



Experimental Investigation of Stochastic Parafoil Guidance Using a Graphics Processing Unit

Nathan Slegers¹

George Fox University, Newberg, OR 97132

Andrew Brown², Jonathan Rogers³

Georgia Institute of Technology, Atlanta, Georgia 30332

Abstract

This paper describes and demonstrates a stochastic parafoil guidance system which couples uncertainty propagation with optimal control to protect against wind and parameter uncertainty in the presence of impact area obstacles. The algorithm uses real-time Monte Carlo simulation performed on a graphics processing unit (GPU) to evaluate robustness of candidate trajectories in terms of delivery accuracy, obstacle avoidance, and other considerations. Prior theoretical developments are built upon in this paper in order to explore performance of the stochastic guidance law, particularly with respect to obstacle avoidance. Development and testing of the technique is summarized in three stages. First, through a comprehensive set of simulation results, key implementation aspects of the stochastic algorithm are explored including tradeoffs between the number of candidate trajectories considered, algorithm runtime, and overall guidance performance. Secondly, flight test results are presented comparing the proposed stochastic guidance algorithm with a standard deterministic one. Finally, techniques are developed so that the demonstrated GPU-based terminal guidance algorithm and hardware can be easily integrated with standard production autopilot systems. Overall, simulation and flight test results demonstrate that the stochastic guidance scheme provides a more robust approach to obstacle avoidance while largely maintaining delivery accuracy.

Keywords: graphics processing unit, GPU, optimal control, parallel processing, parafoil

Nomenclature

$\mathbf{a}_{c/T}$	Acceleration of the parafoil mass center with respect to the inertial frame
C_i	Dimensionless parafoil aerodynamic coefficients
D	Distance of the turn initial point with respect to the target
e_i	Estimated mean miss distance for trajectory candidate i
\mathbf{I}	Inertia matrix of the parafoil-payload vehicle about the mass center
$\mathbf{F}_{AP}, \mathbf{F}_{AS}$	Aerodynamic forces on the parafoil and payload
\mathbf{F}_{AM}	Force on the parafoil-payload due to apparent mass effects
$\mathbf{F}_{WP}, \mathbf{F}_{WS}$	Weight forces on the parafoil and payload

¹ Associate Professor, Department of Mechanical Engineering

² Graduate Research Assistant, Woodruff School of Mechanical Engineering

³ Assistant Professor, Woodruff School of Mechanical Engineering

$\mathbf{i}_r, \mathbf{j}_r, \mathbf{k}_r$	North-East-Down reference frame unit vectors
J_i	Cost associated with trajectory candidate i
k_g	Cost function weight for obstacle avoidance
L	Distance to target along target line
$\mathbf{M}_{AP}, \mathbf{M}_{AS}, \mathbf{M}_{AM}$	Moments from aerodynamics of the parafoil, payload, and apparent mass moments, all about the mass center
M	Number of possible yaw rate values used in discretization
N	Number of possible ψ_F values used in discretization
p_i	Estimated probability of obstacle impact for trajectory candidate i
R	Final turn radius
R_s	Number of candidate trajectories selection for Monte Carlo evaluation
\mathbf{s}	Parafoil 6DOF state vector
TIP	Turn initiation point
t_0	Time final turn begins
t_1	Time final approach begins
t_2	Time of predicted impact
T_{app}	Final approach time
V_h, V_z	Parafoil horizontal airspeed and vertical speed
W_x, W_y	Target frame wind speed components
x, y, z	Parafoil inertial positions in the North-East-Down frame
\dot{x}, \dot{y}	Parafoil inertial velocity components in the North-East-Down frame
ψ_F	Final approach angle
ψ	Parafoil Euler yaw angle
ψ_{max}	Maximum parafoil yaw rate
$\omega_{B/T}$	Angular velocity of the parafoil-payload vehicle with respect to inertial frame

1. Introduction

Over the past five decades, significant effort has focused on development of computationally efficient control algorithms to enable real-time execution on embedded microprocessors for robotic systems. Although computational performance of embedded devices continues to rapidly increase, the emphasis on algorithm computational efficiency remains a key characteristic of modern control design [1-3]. For deterministic systems at least, a variety of linear and nonlinear control algorithms now exist that offer suitable tradeoffs between computational complexity and performance [4]. However, current control algorithms are largely serial in nature. That is in the classical control paradigm, a sequence of serial steps is used to arrive at desired control inputs from feedback measurements

Systems subject to large dynamic uncertainty, i.e. stochastic systems, provide a unique challenge in terms of computational complexity because some element of uncertainty propagation must be inherent in the control formulation to ensure either robustness or optimality. In general, continuous systems subject to stochastic uncertainty do not admit closed-form or exact optimal control solutions and the certainty-equivalence principal does not hold [4]. The exception is linear systems subject to Gaussian disturbances, in which the finite horizon optimal

control problem may be solved exactly using dynamic programming recursion (linear quadratic Gaussian, or LQG, control). In the more general case of continuous nonlinear systems or non-Gaussian systems, new control formulations are needed that avoid the “curse of dimensionality” issues associated with dynamic programming techniques [5]. In such cases, new advancements in high-throughput embedded computing may provide an avenue to practical implementation of optimal control for stochastic systems.

The thesis of this paper is that flexible optimal control algorithms incorporating nonlinear, non-Gaussian uncertainty propagation may be practically realizable for robotic systems through proper division of computational effort across a heterogeneous set of onboard embedded computing devices. By coupling standard microprocessors with emerging massively-parallel computing devices, uncertainty propagation and optimization may be performed in real-time without requiring restrictive assumptions of linear dynamics or Gaussian uncertainty. In this new class of optimal controllers, uncertainty propagation may be performed through real-time Monte Carlo simulation, and an optimal control action is determined by minimizing a cost function conditioned on the resulting probability density. Recent work by Ilg et al. [6] demonstrated the feasibility of executing real-time Monte Carlo simulations of air vehicle trajectories on graphics processing units (GPU’s) given their data-parallel execution model.

The work described here explores this heterogeneous computing approach to stochastic control in the context of autonomous parafoil guidance and control. Parafoils are a type of controllable parachute used to deliver cargo and supplies to a specific target via release from an aircraft. In general, parafoil landing accuracy is adversely affected by unknown wind disturbances, which provide perturbations to the system on the same order as the vehicle airspeed. Largely as a result of wind uncertainty, current parafoil landing accuracy is limited to approximately 100 m, which is unsuitable for landing in environments with complex terrain or obstacles near the target [7,8]. Numerous authors have explored a variety of optimal parafoil guidance strategies including model predictive control [9], direct glide slope control [10], and parametric path optimization [11-13]. Gimadieva [14] and Rademacher *et al.* [15] have proposed alternative optimal control strategies, the latter using modified Dubins paths. A key limitation of these solutions is that they are based on deterministic knowledge of the wind and thus may not be robust in cases of unknown winds or wind shifts during terminal flight. A deterministic solution may be appropriate based on the known mean wind; however, it could be extremely sensitive to variations in the wind, with a small change resulting in potential mission failure. For example, using a deterministic solution the optimal impact may occur close to an obstacle but still be considered acceptable. However, in the presence of uncertain winds, many potential trajectories may actually impact the obstacle. In contrast, a probabilistic solution would determine potential trajectory sensitivity to wind variation and as a result select a solution which reduces the probability of hitting the obstacle by shaping the terminal approach appropriately, even at the expense of slight reductions in landing accuracy.

Recently, the authors proposed a new method for stochastic parafoil terminal guidance in which Monte Carlo simulation is performed in real-time on a GPU co-processor [16,17]. The GPU-derived Monte Carlo predictions are used to minimize a cost function that penalizes both impact point accuracy and other parameters such as drop zone constraint violations. Preliminary simulation results in [16] and [17] showed the ability of this guidance formulation to reshape impact dispersion patterns around arbitrary ground obstacles and terrain. This paper builds upon the theoretical foundations outlined in [16-17] to address various tradeoffs in guidance system performance and explore practical aspects of implementation. Specifically, the contributions of this paper include a discussion of the practical aspects of algorithm implementation on a flight vehicle, flight test results demonstrating performance of the stochastic guidance law in comparison to a standard deterministic guidance scheme, and an exploration of the effects of various guidance system parameters (including cost function weightings, optimization grid size, and the number of candidate trajectories considered).

In the paper, the parafoil guidance problem is first formulated building from prior theoretical results. Next, a description of the flight test vehicle is provided, and the guidance package is described including discussion of workload distribution across the autopilot and GPU co-processor. Through a comprehensive set of simulation results, performance of the stochastic guidance algorithm as a function of various parameters is demonstrated, including cost function values, resolution of the optimization grid, and the number of candidate trajectories selected for uncertainty quantification. Experimental results are then shown comparing performance of the proposed stochastic guidance algorithm with a standard deterministic scheme. Experimental dispersion results demonstrate the practical utility of the proposed guidance scheme in terms of obstacle avoidance and robustness to wind uncertainty in a realistic environment. Finally, techniques are developed and presented so that the demonstrated GPU-based terminal guidance algorithm and hardware can be easily integrated with standard production autopilot systems. In addition, a short description of the newest hardware available is provided.

2. Parafoil Terminal Guidance

2.1. Deterministic Guidance

Parafoil guidance is typically composed of three phases: an energy management phase, in which the parafoil dissipates altitude (usually by tracking some sort of square or circular pattern), a homing phase, in which the parafoil proceeds to the target area, and a terminal guidance phase which consists of final maneuvers such that the parafoil arrives at the target and the ground simultaneously. Terminal guidance is initiated at the so-called turn initiation point (TIP) as depicted in Figure 1, where \mathbf{i}_T and \mathbf{j}_T are axes of the standard North-East-Down target reference frame. Note that in general it is desirable to land directly into the wind to minimize ground velocity at impact.

Upon arrival at the TIP, a final trajectory must be computed in the form of a turning maneuver that faces the parafoil into the wind. At the TIP (time t_0), the parafoil is located a distance D upwind of the target and a distance $2R$ to the left of the target, where R is the turn radius. The terminal path is constrained such that at a time T_{app} prior to ground impact, denoted as t_1 , the parafoil must proceed on a straight line path to the target with yaw angle ψ_F . This is deemed the final approach. In practical scenarios, R , T_{app} , and ψ_F are specified, wind components W_x and W_y are estimated, and altitude z and distance L are measured. The terminal guidance problem is then to find a suitable distance D at which to initiate the turn such that the parafoil arrives at the target at t_2 on a heading of ψ_F at zero altitude above the ground. In [16], the authors provided a closed form solution to this open-loop planning problem by assuming a constant rate turn $\dot{\psi}$, a nearly constant vehicle descent rate V_z , a constant horizontal airspeed V_h , and a turn rate slow enough such that roll and sideslip angles can be ignored. In this case parafoil motion reduces to a kinematic model and a closed-form solution for $\psi(t)$ can be found. First, using the constant turn rate assumption the turn rate solution is a linear function of time according to,

$$\psi(t) = t(\psi_F - \psi_0)/(t_1 - t_0) \quad (1)$$

Then, integrating the horizontal velocity components along \mathbf{i}_T and \mathbf{j}_T from t_0 to t_2 and solving for the final turn time t_1 yields,

$$t_1 = t_0 + \frac{\psi_F(2R - (W_y + V_h \sin \psi_F)T_{app})}{W_y \psi_F + V_h(1 - \cos \psi_F)} \quad (2)$$

Finally, by integrating velocity components along \mathbf{i}_T and \mathbf{k}_T from t_0 to t_2 a solution for the downrange distance of the turn initiation point, D , can be found according to,

$$D = -W_x(t_1 - t_0) - \frac{V_h}{\dot{\psi}} \sin \psi_F - \frac{(W_x + V_h \cos \psi_F)(W_x + V_h)}{V_h(1 - \cos \psi_F)} \left(\frac{-z}{V_z} - (t_1 - t_0) - \frac{L - W_x(t_1 - t_0) - \frac{V_h}{\dot{\psi}} \sin \psi_F}{W_x + V_h} \right) \quad (3)$$

Equations (1)-(3) form the deterministic, open-loop trajectory planner used for the remainder of the paper. Additional details regarding the derivation of this closed form solution can be found in [16] and have been omitted here for brevity.

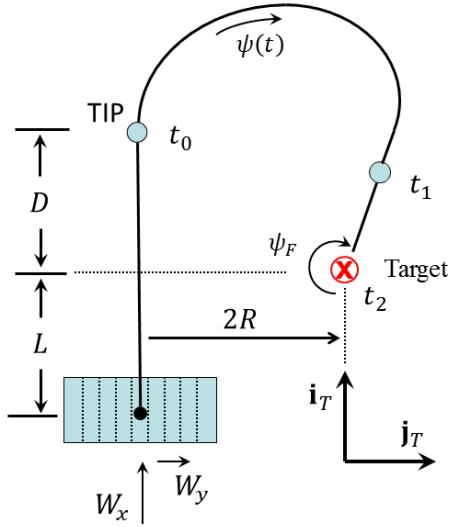


Figure 1. Terminal guidance geometry.

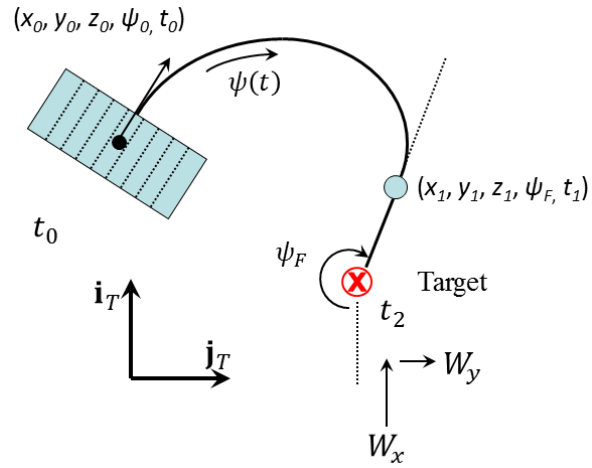


Figure 2. Final turn guidance updates.

The kinematic solution described above is an open-loop planner which yields an optimal turn rate and distance D defining the TIP. Clearly the solution is optimal as long as winds are constant during the final turn. In practical scenarios, gusts or wind shifts during the final turn often cause the parafoil position (x_0, y_0, z_0) and orientation ψ_0 at a given time to vary from the states which satisfied the initial terminal guidance problem. In this case, depicted in Figure 2, a new guidance solution in the form of an updated $\dot{\psi}$ and ψ_F must be computed. This boundary value problem may be formulated as follows: given the parafoil location (x_0, y_0, z_0) at time t_0 with yaw angle ψ_0 , find $\dot{\psi}$ and ψ_F such that the parafoil will land at the target in the presence of winds W_x and W_y . Given the constraints of this problem there is at most one and possibly no exact solution. If in fact no solution exists, it may be reformulated as a nonlinear optimization problem in which an optimal solution for $\dot{\psi}$ and ψ_F is computed given dynamic motion constraints. This optimization problem is described as follows:

Given $x_0, y_0, z_0, \psi_0, t_0$, and estimates of W_x, W_y , find $\dot{\psi}$ and ψ_F that minimizes

$$J_i = f(\mathbf{s}, t) \quad (4)$$

subject to

$$\begin{aligned} x_2 &= x_0 + W_x(t_1 - t_0) + \frac{V_h}{\dot{\psi}}(\sin \psi_F - \sin \psi_0) + (t_2 - t_1)(W_x + V_h \cos \psi_F) \\ y_2 &= y_0 + W_y(t_1 - t_0) - \frac{V_h}{\dot{\psi}}(\cos \psi_F - \cos \psi_0) + (t_2 - t_1)(W_y + V_h \sin \psi_F) \\ t_2 &= \frac{-z_0}{V_z} + t_0 \\ t_1 &= \frac{\psi_F - \psi_0}{\dot{\psi}} + t_0 \end{aligned} \quad (5)$$

The constraints outlined in equation (5) result from the kinematic model approximation and the constant turn rate assumption. In general, the cost function in (4) can represent any desired function of time and parafoil states (\mathbf{s}), and thus the problem described by (4)-(5) is not guaranteed to be convex. Numerical solutions to this problem suffer from well-known issues of any gradient based iterative solver such as convergence, solution speed, and robustness. Nevertheless, given a set of candidate trajectory shapes $(\dot{\psi}, \psi_F)$, equations (4) and (5) may be used to rapidly evaluate candidate paths and select a promising set for further evaluation as described in the next section.

2.2. Stochastic GPU-Based Guidance

While the optimization problem given in (4)-(5) provides a mechanism for feedback planning in the case of wind changes or large tracking error, if a solution can be found it is still only valid in the updated wind conditions.

A potential solution may be appropriate based on the known mean winds; however, it could be extremely sensitive to variations in the wind, with a small change resulting in potentially a large increase in the cost function. Consider the case depicted in Figure 3, where an airdrop target is located within the confines of a designated area represented by walls around an operating base. In this case, landing point accuracy may be far less important than ensuring that the parafoil lands somewhere within the walled boundaries given future wind disturbances. The deterministic guidance formulation in (4)-(5) cannot capture this goal since it evaluates the cost function based on a single, point trajectory prediction. Instead, the guidance algorithm must be able to compute the vehicle response to potential stochastic perturbations and derive a guidance trajectory that, while perhaps landing farther from the target, will at least be robust to wind uncertainty.



Fig. 3. Aerial view of a military forward operating base.

Uncertainty propagation for a dynamic system can be performed in a number of ways from Monte Carlo simulation [18] to direct methods such as the Fokker-Planck-Kolmogorov (FPK) equation [19]. In this case, the point mass model employed in equations (4) and (5) may be insufficient to capture gust response of the vehicle, especially over long distances. Due to its flexibility in handling nonlinear dynamics and non-Gaussian uncertainty, Monte Carlo simulation is chosen to perform uncertainty propagation in real-time for the nonlinear closed-loop system to evaluate robustness of each trajectory candidate. It has been shown [6] that Monte Carlo simulations executed on GPU's exhibit 1-2 orders of magnitude less runtime than serial-based codes on a CPU architecture, leading to the possibility that such simulations can be run in real-time. Experimental results shown here demonstrate that Monte Carlo is indeed a viable real-time method for guidance architectures that include an embedded GPU.

Stochastic parafoil guidance proceeds in three separate stages, as shown in Figure 4. First, a complete set of constant-rate turn trajectory candidates is generated by discretizing $\dot{\psi}$ into M possible values within $[-\dot{\psi}_{\max}, \dot{\psi}_{\max}]$, where $\dot{\psi}_{\max}$ is the parafoil's maximum desired turn rate. Likewise, ψ_F is discretized into N values between 0 and 2π . Given state and wind estimates, these $M \times N$ trajectory candidates are evaluated using the kinematic model in (5), and a cost for each trajectory candidate is computed according to (4). In many cases the cost function used at this stage may be simply the predicted miss distance. These trajectory candidates are then ranked according to their cost, and the best R_s trajectories are selected for further Monte Carlo evaluation. This step is referred to as *prescreening*, and is used to prune trajectory candidates that do not appear promising.

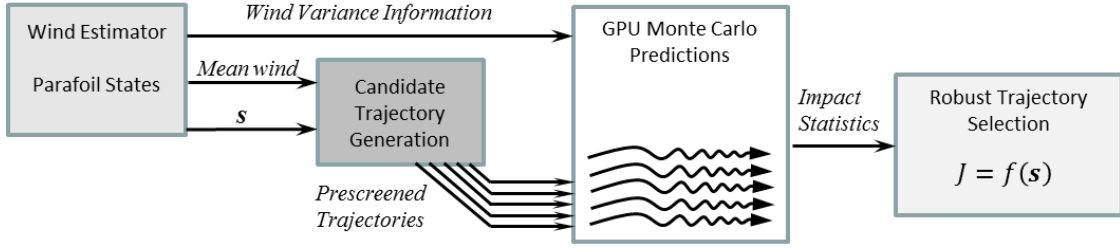


Fig. 4. GPU-based guidance algorithm.

Monte Carlo simulation is performed for each of the R_s trajectory candidates using a closed-loop, 6-degree-of-freedom (6DOF) parafoil model with a model predictive controller that tracks the candidate trajectory. The 6DOF parafoil model is briefly described in the next section, with additional details of the 6DOF and the model predictive controller available in references [20] and [16] respectively. Note that due to the differences in fidelity between the kinematic model (used for prescreening) and the 6DOF model (used for Monte Carlo evaluation), the value R_s must be selected appropriately with respect to the grid resolution M and N so that an adequate number of trajectories are evaluated. This tradeoff is explored later in the paper. Within each Monte Carlo simulation, the steady wind components W_x and W_y are randomized according to a Gaussian distribution with mean equal to the nominal estimated wind, and variance provided by an on-board wind estimator. This Gaussian error model may be viewed as quantifying the effects of uncertainty either in steady-state wind estimates, or due to sudden wind gusts (where the gust is assumed to act at the time that the guidance algorithm is calculated). The result is a dispersion pattern for each of the R_s trajectory candidates. The final step is to select a robust trajectory using a cost function that balances overall miss distance e with other considerations, such as impact velocity or probability of obstacle impact (as described in [16,17]). Here, consider a cost function of the form,

$$J_i = e_i + k_g p_i \quad (6)$$

where J_i is the cost associated with trajectory candidate i , e_i is the mean miss distance, p_i represents the probability of landing inside a constraint (obstacle) region, and k_g is a user-defined gain that weighs the dual goals of accuracy and obstacle avoidance. Note that the obstacle region can be any user-defined polygon. The robust trajectory candidate is then selected as the one yielding minimum cost. This trajectory is used during terminal guidance for tracking within the control algorithm and can be continually updated during terminal maneuvers by repeating this trajectory planning process. Note that feedback planning, in which guidance updates are regularly recomputed, is critical in compensating for error in the 6DOF model, trajectory tracking error, and wind perturbations.

The strengths of the proposed algorithm are two-fold. First, computation time and controller design is largely independent of drop zone features, allowing guidance solutions to be conditioned easily on complex obstacles or terrain. Second, a robust trajectory is selected based on fully nonlinear uncertainty propagation which includes a model of closed-loop system performance. Further details regarding the theoretical foundations of the above GPU-based guidance algorithm, and preliminary simulation results involving obstacles and complex terrain, may be found in [16] and [17]. Furthermore, simulation results comparing the standard deterministic guidance algorithm with the proposed stochastic guidance scheme were provided previously in [16]. In this paper, the effects of various guidance system parameters are explored through simulation for an example scenario and practical implementation aspects are discussed. Experimental performance is then established through a set of flight tests using a small autonomous parafoil system.

2.3 Parafoil 6DOF Model

A brief description of the parafoil 6DOF model used in the stochastic guidance algorithm is presented here. The 6DOF model is formulated by considering the parafoil and payload to be a single rigid body, neglecting aeroelastic effects of the canopy and payload swinging/rotational motion with respect to the canopy. Equations (7) and (8) describe the translational and rotational dynamics of the parafoil respectively, where m_p and m_s denote the parafoil and payload mass:

$$\mathbf{a}_{c/T} = (\mathbf{F}_{AP} + \mathbf{F}_{AS} + \mathbf{F}_{AM} + \mathbf{F}_{WP} + \mathbf{F}_{WS}) / (m_p + m_s) \quad (7)$$

$$\dot{\boldsymbol{\omega}}_{B/T} = \mathbf{I}^{-1} (\mathbf{M}_{AP} + \mathbf{M}_{AS} + \mathbf{M}_{AM}) \quad (8)$$

Note that in the above equations, $\mathbf{a}_{c/T}$ denotes the acceleration of the parafoil-payload center of mass with respect to the inertial frame T , while $\dot{\boldsymbol{\omega}}_{B/T}$ denotes the inertial frame time derivative of the body angular velocity with respect to frame T . The force and moment contributions in (7) and (8) include aerodynamics from the parafoil ($_{AP}$) and payload ($_{AS}$), weight of the parafoil ($_{WP}$) and payload ($_{WS}$), and apparent mass effects ($_{AM}$). Note that the control forces and moments due to brake deflections are included in the aerodynamic force and moment calculations. While higher-fidelity parafoil models have been proposed [20], the 6DOF model used here is judged to be a reasonable compromise between complexity and accuracy for use in guidance calculations, especially since other sources of uncertainty such as winds have a significant effect on parafoil motion and are usually not known precisely. A detailed description of the above model is omitted here for brevity but can be found in Reference [20].

3. Experimental System

To evaluate experimental performance of the proposed guidance algorithm, a parafoil-payload system was constructed and equipped with an electronics package containing an embedded GPU. This section details design and construction of the guidance electronics and provides a brief description of the flight system.

3.1. Guidance Electronics with an Embedded GPU

Figure 5 shows a diagram of the electronics package along with the data flow between each processing element. Typically, GPU processors are combined with a CPU “host” processor for tasking purposes. In this case, the SECO CUDA-on-ARM development board was selected, which includes an NVIDIA Tegra3 quad-core ARM processor coupled with a Quadro 1000M 96-core GPU [21]. The ARM processor is coupled to a standard MEMS autopilot via a serial interface. During each trajectory update, the autopilot sends the 6DOF state and wind estimates to the ARM processor, which performs the initial trajectory prescreening steps. Following prescreening and selection of R_s trajectory candidates, the ARM tasks the GPU to perform Monte Carlo simulations. Once all Monte Carlo simulations are complete, the ARM computes the cost function (6) for each trajectory candidate and passes the optimal trajectory to the autopilot for inner loop control. The advantage of this setup is that the entire system is modular, meaning the autopilot or GPU co-processor components can be replaced or updated without affecting other components. Further note that trajectory optimization is performed completely independently of the inner-loop autopilot, meaning that the latency of guidance optimization calculations do not affect inner-loop control and tracking, since the autopilot continues to track the previous trajectory while a new one is being computed. A picture of the guidance electronics package implemented here is shown in Figure 6. Including the autopilot processor, this embedded electronics package has over 100 cores with a total mass of under 1 kg including batteries.

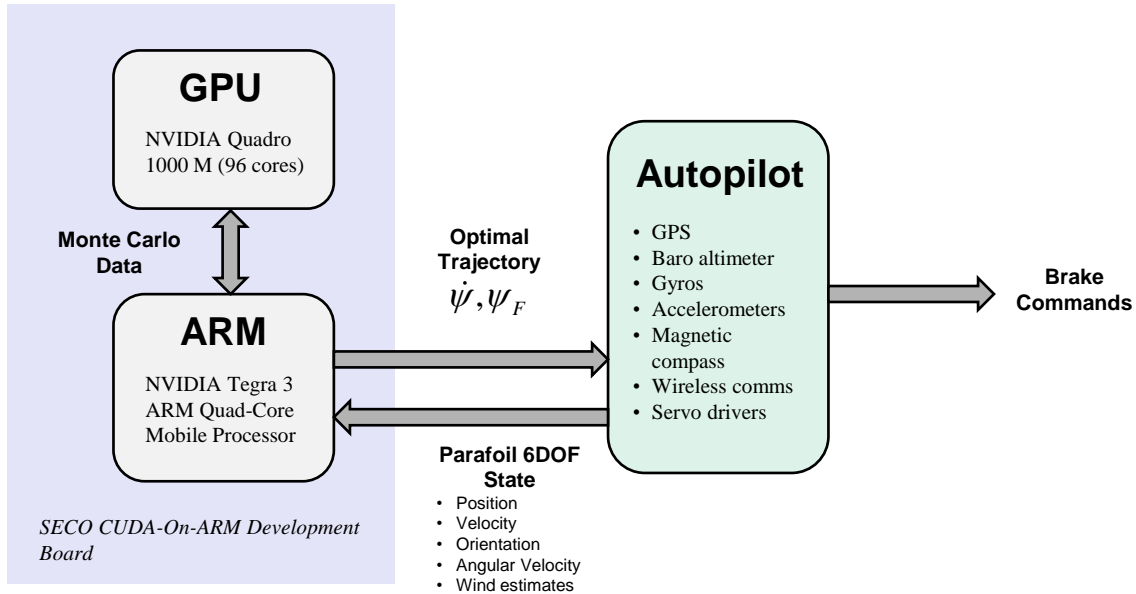


Fig. 5. Flight Control Architecture Design.

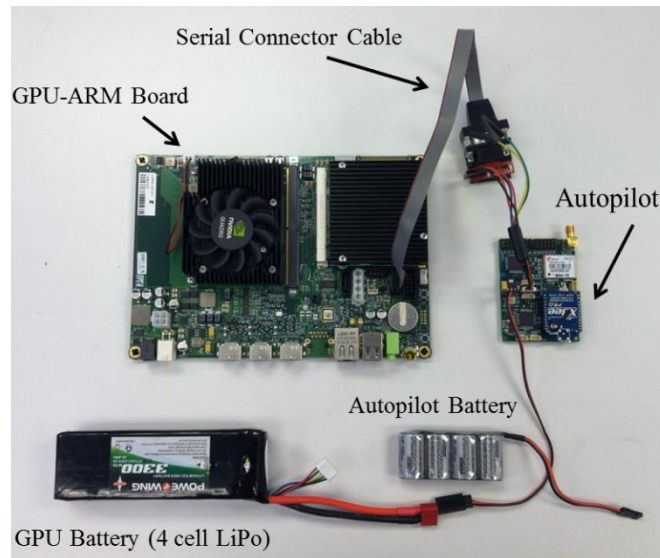


Fig. 6. Flight System Electronics.

In general, it is desired that GPU-based trajectory optimization exhibit latencies of less than about 3 sec so guidance updates can be computed in a feedback manner during terminal guidance [16]. Since Monte Carlo simulations are based on 6DOF trajectory propagation, runtime is a function of altitude. By appropriately selecting the number of trajectory candidates R_s considered for Monte Carlo evaluation, as well as the number of trajectories per Monte Carlo simulation, the desired runtime may be achieved. However, selection of R_s , as well as optimization grid parameters N and M , significantly affect accuracy and obstacle avoidance performance as well. Thus R_s cannot be selected strictly based on runtime limitations. Instead, simulation results for an example scenario presented in the next section are used to obtain a value of R_s that provides a proper tradeoff between runtime, accuracy, and obstacle avoidance.

3.2. Flight Test Vehicle

A small parafoil-payload vehicle was constructed with a canopy span of 2.1 m and mass of 2.72 kg. In power-off glide, the system had a nominal forward flight speed of 7.51 m/s, steady-state descent rate of 4.1 m/s, and maximum turn rate of $\dot{\psi}_{\max} = 30 \text{ deg/s}$. For a given test sequence, the powered system was hand-launched and flown in a climbing pattern to approximately 150 m. At altitude, the motor was turned off and autonomous terminal guidance was activated. A picture of the parafoil-payload vehicle is shown in Figure 7, and a diagram of the flight test pattern is shown in Figure 8 using an aerial photo.



Fig. 7. Photo of Experimental Parafoil-Payload System with Embedded GPU (located in brown box on payload).

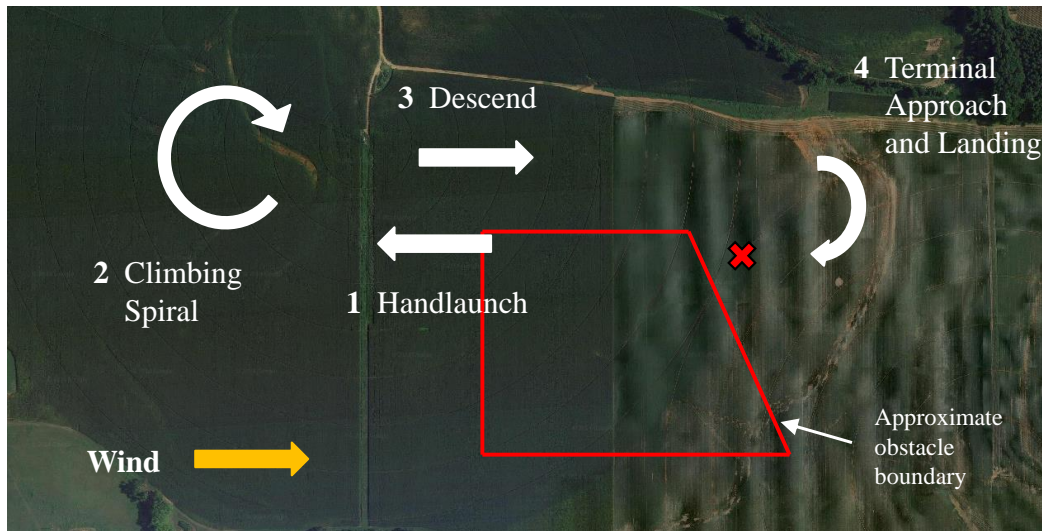


Fig. 8. Aerial Photo of the Test Area and Flight Test Procedure.

The stochastic guidance system utilizes a 6DOF parafoil model for trajectory prediction, and thus system identification is an important aspect of implementation. A specialized parameter estimation method for parafoils outlined in Ward et al. [22] was used to compute critical aerodynamic coefficients for the test vehicle. While it is difficult to obtain a full set of aerodynamic parameters for parafoil-payload aircraft, only a few select parameters

play a significant role in the flight dynamics of the system. These important estimated parameters are listed in Table 1 for this aircraft.

Table 1. Identified Aerodynamic Coefficients for Experimental System.

Parameter	Description	Value (non-dimensional)
C_{D0}	Drag at zero angle of attack	0.15
C_{L0}	Lift at zero angle of attack	0.15
$C_{Y\beta}$	Side force due to sideslip angle	-0.15
$C_{N\delta a}$	Brake control effectiveness	0.0024
$C_{L\delta a}$	Change in lift due to brake deflection	-0.0005
C_{Nr}	Yaw rate damping	-0.02

Wind estimation is performed on the experimental platform prior to terminal guidance using a box-pattern maneuver. For this maneuver the vehicle flies a square pattern recording yaw angle and GPS ground velocity measurements. Using the linear least-squares regression technique described in [22], estimates of the steady-state wind components \bar{W}_x and \bar{W}_y are computed, along with estimated vehicle airspeed \bar{V}_h . Once the mean wind and airspeed estimates are computed, the variance of the wind components can be estimated according to,

$$\bar{\sigma}^2(W_x) = \frac{1}{n-1} \sum_{i=1}^n (W_x^i - \bar{W}_x)^2, \quad \bar{\sigma}^2(W_y) = \frac{1}{n-1} \sum_{i=1}^n (W_y^i - \bar{W}_y)^2 \quad (9)$$

where n is the number of GPS measurements taken during the box pattern, and W_x^i, W_y^i are solutions to the wind triangle at the i^{th} GPS measurement, i.e.,

$$W_x^i = \bar{V}_h \cos(\psi_i) - \dot{x}_i, \quad W_y^i = \bar{V}_h \sin(\psi_i) - \dot{y}_i \quad (10)$$

Note that in equation (10) \dot{x}_i, \dot{y}_i are the i^{th} measured GPS velocities, and ψ_i is the i^{th} measured yaw angle. The estimated mean wind components and wind variances in equation (9) may be used directly in the stochastic guidance algorithm described in Section 2.2.

4. Results

A set of simulation and flight test results is presented for the proposed stochastic guidance scheme highlighting various tradeoffs and performance trends. First, example simulations are provided as well as Monte Carlo simulations demonstrating tradeoffs in selection of guidance system parameters. Based on these simulation studies, a proper set of guidance system parameters are selected for use in flight experiments. A set of experimental results are then presented comprised of 12 flights incorporating the stochastic GPU guidance algorithm, and 11 flights using the open-loop kinematic planner described in Section 2 for comparison purposes. Throughout this section, the kinematic open-loop terminal guidance algorithm is referred to as “standard guidance”, whereas the stochastic guidance algorithm is referred to as “GPU-based guidance”. Two example flights are shown using standard guidance to provide baseline nominal trajectories. Then, several example trajectories using standard guidance and GPU-based guidance are compared. Finally, dispersion results are shown for both standard and GPU-based guidance, and the ability of GPU-based guidance to provide robust obstacle avoidance is demonstrated in a practical setting.

4.1 Simulation Studies

The example scenario employed for this paper uses a trapezoidal constraint in the target area with corners located at (x, y) points of $(-30 \text{ m}, -183 \text{ m})$, $(-30 \text{ m}, -20 \text{ m})$, $(183 \text{ m}, 45 \text{ m})$, and $(183 \text{ m}, -183 \text{ m})$. The target is

located at the origin. Note that while this constraint geometry is simpler than those considered by the authors in [17] (which involved complex terrain), it is selected strictly for the purposes of experimental testing. GPU-based guidance is configured to avoid this constraint area while still landing near the target by selecting an appropriate cost function parameter k_g in equation (6). In computing the cost function in (6), p_i is determined by counting the number of impacts in the Monte Carlo run that land inside the constraint. Based on simulation results presented later in this section, $M = N = 11$ and $R_s = 40$ were chosen, while the number of trajectories per Monte Carlo run was selected as 60. Note that the choice of 60 trajectories per Monte Carlo run is meant to provide a statistically significant sample of the randomized wind model (based on the rule of thumb provided by Van Belle [23] using a standardized distance of 0.5 for the impact miss distance). Figure 9 shows three simulated GPU-based terminal guidance trajectories with different values of k_g starting from an initial location of $x = y = -121$ m, $z = -170$ m, with winds of 3.6 m/s in the downrange direction. Note that, as expected, the terminal guidance paths become more conservative as k_g is increased, since probability of impacting within the constraint is penalized more highly than landing accuracy. Miss distances for these cases are 9 m for $k_g = 0$, 14 m for $k_g = 1.5$, and 53 m for $k_g = 15$.

The effect of k_g is more accurately quantified through Monte Carlo simulations. For various values of k_g , a 200-run Monte Carlo simulation was conducted with randomized wind conditions. Steady-state wind heading and magnitude were sampled according to normal distributions with means of 0 deg and 2.4 m/s, and standard deviations of 30 deg and 1.2 m/s respectively. Furthermore, in each simulation a random gust was introduced at some point in the terminal trajectory. The random gust is modeled as a discrete change in wind lasting the remainder of the flight, with magnitude selected randomly from a normal distribution with a mean equal to the nominal wind value, and standard deviation of 1.8 m/s. The gust direction is chosen from a uniform distribution centered on the nominal wind direction with bounds of ± 45 deg. Figures 10-12 show the results of these simulations. In Figure 10, the impact dispersions become redistributed around the obstacle as k_g grows, so that when $k_g = 15$ only a single impact lands within the constraint. Figure 11 shows that, for values of $k_g > 15$, impacts within the constraint are largely eliminated. At the same time, Figure 12 shows that target-centered Circular Error Probable (CEP) increases until $k_g = 15$, and then largely levels off. Based on these results, a gain of $k_g = 15$ was selected for the flight experiments since only a small number of flight tests would be conducted, and it was desired that obstacle avoidance be emphasized over landing accuracy. Note that, although not shown here, simulation studies verified that a negative k_g value will actually drive the impact dispersion pattern into the constrained region, rather than away from it. By locating the target point inside the constrained area and selecting a negative k_g , the guidance algorithm can be configured to address the constrained drop zone scenario shown in Figure 3.

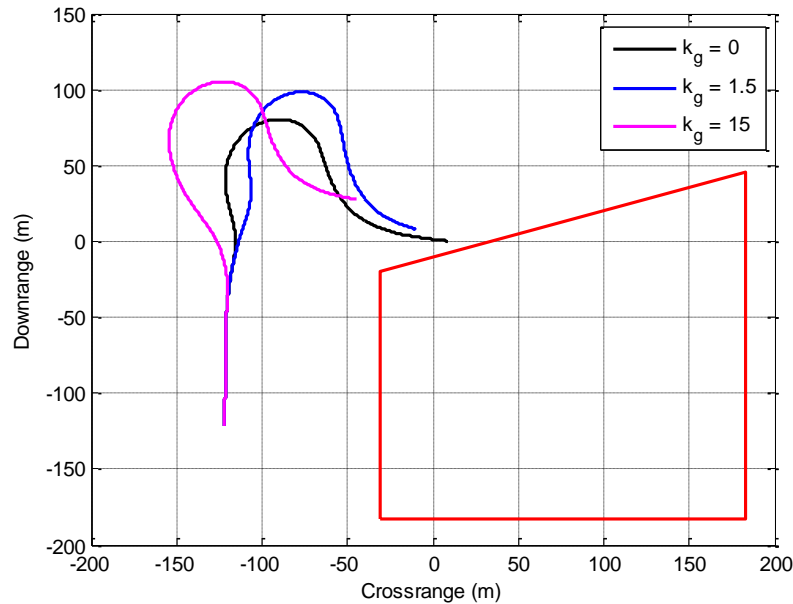


Fig 9. Example Simulated GPU-Based Terminal Guidance Trajectories.

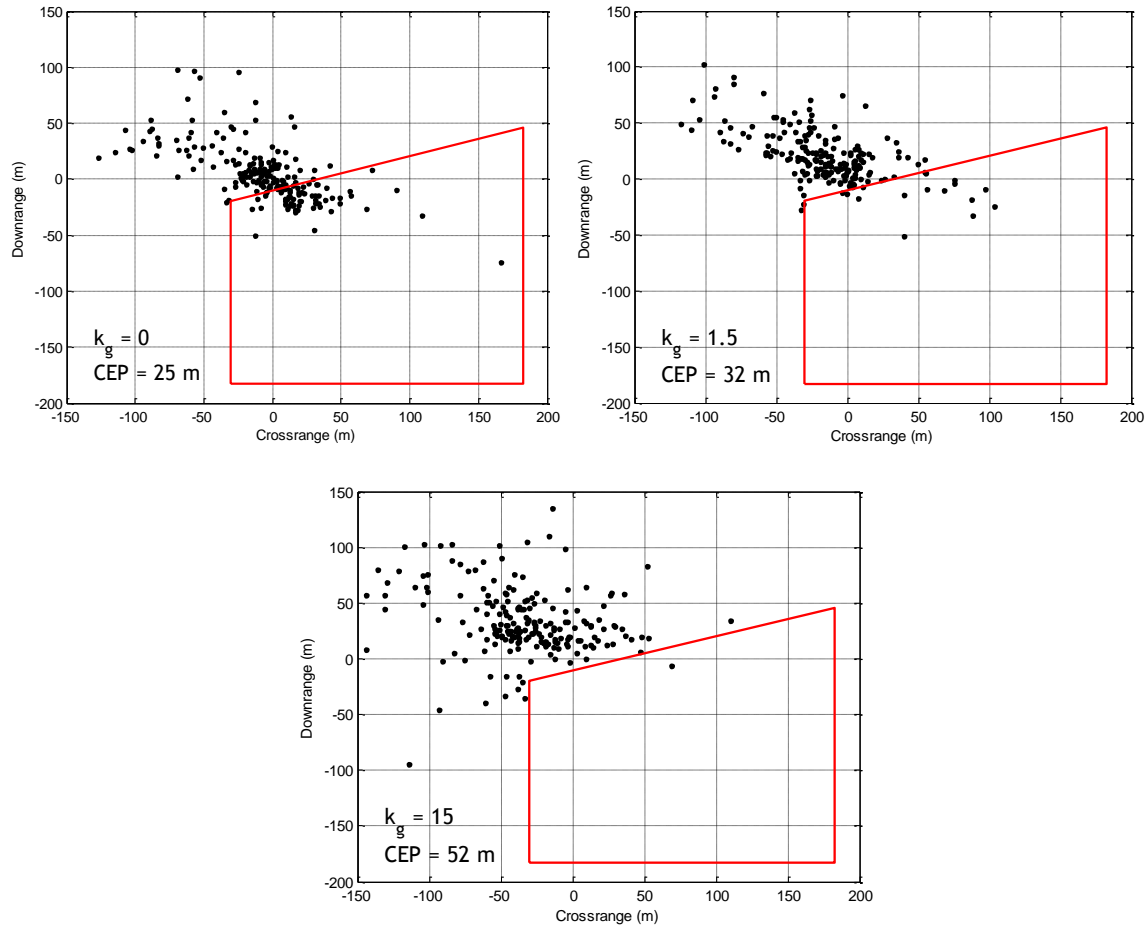


Fig. 10. Impact Point Dispersion Patterns with GPU-Based Terminal Guidance.

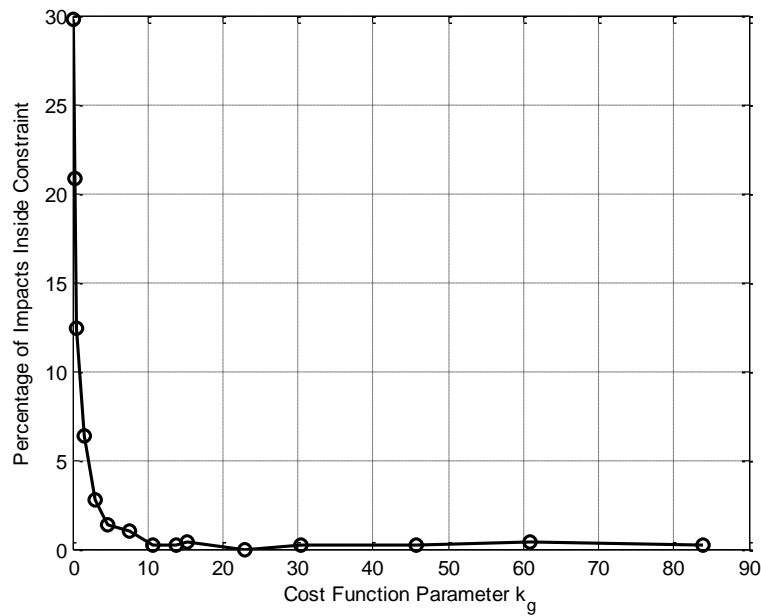


Fig. 11. Percent Impacts within Constraint vs k_g for Example Scenario.

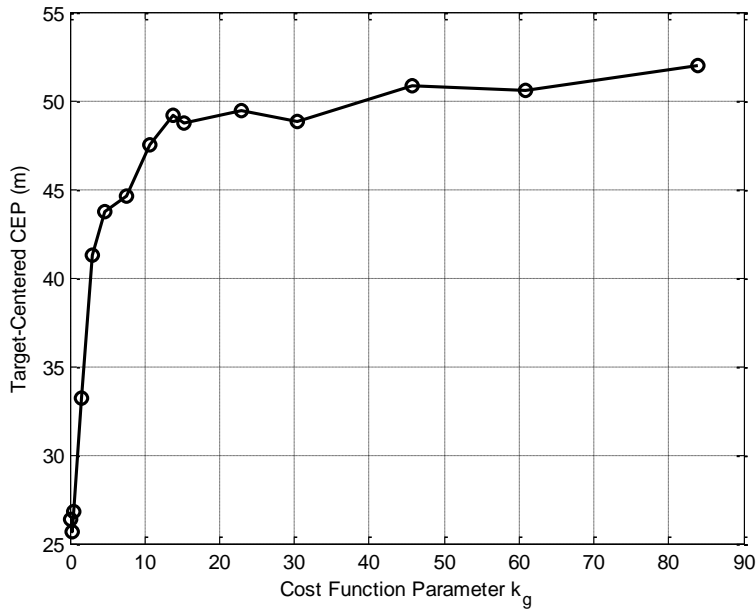


Fig. 12. CEP vs k_g for Example Scenario.

A final trade study involves the size of the optimization grid used to generate candidate trajectories, defined by parameters M and N , and the number of trajectories that are selected for Monte Carlo analysis, R_s . During each guidance cycle, $M \times N$ total trajectories are considered, with the top R_s candidates selected for Monte Carlo evaluation. For the cases considered here, these were selected as the R_s trajectories with the least miss distance based on kinematic predictions (therefore, $J_i = e$ in equation (4), where e is the estimated miss distance from equation (5)). To analyze guidance system performance as a function of M , N , and R_s , Monte Carlo simulations similar to the above were performed for various values of R_s and M , where it was assumed that the number of possible yaw rates M and the number of possible final yaw angles N were the same (i.e., $M = N$). A value of $k_g = 15$ was used. Figures 13 and 14 show percentage of impacts within the constraint and CEP respectively as a function of M and R_s . Note that, for very high resolution grids, both obstacle avoidance and accuracy suffer significantly because the guidance algorithm only evaluates a very small percentage of candidate trajectories, which turn out to be very similar. For instance, with $M = N = 200$ (yielding 40,000 possible paths) and $R_s = 50$, the optimizer evaluates only 0.1% of possible paths which all turn out to be very similar anyway. This is a case of undersampling the high resolution grid. Conversely, if $M = N$ is chosen very small (5 or less), obstacle avoidance is satisfactory but accuracy suffers. This is due to undersampling of the path space, i.e. grid resolution is too coarse to provide an accurate path to the target. Universally, it is seen that as R_s increases, better performance is achieved. However, there are obvious runtime penalties for increasing R_s since more Monte Carlo simulations must be performed. For the experimental tests performed here, values of $M = N = 11$ and $R_s = 40$ were chosen as a suitable balance between obstacle avoidance performance, accuracy, and runtime limitations. Data near these values is highlighted in Figures 13-14. Since Monte Carlo simulations of 60 trajectories are performed for each of the 40 candidates, a total of 2,400 trajectory simulations are generated every guidance cycle. Note that these values are significantly less than those used in [16] since the embedded GPU used here has significantly less processing power than the desktop model used in prior simulation studies.

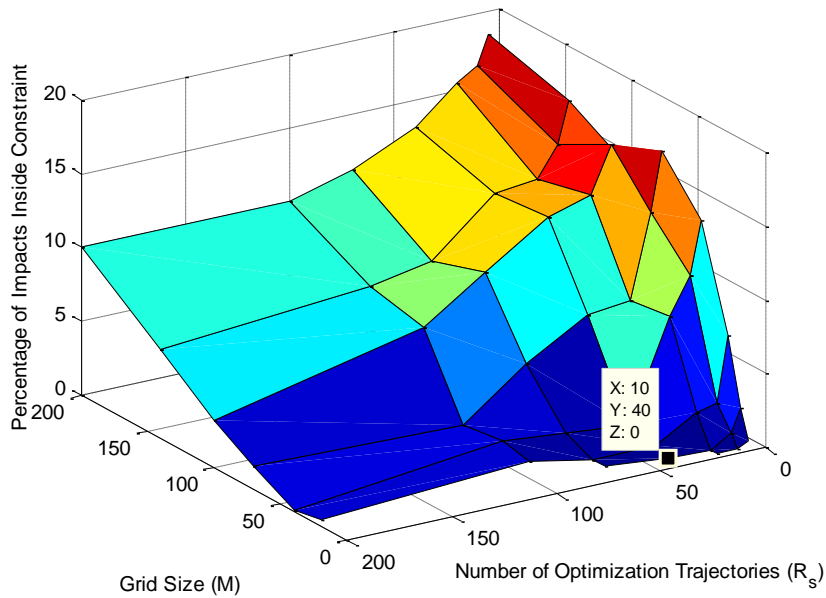


Fig. 13. Percentage of Impacts within Constraint vs M and R_s for Example Scenario.

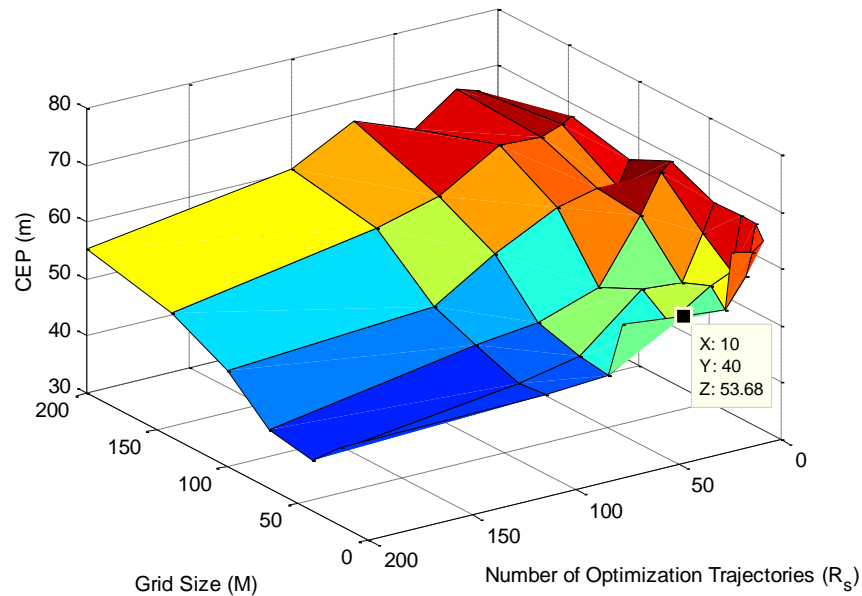


Fig. 14. 50% Target-Centered CEP vs M and R_s for Example Scenario.

4.2. Experimental Results: Standard Guidance

A set of 11 flight experiments was performed using standard guidance. Figures 15 and 16 show two example trajectories. In these cases, once the system enters autonomous mode the autopilot uses wind estimates to compute the optimal turn using (1)-(3) and proceeds to the TIP. Note that in all experimental cases, a false floor was used and thus the exact impact point on the ground must be projected given the final recorded position and velocity vector. A straight line prediction is used to compute the projected impact point at the target altitude. Also note that

in all examples below, data is plotted in the guidance reference frame such that the estimated wind blows directly along the $+y$ axis, i.e., exactly downrange, and the initial parafoil location is roughly at the bottom of each figure. However, these wind estimates are not entirely accurate and are computed at the beginning of terminal guidance – thus, an unanticipated cross wind component causes both trajectories to turn at a rate less than that required. Furthermore, in the case shown in Figure 16, poor wind estimates cause the parafoil to begin the final turn too early and overfly the target. Miss distances for each of these cases, computed from the projected impact point, are 23 m and 72 m respectively.

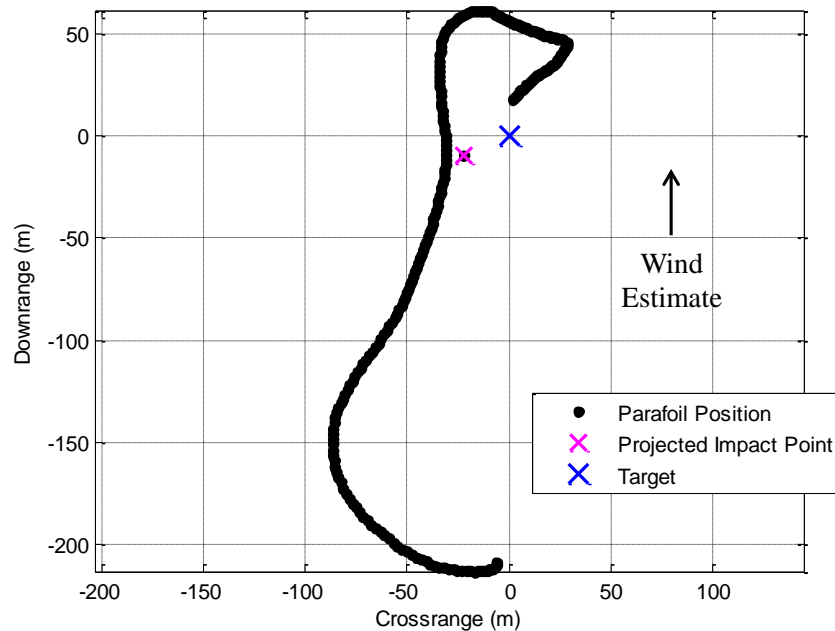


Fig. 15. Standard Guidance Trajectory, Example 1.

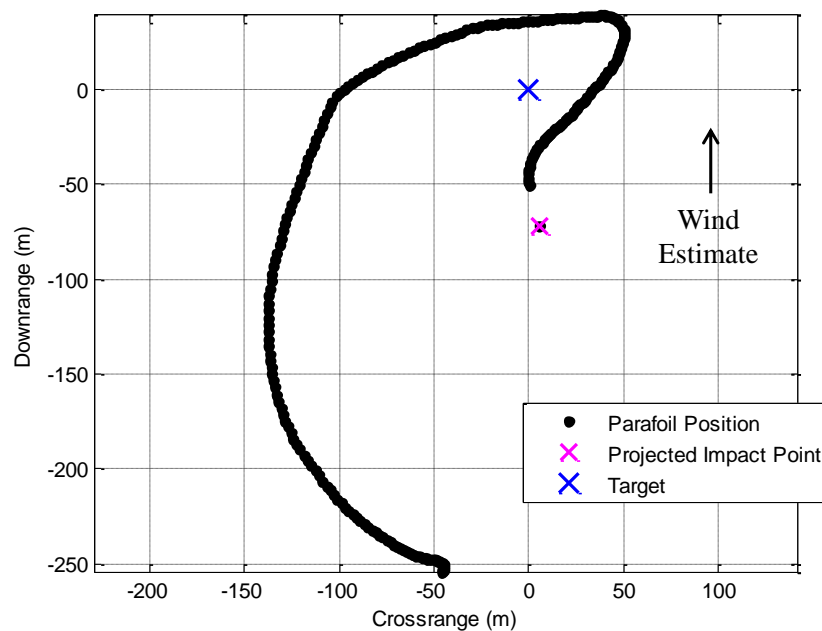


Fig. 16. Standard Guidance Trajectory, Example 2.

4.3. Experimental Results: GPU-Based Guidance

A primary goal of the flight experiments was to compare performance of standard guidance and GPU-based guidance in an experimental setting. Note that the standard guidance flight shown in Figure 16 lands long of the target, and as a result would have landed in the constraint area defined above. This is a clear example of where GPU-based guidance would be beneficial. Alternatively, Figure 17 shows a GPU-based guidance experimental case. In this flight, GPU updates are initiated as the parafoil nears the constraint. Figure 18 shows the same flight, with dashed lines indicating the path commanded by the guidance algorithm during terminal guidance. Note that the initial path commanded by the guidance algorithm is a left turn in order to dissipate altitude away from the constraint. Shortly afterward, right turns are commanded toward the target. Impact occurs short of the target on the proper side of the constraint, with a final miss distance of 16 m. Figure 17 also shows a simulated trajectory from the exact initial conditions used in the flight experiment at the first GPU update. Note that the simulated trajectory is similar in nature to that observed during the flight test. However, in Figures 17 and 18 it is clear that the experimental system did not track the desired turn commands well – the vehicle consistently turned faster than the simulation predicted due to poor tracking error by the inner loop controller. This tracking error was mitigated somewhat through feedback path planning, but in general had a detrimental effect on overall performance. Figure 19 shows the optimal trajectories provided by the GPU co-processor, where a negative turn rate indicates a left turn.

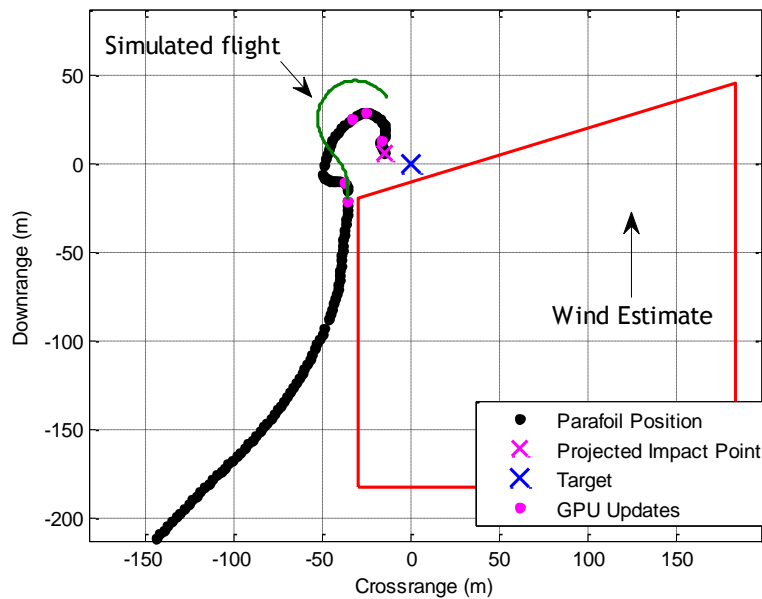


Fig. 17. GPU-Enabled Guidance, Example Flight Trajectory 1.

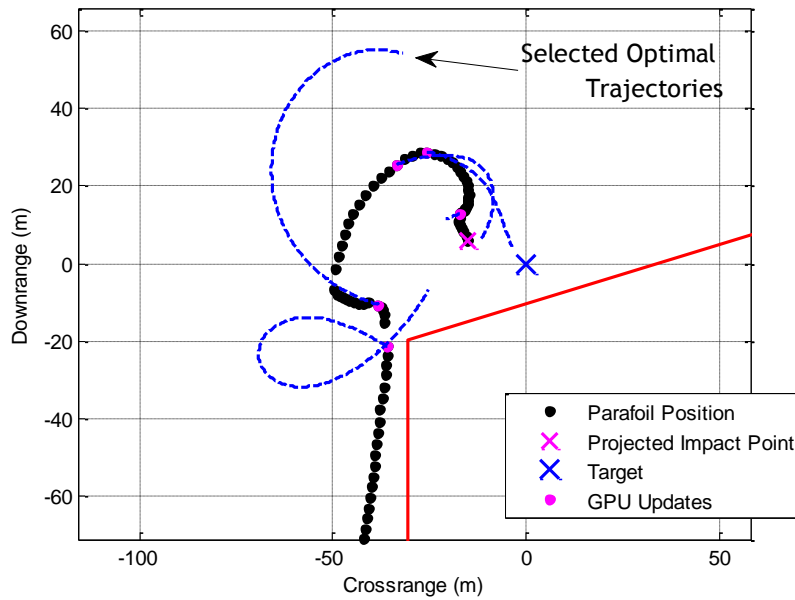


Fig. 18. Optimal Guidance Paths Selected by GPU-Based Guidance, Example Flight Trajectory 1.

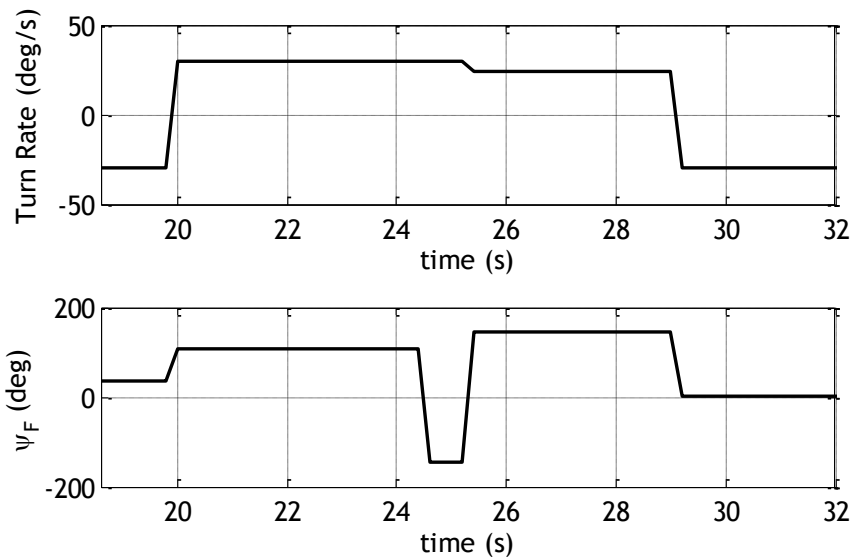


Fig. 19. GPU-Based Guidance Trajectory Solutions, Example Flight Trajectory 1.

Two additional GPU-based guidance examples are provided to highlight behavior of the algorithm in several scenarios. In the first example, shown on the right in Figure 20, the initial GPU trajectory update is requested while the parafoil is already overflying the constraint. In this case, the algorithm provides a shallow left turn such that the parafoil exits the constraint just prior to impact. The final approach angle is roughly along the constraint, which is consistent with behavior observed in simulation studies in [17]. In these cases where the vehicle is in or very near the constraint, the GPU-based trajectories prefer to land heading away from the constraint rather than into it. Total miss distance for this example is a rather large 112 m, partly due to poor geometry at initiation of terminal guidance and partly due to unanticipated deadband in the inner loop control. A final example trajectory

shown on the left in Figure 20 demonstrates performance when terminal guidance is initiated rather far from the target. Because of the conservative weighting of the control system, which values constraint avoidance much more than accuracy, the GPU-based algorithm commands a tight turn far from the target, such that altitude is dissipated well away from the constraint. This behavior was observed in several tests, where the GPU-based algorithm commanded tight turns downwind and to the left of the constraint to significantly reduce the probability of constraint impact.

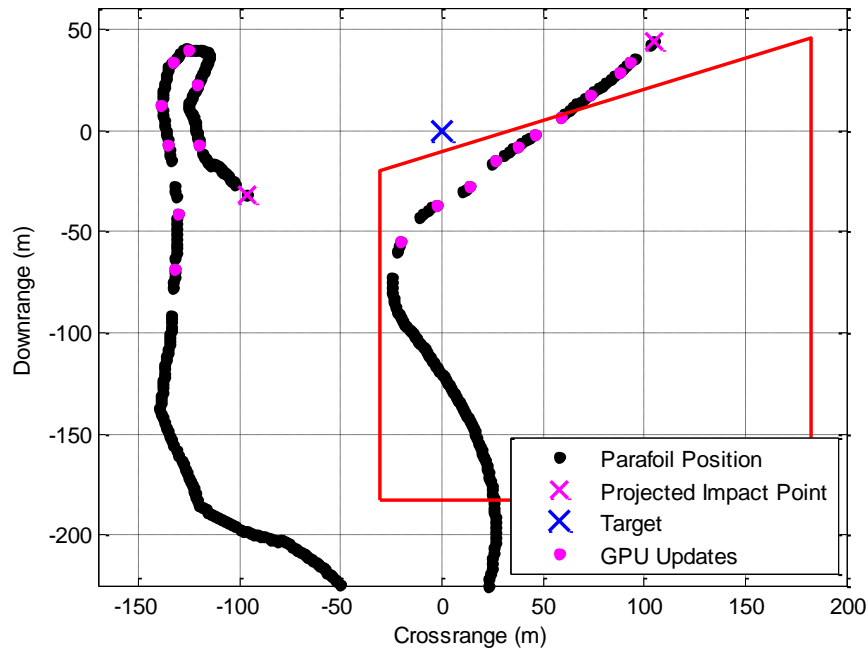


Fig. 20. GPU-Enabled Guidance, Example Trajectories 2 (right) and 3 (left).

Overall, 11 standard guidance and 12 GPU-based guidance flights were conducted. Wind direction shifts between flights were accounted for by rotating the guidance reference frame with respect to the North-East-Down frame by with the estimated wind direction, and thus all impact locations can be plotted together in the same guidance system coordinate frame. Figure 21 shows impact dispersion plots for standard guidance (left) and GPU-based guidance (right). In the standard guidance cases, a target-centered CEP of 46 m is observed, and 8 of 11 cases landed within the constraint. GPU-based guidance cases exhibited a CEP of 77 m, and 2 of the 12 cases landed within the constraint. In the standard guidance case, the tendency was for the vehicle to land upwind of the target (inside the constraint) due to a slight, but consistent, overprediction in wind magnitude. This is typical of guided parafoil performance since winds are usually stronger at higher altitudes where the initial terminal guidance path is first computed. However, in the GPU-based guidance cases, the trend is clear that the vehicle tended to miss in directions away from the constraint (i.e., downwind and to the left of the target area). Miss distance clearly grows in the GPU-based guidance cases since obstacle avoidance is considered, which corroborates trends in simulation predictions shown in Figures 11 and 12. Furthermore, note the high degree of similarity between the experimental GPU-based guidance dispersion pattern and the simulated dispersion pattern shown in the bottom of Figure 10, providing validation for the simulation results. Finally, note that two GPU-based guidance trajectories in Figure 21 landed in the constraint. It was determined that in these cases, and other cases as well, poor inner loop tracking performance and deadband in the control system played a major role in failing to execute the optimal guidance path. Thus, shallow turns commanded by the guidance algorithm typically were not executed well, and more moderate turn commands sometimes resulted in very sharp turns. Potentially, an improved inner-loop control and/or an improved turn rate model within the 6DOF simulation may yield better accuracy and constraint avoidance performance beyond that observed here. Note that, due to this poor tracking performance, the CEP of 77 m observed in experiment was somewhat higher than the 52 m predicted in simulation as shown in Figure 10.

Overall, results indicate the proposed stochastic guidance solution is a practical algorithm that can be successfully implemented using commercial off-the-shelf components. Use of emerging embedded GPU

architectures, more powerful than the system used here, is likely to lead to improved performance since a richer set of candidate trajectories may be evaluated (corresponding to larger R_s as shown in Figures 13 and 14). Experimental performance largely matched that predicted in simulation, demonstrating the ability of the proposed guidance scheme to avoid obstacle regions near the target in the presence of wind uncertainty at the expense of increased miss distance.

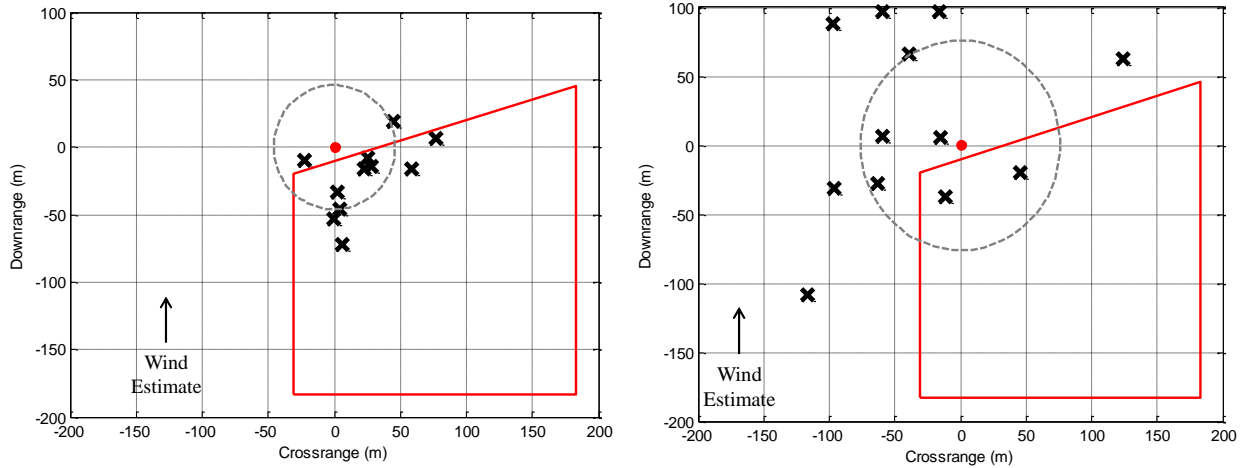


Fig. 21. Impact Point Dispersions for Standard Guidance (left), and GPU-Based Guidance (right). Dashed circle indicates 50% target-centered CEP.

5. Guidance Algorithm Compatibility

A primary strength of the proposed guidance algorithm is its compatibility with existing flight platforms. This compatibility is enabled through the decoupling of the guidance and control loops, which allows the GPU to function as a co-processor alongside existing autopilot hardware to generate terminal trajectory paths. This section describes an additional algorithm extension that allows the GPU guidance law to interface with existing guidance logic, as well as additional simulation results of a larger-scale parafoil which demonstrate scalability of the algorithm to practical airdrop systems. Finally, some recent hardware advancements are discussed that allow the GPU-based algorithm to be fielded at lower size, weight, and power compared to the flight test system described above.

A. Approach Phase Logic

While the GPU-based guidance algorithm formulated in Section 2.2 will provide an optimal solution from any initial location and heading, best performance is obtained when terminal guidance is initiated with the parafoil heading downwind at a lateral offset from the target. The primary reason for this is that the path parameterization used in the guidance law consists of a constant-rate turn followed by a straight section, also known as a standard hook-turn approach. If GPU guidance is initiated while heading downwind at a lateral offset from the target, it can provide a relatively gentle hook-turn geometry that will allow the parafoil to land into the wind (which is generally desirable for larger systems).

Many existing autopilot systems use a multi-phase approach geometry consisting of a loiter pattern, approach phase, and terminal maneuver phase. While some autopilot algorithms transition directly from loiter to terminal approach, more advanced strategies may leverage an intermediate approach stage that allows the parafoil to dissipate altitude far from the drop zone and then proceed to the target area at the appropriate time prior to terminal guidance. This section describes a terminal approach algorithm meant to link a general loiter pattern with the GPU-based terminal guidance algorithm described above.

Figure 22 shows a general approach geometry consisting of four phases: a homing phase in which the parafoil heads to the loiter area, a loiter pattern, approach phase, and terminal guidance path. Typically production autopilot systems include relatively simple logic to perform the homing and loiter phases. Upon reaching a certain

altitude, the vehicle exits the loiter pattern at the Loiter Exit Point (LEP) and proceeds through the approach phase to the turn initiation point (TIP), at which terminal maneuvers commence. The TIP is located a distance R laterally from the target (with respect to the wind) and a distance D upwind of the target. The approach path is configured such that the parafoil reaches the TIP heading exactly downwind at an altitude of z_0 .

To maintain compatibility with any general loiter configuration, an approach path generation algorithm is formulated to provide a flyable trajectory from the LEP to the TIP. This trajectory is given by a Dubins path [24] consisting of an initial constant-rate turn, a straight segment, and a final constant-rate turn as shown in Figure 23. The approach path generation problem can thus be stated as follows:

Given: Initial parafoil state at LEP (x_a, y_a, z_a, ψ_a) , final parafoil state at TIP (x_d, y_d, z_d, ψ_d) ,
and desired turn rate $\dot{\psi}_{des}$

Find elapsed times for each turn and for the straight segment given by t_a , t_b , and t_c respectively.

Since each of the two turns can be taken to the right or left, there are four solutions for (t_a, t_b, t_c) which obviously differ in the total time required to complete the path. To solve the trajectory generation problem, let the first and second turn rates be given as $\dot{\psi}_{des}^{(1)} = \pm \dot{\psi}_{des}$ and $\dot{\psi}_{des}^{(2)} = \pm \dot{\psi}_{des}$ respectively. Note that (x_b, y_b) may be written as,

$$x_b = x_a + \int_0^{t_a} W_x + V_{ss} \cos \psi dt = x_a + W_x t_a + \frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} (\sin(\dot{\psi}_{des}^{(1)} t_a + \psi_a) - \sin \psi_a) \quad (11)$$

$$y_b = y_a + \int_0^{t_a} W_y + V_{ss} \sin \psi dt = y_a + W_y t_a - \frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} (\cos(\dot{\psi}_{des}^{(1)} t_a + \psi_a) - \cos \psi_a) \quad (12)$$

where for simplicity it is assumed that the parafoil arrives at the LEP at $t = 0$. Likewise, (x_c, y_c) is given by,

$$x_c = x_b + W_x (t_b - t_a) + V_{ss} (t_b - t_a) \cos(\dot{\psi}_{des}^{(1)} t_a + \psi_a) \quad (13)$$

$$y_c = y_b + W_y (t_b - t_a) + V_{ss} (t_b - t_a) \sin(\dot{\psi}_{des}^{(1)} t_a + \psi_a) \quad (14)$$

Given the expression for (x_c, y_c) , the location of the turn initiation point can be written as,

$$x_d = x_c + W_x (t_c - t_b) + \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} (\sin \psi_w - \sin(\dot{\psi}_{des}^{(1)} t_a + \psi_a)) \quad (15)$$

$$y_d = y_c + W_y (t_c - t_b) - \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} (\cos \psi_w - \cos(\dot{\psi}_{des}^{(1)} t_a + \psi_a)) \quad (16)$$

where ψ_w is the wind heading. Substituting equations (11)-(14) into (15) and (16) yields:

$$x_d = x_a + W_x (t_a + t_b + t_c) + \left(\frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} - \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \right) \sin(\dot{\psi}_{des}^{(1)} t_a + \psi_a) - \frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} \sin \psi_a + \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \sin \psi_w \quad (17)$$

$$y_d = y_a + W_y (t_a + t_b + t_c) + \left(-\frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} + \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \right) \cos(\dot{\psi}_{des}^{(1)} t_a + \psi_a) + \frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} \cos \psi_a - \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \cos \psi_w \quad (18)$$

Recognizing that $t_c = (\psi_w - \dot{\psi}_{des}^{(1)} t_a - \psi_a) / \dot{\psi}_{des}^{(2)}$, equations (17) and (18) can be rewritten as,

$$x_d = x_a + W_x \left(t_a + t_b + \frac{\psi_w - \dot{\psi}_{des}^{(1)} t_a - \psi_a}{\dot{\psi}_{des}^{(2)}} \right) + \left(\frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} - \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \right) \sin(\dot{\psi}_{des}^{(1)} t_a + \psi_a) - \frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} \sin \psi_a + \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \sin \psi_w \quad (19)$$

$$y_d = y_a + W_y \left(t_a + t_b + \frac{\psi_w - \dot{\psi}_{des}^{(1)} t_a - \psi_a}{\dot{\psi}_{des}^{(2)}} \right) + \left(-\frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} + \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \right) \cos(\dot{\psi}_{des}^{(1)} t_a + \psi_a) + \frac{V_{ss}}{\dot{\psi}_{des}^{(1)}} \cos \psi_a - \frac{V_{ss}}{\dot{\psi}_{des}^{(2)}} \cos \psi_w \quad (20)$$

Since the TIP location (x_d, y_d) is given, the only unknowns in equations (19) and (20) are t_a and t_b . By constraining $t_a \in [0, 2\pi / \dot{\psi}_{des}^{(1)}]$, these algebraic equations may be solved numerically for t_a and t_b (using for instance a Levenberg-Marquardt algorithm [25]) thereby generating the approach trajectory.

Four possible trajectory solutions always exist for (19) and (20) corresponding to the first turn in the right or left direction, and the second turn in the right or left direction. Each solution has a total elapsed time to completion given as $t_{total} = t_a + t_b + t_c$. Assuming a constant descent rate V_z , the completion time may be transformed into a completion altitude given by V_z / t_{total} . In practical implementations, each of the four possible trajectories can be computed at regular intervals while in the loiter pattern, and the path with the minimum time to completion (t_{total}^{min}) selected. If it is desired that the parafoil arrive at the TIP at altitude z_d , then the critical altitude for loiter exit can be computed as $z_{crit} = z_d + V_z / t_{total}^{min}$. Thus, if at any time in the loiter pattern the altitude drops below the computed z_{crit} for the current parafoil state, the parafoil exits the loiter pattern and begins the approach trajectory.

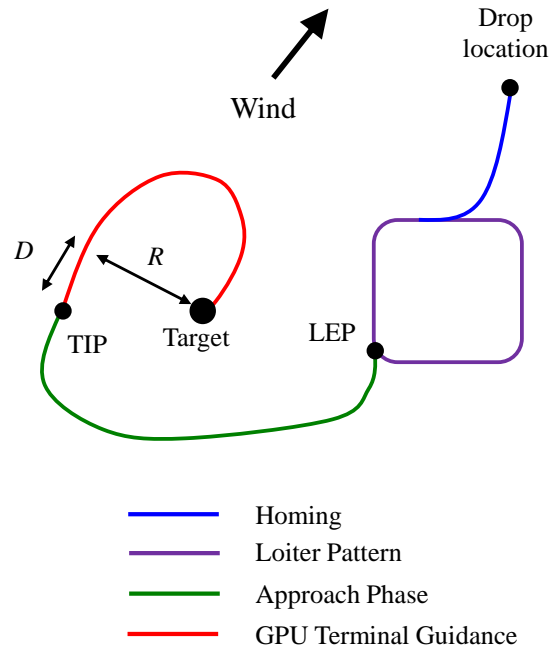


Figure 22. Multi-Phase Parafoil Approach Geometry. Any type of loiter pattern may be used (circular, square, figure 8, etc.).

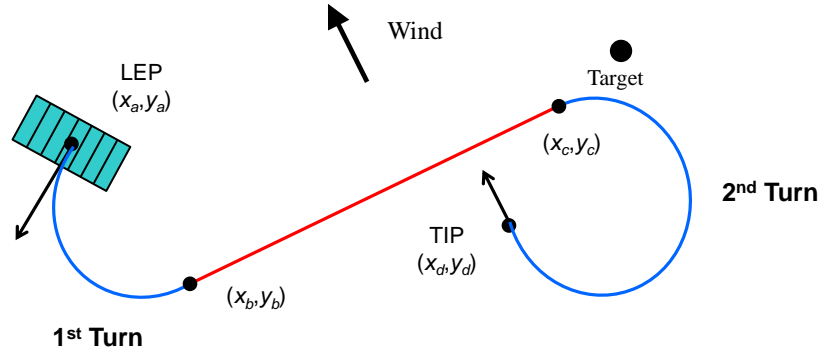


Figure 23. Dubins Path Approach Phase Segments.

B. Example Simulations

Example simulations are performed using a larger parafoil system than that described in Section 3. The purpose of these simulations is to show how the approach generation algorithm can link GPU-based terminal guidance with standard loiter paths flown by production autopilot systems. The larger example vehicle used here has a canopy span of 2.1 m, total mass of 14 kg, and nominal forward flight speed of 11.3 m/s. The steady-state descent rate of this system is approximately 5.5 m/s in level flight. For the simulations conducted here, the system is released at an altitude of 915 m above the target, and 400 m offset laterally to the northeast. For the Dubins path approach, a desired turn rate of $\dot{\psi}_{des} = 30$ deg/s is used, and the TIP is located 75 m laterally and 30 m upwind from the target with respect to the wind direction. Furthermore, the desired altitude upon reaching the TIP is $z_d = 215$ m. Figure 24 shows the results of two example trajectories with winds directly to the North at a speed of 3.1 m/s. In the first trajectory (no obstacle), the cost function penalty in (6) penalizes accuracy only, and thus $k_g = 0$. In this case, the parafoil enters a predetermined loiter pattern northeast of the target (in green). At 1 sec intervals in the loiter pattern, Dubins approach paths are generated and the critical altitude computed. When the altitude reaches z_{crit} , the parafoil exits the loiter pattern and begins the approach phase (shown in blue). As shown in Figure 25, this occurred at an altitude of approximately 415 m, and the parafoil reached the TIP at an altitude of 223 m which is close to the desired value of $z_d = 215$ m. In this case, a sequence of two right turns was the optimal solution for the Dubins path approach. Upon reaching the TIP, the parafoil initiates GPU terminal guidance. The resulting trajectory path resembles a hook turn and, in the no-obstacle case (shown in black), results in a miss distance of 7 m. A second example case is performed using a rectangular obstacle placed in the southeast quadrant near the target with a cost function penalty of $k_g = 1000$. This terminal trajectory (shown in purple) takes a slightly more conservative approach path resulting in an increased miss distance of 15 m.

Monte Carlo simulations of 200 trajectories are performed for this example vehicle by randomly varying the drop altitude between 670 m and 1200 m, and by randomizing the winds between 0 and 8.5 m/s in magnitude and 0 and 2π in direction, with all distributions being uniform. Randomly varying the drop altitude ensures that the vehicle exits the loiter pattern in a random location for each trajectory. The results of this Monte Carlo simulation are shown in Figure 26. For the no-obstacle case, the target-centered 50% CEP is approximately 8 m. Using the same obstacle geometry and weighting as shown in Figure 24, the obstacle cases exhibited a 50% CEP of 13 m, while avoiding the constraint in 100% of cases. Note that this obstacle is intentionally placed very close to the target (within 3 m) to highlight the rather favorable performance of the guidance system, even in moderate winds up to 8.5 m/s.

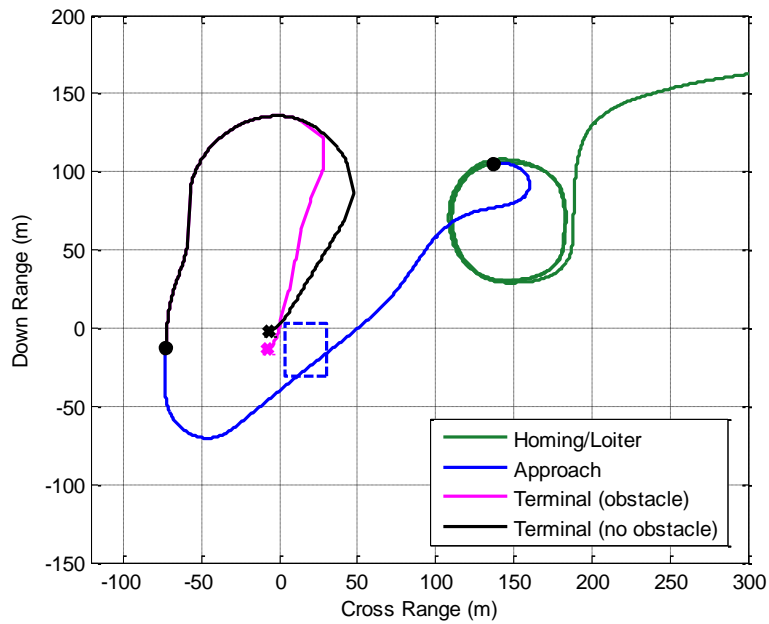


Figure 24. Example Complete Guidance Trajectories Including Loiter and Approach Segments.

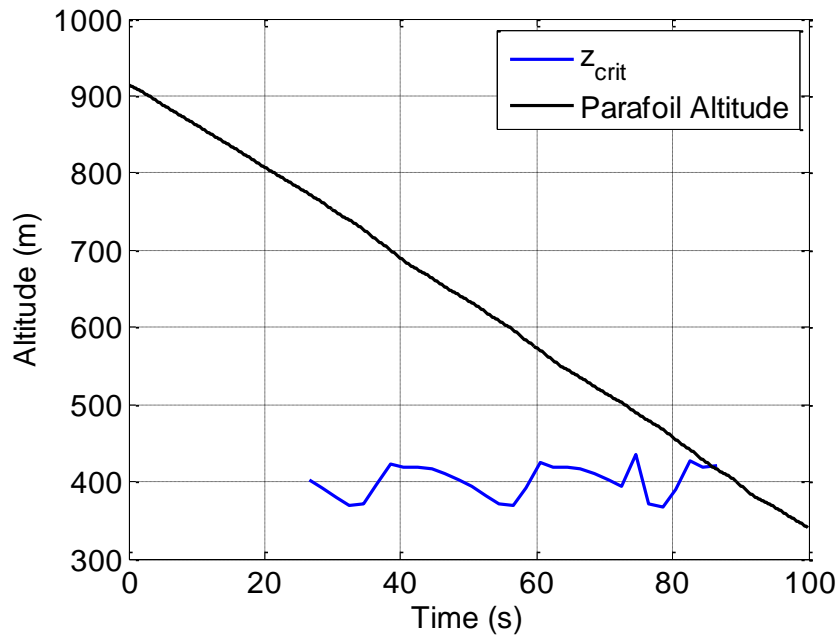


Figure 25. Computed Critical Altitude vs Time for Dubins Path Approach.

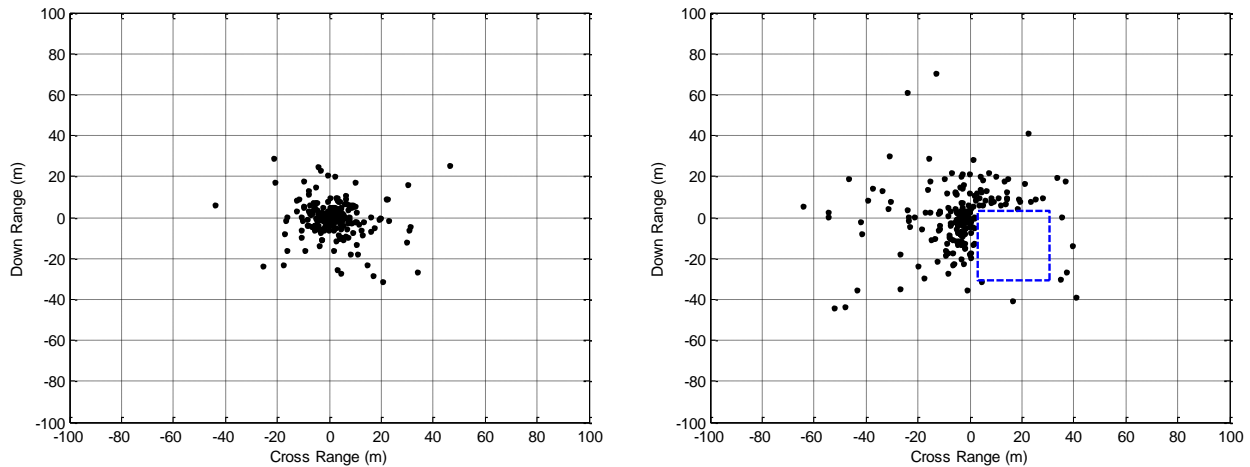


Figure 26. Monte Carlo Simulation Results for Larger Vehicle Full Guidance Simulation.

C. Improved Embedded GPU Hardware

Since the hardware experiments described in Sections 3 and 4 were performed, new embedded GPU hardware has become available in the form of the NVIDIA Jetson TK1 development kit. This development board, shown in Figure 27, features a quad-core ARM processor and 192-core Kepler-class GPU with 1.7 GB of GPU global memory and a GPU clock speed of 852 MHz. The current cost of a single board is \$192. These specifications mean the onboard GPU is significantly cheaper and more powerful than the Quadro-class GPU used in the flight experiments described in Sections 3 and 4. In addition the board footprint is significantly smaller at 4.5 in \times 4.5 in, and weighs less than 1 lb. An autopilot interface to the board may be easily established using the RS232 port on the Jetson board. Initial experiments in attaching this new GPU board to a standard parafoil autopilot and running the proposed terminal guidance algorithm has been successful, with hardware-in-the-loop simulations demonstrating significantly improved computational performance compared to the previous experimental setup. Note that inclusion of additional cores (192 with the Jetson vs 96 with the Quadro) allows for the use of a larger number of candidate trajectories R_s , which, as shown in Figures 13 and 14, generally translates to better algorithm performance in terms of accuracy and obstacle avoidance. As embedded GPU hardware becomes more advanced in terms of SWAP and computational capability, it is expected that the algorithm proposed here will provide better real-world performance and become increasingly practical.



Figure 27. NVIDIA's Jetson TK1 Development Board [26].

6. Conclusions

A stochastic parafoil terminal guidance algorithm has been presented and implemented on an experimental system. The algorithm selects a path to the target which is robust to wind disturbances through a statistical optimization procedure involving real-time Monte Carlo simulation. The optimization cost function trades delivery accuracy for obstacle avoidance, and may be tailored for specific terminal guidance scenarios. A practical implementation of the stochastic guidance scheme is demonstrated by integrating a standard autopilot processor with a GPU co-processor for real-time uncertainty propagation. Experimental results with a small flight test system demonstrate the guidance algorithm's capability to weigh the dual goals of accuracy and obstacle avoidance given wind uncertainty. These experimental results compare favorably to results using a deterministic guidance approach that assumes perfect knowledge of the winds, generally corroborating trends seen in simulation. Comprehensive simulation results demonstrate the guidance system's ability to avoid obstacle areas near the target in the presence of wind gusts, and the effect of various guidance system parameters on landing performance is quantified. Overall, results show that stochastic control through real-time GPU-based uncertainty propagation is a practical technique that can be implemented for robust guidance of autonomous parafoil systems.

Acknowledgements

This material is based upon work supported by NASA under Cooperative Agreement NNX13AP90A.

References

- [1] Hellstrom, E., Aslund, J., Nielsen, L., "Design of an Efficient Algorithm for Fuel-Optimal Look-Ahead Control," *Control Engineering Practice*, Vol. 18, No. 11, November 2010, pp. 1318-1327.
- [2] Kothare, M., Wan, Z., "Computationally Efficient Scheduled Model Predictive Control Algorithm for Control of a Class of Constrained Nonlinear Systems," in *Lecture Notes in Control and Information Sciences*, Vol. 358, 2007, pp. 49-62.
- [3] Duchaine, V., Bouchard, S., Gosselin, C., "Computationally Efficient Predictive Robot Control," *IEEE/ASME Transactions on Mechatronics*, Vol. 12, No. 5, October 2007, pp. 570-578.
- [4] Blondel, V., Tsitsiklis, J., "A Survey of Computational Complexity Results in Systems and Control," *Automatica*, Vol. 36, 2000, pp. 1249-1274.
- [5] Bars, R., Colaneri, P., de Souza, C., Dugard, L., Allgower, F., Kleimenov, A., Scherer, C., "Theory, Algorithms, and Technology in the Design of Control Systems," *Annual Reviews in Control*, Vol. 30, 2006, pp. 19-30.
- [6] Ilg, M., Rogers, J., Costello, M., "Projectile Monte Carlo Analysis Using A Graphics Processing Unit," 2011 AIAA Atmospheric Flight Mechanics Conference, Portland, OR, 2011.
- [7] Benney, R., Meloni, A., Cronk, A., Tiaden, R., "Precision Airdrop Technology Conference and Demonstration," 20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, 2009, Seattle, WA.
- [8] Tavan, S., "Status and Context of High Altitude Precision Aerial Delivery Systems," AIAA Guidance, Navigation, and Control Conference and Exhibit, 2006, Keystone, Colorado.
- [9] Slegers, N., Costello, M., "Model Predictive Control of a Parafoil and Payload System," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 4, 2005, pp. 816-821.

- [10] Slegers, N., Beyer, E., Costello, M. "Use of Dynamic Incidence Angle for Glide Slope Control of Autonomous Parafoils," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 3, 2008, pp. 585–596.
- [11] Slegers, N., Yakimenko, O. "Terminal Guidance of Autonomous Parafoils in High Wind-to-Airspeed Ratios," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 225, No. 3, 2011, pp. 336–346.
- [12] Ward, M. "Adaptive Glide Slope Control for Parafoil and Payload Aircraft," Doctoral Dissertation, Georgia Institute of Technology, Atlanta, GA, 2012.
- [13] Fowler, L., Rogers, J., "Bezier Curve Path Planning for Parafoil Terminal Guidance," *Journal of Aerospace Information Systems*, In Press, 2013.
- [14] Gimadieva, T. Z., "Optimal Control of a Guiding Gliding Parachute System," *Journal of Mathematical Sciences*, Vol. 103, No. 1, 2001, pp. 54–60.
- [15] Rademacher, B., Lu, P., Strahan, A., Cerimele, C., "In-flight Trajectory Planning and Guidance for Autonomous Parafoils," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 6, 2009, pp. 1697–1712.
- [16] Rogers, J., Slegers, N., "Robust Parafoil Terminal Guidance Using Massively Parallel Processing," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 5, September-October 2013, pp. 1336-1345.
- [17] Slegers, N., Rogers, J., "Terminal Guidance for Complex Dropzones Using Massively Parallel Processing," AIAA Aerodynamic Decelerator Systems Technology Conference and Exhibit, Daytona Beach, FL, March 25-28, 2013.
- [18] Doucet, A., de Freitas, N., and Gordon, N., Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York, 2001, pp. 6–14.
- [19] Risken, H., The Fokker-Planck Equation: Methods of Solution and Applications, Springer, New York, 1989, pp. 32–62.
- [20] Gorman, C., Slegers, N., "Comparison and Analysis of Multi-body Parafoil Models With Varying Degrees of Freedom," 21st AIAA Aerodynamic Decelerator Systems Technology Conference, 2011, Dublin, Ireland.
- [21] SECO Corporation, "CARMA Devkit – SECO," <http://www.nvidia.com/object/carma-devkit.html>, obtained December 20, 2012.
- [22] Ward, M., Costello, M., Slegers, N., "Specialized System Identification for Parafoil and Payload Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 2, March-April 2012, pp. 588-597.
- [23] van Belle, G., Statistical Rules of Thumb, Wiley Series in Probability and Statistics, John Wiley and Sons, Hoboken, NJ, 2008, p. 29.
- [24] Dubins, L. E., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, Vol. 79, No. 3, July 1957, pp. 497-516.
- [25] Levenberg, K., "A Method for the Solution of Certain Non-Linear Problems in Least Squares," *Quarterly of Applied Mathematics*, Vol. 2, 1944, pp. 164-169.
- [26] NVIDIA Corporation, "NVIDIA Jetson TK1 Development Kit," http://developer.download.nvidia.com/embedded/jetson/TK1/docs/Jetson_platform_brief_May2014.pdf, obtained March 4, 2015.