t_I-m-1}^{t_I} \mathbf{w}_i, \tag{18}$$

where m is the filter window width.

The filter width m is chosen by optimizing a metric representing the predictive accuracy of the filter[24]. Consider propagating the parafoil dynamics from some initial state $\mathbf{p}_I$ to ground, assuming zero input ($u \equiv 0$), zero lag dynamics ($\mathbf{s} \equiv 0$), and flat terrain ($T(p_x, p_y) \equiv 0$). Assume that previous observations of the wind profile have been observed prior to the parafoil reaching $\mathbf{p}_I$, such that the filter (18) could be applied. The dynamics are propagated from the same initial state and observations according to each available 3D wind profile, including updrafts and downdrafts[23]. For the wth wind profile, three possible landing positions are of interest:

- The landing position under true wind, $\mathbf{p}_T^{(w)}$;
- The landing position under zero wind, $\mathbf{p}_0^{(w)}$; and
- The landing position under constant wind as predicted by (18) with width m, $\mathbf{p}_m^{(w)}$.

Define the quantity

$$\delta d_m^{(w)} = \left\| \mathbf{p}_0^{(w)} - \mathbf{p}_T^{(w)} \right\| - \left\| \mathbf{p}_m^{(w)} - \mathbf{p}_T^{(w)} \right\|, \tag{19}$$

which takes the difference in accuracy between the zero-wind model and the impulse-filter model in predicting the true landing position. For those wind profiles in which prediction accuracy degrades with the impulse-filter model, $\delta d_m^{(w)} < 0$; denote $D_m = \{\delta d_m^{(w)} \mid \delta d_m^{(w)} < 0\}$. The filter width is then chosen as

$$m = \arg\max_{m>0} \left\{ \min(D_m) + \beta \ \text{mean}(D_m) - \lambda m \right\}, \tag{20}$$

where $\beta, \lambda > 0$; in this work, $\beta = 2$ and $\lambda = 1$. This cost function includes terms for the worst-case and average-case accuracy in D_m, as well as a regularization term[24].

The variational estimate $\delta\mathbf{w}_t$ is represented as multi-modal linear dynamics subject to Gaussian noise,

$$\delta\mathbf{w}_{t+1} = (I + dtA_c)\delta\mathbf{w}_t + dtB_c\mathbf{v}_t, \quad c \in \{1, \ldots, N_C\}, \tag{21}$$

where N_C is the number of modes/classifications used, A_c and B_c are the tuned matrices used for the cth wind classification, corresponding to the continuous-time state-space system (A_c, B_c) (Section IV.B), and

$\mathbf{v}_t \in \mathcal{N}(0,1)$, *i.e.,* zero-mean, unit-variance Gaussian noise. This colored noise process is chosen to reflect the idea that, while wind at lower altitudes is correlated with the wind measured at the current altitude, this correlation degrades with further separation.

By substituting (17) into (21), the wind model function (16) can be written as

$$f_w(\mathbf{w}_t, \overline{\mathbf{w}}, \mathbf{v}_t) = \overline{\mathbf{w}} + (I + dtA_c)(\mathbf{w}_t - \overline{\mathbf{w}}) + dtB_c\mathbf{v}_t, \quad c \in \{1, \ldots, N_C\}. \tag{22}$$

The remaining questions, then, are (1) how to identify the number of classifications N_C and corresponding wind model dynamics (A_c, B_c), $c \in \{1, \ldots, N_C\}$, and (2) how to select the appropriate classification online. These topics are discussed next.

IV.B. Wind Model Training

The wind profiles used for training[23] list each component of the measured wind velocity vector as a function of altitude, *i.e.,* $\{w_x(p_z), w_y(p_z), w_z(p_z)\}$. This can pose problems for clustering and classification algorithms, which are typically designed to operate on observations, rather than functions. By using feature selection, the dimension of the system model can be reduced, allowing for the use of many efficient clustering and classification schemes[25]. Denote $\rho = \sqrt{w_x^2 + w_y^2 + w_z^2}$ and $\theta = \text{atan2}(w_y, w_x)$; for this work, we use the feature vector

$$\Phi = \left[\begin{array}{ccccccc} \text{mean}(\rho) & \max(\rho) & \text{mean}\left(\frac{d\rho}{dp_z}\right) & \max\left(\frac{d\rho}{dp_z}\right) & \text{mean}\left(\frac{d\theta}{dp_z}\right) & \max\left(\frac{d\theta}{dp_z}\right) \end{array} \right]. \tag{23}$$

This feature vector takes the mean and maximum value, over all data points in a wind profile, of three quantities: the wind magnitude, the rate of change of wind magnitude, and the rate of change of wind direction. Collectively, these features were chosen to represent the power and variability inherent in each profile.

The objective now is to use the feature-representation of each wind profile (23), denoted for the wth wind profile as x_w, to classify the N_W wind profiles into N_C classes. We represent each possible disjoint partition of these profiles as $\mathbf{S} = \{S_1, S_2, \ldots, S_{N_C}\}$. The partition is chosen so as to minimize the squared sum of the distance from the mean within each cluster, μ_i, such that

$$\mathbf{S} = \arg\min_{\mathbf{S}} \left(\sum_{i=1}^{N_C} \sum_{x_w \in S_i} \|x_w - \mu_i\|^2 \right) + \lambda k, \tag{24}$$

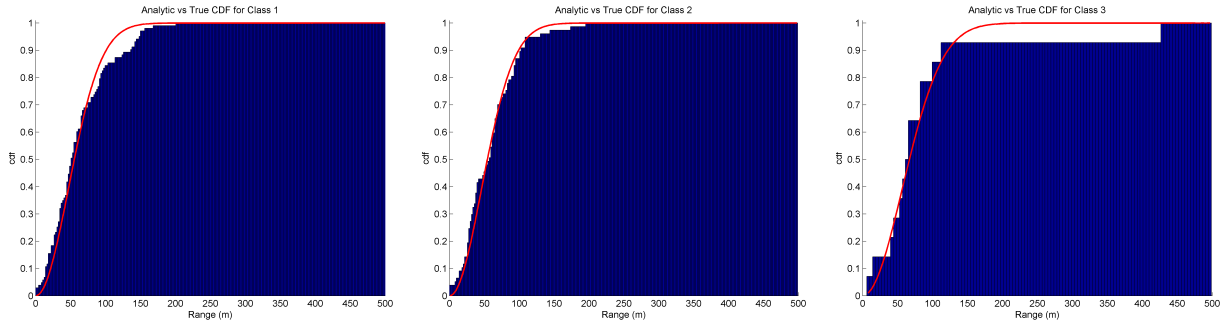
where a regularization term is included. This algorithm might be conventionally solved using the k-means algorithm[26], which assumes the number of clusters N_C is known *a priori*. Here, this optimization is solved using the DP-means algorithm[27], which extends k-means such the appropriate number of clusters N_C can be incrementally identified without *a priori* knowledge. During the DP-means assignment step, if an observation is further than λ from the nearest cluster center, a new cluster is added, with the center defined as the observation x_w which created it. Using the aforementioned Draper wind profiles[23], three distinct classes were identified, representing successively more variational (and hence more conservative) models for the evolution of the wind distribution.

For each classification, the variational wind model dynamics (A_c, B_c) are constructed by matching the analytic uncertainty distribution to the empirical distribution identified from the wind profiles. We construct the variational wind model to be two-dimensional, *i.e.,* $\delta w_z = 0$, though incorporating vertical components will be considered in future work. (Observed updrafts and downdrafts are still incorporated in the wind model, via the mean wind $\overline{\mathbf{w}}$.) We further assume that δw_x and δw_y are independent and symmetric. As a result, the variational wind model dynamics $A_c \in \mathbb{R}^{3\times3}$ and $B_c \in \mathbb{R}^{3\times2}$ can alternatively be written as

$$A_c = \alpha_c \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_c = \beta_c \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \tag{25}$$

where $\alpha_c, \beta_c \in \mathbb{R}$.

For each wind profile in the cluster, compute the miss distances $\left\| \mathbf{p}_m^{(w)} - \mathbf{p}_T^{(w)} \right\|$ as used in (19); let d_i denote the ith largest miss distance, and n_i the fraction of profiles with a miss distance less than or equal



(a) Class 1 (least variation) (b) Class 2 (moderate variation) (c) Class 3 (most variation)

Figure 1. Comparison of miss distance CDFs for the wind profiles (blue) and tuned wind model (red) for each wind class.

to d_i. These characteristics of the cumulative density function (CDF) will be compared against the analytic wind model for tuning.

Represent the wind model (21) in continuous-time form as

$$\dot{\delta\mathbf{w}} = A_c\delta\mathbf{w} + B_c\mathbf{v}, \tag{26}$$

and define the position variation $\delta\mathbf{p} = \begin{bmatrix} p_x - E[p_x] & p_y - E[p_y] & p_z - E[p_z] \end{bmatrix}^T$. We can then construct the augmented dynamics

$$\begin{bmatrix} \dot{\delta\mathbf{p}} \\ \dot{\delta\mathbf{w}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0_3 & I_3 \\ \mathbf{0}_3 & A_c \end{bmatrix}}_{A_{\text{aug}}} \begin{bmatrix} \delta\mathbf{p} \\ \delta\mathbf{w} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 2} \\ B_c \end{bmatrix}}_{B_{\text{aug}}} \mathbf{v}. \tag{27}$$

The covariance at impact time t_F, $\Sigma_F \equiv \Sigma(t_F)$, can then be propagated forward using the dynamics

$$\dot{\Sigma} = A_{\text{aug}}\Sigma + \Sigma A_{\text{aug}}^T + B_{\text{aug}}B_{\text{aug}}^T, \tag{28}$$

$$\Sigma(t_I) = \begin{bmatrix} \Sigma_p & 0_3 \\ 0_3 & \Sigma_w \end{bmatrix}, \tag{29}$$

where Σ_p and Σ_w are the initial covariance for the position and wind, respectively. Either quantity is zero if perfectly known, but may be non-zero if, for example, an estimator is providing data to the system.

For comparison with the empirical wind profile data, the covariance in lateral position is isolated via

$$\Sigma_F' = C_T\Sigma_F C_T^T, \tag{30}$$

$$C_T = \begin{bmatrix} I_2 & 0_{2\times 4} \end{bmatrix}. \tag{31}$$

Given the independence and symmetry assumptions on δw_x and δw_y, Σ_f' can be written as $\Sigma_f' = \sigma^2 I_2$, where $\sigma > 0$ is a scalar. This can be thought of as a distribution on landing miss distances, represented as a χ distribution with standard deviation σ; denote its CDF for dynamics (A_c, B_c) as $\chi(x, A_c, B_c)$. This CDF can be matched directly to the empirical wind profile CDF with characteristics (d_i, n_i) as described above. For the cth wind classification, the dynamics (A_c, B_c) are identified by minimizing the root mean square error between the two CDFs,

$$(A_c, B_c) = \underset{(A,B)}{\text{argmin}} \sum_{i\in S_c} \left(n_i - \chi(d_i, A, B)\right)^2, \tag{32}$$

implemented in MATLAB via `fminunc`.

Figure 1 compares the analytic and "true" (empirical) CDFs for each of the classes identified from the Draper wind profiles. Class 1 represents the "optimistic" class, where the wind is believed to have low power

and/or variability. Class 2 represents a "moderate" class, in which the wind is believed to have more of an effect relative to Class 1. Finally, Class 3 represents the "pessimistic" class, in which the wind is believed to have the most significant effect on the parafoil. Section VIII demonstrates that by adjusting the level of conservatism online, the multi-classification wind model achieves better performance (in terms of miss distance) than using any one wind class alone.

IV.C. Online Classification Selection

In order to utilize the varying levels of uncertainty associated with the C classifications determined by DP-means in Section IV.B, the planner must have a methodology for using the observed wind estimates to assign the wind that is being experienced by the vehicle to a classification, *i.e.,* statistical classification. This is accomplished using support vector machines (SVM) [28]. An SVM binary inclusion classifier is generated for each of the C classes, which can be used to identify if the wind estimates being received are a member of a particular class. It is possible for multiple classifier to result in affirmative classifications. If this is the case, the algorithm chooses the class which is the most conservative, *i.e.,* has the fastest-growing uncertainty [24]. The trained SVM classifier then classifies the observed wind profile online.

V. Analytic Uncertainty Sampling and Robust Planning

This section presents a novel framework for modeling future uncertainty in trajectory predictions, such that robustness to possible future variation in disturbances can be achieved. Recall that in the formulation of the parafoil terminal guidance problem (Section III), satisfaction of state constraints is specified via the chance constraint (9), restated below:

$$\mathbb{P}_{\mathbf{v}}\left(\mathbf{x} \in \mathcal{X}\right) \geq p_{\text{safe}}.$$

where p_{safe} is a risk parameter specified by the user. This represents a minimum likelihood that all state constraints, here consisting of the terrain surface $p_z \geq T(p_x, p_y)$, be satisfied with a minimum probability of p_{safe} along each trajectory. In CC-RRT, the chance constraint (9) must be satisfied at each *timestep*, rather than over the entire path, and is converted to a tightened, deterministic constraint [1]. Under the assumptions of linear dynamics and Gaussian noise, these tightened constraints are shown to guarantee probabilistic feasibility to polyhedral constraints at each timestep. Whereas the traditional RRT algorithm grows a tree of states which are known to be feasible, CC-RRT grows a tree of state distributions which are known to satisfy the lower bound on feasibility p_{safe}. Further, due to RRT's trajectory-wise constraint checking, a risk bound can be explicitly computed online against each uncertainty source at each timestep.

In previous work, removing the assumptions of linear dynamics and/or Gaussian dynamics requires either linearizing the dynamics at each timestep or performing full time-series simulations of particle-based uncertainty approximations [5]. While such formulations allow the evaluation of path-wise feasible, they can be computationally intensive, cannot be simulated *a priori* (*i.e.,* independently of the individual RRT trajectories), and can only approximate theoretical guarantees for probabilistic feasibility. It is shown here that, though the parafoil dynamics are nonlinear, the effect of the wind uncertainty is linear-Gaussian. As a result, uncertainty distributions can be derived analytically *a priori* at all future timesteps, and theoretical guarantees maintained – but subject to polyhedral state constraints. In subsequent developments, we choose to take equi-spaced samples of the uncertainty distributions, such that they can be checked for collision against arbitrary, possibly aggressive terrain map functions of the form $T(p_x, p_y)$ (Section III). Though probabilistic guarantees are approximated statistically, uncertainty samples are obtained without dynamic state propagation. As a result, this variant of CC-RRT is more efficient, and better representative of uncertainty distributions, than previous particle-based formulations [5]. Furthermore, the use of sampling allows for path-wise probabilistic feasibility to be evaluated.

The wind uncertainty is assumed to follow the multi-modal wind model introduced in Section III and detailed in Section IV, in particular (22). Each wind classification represents the estimated possible variation in the wind.

V.A. Analytic Uncertainty Derivation

Consider the dynamics for the vehicle state (11)-(15), written here in discrete-time form as

$$\begin{aligned} p_{x,t+1} &= p_{x,t} + dt\left[v(p_{z,t})\cos\psi_t + w_{x,t}\right], && (33)\\ p_{y,t+1} &= p_{y,t} + dt\left[v(p_{z,t})\sin\psi_t + w_{y,t}\right], && (34)\\ p_{z,t+1} &= p_{z,t} + dt\left[v(p_{z,t})\left(-L_D^{-1}\right) + w_{z,t}\right], && (35)\\ \mathbf{s}_{t+1} &= \mathbf{s}_t + dt\left[A\mathbf{s}_t + Bu_t\right], && (36)\\ \psi_{t+1} &= \psi_t + dt\cdot\mathrm{sat}(C\mathbf{s}_t + Du_t, -\omega_{\max}, \omega_{\max}). && (37) \end{aligned}$$

The final two equations (36)-(37) are unaffected by the uncertainty $\mathbf{v}_t$. As in Section IV.B, take the variation $\delta\mathbf{p} = \begin{bmatrix} p_x - E[p_x] & p_y - E[p_y] & p_z - E[p_z] \end{bmatrix}^T$. Recalling that $\delta w_z = 0$ (and thus $p_z = E[p_z]$), combining (33)-(35) with (26) in discrete-time form yields

$$\begin{aligned} \delta p_{x,t+1} &= \delta p_{x,t} + dt\left(\delta w_{x,t}\right), && (38)\\ \delta p_{y,t+1} &= \delta p_{y,t} + dt\left(\delta w_{y,t}\right), && (39)\\ \delta p_{z,t+1} &= \delta p_{z,t}, && (40)\\ \delta w_{x,t+1} &= (I + dt\alpha_c)\delta w_{x,t} + dt\beta_c v_{x,t}, && (41)\\ \delta w_{y,t+1} &= (I + dt\alpha_c)\delta w_{y,t} + dt\beta_c v_{y,t}; && (42) \end{aligned}$$

note that (40) has decoupled from the other variational dynamics. By defining the variation state vector $\delta\mathbf{x}_t = \begin{bmatrix} \delta p_{x,t} & \delta p_{y,t} & \delta w_{x,t} & \delta w_{y,t} \end{bmatrix}^T$, the variation dynamics (38)-(39),(41)-(42) can be clearly written in the linear form

$$\begin{aligned} \delta\mathbf{x}_{t+1} &= \mathbb{A}\delta\mathbf{x}_t + \mathbb{B}\mathbf{v}_t, && (43)\\ \mathbb{A} &= \begin{bmatrix} 1 & dt & 0 & 0\\ 0 & 1+dt\alpha_c & 0 & 0\\ 0 & 0 & 1 & dt\\ 0 & 0 & 0 & 1+dt\alpha_c \end{bmatrix}, \quad \mathbb{B} = \begin{bmatrix} 0 & 0\\ dt\beta_c & 0\\ 0 & 0\\ 0 & dt\beta_c \end{bmatrix}. && (44) \end{aligned}$$

Because this linear system is driven by Gaussian noise, all future state distributions take the form $\mathbf{x}_t \in \mathcal{N}(\widehat{\mathbf{x}}_t, P_t)$, *i.e.,* Gaussian with mean $\widehat{\mathbf{x}}_t$ and covariance $P_t \equiv E[\delta\mathbf{x}_t\delta\mathbf{x}_t^T]$. The mean can be computed using the disturbance-free dynamics (50)-(51), while the covariance can be represented either implicitly as

$$P_{t+1} = \mathbb{A}P_t\mathbb{A}^T + \mathbb{B}\mathbb{B}^T \tag{45}$$

or explicitly as

$$P_t = \mathbb{A}^t P_0 (\mathbb{A}^T)^t + \sum_{k=0}^{t-1} \mathbb{A}^{t-k-1}\mathbb{B}\mathbb{B}^T(\mathbb{A}^T)^{t-k-1}, \tag{46}$$

where P_0 depends on whether the initial state and wind measurement are perfectly known or not (Section IV-IV.B). Note that, as in the conventional CC-RRT framework, (45)/(46) can be computed *a priori*, independently of any individual trajectory simulated[1]. Finally, the covariance of the position states can be isolated via the transformation

$$Q_t = C_T P_t C_T^T, \quad C_T = \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{47}$$

V.B. Covariance Sampling Generation

In order to efficiently check path-wise feasibility of the terrain constraint $p_z \geq T(p_x, p_y)$, the likelihood of collision with terrain is approximated by generating equi-spaced samples at specified levels of the uncertainty distribution at each prospective trajectory node. Sampling the distribution in this way allows for coverage of the uncertainty space with relatively few samples, as well as removing the need to dynamically propagate each sample, significantly reducing computation time. The discretization level of the samples N_s, as well the the minimum probability of feasibility p_{safe}, can both be specified by the user to allow for tunable levels of robustness.

The covariance samples are placed at a series of equi-spaced points along uncertainty ellipses. The covariance matrix Q_t describes a contour of equal probability of points $\delta\mathbf{p}_t = \begin{bmatrix} \delta p_{x,t} & \delta p_{y,t} \end{bmatrix}^T$, relative to the nominally propagated trajectory $\widehat{\mathbf{x}}_t$ by the conic relaxation $\delta\mathbf{p}_t^T Q_t^{-1} \delta\mathbf{p}_t = 1$. Denote the elements of Q_t as

$$Q_t = \begin{bmatrix} \sigma_{x,t}^2 & \sigma_{xy,t} \\ \sigma_{xy,t} & \sigma_{y,t}^2 \end{bmatrix}, \tag{48}$$

with eigenvalues σ_a^2 and σ_b^2 ($\sigma_a > \sigma_b$). The principle axis of the uncertainty ellipse has angle $\theta_P = \frac{1}{2}\tan^{-1}\left(\frac{2\sigma_{xy,t}}{\sigma_{x,t}^2 - \sigma_{y,t}^2}\right)$. The N_s samples are spaced at equal angular intervals relative to this principle axis; the jth sample $\delta\mathbf{p}_t^{(j)}$ has angle $\theta_j = \frac{2\pi}{N_s - 1} j$, relative to θ'. The samples are then placed at

$$\delta\mathbf{p}_t^{(j)} = \sigma R_Q \mathbb{R}(-\theta') \begin{bmatrix} \cos(\theta_j - \theta') \\ \sin(\theta_j - \theta') \end{bmatrix}, \tag{49}$$

$$R_Q = \frac{\sigma_a \sigma_b}{\sqrt{(\sigma_b \cos(\theta_j - \theta'))^2 + (\sigma_a \sin(\theta_j - \theta'))^2}},$$

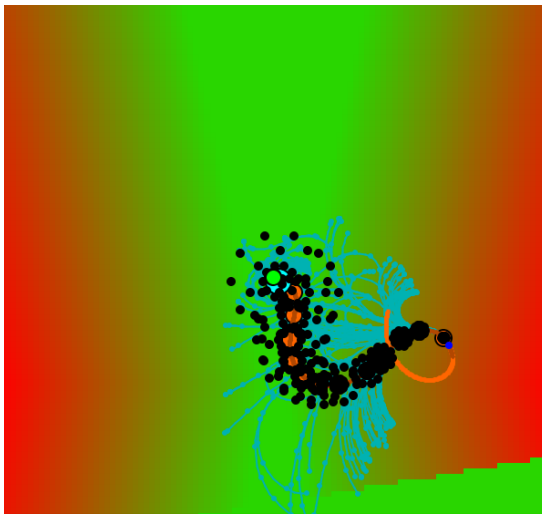


Figure 2. Screenshot of a simulation in progress, showing covariance samples. The planner constructs a tree of feasible trajectories (teal), and executes a path (orange) which guides the parafoil (blue) to land on the terrain (background; changes from green to red with increasing altitude) at the goal (green circle). Covariance samples for each path node are shown in black, for $\sigma = 1.5$.

where $\sigma > 0$ is the covariance scale factor and $\mathbb{R}(\alpha)$ is the 2D rotation matrix for a counterclockwise rotation of α. The parameter σ represents the number of standard deviations within the uncertainty ellipse; empirical analysis has shown a value of about $\sigma = 1.75$, used in subsequent work, to be a reasonable approximation of the uncertainty environment in practice[24]. Note that using this approach, nodes can be generated at a rate 3-6 times faster than particle CC-RRT[24].

Samples may also be distributed across multiple values of σ. In subsequent results, two rings of covariance samples are used, one at σ and another at 0.4σ, a ratio empirically found to work well. Figure 2 shows an example of a parafoil CC-RRT tree generated for the valley terrain (Section VIII.A) for $\sigma = 1.5$. Covariance samples are shown for the path currently selected for execution.

VI. Parafoil CC-RRT Path Planning

This section presents the parafoil CC-RRT algorithm, for robust motion planning using the previously-developed wind model (Section IV) and covariance samples (Section V). The core algorithm upon which the parafoil CC-RRT framework builds is the rapidly-exploring random tree (RRT)[2], which incrementally constructs a tree of dynamically feasible trajectories from the current state. While many algorithms are available for motion planner problems of this nature[29,30], an RRT-based approach is particularly well-suited to this application. The lack of controllability in the altitude state p_z largely precludes the use of graph-based approaches. An RRT can quickly identify and refine feasible solutions online within the 9-dimensional configuration space (3 for position, 1 for heading, 5 for lag dynamics), without discretizing the solution space. Furthermore, its incremental construction and constraint checking allows it to scale with both problem complexity and available computational resources.

Let the current time step be t; the tree is rooted at the current vehicle state, $\mathbf{x}_t$, and the most recent wind measurement is denoted as $\mathbf{w}_t$. Each simulated trajectory within the tree uses the nominal dynamics (5) and wind model (7), written as

$$\widehat{\mathbf{x}}_{t+k+1|t} = f(\widehat{\mathbf{x}}_{t+k|t}, \mathbf{u}_{t+k|t}, \widehat{\mathbf{w}}_{t+k|t}), \quad \widehat{\mathbf{x}}_{t|t} = \mathbf{x}_t, \tag{50}$$

$$\widehat{\mathbf{w}}_{t+k+1|t} = f_w(\widehat{\mathbf{w}}_{t+k|t}, \overline{\mathbf{w}}_t, 0), \quad \widehat{\mathbf{w}}_{t|t} = \mathbf{w}_t, \tag{51}$$

where the subscript $(\cdot)_{\alpha|\beta}$ denotes simulation time step α and execution time step β ($< \alpha$).

Whereas the nominal RRT algorithm grows a tree of states which are known to be feasible, with any uncertainties assumed to maintain nominal values (in this case, $\mathbf{v} \equiv 0$), CC-RRT grows a tree of *state*

Algorithm 1 Parafoil CC-RRT: Tree Growth

1: Take a sample $\mathbf{x}_{\text{samp}}$ from the environment
2: Identify the m nearest nodes using heuristics
3: **for** $m \leq M$ sorted nearest nodes **do**
4: $N_{near} \leftarrow$ current node, $(\widehat{\mathbf{x}}_{t+k|t}, \widehat{\mathbf{w}}_{t+k|t}) \leftarrow$ final vehicle and wind state of N_{near}
5: $p_{\text{collide}} \leftarrow 0$
6: **while** $p_{\text{collide}} < p_{\text{safe}}$ **and** (54) true **and** $\widehat{\mathbf{x}}_{t+k|t}$ has not reached $\mathbf{x}_{\text{samp}}$ **do**
7: Select input $\mathbf{u}_{t+k|t} \in \mathcal{U}$
8: Simulate $(\widehat{\mathbf{x}}_{t+k+1|t}, \widehat{\mathbf{w}}_{t+k+1|t})$ using (50),(51)
9: Create intermediate nodes as appropriate
10: Compute/retrieve $P_{t+k+1|t}$ using (45)
11: Compute p_{collide} using (49),(53)
12: $k \leftarrow k+1$
13: **for** each identified node N **do**
14: Add N to tree
15: Try connecting N to $\mathbf{x}_G$

distributions, which are checked for feasibility by satisfying an upper bound on probability of collision [1] at each timestep. The parafoil CC-RRT algorithm similarly generates a tree of uncertainty distributions around trajectories, rather than just the nominal trajectories itself. Instead of evaluating collision probabilities using the full distributions, however, we sample these analytic uncertainty distributions, using the novel sampling technique introduced in Section V.A. This allows path-wise probabilistic feasibility checks to be enforced against arbitrary terrain maps.

As shown in Section V.A, the uncertainty at each prediction timestep $t+k$, relative to execution timestep t, can be represented as a Gaussian state distribution

$$\mathbf{x}_{t+k|t} \sim \mathcal{N}(\widehat{\mathbf{x}}_{t+k|t}, P_{t+k|t}). \tag{52}$$

The mean $\widehat{\mathbf{x}}_{t+k|t}$, with position $(\widehat{p}_{x,t+k|t}, \widehat{p}_{y,t+k|t}, \widehat{p}_{z,t+k|t})$, can be simulated using the disturbance-free dynamics (50)-(51), while the covariance $P_{t+k|t}$ can be computed offline via (45). Using (49), the samples are placed at offsets $\delta\mathbf{p}^{(j)}_{t+k|t} = \left(\delta p^{(j)}_{x,t+k|t}, \delta p^{(j)}_{y,t+k|t}, 0\right), \quad j \in \{1, \ldots, N_s\}$.

Probabilistic feasibility is checked statistically, by seeing whether the fraction of covariance samples for a given trajectory point $\mathbf{x}_{t+k|t}$ that have intersected with the terrain exceeds $1 - p_{\text{safe}}$, at this or any previous simulation step. Given the terrain map $T(p_x, p_y)$, the probability of terrain collision p_{collide} is estimated to be

$$p_{\text{collide}} = \frac{1}{N_s} \sum_{j=1}^{N_s} \mathbb{I}\left[\bigwedge_{i=0}^{k} \widehat{p}_{z,t+i|t} \leq T\left(\widehat{p}_{x,t+i|t} + \delta p^{(j)}_{x,t+i|t}, \widehat{p}_{y,t+i|t} + \delta p^{(j)}_{y,t+i|t}\right)\right], \tag{53}$$

where $\mathbb{I}[\cdot]$ is the indicator function, *i.e.,* 1 if the contained statement is true and 0 otherwise, and $\bigwedge$ represents a conjunction of the indexed constraints. If $p_{\text{collide}} > 1 - p_{\text{safe}}$, then the trajectory is considered to have landed.

In addition to the uncertainty-based feasibility check, a trajectory is also considered to have landed if the nominal trajectory intersects the terrain. In other words,

$$\widehat{p}_{z,t+k|t} > T\left(\widehat{p}_{x,t+k|t}, \widehat{p}_{y,t+k|t}\right) \tag{54}$$

is added as an additional state constraint.

As with the conventional RRT algorithm, the parafoil CC-RRT algorithm consists of two core components: a "tree growth" step that incrementally constructs the tree, and an "execution" step that selects paths from the tree for parafoil execution with some frequency. The tree growth step is detailed in Algorithm 1.

Each time Algorithm 1 is called, first a sample state is taken from the environment (line 1; Section VI.A), and the nodes nearest to this sample (Section VI.B), in terms of some heuristic(s), are identified as candidates for tree expansion (line 2). An attempt is made to form a connection from the nearest node(s) to the sample by generating a probabilistically feasible trajectory between them (lines 6–12). This trajectory

Algorithm 2 Parafoil CC-RRT: Execution

Require: Initial vehicle state $\mathbf{x}_I$, initial wind measurement $\mathbf{w}_I$, goal state $\mathbf{x}_G$

1: $t \leftarrow 0$, $\mathbf{x}_t \leftarrow \mathbf{x}_I$, $\mathbf{w}_t \leftarrow \mathbf{w}_I$
2: Initialize tree with node at $\mathbf{x}_t$
3: **while** $\mathbf{x}_t \notin \mathcal{X}$ **do**
4: Update current vehicle state $\mathbf{x}_t$, wind measurement $\mathbf{w}_t$, and mean wind estimate $\overline{\mathbf{w}}_t$
5: Propagate mean state $\mathbf{x}_t$ by the computation time $\rightarrow \mathbf{x}_{t+\Delta t}$ using (5),(7)
6: Update tree feasibility and costs using (55)
7: **while** time remaining for this time step **do**
8: Expand the tree by adding nodes (Algorithm 1)
9: Use cost (55) to identify lowest-cost path $\{N_{\text{root}}, \ldots, N_{\text{target}}\}$
10: **if** at least one path exists **then**
11: Apply best path
12: **else**
13: Apply "safe" action
14: $t \leftarrow t + \Delta t$
15: Mark vehicle as landed at $\mathbf{x}_t$

is incrementally simulated by selecting some feasible input (line 7), then simulating the disturbance-free vehicle/wind dynamics (line 7). This input may be selected at the user's discretion, such as through random sampling or a closed-loop controller[31], but should guide the state distribution toward the sample; here, we use a circular-arc-based reference model to construct trajectories to samples (Section -VI.C). Note that the propagated vehicle state and wind state are represented as $\widehat{\mathbf{x}}_{t+k|t}$ and $\widehat{\mathbf{w}}_{t+k|t}$, reflecting that parafoil CC-RRT is propagating the mean values of the uncertainty distributions, rather than the states themselves. Intermediate nodes may be occasionally inserted during the trajectory generation process (line 9), to encourage future expansion.

To check feasibility, the algorithm computes or retrieves (if previously computed offline) the covariance $P_{t+k+1|t}$ at each simulation step (line 10), then computes p_{collide} based on the covariance samples that are feasible up to that simulation step (line 11). This is used to check probabilistic feasibility via $p_{\text{collide}} < p_{\text{safe}}$ along with (54) (line 6). Trajectory simulation continues until the state has either reached the sample or is no longer feasible (line 6). As each node is added to the tree (line 14), an attempt is made to connect it directly to the goal state (line 15). Note that unlike conventional RRT algorithms, every simulated node is added to the tree, even if it does not reach its target (p_x, p_y)-sample before intersecting the terrain. Many of these nodes may still be useful as the tree is updated and new paths are selected for execution, discussed next.

Algorithm 2 details the execution step, which executes some portion of the tree while continuing to grow and update it. The planner updates the current best path to be executed by the system every Δt seconds (here, $\Delta t = 1$ s). After updating the current vehicle state, wind measurement, and mean wind estimate (lines 4–5), the tree is updated via re-propagation (line 6). All nodes are re-checked for probabilistic feasibility; any nodes which have become infeasible are pruned, along with their descendants. Additionally, costs are updated for each node using the cost function (3), written in discrete form as

$$J(N_{\text{target}}) = \phi_F(\mathbf{x}_{t_F|t+\Delta t}, \mathbf{x}_G) + \Delta t \sum_{i=t+\Delta t}^{t_F} \phi(\mathbf{x}_{i|t+\Delta t}, \mathbf{x}_G). \tag{55}$$

After these updates, the tree is repeatedly expanded using Algorithm 1 for the duration of the time step (lines 7–8). Following this tree growth, (55) is used to select the lowest-cost path in the tree for execution (lines 9–11). If no path exists in the tree, some "safe" motion primitive is applied to attempt to keep the vehicle safe (lines 12–13).

In this approach, we have chosen to re-propagate the entire tree for both feasibility and cost, rather than simply re-check the feasibility of the lowest-cost path via "lazy check"[31] without updating costs. In the context of the parafoil terminal guidance problem, where feasibility and cost are both inextricably linked and highly dynamic as a function of the uncertainty, it is useful to update all possible trajectories with changing conditions, in order to achieve reliable performance. While additional computation is required to perform this update, in practice this computation will be balanced over time with the tree size, via the amount of

time spent in lines 7–9 of Algorithm 2. Section VIII demonstrates the effectiveness of the analytic CC-RRT algorithm in improving worst-case planning for parafoil terminal guidance, particularly subject to complex terrain and pathological wind conditions.

VI.A. Sampling Strategy

The sampling strategy used by the planner (Algorithm 1, line 1) is designed specifically for the parafoil guidance problem, and can be broken up into two parts: random sampling, which generates randomly across the entire planning space, and directed sampling, which creates samples geared toward incorporating specifically planning options into the tree.

The random sampling consists of four regions of interest within the 3-D Euclidean planning space; a region is selected for each sample probabilistically based on tunable probabilities.

1. **Goal sampling:** Samples an upper hemisphere centered on the goal $\mathbf{p}_G$. The radius is sampled from a folded-normal distribution, while the azimuth and elevation angles are sampled uniformly.

2. **Local sampling:** Samples a sphere centered on the current parafoil location $\mathbf{p}_t$. The radius is sampled from a folded-normal distribution, while the azimuth and elevation angles are sampled uniformly.

3. **Line sampling:** Samples a sphere in the same manner as local sampling, but centered on a convex combination of the parafoil and goal locations, $\lambda\mathbf{p}_t + (1-\lambda)\mathbf{p}_G,\ \lambda \in [0,1]$, sampled uniformly.

4. **Global sampling:** Samples uniformly in a large, rectangular 3D region around the goal $\mathbf{p}_G$.

Figure 3 shows a "heat map," showing relative sample density, of a typical RRT sampling distribution. In this example, the parafoil is located at $\mathbf{x}_I = (-300, -300, 500)$ m, while the goal is located at $\mathbf{x}_G = (0, 0, 0)$. After connecting new nodes to the tree via random sampling, two kinds of directed sampling may be additionally, and immediately, applied to the newest node. In addition to directing all newly-connected nodes to goal samples (Algorithm 1, line 13), samples are sometimes generated to direct the parafoil to "turn around." Denote the final state of the newest node's trajectory as $\mathbf{p}_{\text{last}} = (p_{xl}, p_{yl}, p_{zl})$ with heading ψ_l, and let $\Delta\psi = \psi_l - \psi_G$ (wrapped). When such a sample is selected, it is placed in position

$$\mathbf{p}_{\text{samp}} = \mathbf{p}_{\text{last}} + \mathbb{R}\left(\psi_l\right)\left(-R_{\min}, \text{sign}(\dot{\psi})R_{\min}, 0\right), \quad (56)$$

placing the sample behind the parafoil in a direction matching the most recent rotation rate. The likelihood of such a sample being chosen is

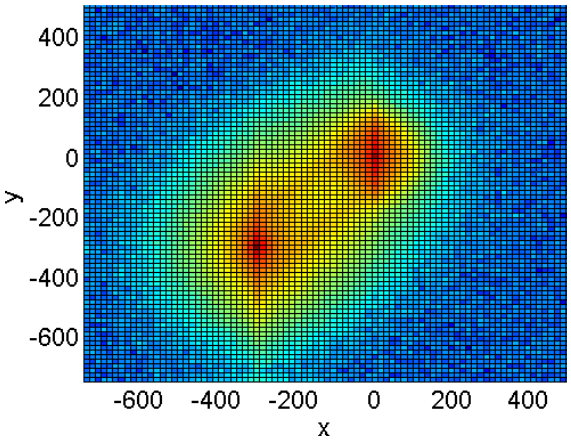


Figure 3. Typical sampling distribution for RRT planner (blue = fewer samples, red = more samples)

$$p_{\text{turn}} \propto \frac{\sqrt{p_{xl}^2 + p_{yl}^2}}{R_{\min}} \left(\frac{\Delta\psi}{\pi}\right)^2. \quad (57)$$

Note that p_{turn} increases with both distance from goal and heading offset, conditions in which it is favorable to bias sampling toward turning the parafoil around.

VI.B. Nearest Node Metric

The nearest node metric (Algorithm 1, line 2) is a distance metric over the state space which prioritizes which nodes in the tree should be connected to a new sample. For this work, the distance metric $d(n)$ used is a variant of Euclidean distance designed to not penalize altitude above the sample, based on propagation

along the glide slope. Suppose the sample has position $\mathbf{p}_s = (p_{xs}, p_{ys}, p_{zs})$, while a node's trajectory ends at $\mathbf{p}_n = (p_{xn}, p_{yn}, p_{zn})$. The distance metric is then

$$d(n) = \|\mathbf{p}_n - [\mathbf{p}_s + (0, 0, \delta)]\|_2, \tag{58}$$

$$\delta = \frac{\sqrt{(p_{xn} - p_{xs})^2 + (p_{yn} - p_{ys})^2}}{L_D}. \tag{59}$$

This effectively "shifts" $\mathbf{p}_s$ upward to the point the parafoil would reach if it glided directly over sample, and computes the Euclidean 2-norm from there. Additionally, this metric has the effect of biasing toward connections when a sample is below the node. This is an advantageous bias, as the parafoil has no method of vertical control, and therefore must descend to all future states.

VI.C. Reference Model

Recall that for the parafoil model, the input is $u = \dot{\psi}_d$, the desired angular rate. A reference model is used to generate a sequence of inputs u (Algorithm 1, line 7) to guide a trajectory from a nearest node, with position $\mathbf{p}_n$ and heading ψ_n, to a sample with position $\mathbf{p}_s$. Since the altitude state p_z is uncontrollable, this is accomplished by generating the 2D circular arc to the sample from the nearest node, while tangent to the heading at the nearest node. Such a circular arc can be followed by the ideal (*i.e.,* no lag dynamics) parafoil in 2D, by applying a constant input $\overline{u}$ for some time duration $\overline{t}$. Resulting trajectories within the RRT framework are thus sequences of piecewise-constant-angular-rate segments. The number of segments and the duration of each segment are not fixed, such that very complex trajectories can still be specified within this reference model. The radius of this circle can be shown to be

$$R = \frac{\delta^2}{2(\delta_y \cos\psi_n - \delta_x \sin\psi_n)}, \tag{60}$$

$$\delta = \sqrt{\delta_x^2 + \delta_y^2}, \tag{61}$$

$$\delta_y = y_s - y_n, \tag{62}$$

$$\delta_x = x_s - x_n; \tag{63}$$

note that the sign of R in (60) encodes the turn direction. This is then used with the velocity model (10) to yield the desired angular rate,

$$\overline{u} = \frac{v(p_{zn})}{R}. \tag{64}$$

The duration of the reference command is computed by computing the arc subtended angle θ:

$$\theta = 2\sin^{-1}\left(\frac{\delta}{|2R|}\right), \tag{65}$$

$$\overline{t} = \theta/\overline{u}. \tag{66}$$

Alternative reference generation models could be considered; options might include Bezier curves[14], B-splines, or piecewise linear (rather than piecewise constant) angular rate commands. Closed-loop propagation may also be used to limit uncertainty growth over time; this will be explored further in future work.

VII. Reachability-based Cost-to-go

One of the advantages of using RRTs, due to their incremental construction, is the capability to select a path which has not yet terminated in planning, use it as the basis for vehicle execution, and complete the path during later planning cycles. Critical to this capability is an informative cost-to-go function, which should provide the ability to compare two prospective paths which have not yet terminated, as well as to compare paths that have not terminated with actual path costs of paths that have terminated in planning. This section develops a cost-to-go formulation (55) which combines the cost at the point of interest, the end of a trajectory within the tree, with a discrete approximation of the reachability set beyond that point. In this manner, the cost-to-go function weighs the value of the system's present state against possible near-term future states, which are heavily constrained by the present state and particularly heading.

In this formulation, it is assumed that the cost is purely a function of the state at the terminus of a trajectory or trajectory segment, *i.e.,* it is path-independent. The primary quantity of interest in this cost is the location where interaction with the terrain occurs, whether intentional (*e.g.,* nominal behavior) or not (*e.g.,* off-nominal behavior). In view of (55), this implies that $\phi \equiv 0$, and we are focusing on functions of the form $\phi_F(\mathbf{x}_{t'|t+\Delta t}, \mathbf{x}_G)$, where t' is the terminal time of the current trajectory segment. Under many circumstances, it may make sense to incorporate other conditions within a path cost, such as penalizing large heading rates, though this is not considered further in this work.

The use of a reachability set addresses the intention to incorporate the effect of approach direction on the cost of a particular node. The full reachability set of a state is defined as all possible future states that can be reached by the system from that state, by applying the appropriate sequence of future inputs. In the case of the parafoil, this set can be constructed by propagating all input sequences from the initial state until intersection with the terrain. Constructing such a set is intractable, at best, so for the purposes of utilizing it within the cost-to-go function, several simplifying assumptions are made.

First, it is assumed for the purposes of constructing the reachability set that the parafoil dynamics (11)-(15) have "ideal," lag-free angular rate dynamics, *i.e.,* $A = B = C = 0$ and $D = 1$. This eliminates (14) and simplifies (15) to $\dot{\psi} = u$, where $|u| \leq \omega_{\max}$. Second, the reachability set is bounded to consider the possible evolutions of the system state over a finite propagation time t_P.

Finally, rather than consider all possible input sequences over the finite time window, only a finite number N_P of *constant* heading rates are propagated. These heading rates should be representative of the full input set, such that the resulting points are representative of the full reachability set. With this in mind, we elect to use a set of equi-spaced inputs, sampled evenly between the minimum and maximum angular rates (which are equal and opposite):

$$\omega_i = -\omega_{\max} + \frac{2\omega_{\max}}{N_P - 1} i, \quad i \in \{0, \ldots, N_P - 1\}. \tag{67}$$

It is generally desirable to choose an odd value for N_P, such that the propagation with zero angular rate ($\omega_i = 0$, for $i = (N_P - 1)/2$) is included. The system dynamics will be simulated from the current state at the angular rates ω_i to identify points on the frontier of the reachability set.

Consider a given parafoil state with position $\mathbf{x}_0 = (p_{x0}, p_{y0}, p_{z0})$ and heading ψ_0. Given that the parafoil is descending at the rate $v_z = v(p_z)/L_D$, the goal altitude p_{zG} will be reached after time $t_{zG} = (p_z - p_{zG})/v_z$. The propagation time is chosen to be the lesser of this value and t_P,

$$\tau \equiv \min\{t_{zG}, t_P\}. \tag{68}$$

For the ith control input ω_i and propagation time τ, the ith point $\mathbf{x}_i = (p_{xi}, p_{yi}, p_{zi})$ in the frontier of the reachability set approximation is computed to be

$$p_{xi} = p_{x0} + \left|\frac{v(p_{z0})}{\omega_i}\right| \left(\cos(\psi_0 + \text{sign}(\omega_i)\frac{\pi}{2}) + \cos(\psi_0 + (2 + \text{sign}(\omega_i))\,\frac{\pi}{2} + \omega_i \tau)\right), \tag{69}$$

$$p_{yi} = p_{y0} + \left|\frac{v(p_{z0})}{\omega_i}\right| \left(\sin(\psi_0 + \text{sign}(\omega_i)\frac{\pi}{2}) + \sin(\psi_0 + (2 + \text{sign}(\omega_i))\,\frac{\pi}{2} + \omega_i \tau)\right), \tag{70}$$

$$p_{zi} = p_{z0} - \frac{v(p_{z0})}{L_D}\tau. \tag{71}$$

Figure 4 illustrates the result of the reachability set approximation for $N_P = 3$ and $\tau = t_P = \frac{\pi}{2\omega_{\max}}$, representing a quarter-turn at maximum angular rate.

The cost-to-go function is constructed by assigning costs J_i to each state $\mathbf{x}_i$, $i \in \{0, \ldots, N_P\}$, then comparing those values. The cost J_i is the Euclidean distance from the point $\mathbf{x}_i$ to the goal $\mathbf{x}_G$ at position $\mathbf{p}_G = (p_{xG}, p_{yG}, p_{zG})$, modified to account for drift due to the persistent wind estimate $\overline{\mathbf{w}} = (\overline{w}_x, \overline{w}_y, \overline{w}_z)$ as discussed in Section IV. The cost takes the form

$$J_i = \sqrt{(p_{xi} - p_{zG} - t_{zG}\overline{w}_x)^2 + (p_{yi} - p_{yG} - t_{zG}\overline{w}_y)^2 + (p_{zi} - p_{zG})^2}. \tag{72}$$

The final cost-to-go function takes the maximum between the cost of the initial point, J_0, and the minimum of the cost of the points of the reachability set approximation (69)-(71):

$$\phi_F(\mathbf{x}_0, \mathbf{x}_G) = \max(J_0, \min(J_1, J_2, \ldots, J_{N_P})). \tag{73}$$

Each piece of the cost-to-go function (73) incorporates a different understanding about the parafoil planning problem. The first piece, J_0, encourages the planner to situate the vehicle directly above the goal (after correcting for wind), which is as far from the glide-slope boundary as possible while remaining inside it. Such a placement enables significant disturbance rejection later in the planning sequence. The second piece, $\min(J_1, J_2, \ldots, J_{N_P})$, represents the most favorable state the vehicle can reach within the (approximated) finite-time reachability set. The cost-to-go function considers the minimum cost of these points, rather than the maximum or average, as the planner has the control authority to choose the executed trajectory (assuming idealized heading rate dynamics) and could therefore choose to execute the control yielding the lowest cost. When this piece of the cost-to-go function is active, *i.e.,* $\min(J_1, J_2, \ldots, J_N) > J_0$, it is implied that all possible choices available to the planner are less desirable than remaining at the current state. Simulation results in Section VIII show that this combined cost-to-go is more effective that using either of the individual components alone.

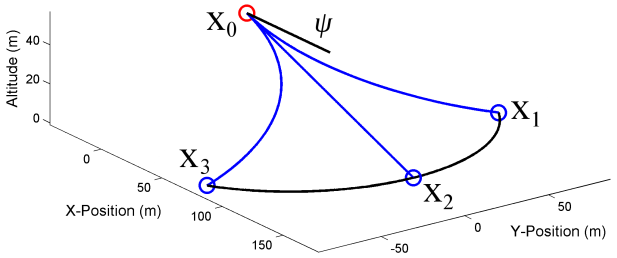


Figure 4. Diagram of the reachability set approximation for $N_P = 3$ and $\tau = t_P = \frac{\pi}{2\omega_{\max}}$.

Note that the cost-to-go function as presented does not include cost terms on the desired heading at landing ψ_G. It is typically desirable, or even critical, in many parafoil applications to land aligned with the upwind heading, in order to minimize impact speed. Preliminary results have shown that augmenting this cost function with a penalty on an impact speed can achieve landing heading accuracy comparable to existing algorithms, with minimal effect on performance; this will be explored further in future work.

Extensive analysis has been performed to determine suitable values for t_P and N_P[24]. Analysis for t_P is based on which component of the cost-to-go (73) is active when the state is on the glide-slope boundary. When on the glide-slope boundary facing the goal (*i.e.,* a desirable state), the J_0 term should be active, whereas if on the boundary facing away from the goal (*i.e.,* a highly undesirable state), the $\min(J_1, J_2, \ldots, J_{N_P})$ term is active. It can be shown that this asymmetry only occurs for values of $t_P < t^*$, where

$$L_D^2 \sin \omega_{\max} t^* = \omega_{\max} t^*. \tag{74}$$

For $L_D = 2.8$ and $\omega_{\max} = \pi/15$ rad/s (Section IV.A), $t^* \approx 13.27$s. Empirical results have demonstrated that a value approximately half that is effective in simulation[24]; we choose $t_P = 7.5$ s, representing one quarter-turn in the reachability set. Regarding N_P, analysis has shown that there is a significant information gain in moving from $N_P = 1$ to $N_P = 3$, whereas increasing N_P beyond 3 has minimal-to-no effect on the shape of the cost-to-go function. Thus $N_P = 3$ is chosen[24] (Figure 4).

VIII. Simulation Results

After providing details of the algorithm software implementation, this section presents simulations demonstrating that parafoil CC-RRT, also referred to here as "analytic CC-RRT," achieves superior landing accuracy to both nominal RRT and BLG[6] on a challenging terrain scenario. In particular, parafoil CC-RRT demonstrates significant improvement in mean accuracy over BLG, and superior worst-case landing accuracy to both RRT and BLG. Further analysis shows that while BLG performance tends to degrade as the difficulty of the terrain increases, parafoil CC-RRT is largely invariant to changes in terrain, in both the mean and worst-case. Parafoil CC-RRT is also shown to be not only capable of use at higher initial drop altitudes, but also invariant to initial drop altitude. Additional simulation results show that the multi- classification wind model (Section IV) and max-min cost-to-go formulation (Section VII) each yield better accuracy than their individual components.

Three algorithms are compared throughout this section:

RRT with mean wind, which represents a nominal RRT planner in which uses the mean wind estimate $\overline{\mathbf{w}}$, but assumes no future wind variation (*i.e.,* $\delta\mathbf{w} \equiv 0$, $\mathbf{w}_t \equiv \overline{\mathbf{w}}$). This approach makes no active attempt at robustness against uncertainty, but does utilize replanning at every timestep to try to counteract system disturbances.

Analytic CC-RRT, the full parafoil CC-RRT algorithm presented throughout this paper and specified in Algorithms 1 and 2. It utilizes the multi-classification wind model (Section IV) to inform the choice

of analytic covariance samples, used to select paths which are robust to future wind uncertainty (Section V). The reachability-based cost-to-go (Section VII) is also incorporated.

BLG, or band-limited guidance, which utilizes band-limited control to ensure accurate tracking and prediction, as well as knowledge of the mean wind estimate $\overline{\mathbf{w}}$ and replanning to account for system disturbances[6]. (The implementation of BLG is detailed below.)

Most results in this section are presented in the form of CDFs on radial 2D miss distance, aggregated over a large collection of trials in which the initial conditions, wind profile, and (for the RRT-based algorithms) sample randomization are varied. On such plots, each data point (q, r) implies that the fraction r of all trials performed achieved a miss distance of q or less. Thus, when comparing relative performance, it is desirable for an algorithm's CDF to move "upward" (*i.e.,* larger fraction achieves the same miss distance) and "leftward" (*i.e.,* same fraction achieved a lower miss distance). The "tail" of the CDF, representing the worst-performing trials, is also of particular interest; a long tail may indicate that performance is inconsistent, or not robust to certain pathological conditions. Confidence bounds at 95% (computed using MATLAB's `norminv` function) and tabular representation of the data, which includes mean accuracy, are also provided.

VIII.A. Implementation

For each scenario, each algorithm is tested on a large series of simulation trials, which vary in the combination of wind profile and initial conditions used. Of the hundreds of wind profiles released by Draper Laboratories[23], a set of 25 representative wind profiles are used. These 25 wind profiles consist of 18 profiles from collected drop data and 7 artificially-generated profiles. Of the 7 artificially-generated profiles, 6 are constant-wind profiles moving in the cardinal directions, varying in intensity from zero wind to 25 knots (over 70% of the parafoil airspeed). The 7th artificially-generated profile represents an exponentially-decaying wind, with average and maximum wind speed changes (with respect to altitude) of 0.0025 $\frac{\text{m/s}}{\text{m}}$ and 0.05 $\frac{\text{m/s}}{\text{m}}$, respectively. The actual drop wind profiles are significantly more aggressive, with an average overall intensity of 6.7 m/s and gusts up to 17.1 m/s (nearly matching the parafoil airspeed). These profiles are subject to average and and maximum wind speed changes (with respect to altitude) of 0.025 $\frac{\text{m/s}}{\text{m}}$ and 2.4 $\frac{\text{m/s}}{\text{m}}$, respectively. They are also subject to rapid directional changes, potentially as large as 115°/m.

In each trial, the parafoil state is initialized 500 meters above the goal (assumed in all scenarios to be located at $\mathbf{p}_G = (0, 0, 0)$) with a random heading and a lateral distance between 100 and 400 meters from the goal. Each algorithm is subject to the same sequence of wind profile/initial condition combinations. A total of 500 trials are performed for each algorithm in each scenario, representing 20 uses of each wind profile for different initial conditions.

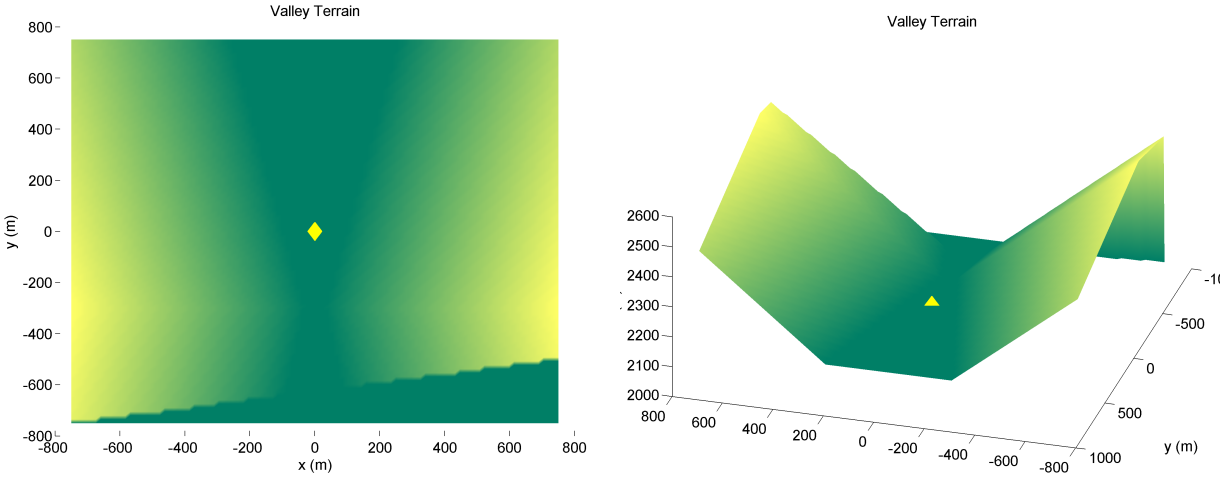
The primary terrain used in the simulations is the valley terrain, $T_{\text{valley}}(p_x, p_y)$, pictured in Figure 5. This represents a particularly challenging terrain for the parafoil terminal guidance problem, for several reasons. First, the slope of the valley is greater than the glide-slope of the parafoil, making approach from either side impossible. Second, the large low-altitude regions away from the goal (bottom-right and top in Figure 5(a)), where terrain collisions can be avoided for longer path durations, are likely to lead to terrain interactions as the parafoil's path crosses in and out of those regions.

To test how algorithmic performance varies with terrain "difficulty," this section also considers scaled-down versions of the valley terrain, *i.e.,* $\alpha T_{\text{valley}}(p_x, p_y)$ for $\alpha \in [0, 1]$. In particular, simulations are performed for $\alpha = 0$, representing completely flat terrain, and $\alpha = 0.75$, representing intermediate conditions.

The CC-RRT algorithm has been implemented as a single-threaded Java application. To simplify comparisons, a fixed number of samples, or iterations of Algorithm 1, are performed per loop of Algorithm 2 (lines 3–17). In the subsequent results, 165 samples per loop are used, representing the average number of samples generated in a 1 Hz planning cycle with 60% duty cycle by the nominal RRT algorithm. The mean wind impulse filter (Section IV) has a width $m = 8$, while two rings of 10 covariance samples each are used with an overall $p_{\text{safe}} = 0.9$ (Section V).

The BLG algorithm, against which parafoil CC-RRT is compared, determines an optimal control by choosing coefficients ψ_k for the heading rate profile,

$$\psi'(h) = \sum_{k=0}^{N} \psi_k \frac{\sin(\pi(z - k\Delta z)/\Delta h)}{\pi(z - k\Delta z)/\Delta h}, \tag{75}$$



(a) top view (b) skew-side view

Figure 5. Valley terrain used in several of the parafoil terminal guidance scenarios. Green shades indicate lower altitudes; the goal is located at the yellow diamond.

based on simulating forward simplified dynamics,

$$\begin{aligned} x' &= -L_D \cos(\psi) + w_x/\dot{z}, \\ y' &= -L_D \sin(\psi) + w_y/\dot{z}, \\ (\cos(\psi))' &= -\psi(z)' \sin(\psi), \\ (\sin(\psi))' &= \psi(z)' \cos(\psi), \end{aligned} \tag{76}$$

where $(\cdot)'$ denotes a derivative with respect to altitude z. It then optimizes a cost function consisting of a weighted sum of x^2, y^2, and $(\sin(\Delta\psi/2))^2$, where $\Delta\psi$ is the difference between final heading and desired heading, using the Nelder-Mead simplex optimization algorithm[6]. This vehicle model is fundamentally different from the one used by analytic CC-RRT (Section IV.A) in the way heading rate is handled. Whereas CC-RRT assumes heading rate is the output of a linear lag-dynamics model, the BLG vehicle model assumes lag-free control over the heading rate, provided that the controls are bounded by (75).

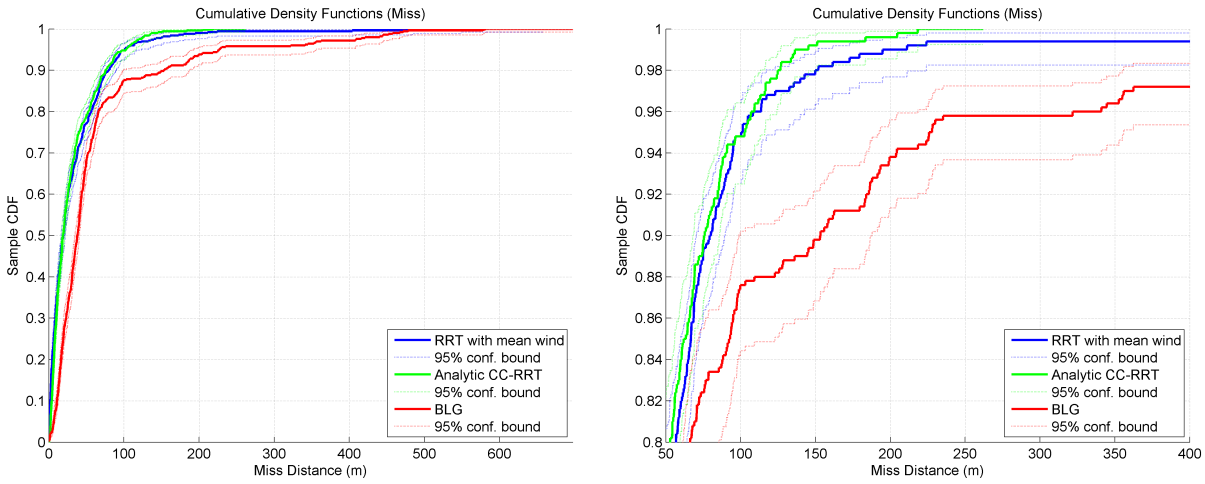
The BLG algorithm has been implemented in MATLAB for comparison. For comparison with CC-RRT, rather than using a tolerance-based stopping criterion, BLG is permitted to simulate the parafoil to the ground through 75 iterations per planning cycle. This number of iterations requires a comparable amount of computation to the number of RRT samples per planning cycle, as described above.

VIII.B. Valley Terrain Simulations

First, 500 trials were run for each algorithm – RRT with mean wind, analytic CC-RRT, and BLG – on the valley terrain scenario (Figure 5). Figure 6 and Table 1 show the CDF (with 95% error bounds) and statistics, respectively, for these trials. Analytic CC-RRT demonstrates matching or improved landing accuracy, relative to RRT with mean wind and BLG, at nearly all percentiles.

Both RRT-based algorithms show significant improvement over BLG for all but the worst-case trials. The mean landing accuracy for both is lower than BLG by a factor of 2. This ratio continues up to the 95th percentile, and increases to a factor of 3-4 by the 98th percentile (Table 1). In particular, about 12% of BLG trials have an miss distance exceeding 100m, whereas only about 5% of the RRT-based trials have a miss distance exceeding 100m (Figure 6(b)). The BLG algorithm also demonstrates a "long tail": 4% of trials have a miss distance of 300m or worse, while the worst-case trial misses by 581m.

Landing accuracy is comparable between RRT with mean wind and analytic CC-RRT up to the 95th percentile; however, analytic CC-RRT demonstrates superior performance over both RRT and BLG over the worst 5% of trials. Up to the 95th percentile, performance is similar between the two algorithms, though analytic CC-RRT shows slight improvement at most percentiles (and also has a 10% better mean accuracy). This suggests that for those trials in which terrain interaction is unlikely, the robustness-based



(a) full CDF

(b) zoomed in to $(0, 400)$m $\times$ $(0.8, 1)$

Figure 6. Miss distance CDF for valley terrain comparison, over 500 trials.

Table 1. Miss distance data for valley terrain comparison, over 500 trials (in meters).

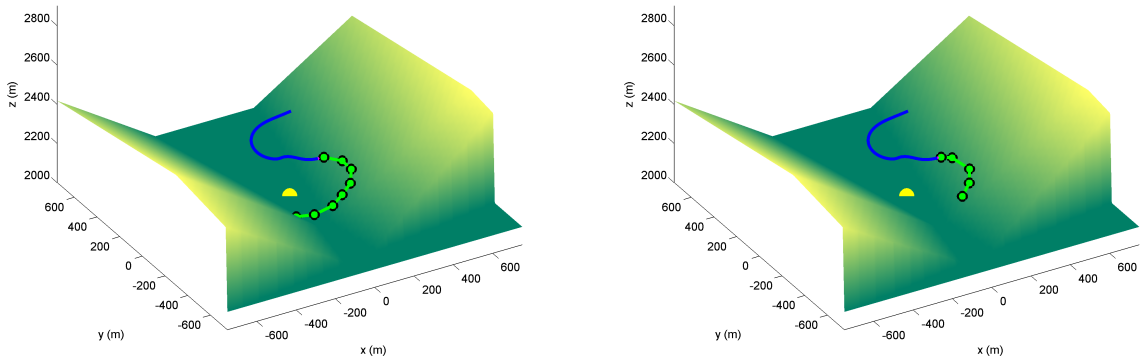
Algorithm	**Mean**	**StDev**	**50%**	**80%**	**90%**	**95%**	**98%**	**Max**
RRT with mean wind	33.9	49.3	19.3	56.6	79.7	101	151	548
Analytic CC-RRT	30.8	32.8	18.7	52.5	75.8	103	126	218
BLG	63.5	89.0	37.9	66.1	153.2	226.9	430.5	581

enhancements in analytic CC-RRT do not significantly influence performance relative to RRT with mean wind prediction alone. However, the two CDF curves diverge beyond the 95th percentile (Figure 6(b)). At the 98th percentile, analytic CC-RRT miss distance is 17% lower than RRT. By the worst-case trial (*i.e.,* 99.8%), analytic CC-RRT miss distance is 60% lower. All trials of analytic CC-RRT have an accuracy of 218m or less, whereas RRT demonstrates some trials exceeding 500m (Table 1). This "shorter tail" for analytic CC-RRT, relative to RRT with mean wind (Figure 6), demonstrates the robustness of the algorithm to pathological uncertainty conditions, which might otherwise drive the vehicle prematurely into the terrain.

Both RRT with mean wind and BLG encounter trials where landing accuracy exceeds 500m. Such situations are the product of an interaction between the uncertain wind and the difficult terrain encountered by the parafoil. Figures 7 and 8 demonstrate how changing wind conditions can cause selected/executed paths from RRT and BLG, respectively, to become infeasible despite replanning.

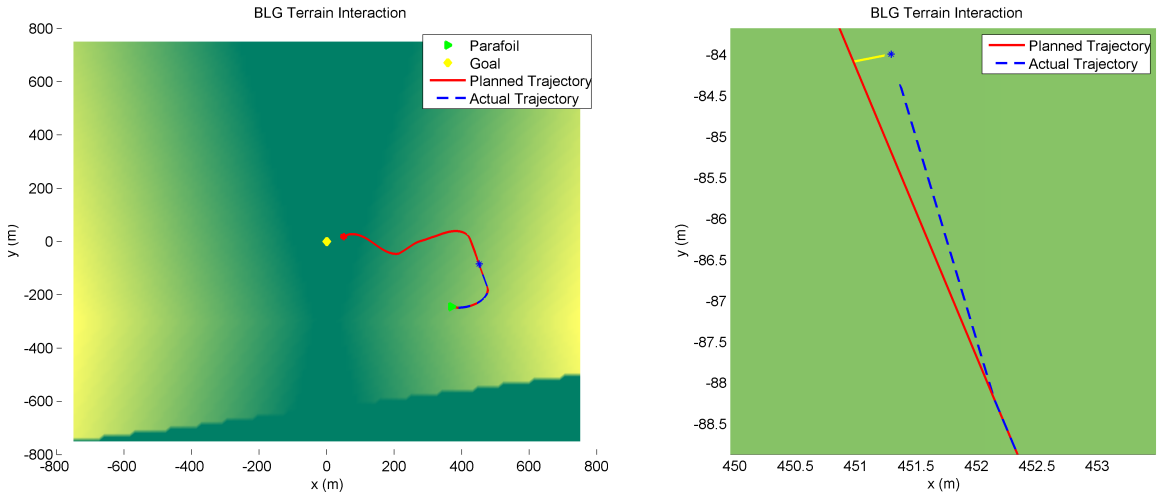
Figure 7 shows the planned paths (green) for nominal RRT on successive timesteps. On the first timestep (Figure 7(a)), the RRT planner has identified a semi-circular path which brings the parafoil relatively close to the goal (yellow circle). However, after a new wind measurement, this trajectory is now predicted to collide with the terrain only about halfway through this path (Figure 7(b)). This causes the second half of the path to be pruned, leaving the parafoil on a trajectory which now has poor terminal accuracy. The issue, in this case, is that several of the intermediate path nodes are very close to the terrain, such that a wind shift causes them to become infeasible.

Figure 8 compares a planned trajectory (solid red line) and executed trajectory (dashed blue line) for a parafoil using a BLG planner. The planned, nominal trajectory (based on the mean wind estimate) takes the parafoil very close to the goal, but also comes very close to the terrain surface on the right side of the valley before turning back toward the goal (Figure 8(a)). About one-quarter of the way through execution, a small wind shift takes place, resulting in a deviation between prediction and execution (Figure 8(b)) that yields a mismatch of less than 1m (yellow line). Yet this mismatch is sufficient to cause the parafoil to collide with the terrain (blue star), resulting in a miss distance exceeding 450m. The direct optimization technique of BLG does not consider off-nominal, future terrain interactions caused by changing wind conditions. As a result, such adverse terrain interactions are possible in the worst case. In the analytic CC-RRT formulation, such proximity to this sloping terrain would be captured by the analytic samples (Section V), such that the original path with low robustness margin would not have been chosen for execution.



(a) Timestep 1 (b) Timestep 2

Figure 7. Trajectory planned using RRT with mean wind, during the first two timesteps. Trajectory history is shown in blue, while the current, feasible path being executed is shown in green.



(a) top view (b) zoomed-in view

Figure 8. Trajectory planned using BLG (red) vs. executed trajectory (blue); note that the premature crash is caused by a mismatch of less than 1 meter (yellow line).

Table 2 shows the average computation time per node generated for both RRT with mean wind (*i.e.*, 0 samples) and analytic CC-RRT, for various numbers of covariance samples. The time per node for analytic CC-RRT scales favorably with both the number of covariance samples and relative to RRT alone.

VIII.C. CC-RRT Invariance to Terrain

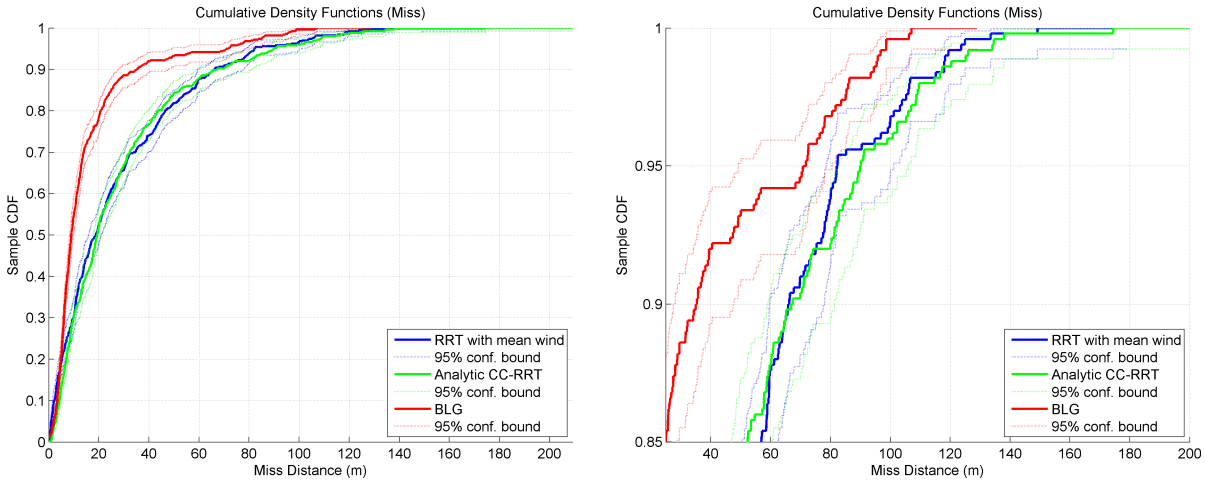
Figure 9 and Table 3 show the CDF and statistics, respectively, for 500 trials performed on a completely flat terrain, *i.e.*, $T(p_x, p_y) \equiv 0$. Though the terrain poses less challenges to the planner, this is still a challenging planning problem, due to the parafoil's underactuated dynamics and the wind uncertainty.

Here, the miss distance accuracy of RRT with mean wind and analytic CC-RRT have converged to approximately the same CDF, with RRT with mean wind showing only a slight decrease in miss distance relative to analytic CC-RRT at most percentiles, including the worst-case. This is consistent with using obstacle-free terrain: because the covariance sampling is performed in the 2D horizontal plane relative to the prospective trajectories, said samples will either all be feasible ($p_z > 0$) or all be infeasible ($p_z \leq 0$) at any given timestep, and thus provide no new information to the planner. As a result, analytic CC-RRT functions identically to RRT with mean wind, which has a binary feasibility check, in this scenario. There has also been a slight improvement in the performance of both algorithms relative to the valley terrain, though the improvement is more pronounced for RRT with mean wind.

On the other hand, the BLG algorithm demonstrates significant improvements in accuracy, relative to

Table 2. Average node generation times for nominal RRT and analytic CC-RRT.

# samples	Time per Node (ms)
0 (RRT)	10.4
10	17.00
20	26.24
30	33.19
50	46.00



(a) full CDF

(b) zoomed in to $(0, 200)$m $\times$ $(0.85, 1)$

Figure 9. Miss distance CDF for flat terrain comparison, over 500 trials.

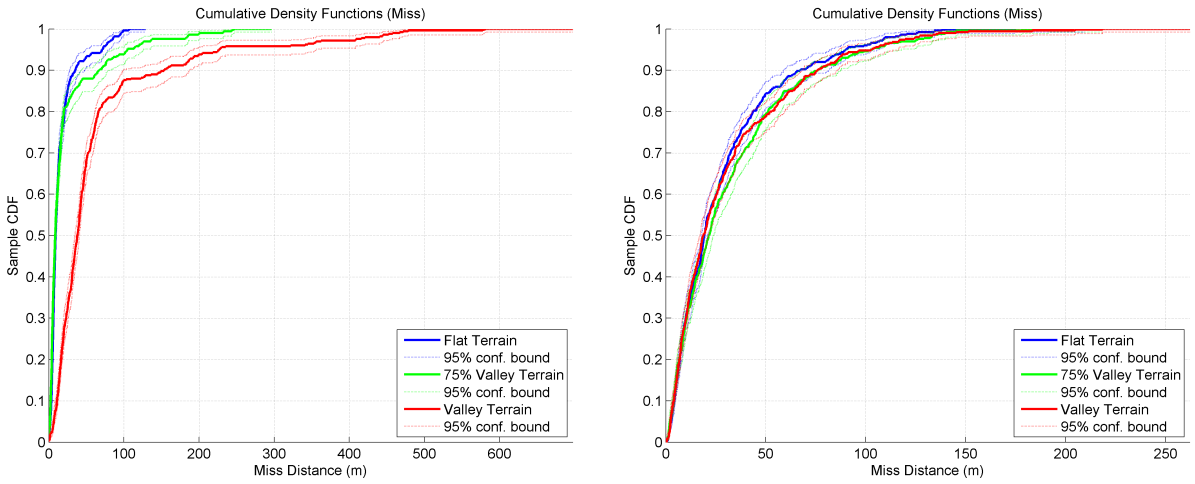
Table 3. Miss distance data for flat terrain comparison, over 500 trials (in meters).

Algorithm	**Mean**	**StDev**	**50%**	**80%**	**90%**	**95%**	**98%**	**Max**
RRT with mean wind	27.8	27.7	18.7	46.5	66.1	82.2	107	149
Analytic CC-RRT	28.4	28.1	19.3	43.6	67.1	90.0	112	174
BLG	15.9	19.7	8.9	20.7	35.2	71.3	86.1	107

both the RRT-based algorithms and its own performance on the valley terrain. Compared to the valley terrain, the mean miss distance has decreased by 75%, while the worst-case miss distance has decreased by over 80% (from nearly 600m to just over 100m). BLG also demonstrates up to 40% better accuracy than RRT with mean wind and analytic CC-RRT, in both mean and worst-case miss distance (Table 3). The improvement is most noticeable between the 50th and 90th percentiles (Figure 9). Because of the absence of terrain features, the BLG algorithm does not have to consider off-nominal terrain interactions in this scenario, creating the ideal environment for the algorithm to converge on optimal solutions. Here, finding feasible solutions (the primary strength of RRT-based algorithms) is a relatively simple task compared to more complex terrain, such that optimizing the planned trajectory (a task BLG is more effective at) is a more efficient use of available computational resources. Replanning alone, without robustness modifications, is sufficient to counteract the shifting wind conditions.

Based on this analysis, we consider how the performance of both BLG and analytic CC-RRT varies as the "difficulty" of the terrain is changed. As stated in Section VIII.A, this is done by considering scalings of the valley terrain, $\alpha T_{\text{valley}}(p_x, p_y)$. In addition to the cases of $\alpha = 1$ and $\alpha = 0$ already considered, we also consider "75% Valley Terrain," in which $\alpha = 0.75$, representing a terrain of intermediate difficulty. Figures 10(a) and 10(b) show the resulting CDFs over 500 trials for BLG and analytic CC-RRT, respectively, with tabular data provided in Tables 4 and 5, respectively.

Based on Figure 10(a) and Table 4, it is clear that the BLG algorithm is highly sensitive to the complexity and steepness of the terrain. As the terrain becomes more complex, feasible paths become more difficult to find, and thus cannot be optimized to the same extent. On the 75% valley terrain, there exists a regime of nominal performance, up to around the 80th percentile, where BLG performance matches or even exceeds performance on flat terrain, with a miss distance under 40m (Figure 10(a)). It is in these cases where



(a) BLG (b) Analytic CC-RRT

Figure 10. Miss distance CDF for BLG and Analytic CC-RRT on various terrain scalings, over 500 trials.

Table 4. Miss distance data for BLG on various terrain scalings, over 500 trials (in meters).

BLG	Mean	StDev	50%	80%	90%	95%	98%	Max
Flat Terrain	15.9	19.7	8.9	20.7	35.2	71.3	86.1	107
75% Valley Terrain	22.5	39.5	8.5	20.1	67.1	105	184	247
Valley Terrain	63.5	89.0	37.9	66.1	153	227	431	581

Table 5. Miss distance data for Analytic CC-RRT on various terrain scalings, over 500 trials (in meters).

Analytic CC-RRT	Mean	StDev	50%	80%	90%	95%	98%	Max
Flat Terrain	28.4	28.1	19.3	43.6	67.1	90.0	112	174
75% Valley Terrain	32.0	32.2	21.6	51.2	76.0	103	133	191
Valley Terrain	30.8	32.8	18.7	52.5	75.8	103	126	218

finding a feasible solution is relatively straightforward, and BLG is able to spend significant time optimizing the solution. For the remaining approximately 100 trials, however, terrain interactions are a serious issue, and BLG miss distance begins to increase significantly relative to the flat terrain (Figure 10(a)). At the 98th-percentile, both the miss distance and worst-case miss distance on 75% valley terrain are more than twice their flat-terrain counterparts. Once the terrain scaling is increased to the full valley terrain (Section VIII.B), the possibility of terrain interactions becomes much more significant, and miss distances increase at all percentiles, especially the worst-case trials.

On the other hand, the performance of the analytic CC-RRT algorithm (Figure 10(b) and Table 5) is largely invariant to the terrain scaling considered. The gap in mean performance between all terrains considered is only 4m or about 12%, while the gap in the worst-case is only 45m or about 20%. Though the best performance (by a slight margin) is observed on the flat terrain, there is no statistically significant distinction between the 75% and 100% valley terrain at any percentile (Table 5). Indeed, there is little discernible difference between the shapes of the CDF curves for any terrain (Figure 10(b)). This suggests that analytic CC-RRT is able to maintain consistent performance, regardless of the difficulty of the terrain scenario. While other algorithms may be able to leverage highly simplified terrain to improve accuracy, such as via BLG's direct optimization, analytic CC-RRT can ensure reasonable performance even under worst-case terrain and wind conditions.

VIII.D. CC-RRT Invariance to Initial Altitude

One of the key advantages of analytic CC-RRT (and RRT algorithms in general), relative to other parafoil terminal guidance algorithms, is the ability to start planning from any initial altitude. Other approaches in the literature require an upper limit on the initial altitude for terminal guidance to remain computationally tractable[6,8–10,12,13]. Figure 11 and Table 6 present simulation results on the valley terrain when the initial

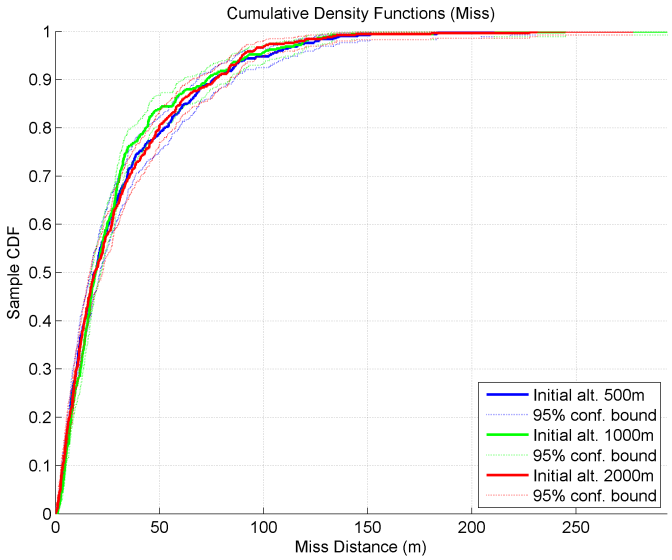


Figure 11. Miss distance CDF for Analytic CC-RRT from various initial altitudes, over 500 trials.

Table 6. Miss distance data for Analytic CC-RRT from various initial altitudes, over 500 trials (in meters).

Init. Alt.	Mean	StDev	50%	80%	90%	95%	98%	Max
$p_{zI} = 500$m	30.8	32.8	18.7	52.5	75.8	103	126	218
$p_{zI} = 1000$m	29.8	32.9	19.2	43.5	72.7	92.6	120	245
$p_{zI} = 2000$m	30.7	33.4	19.0	49.8	76.7	89.8	118	231

altitude, p_{zI}, is varied from 500m (as in Section VIII.B) to 1000m and 2000m; all other conditions are the same as in Section VIII.B. As when varying the terrain, the performance of analytic CC-RRT is seen to be largely independent of the starting altitude. Compared to starting at 500m, worst-case performance only increases 12% at 1000m and 6% at 2000m, while the mean miss distance actually decreases slightly (less than 4%) at higher initial altitudes (Table 6). Again, the shape of the CDF curves for all three cases are nearly the same (Figure 11). This data suggests that analytic CC-RRT is capable of operating at higher altitudes without any deterioration in performance.

VIII.E. Multi-Classification Wind Model

Simulation results in this section compare performance of the multi-classification wind model (Section IV), used in all previous simulation results, to when only a single wind classification/model is used. Here, tests are performed on the same conditions as in Section VIII.B, including the use of the valley terrain, with analytic CC-RRT and 3 wind classes. In the "Combined" case, the full wind model is used, whereas in the "Class X only" cases, the algorithm is artificially forced to classify all wind profiles into a single classification, Class X. This can be considered to be a forced misclassification of the observed wind profile.

Figure 12 and Table 7 give the miss distance CDF and statistics, respectively, for these simulations. When all wind profiles are classified as Class 1, which assumes low power and/or variability (Section IV.B), the planning algorithm is expected to take considerable risks, some of which would pay off with low miss distances, whereas others may result in poor worst-case performance. Indeed, compared to the other single-class results, using Class 1 demonstrates the lowest mean and the highest worst-case miss distance. Moving to Class 2 and Class 3, the planner takes fewer risks and plans more conservative paths. As a result, the mean miss distance increases, but the worst-case performance decreases (Table 7).

On the other hand, the combined wind model achieves the best performance in both the mean and the worst-case. Compared to the least conservative Class 1, the full wind model has only a slightly better mean (4% decrease), but a significantly better worst-case accuracy (38% decrease). Compared to the most conservative Class 3, the full wind model has the same worst-case miss distance, but a significantly better mean miss distance (25% decrease). In the context of the CDFs (Figure 13), the combined CDF nearly matches the Class 1 CDF at most percentiles, but has a significantly shorter tail. By identifying the wind

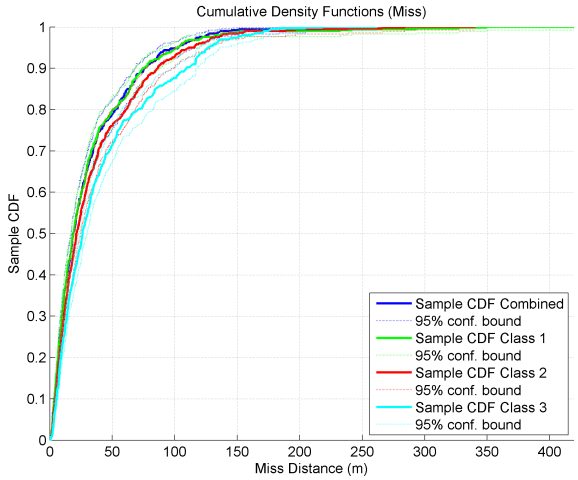


Figure 12. Miss distance CDF for combined wind model and its individual components, over 500 trials.

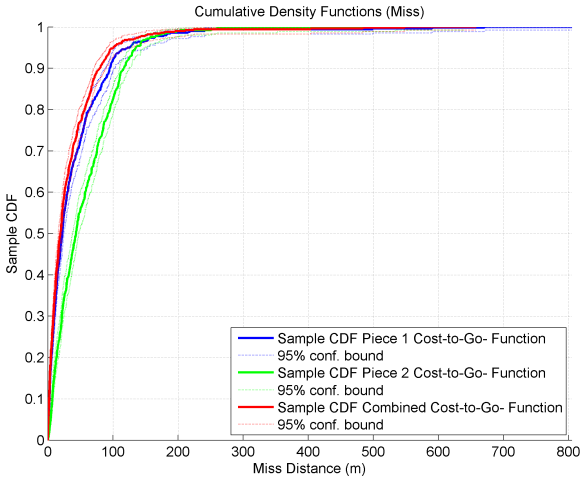


Figure 13. Miss distance CDF for combined cost-to-go function and its individual components, over 500 trials.

Table 7. Miss distance data for combined wind model and its individual components, over 500 trials (in meters).

Components	Mean	StDev	50%	80%	90%	95%	98%	Max
Combined	30.8	32.8	18.7	52.5	75.8	103	126	218
Class 1 only	32.0	39.7	18.8	50.4	74.8	102	134	349
Class 2 only	35.4	38.6	21.2	61.1	86.2	113	138	284
Class 3 only	41.2	41.2	26.1	70.6	109	127	157	218

Table 8. Miss distance CDF for combined cost-to-go function and its individual components, over 500 trials (in meters).

Components	Mean	StDev	50%	80%	90%	95%	98%
Piece 1 only	40.0	59.3	22.0	64.1	95.1	124	181
Piece 2 only	55.9	51.1	43.2	95.3	117	135	170
Combined	33.9	49.3	19.3	56.6	79.7	101	151

classification online, and adapting the wind variation model accordingly, analytic CC-RRT is able to plan robustly while only using as much conservatism as the current wind conditions dictate.

VIII.F. Cost-to-Go Components

Simulation results in this section compare performance of the reachability-based cost-to-go (Section VII), used in all previous simulation results, to accuracy when only a subset of the cost-to-go points are used. Here, tests are performed using nominal RRT with mean wind on the valley terrain (Section VIII.B). In the "Combined" case, the full cost-to-go function as presented in Section VII is used. In the "Piece 1 only" case, only the current state is used in the cost-to-go function, represented as the J_0 piece of (73). In the "Piece 2 only" case, only the future, propagated states are used in the cost-to-go function, represented as the $\min(J_1, J_2, \ldots, J_{N_P})$ piece of (73).

Figure 13 and Table 8 give the miss distance CDF and statistics, respectively, for these simulations. Here, it can be seen that the combined cost-to-go yields superior performance to the individual cost components, along the entire CDF (Figure 13) and at all percentiles of interest (Table 8). Of the individual components, the "Piece 2 only" cost has slightly better worst-case performance than "Piece 1 only," but worse mean performance. The combined CDF more closely tracks Piece 1, but has better performance at the higher percentiles.

IX. Conclusions

This paper has presented a new approach to online trajectory planning and robust obstacle avoidance for the parafoil terminal guidance problem. This algorithm, analytic CC-RRT, robustly executes collision

avoidance with arbitrary, non-convex, mapped terrain, subject to wind disturbances which may be highly dynamic through the terminal approach. The wind is modeled as the sum of a filtered mean wind and a multi-modal variation model. Through online classification, this approach achieves superior performance to a single-class model, by adjusting the level of conservatism to reflect current wind conditions. This wind model is utilized to derive and sample the uncertainty around each simulated trajectory, extending the previously-developed CC-RRT framework for robust planning against arbitrary terrain. Finally, a cost-to-go function is developed which considers both the current state and possible future states on the frontier of an approximated finite-time reachability set, allowing for more intelligent partial path selection (and better landing accuracy) than either component alone. Simulation results have demonstrated that analytic CC-RRT achieves smaller miss distances, both in the mean and in the worst-case, than nominal RRT and BLG (both of which use replanning) on complex terrain scenarios. It is further shown that analytic CC-RRT is largely invariant to both terrain complexity and initial altitude, key deficiencies of existing algorithms.

Future work will focus on considering other kinds of uncertainty that may be acting on the parafoil, extending their use in components of the planning algorithm, and considering other problem constraints. More complex models of the parafoil dynamics will be considered, including descent rates that are partially influenced by the vehicle kinematics (particularly the turn rate), as well as model uncertainty. Depending on the nature of the model uncertainty, the assumption of Gaussian uncertainty at future timesteps (Section V) may no longer hold. Though the mean wind filter already considers 3D wind, the variation wind model will be extended to three dimensions (*e.g.,* covariance samples from a 3D uncertainty ellipsoid), such that the possible future effect of updrafts and downdrafts can be explicitly modeled. We will explore modifications to the cost function, including incorporating statistics from the covariance samples [13], adding path costs, and imposing heading constraints (eg upwind) at landing via penalties on impact speed. Finally, we will explore extensions which incorporate optimality, such as the newly-developed CC-RRT* algorithm [32] which builds on the asymptotic optimality guarantees of RRT* [33].

Acknowledgments

Research funded by Draper Laboratories through their Independent Research & Development (IR&D) program.

References

[1] B. Luders, M. Kothari, and J. P. How. Chance constrained RRT for probabilistic robustness to environmental uncertainty. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Toronto, Canada, August 2010. (AIAA-2010-8160).

[2] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, October 1998.

[3] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference (ACC)*, 2006.

[4] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013.

[5] B. Luders and J. P. How. Probabilistic feasibility for nonlinear systems with non-Gaussian uncertainty using RRT. In *AIAA Infotech@Aerospace Conference*, St. Louis, MO, March 2011. (AIAA-2011-1589).

[6] D. Carter, L. Singh, L. Wholey, S. Rasmussen, T. Barrows, S. George, M. McConley, C. Gibson, S. Tavan, and B. Bagdonovich. Band-limited guidance and control of large parafoils. In *AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, 2009 (AIAA 2009-2981).

[7] N. A. Melchior and R. Simmons. Particle RRT for path planning with uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[8] A. Calise and D. Preston. Swarming/flocking and collision avoidance for mass airdrop of autonomous guided parafoils. In *Guidance, Navigation, and Control and Co-located Conferences*. American Institute of Aeronautics and Astronautics, August 2005.

[9] N. Slegers, E. Beyer, and M. Costello. Use of variable incidence angle for glide slope control of autonomous parafoils. *Journal of Guidance, Control, and Dynamics*, 31(3):585–596, May 2008.

[10] S. Bergeron, K. Tavan and A. Fejzic. Accuglide: Precision airdrop guidance and control via glide slope control. In *Aerodynamic Decelerator Systems Technology Conferences*. American Institute of Aeronautics

and Astronautics, May 2011.
[11] T. Gimadieva. Optimal control of a gliding parachute system. 103(1):54–60–, 2001.
[12] O. A. Yakimenko and N. Slegers. Using direct methods for terminal guidance of autonomous aerial delivery systems. Technical report, DTIC Document, 2009.
[13] J.D. Rogers and N. Slegers. Terminal guidance for complex drop zones using massively parallel processing. In *Aerodynamic Decelerator Systems Technology Conferences*. American Institute of Aeronautics and Astronautics, March 2013.
[14] L. Fowler and J. Rogers. Bezier curve path planning for parafoil terminal guidance. In *Aerodynamic Decelerator Systems Technology Conferences*. American Institute of Aeronautics and Astronautics, March 2013.
[15] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, July 1957.
[16] B.J. Rademacher. *In-flight trajectory planning and guidance for autonomous parafoils*. PhD thesis, Iowa State University, January 2009.
[17] D. Carter, S. George, P. Hattis, M. McConley, S. Rasmussen, L. Singh, and S. Tavan. Autonomous large parafoil guidance, navigation, and control system design status. In *Aerodynamic Decelerator Systems Technology Conferences*. American Institute of Aeronautics and Astronautics, May 2007.
[18] D. Carter, S. George, P. Hattis, L. Singh, and S. Tavan. Autonomous guidance, navigation and control of large parafoils. In *Aerodynamic Decelerator Systems Technology Conferences*. American Institute of Aeronautics and Astronautics, May 2005.
[19] D. Delahaye and S. Puechmorel. Aircraft local wind estimation from radar tracker data. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 1033–1038, 2008.
[20] J. Petrich and K. Subbarao. On-board wind speed estimation for uavs. In *Guidance, Navigation, and Control and Co-located Conferences*. American Institute of Aeronautics and Astronautics, August 2011.
[21] K. Hunt and G. P. Nason. Wind speed modelling and short-term prediction using wavelets, 2001.
[22] X. Jiang, B. Dong, L. Xie, and L. Sweeney. Adaptive gaussian process for short-term wind speed forecasting. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 661–666, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.
[23] Jonathan P. How, Louis Breger, Andreas Kellas, and Chris Dever. Personal correspondence. E-mails dated October 11th, 2011 and March 29th, 2012.
[24] Ian Sugel. Robust planning for autonomous parafoil. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, September 2013.
[25] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.
[26] J. Hartigan and M. Wong. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.
[27] B. Kulis and M.I. Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. *CoRR*, abs/1111.0352, 2011.
[28] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, first edition, 2007.
[29] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.
[30] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.
[31] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, September 2009.
[32] B. D. Luders, S. Karaman, and J. P. How. Robust sampling-based motion planning with asymptotic optimality guarantees. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Boston, MA, August 2013.
[33] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, June 2011.