3-2015

Experimental Investigation of Stochastic Parafoil Guidance using a Graphics Processing Unit

Nathan Slegers  
George Fox University, nslegers@georgefox.edu

Andrew Brown  
Georgia Institute of Technology - Main Campus

Jonathan Rogers  
Georgia Institute of Technology - Main Campus

Follow this and additional works at: http://digitalcommons.georgefox.edu/mece_fac

Recommended Citation  
Slegers, Nathan; Brown, Andrew; and Rogers, Jonathan, "Experimental Investigation of Stochastic Parafoil Guidance using a Graphics Processing Unit" (2015). Faculty Publications - Department of Mechanical and Civil Engineering. Paper 54.  
http://digitalcommons.georgefox.edu/mece_fac/54

This Article is brought to you for free and open access by the Department of Mechanical and Civil Engineering at Digital Commons @ George Fox University. It has been accepted for inclusion in Faculty Publications - Department of Mechanical and Civil Engineering by an authorized administrator of Digital Commons @ George Fox University. For more information, please contact arofe@georgefox.edu.
Experimental investigation of stochastic parafoil guidance using a graphics processing unit

Nathan Slegers a, Andrew Brown b, Jonathan Rogers b,*

a Department of Mechanical Engineering, George Fox University, Newberg, OR 97132, United States
b Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, United States

ARTICLE INFO

Keywords:
Graphics processing unit
GPU
Optimal control
Parallel processing
Parafoil

ABSTRACT

Control of autonomous systems subject to stochastic uncertainty is a challenging task. In guided airdrop applications, random wind disturbances play a crucial role in determining landing accuracy and terrain avoidance. This paper describes 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. Building upon prior theoretical developments, this paper explores performance of the stochastic guidance law compared to standard deterministic guidance schemes, particularly with respect to obstacle avoidance. Flight test results are presented comparing the proposed stochastic guidance algorithm with a standard deterministic one. 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. 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.

1. Introduction

Current control algorithms for robotic systems are largely serial in nature. In the classical control paradigm, a sequence of serial steps is used to arrive at desired control inputs from feedback measurements. 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 (for example, see (Hellstrom, Aslund & Nielsen 2010; Kothare & Wan, 2007; Duchaine, Bouchard & Gosselin 2007)). For deterministic systems at least, a variety of linear and nonlinear control algorithms now exist that offer suitable tradeoffs between computational complexity and performance (Blondel & Tsitsiklis, 2000).

Stochastic systems, or systems subject to large dynamic uncertainty, 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 (Blondel & Tsitsiklis, 2000). 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 (Bars et al., 2006). 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, Rogers, and Costello (2011) demonstrated the feasibility of executing real-time
Monte Carlo simulations of air vehicle trajectories on graphics processing units (GPUs) give their data-parallel execution model.

The work described here explores this heterogeneous computing approach to stochastic control in the context of autonomous paraplane guidance and control. Paraplanes are a type of controllable parachute used to deliver cargo and supplies to a specific target via release from an aircraft. In general, paraplane 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 paraplane landing accuracy is limited to approximately 100 m, which is unsuitable for landing in environments with complex terrain or obstacles near the target (Benney, Meloni, Cronk & Tiaden, 2009; Tavan, 2006). Numerous authors have explored a variety of optimal paraplane guidance strategies including model predictive control (Slegers & Costello, 2005), direct glide slope control (Slegers, Beyer & Costello, 2008), and parametric path optimization (Slegers & Yakimenko, 2011; Ward, 2012; Fowler & Rogers, 2014). Gimadieva (Gimadieva, 2001) and Rademacher, Lu, Strahan, and Cerimele (2009) 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 uncertain 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 paraplane terminal guidance in which Monte Carlo simulation is performed in real-time on a GPU co-processor (Rogers & Slegers, 2013; Slegers & Rogers, 2013). 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 Rogers and Slegers (2013) and Slegers and Rogers (2013) 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 Rogers and Slegers (2013) and Slegers and Rogers (2013) 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

Nomenclature

\[ \mathbf{a}_{\text{T}} \] Acceleration of the paraplane mass center with respect to the inertial frame
\[ C_i \] Dimensionless paraplane aerodynamic coefficients
\[ D \] Distance of the turn initial point with respect to the target
\[ e_i \] Estimated mean miss distance for trajectory candidate \( i \)
\[ I \] Inertia matrix of the paraplane-payload vehicle about the mass center
\[ F_{\text{AP}}, F_{\text{AS}} \] Aerodynamic forces on the paraplane and payload
\[ F_{\text{AM}} \] Force on the paraplane-payload due to apparent mass effects
\[ F_{\text{WP}}, F_{\text{WS}} \] Weight forces on the paraplane and payload
\[ i_{\text{T}}, j_{\text{T}}, k_{\text{T}} \] North-East-Down reference frame unit vectors
\[ j_i \] Cost associated with trajectory candidate \( i \)
\[ k_i \] Cost function weight for obstacle avoidance
\[ L \] Distance to target along target line
\[ M_{\text{AP}}, M_{\text{AS}}, M_{\text{AM}} \] Moments from aerodynamics of the paraplane, payload, and apparent mass moments, all about the mass center
\[ M \] Number of possible yaw rate values used in discretization
\[ N \] Number of possible yaw rate values used in discretization
\[ n_{\text{ip}} \] Estimated probability of obstacle impact for trajectory candidate \( i \)
\[ R \] Final turn radius
\[ s \] Paraplane 6DOF state vector
\[ \text{TIP} \] Turn initiation point
\[ t_0 \] Time of predicted impact
\[ t_1 \] Time final turn begins
\[ t_2 \] Time final approach begins
\[ T_{\text{app}} \] Time of final predicted impact
\[ V_{\text{h}}, V_{\text{z}} \] Paraplane horizontal airspeed and vertical speed
\[ W_{\text{c}}, W_{\text{y}} \] Target frame wind speed components
\[ x, y, z \] Paraplane inertial positions in the North-East-Down frame
\[ x, y \] Paraplane inertial velocity components in the North-East-Down frame
\[ \zeta_i \] Final approach angle
\[ \psi \] Paraplane Euler yaw angle
\[ \psi_{\text{max}} \] Maximum paraplane yaw rate
\[ \omega_{\text{fl}} \] Angular velocity of the paraplane-payload vehicle with respect to inertial frame
exploration of the effects of various guidance system parameters (including cost function weightings, optimization grid size, and the number of candidate trajectories considered).

The paper is organized as follows. First, the parafoil guidance problem is formulated and the stochastic guidance law is summarized. A description of the flight test vehicle is provided, and the guidance electronics package is described including discussion of workload distribution across the autopilot and GPU co-processor. A comprehensive set of simulation results is presented demonstrating performance of the stochastic guidance algorithm as a function of various parameters, including cost function values, resolution of the optimization grid, and the number of candidate trajectories selected for uncertainty quantification. Finally, experimental results are shown comparing performance of the proposed stochastic guidance algorithm with a standard deterministic scheme. Experimental dispersion results demonstrate the practical utility of the guidance algorithm with a standard deterministic scheme. Experiments are shown comparing performance of the proposed stochastic guidance algorithm as a function of various parameters, including cost function values, resolution of the optimization grid, and the number of candidate trajectories considered.

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 Fig. 1, where $i_T$ and $j_T$ are the 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 $\psi_F$. This is deemed the final approach. In practical scenarios, $R$, $T_{app}$, and $\psi_F$ 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 $\psi_F$ at zero altitude above the ground. In Rogers and Slegers (2013), the authors provided a closed form solution to this open-loop planning problem by assuming a constant rate turn $\psi$, a nearly constant vehicle descent rate $V_h$, a constant horizontal airspeed $V_W$, 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 the following equation:

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

Then, integrating the horizontal velocity components along $i_T$ and $j_T$ from $t_0$ to $t_2$ and solving for the final turn time $t_1$ yields

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

Finally, by integrating velocity components along $i_T$ and $k_T$ from $t_0$ to $t_2$ a solution for the downrange distance of the turn initiation point, $D$, can be found according to the following equation:

$$D = -W_x(t_1 - t_0) - (V_h/\psi) \sin \psi_F - \frac{(W_x + V_x \cos \psi_F)(W_z + V_z)}{V_h (1 - \cos \psi_F)} \left( \frac{1}{2} - \frac{1}{2} \frac{1}{W_x + V_x} - \frac{1}{W_z + V_z} \right)$$

Eqs. (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 Rogers and Slegers (2013) and have been omitted here for brevity.

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 $(x_T, y_T, z_T)$ and orientation $\psi_0$ at a given time to vary from the states which satisfied the initial terminal guidance problem. In this case, depicted in Fig. 2, a new guidance solution in the form of an updated $\psi$ and $\psi_F$ must be computed. This boundary value problem may be formulated as follows: given the parafoil location $(x_T, y_T, z_T)$ at time $t_0$ with yaw angle $\psi_0$, find $\psi$ and $\psi_F$ 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 $\psi$ and $\psi_f$ is computed given dynamic motion constraints. This optimization problem is described as follows:

Given $x_0$, $y_0$, $z_0$, $\psi_0$, $f_0$, and estimates of $W_x$, $W_y$, find $\psi$ and $\psi_f$ that minimizes

$$J_f = f(s, t)$$

subject to

$$x = x_0 + W_x(t_1 - t_0) + \frac{V_y}{\psi_f}(\sin \psi_f - \sin \psi_0) + (t_2 - t_1)(W_x + V_h \cos \psi_f)$$

$$y = y_0 + W_y(t_1 - t_0) - \frac{V_y}{\psi_f}(\cos \psi_f - \cos \psi_0) + (t_2 - t_1)(W_y + V_h \sin \psi_f)$$

$$t_2 = \frac{t_1}{\psi_f} + t_0$$

$$t_1 = \frac{W_x - W_y}{\psi_f} + t_0$$

The constraints outlined in Eq. (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 ($s$), and thus the problem described by (4) and (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 ($\psi$, $\psi_f$), Eqs. (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) and (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 Fig. 3, where an airfoil 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) and (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.

Uncertainty propagation for a dynamic system can be performed in a number of ways from Monte Carlo simulation (Doucet, de Freitas & Gordon, 2001) to direct methods such as the Fokker–Planck–Kolmogorov (FPK) equation (Risken, 1989). In this case, the point mass model employed in Eqs. (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 (Ilg et al., 2011) 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 Fig. 4. First, a complete set of constant-rate turn trajectory candidates is generated by discretizing $\psi$ into $M$ possible values within $[-\psi_{\text{max}}, \psi_{\text{max}}]$, where $\psi_{\text{max}}$ is the parafoil’s maximum desired turn rate. Likewise, $\psi_f$ is discretized into $N$ values between 0 and $2\pi$. 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 $R_t$ 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.

Monte Carlo simulation is performed for each of the $R_t$ 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 Gorman and Slegers (2011) and Rogers and Slegers (2013) 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 $R_t$ 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

![Fig. 3. Aerial view of a military forward operating base.](image)

![Fig. 4. GPU-based guidance algorithm.](image)
the $R_i$ trajectory candidates. The final step is to select a robust trajectory among possible trajectories, considering both 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 providing the smallest cost. This trajectory is used during terminal guidance for tracking 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 errors in the 6DOF model, trajectory tracking errors, 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 Rogers and Slegers (2013). In this paper, 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


\[ J_i = e_i + k_p p_i \]  

(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 $k_p$ is a user-defined constant 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 providing the smallest cost. This trajectory is used during terminal guidance for tracking 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 errors in the 6DOF model, trajectory tracking errors, 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 Rogers and Slegers (2013). In this paper, 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.

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

Fig. 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 (SECO Corporation, 2012). 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 $R_i$, 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 Fig. 6. Including the autopilot processor, this embedded electronics package has over 100 cores with a total mass of under 1 kg including batteries.

In general, it is desired that GPU-based trajectory optimization exhibit latencies of less than about 3 s so guidance updates can be computed in a feedback manner during terminal guidance (Rogers & Slegers, 2013). 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 $\psi_{\text{max}} = 30$ 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 flight test pattern is shown in Fig. 8 using an aerial photo.

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, Costello, and Slegers (2012) 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.

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 (Ward et al., 2012), estimates of the steady-state wind components $\vec{W}_x$ and $\vec{W}_y$ are computed, along with estimated vehicle airspeed $\vec{V}_h$. Once the mean wind and airspeed estimates are computed, the variance of the wind components can be estimated according to the following equation:

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

where $n$ is the number of GPS measurements taken during the box pattern, and $W^i_x,W^i_y$ are solutions to the wind triangle at the $i$th GPS measurement, i.e.

$$W^i_x = \vec{V}_h \cos(\psi^i) - \dot{x}_i \quad W^i_y = \vec{V}_h \sin(\psi^i) - \dot{y}_i$$

Note that in Eq. (10) $\dot{x}_i, \dot{y}_i$ are the $i$th measured GPS velocities, and $\psi^i$ is the $i$th measured yaw angle. The estimated mean wind components and wind variances in Eq. (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.
Note that, as expected, the terminal guidance paths become more
z
other than what provided by van Belle (2008) using a standard
d values of
simulated GPU-based terminal guidance trajectories with differ-
dized distance of 0.5 for the impact miss distance). Fig. 9 shows three
trajectories per Monte Carlo run was selected as 60. Note that the
in Eq. (6). In computing the cost function in (6),
section, the kinematic open-loop terminal guidance algorithm is
ferred 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 guid-
ance 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 avoid-
adence 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 m, −183 m), (−30 m, −20 m), (183 m, 45 m), and (183 m, −183 m). The target is located at the origin. Note that while this
constraint geometry is simpler than those considered by the authors
in Slegers and Rogers (2013) (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 Eq. (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 (2008) using a standard-
dized distance of 0.5 for the impact miss distance). Fig. 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° and 2.4 m/s, and standard deviations
of 30° 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°. Figs. 10–12 show the results of these simulations. In
Fig. 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. Fig. 11 shows that, for values of $k_g > 15$, impacts
within the constraint are largely eliminated. At the same time, Fig. 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 Fig. 3.
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 Eq. (4), where \( e \) is the estimated miss distance from Eq. (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 \( R_s = 15 \) was used. Figs. 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 Figs. 13–14. Since Monte Carlo simulations of 60 trajectories are performed for each of the 40 candidates, a total of 2400 trajectory simulations are generated every guidance cycle. Note that these values are significantly less than those used in Rogers and Slegers (2013) since the embedded GPU used here has significantly less processing power than the desktop model used in prior simulation studies.

4.2. Experimental results: standard guidance

A set of 11 flight experiments was performed using standard guidance. Figs. 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 $\pm y$ 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 Fig. 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.

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 Fig. 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, Fig. 17 shows a GPU-based guidance experimental case. In this flight, GPU updates are initiated as the parafoil nears the constraint. Fig. 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. Fig. 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 Figs. 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. Fig. 19 shows the optimal trajectories provided by the GPU co-processor, where a negative turn rate indicates a left turn.

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 Fig. 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 Slegers and Rogers

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.

Fig. 20. GPU-enabled guidance, Example Trajectories 2 (right) and 3 (left).
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 Fig. 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.

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. Fig. 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 Figs. 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 Fig. 10, providing validation for the simulation results. Finally, note that two GPU-based guidance trajectories in Fig. 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 Fig. 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 that 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, as shown in Figs. 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.

5. 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.
Acknowledgments

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

References


