JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 30, No. 6, November–December 2007

Simulation and Flight Test of Autonomous Aircraft Estimation, Planning, and Control Algorithms

Mark E. Campbell,* Jin-woo Lee,† and Eelco Scholte‡
Cornell University, Ithaca, New York 14853
and
David Rathbun§
The Insitu Group, Bingen, Washington 98605

DOI: 10.2514/1.29719

The development, summary, and validation of real time autonomous estimation, planning, and control algorithms for nonlinear aircraft are presented. Two square root estimation algorithms are developed and used to estimate the state dynamics, parameters, and uncertainty bounds of a nonlinear aircraft in real time. A square root sigma point filter approximates the uncertain state using a finite set of points, while a square root extended set-membership filter bounds the states and parameters of the nonlinear system using ellipsoidal sets. The estimation methods are integrated into online optimized model predictive control methodology for fast reconfigurable control, and a path planner based on streamlines for fast path planning around obstacles and for target pursuit. The performances of the estimation, planning, and control techniques are tested in flight using the SeaScan autonomous aircraft. The SeaScan platform is briefly described, and real time hardware in the loop and flight data are presented for a variety of applications using each of the algorithms.

Introduction

AUTONOMY implies a degree of self-regulation inherent in a system's operation and is a key technology for future high-performance and remote operation applications [1]. The promise of autonomy includes 1) a reduced need for human intervention, which is especially important for dull and/or dangerous missions; 2) increased performance range and capabilities; 3) extended operation life; and 4) decreased costs. Examples of important future applications that will rely on developments in the area of autonomy and control include satellite clusters, deep space exploration, air traffic control, and battlefield management with multiple uninhabited aerial vehicles (UAVs) and ground operations.

One of the most challenging applications is real time flight planning and control for single and multiple UAVs. For this application, an autonomous software architecture requires many online functions such as trajectory planning [2–4], control reconfiguration [5–8], estimation and fault detection [9,10], and mission planning [4,11]. In addition, effective human command of many vehicles requires information flow and fusion at several levels to facilitate (semi)autonomous operation. Reference [12] presents a summary of recently developed autonomous control algorithms for single vehicles, whereas [13] yields similar summaries for multiple vehicles.

As these algorithms have matured, one of the biggest challenges in UAV research has emerged: how to validate the algorithms in a realistic setting. This is especially challenging for UAVs because of the enormous overhead involved in their development, cost and robustness, and safety during operations. Several recent sets of flight tests are worth noting. Several groups in the Defense Advanced Research Projects Agency (DARPA) software enabled control (SEC) program worked with the Boeing Company to demonstrate a wide variety of algorithms (model predictive control, hybrid control, fault detection) using Boeing's J-UCAS T-33 flying test bed, along with an F-15E Strike Eagle fighter. With the T-33's J-UCAS avionics operating in an autonomous mode and without the use of a ground controller, a weapons systems operator in the back seat of the F-15E successfully tasked the T-33 through a series of flight maneuvers specifically designed to address challenges that will face more maneuverable UAVs. The real time implementation of these algorithms was an excellent contribution. A key element of these tests was the use of the open control platform (OCP), proposed [14] and realized [15] by the Boeing Company. The middleware of the OCP provides the software layer isolating the application from the underlying target platform, thus allowing the integration of different engineering tools.

In the multiple-UAV area, several universities have developed test beds for multiple vehicle flight operations [16–18]. These test beds, which typically have small windows for flight (∼30 min) and most of the computing power on the ground, still enable successful demonstrations of algorithms such as cooperative path planning and mission planning for small numbers of vehicles.

This paper takes a set of autonomous control algorithms from theory to real time development to hardware in the loop implementation and finally to demonstration in real time flight controls. The algorithms include 1) a square root stochastic nonlinear sigma point estimator [19], used for real time state and aerodynamic parameter estimation; 2) a square root nonlinear set-membership filter [20], used to provide state estimates with bounded uncertainties for control; 3) a nonlinear model predictive control implementation coupled with a bounded estimator [21], used to integrate the bounded set estimation as constraints and a computational implementation that enables the production of real time control signals even if computational resources are limited; and 4) a streamline path planner [2,22], used to provide fast path options in uncertain environments. This paper presents unique, integrated real time implementations of the algorithms along with their demonstration in end-to-end real time hardware in the loop and flight experiments.

The test bed used in this work is the SeaScan UAV. The SeaScan is a long endurance UAV developed by the Insitu Group for weather

Received 12 January 2007; revision received 4 June 2007; accepted for publication 6 June 2007.

*Associate Professor, Sibley School of Mechanical and Aerospace Engineering; mc288@cornell.edu. Associate Fellow AIAA.
†Research Associate, Sibley School of Mechanical and Aerospace Engineering; jl206@cornell.edu.
‡Research Assistant, Sibley School of Mechanical and Aerospace Engineering; es244@cornell.edu.
§Chief Engineer for Software, 118 East Columbia River Way; david.rathbun@insitugroup.com.

reconnaissance, crossing the Atlantic, and most recently deployment in Iraq. As part of this work, the authors developed a payload capability for the SeaScan [23,24] which interfaces directly to the SeaScan's primary processor to provide capability for real time estimation, planning, and control in flight. All hardware in the loop and flight test results presented in this paper were implemented in real time, within this payload processor.

The outline of the paper is as follows. First, real time implementations of each of the four algorithms for autonomous precision control are detailed. Next, the SeaScan UAV is described, including the hardware, payload development and implementation, hardware in the loop simulator, and models used in this work. Finally, hardware in the loop and flight test results are presented for three cases: 1) state and aerodynamic model estimation with and without an actuator failure, 2) target tracking and pursuit, and 3) sensing and avoidance of a pop-up threat.

Algorithms for Real Time Autonomous Control

The concept of autonomous control is typically described as a layered, feedback based architecture with degrees of autonomy [1]. Figure 1 shows the lower and middle layers of this architecture and interconnections. Low and midlevel control provides feedback compensation and trajectory generation, respectively; fault detection provides information to the control/planning blocks in order to recover from typical failures within the system; active state models [9] provide accurate model based information to the other components so that they can perform adequately.

Most control and fault detection methods are model based; therefore, a key building block in the autonomous control architecture is estimation/modeling technology. The focus of this paper is on the development of estimation algorithms, their integration into control and planning, and the implementation of the full autonomous system in real time. More specifically, this paper develops real time implementations of the following algorithms:

1) square root sigma point filter (SR-SPF): A square root nonlinear stochastic estimator used to estimate aircraft states and aerodynamics in real time.

2) square root extended set-membership filter (SR-ESMF): A square root nonlinear set estimator used to provide estimates and bounded constraints to online model predictive control.

3) robust nonlinear model predictive control (NMPC): A robust implementation for low level aircraft control that integrates uncertainty bounds from the SR-ESMF with a contractive model predictive approach.

4) streamline path planner (SPP): A fast, accurate, path planning approach in an uncertain environment, rooted in aerodynamics theory; the SPP can stand alone or be integrated into the NMPC.

This section summarizes each of these algorithms.

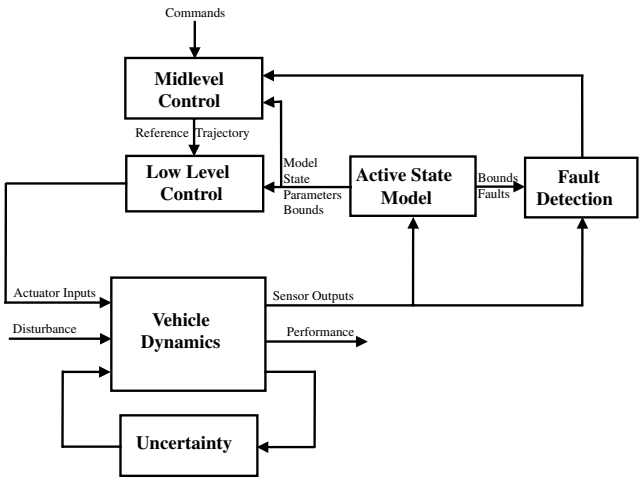


Fig. 1 Function blocks of the SeaScan control system.

Square Root Sigma Point Filter (SR-SPF)

A stochastic active state model requires a nonlinear estimation algorithm that performs well online, does not diverge if there are jumps in the sensor (such as when there is damage), and is robust to various parameters such as tuning and initial conditions. The sigma point filter (SPF), originally introduced as the unscented Kalman filter (UKF) [25], is a unique approach to nonlinear filtering where the distributions are approximated with a finite set of points. These points, called sigma points, are then propagated through the nonlinear dynamics. The mean and covariance of the distribution are then calculated as a weighted sum and outer product of these propagated points. Because nonlinear dynamics are used without approximation [i.e., no derivatives are calculated as in the extended Kalman filter (EKF)], it is much simpler to implement and better results are expected.

The performance of the SPF has been shown to be analytically similar to a truncated second-order EKF, but without the need to calculate Jacobians or Hessians of the dynamics [25]. Past work and the study in [26] include more details on comparing the SPF to the EKF. The results and observations indicate that the SPF is 1) more accurate from one time step to the next, 2) less susceptible to divergence, 3) equivalent in numerical computation, 4) less sensitive to tuning of process noise, 5) less sensitive to slow sampling rates, and 6) more robust to initial uncertainties (and jumps in the data) as compared to the EKF. These results support the conclusion that the SPF is better than the EKF for real time estimation applications such as UAVs.

The real time implementation of the SPF requires a numerically stable square root implementation, as shown in [19] for real time aerodynamic model estimation [26]. Because the SPF algorithm uses the covariance matrix square root at each time step, a square root implementation is an elegant as well as numerically robust solution.

Consider the general discrete, nonlinear state-space system:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \tag{1}$$

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k) \tag{2}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{w}_k \in \mathbb{R}^n$ is the disturbance, $\mathbf{y}_{k+1} \in \mathbb{R}^{n_y}$ is the output, $\mathbf{u}_k \in \mathbb{R}^{n_u}$ is the control input, $\mathbf{v}_{k+1} \in \mathbb{R}^{n_y}$ is the sensor noise, and $k \in \mathbb{Z}^+$ is the discrete time index. The disturbance and sensor noises are considered zero mean Gaussian processes with covariances Q_k, R_{k+1}, respectively. The state vector $\mathbf{x}_k$ is quite general, in that it can include a combination of the system state and/or internal model parameters.

An augmented state vector is defined as

$$\mathbf{x}_k^a = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{w}_k \\ \mathbf{v}_k \end{bmatrix} \tag{3}$$

which has dimension $\mathbf{x}_k^a \in \mathbb{R}^{n_a}$, $n_a = n + n_w + n_v$. The initial augmented state estimate and square root covariance are assumed to be

$$\mathbf{x}_0^a = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$S_{xx,0}^a = \sqrt{E\left[\left(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a\right)\left(\mathbf{x}_0^a - \hat{\mathbf{x}}_k^a\right)^T\right]} = \begin{bmatrix} S_{xx,0} & 0 & 0 \\ 0 & \sqrt{Q_0} & 0 \\ 0 & 0 & \sqrt{R_0} \end{bmatrix} \tag{4}$$

where $S_{xx,0}$, $\sqrt{Q_0}$, and $\sqrt{R_0}$ are the initial square root state, process noise, and sensor noise covariances.

Next, $2n_a + 1$ sigma points are defined as

$$\mathcal{X}_0^a = \left[\hat{\mathbf{x}}_0^a \;\; \hat{\mathbf{x}}_0^a e_{n_a} + \sigma_f S_{xx,0}^a \;\; \hat{\mathbf{x}}_0^a e_{n_a} - \sigma_f S_{xx,0}^a\right] = \begin{bmatrix} \mathcal{X}_0^x \\ \mathcal{X}_0^w \\ \mathcal{X}_0^v \end{bmatrix} \tag{5}$$

where σ_f is a scaling for the distance of the sigma points from the mean, and e_{n_a} is a $n_a \times 1$ vector of all ones. The sigma points have an associated set of associated weights,

$$W_0^m = \frac{\sigma_f^2 - n}{\sigma_f^2}, \qquad W_0^c = \frac{\sigma_f^2 - n}{\sigma_f^2} + 3 - \frac{\sigma_f^2}{n}, \qquad W^c = W \cdot I_{2n_a}$$
$$W_i^m = W_i^c = W = \frac{1}{2\sigma_f^2}, \qquad i = 1, \ldots, 2n_a \tag{6}$$

where σ_f is a scaling for the distance of the sigma points from the mean, and m and c denote mean and covariance, respectively.

The SR-SPF algorithm is summarized in Appendix A. The SR-SPF prediction step propagates each of the $2n_a + 1$ sigma points through the nonlinear dynamics and evaluates the sample mean and covariance. A key step in the formulation of the SR-SPF is the update of the covariance, which makes use of two functions: 1) orthogonalization or triangularization of a rectangular set of sigma points into an $n \times n$ matrix ($A = SS^T = RR^T$, $S \in \mathbb{R}^{n \times 2n_a+1}$, $R \in \mathbb{R}^{n \times n}$, $2n_a + 1 > n$), and 2) rank-one update of this Cholesky factor, denoted as $X_u = \text{up}\{X, v, \pm\sqrt{r}\}$ ($A = X_u X_u^T = XX^T \pm \sqrt{r} vv^T$). As shown in [19], it is best to use an orthogonalization to find $S_{xx,k}$, rather than $2n_a + 1$ rank-one updates because of numerical round-off problems when the sigma points are large; one rank-one update is required when $W_0^c < 0$, however.

Square Root Extended Set-Membership Filter (SR-ESMF)

The ESMF is a guaranteed estimator for a general class of nonlinear systems and online usage [20]. This filter recursively estimates an ellipsoidal set in which the true state lies. The ESMF bounds the linearization error of the estimator, then applies a linear set-membership filter such that stability guarantees hold for nonlinear systems. A tight bound on the linearization error is found using interval analysis. General assumptions include the use of bounded noises and twice continuously differentiable dynamics. When the system is uniformly observable, it is proven that the nonlinear set-membership filter is stable. In addition, if no noise is present and the initial error is small, the error between the center of the estimated set and the true value converges to zero. The result is an estimator which is computationally attractive and can be implemented robustly in real time. Full derivation of the filter and its convergence properties are given in [20]. A unique square root version for real time implementation is presented here.

Consider the general discrete, nonlinear state-space system but now with additive noise,

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{7}$$

$$\mathbf{y}_{k+1} = h(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) + \mathbf{v}_{k+1} \tag{8}$$

Both f and h are assumed to be smooth $\mathcal{C}^2$ functions. For the ESMF, the initial state $\mathbf{x}_0$ and disturbance/noise are known to be bounded by ellipsoids given as

$$\mathbf{x}_0 \in \mathcal{E}(\hat{\mathbf{x}}_0, \Sigma_{0,0}), \qquad \mathbf{w}_k \in \mathcal{E}(0, Q_k), \qquad \mathbf{v}_k \in \mathcal{E}(0, R_k) \tag{9}$$

where the notation $\mathbf{x} \in \mathcal{E}(\hat{\mathbf{x}}, P)$ defines an ellipsoid, $[\mathbf{x} - \hat{\mathbf{x}}]P^{-1}[\mathbf{x} - \hat{\mathbf{x}}]^T \leq 1$. Note that no assumptions on the structure of the noise are made except that it is bounded; hence, many types of noises are included within this framework including random and biased signals. Linearizing Eq. (7) about the current state estimate $\hat{\mathbf{x}}_k$ (and dropping the input $\mathbf{u}_k$ for now) yields

$$\mathbf{x}_{k+1} = f(\hat{\mathbf{x}}_k) + \left.\frac{\partial f(\mathbf{x}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k} (\mathbf{x}_k - \hat{\mathbf{x}}_k) + \mathcal{O}(\mathbf{x}^2) + \mathbf{w}_k \tag{10}$$

The traditional EKF formulation ignores the higher order terms $[\mathcal{O}(\mathbf{x}^2)]$. The approach here is to combine the higher order terms and process noise into one bound such that Eq. (10) can be rewritten as

$$\mathbf{x}_{k+1} = f(\hat{\mathbf{x}}_k) + \left.\frac{\partial f(\mathbf{x}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_k} (\mathbf{x}_k - \hat{\mathbf{x}}_k) + \hat{\mathbf{w}}_k \tag{11}$$

where $\hat{\mathbf{w}}_k$ is a new noise term that bounds both the original noise and the linearization remainder, and is defined as

$$\hat{\mathbf{w}}_k \in \mathcal{E}(0, \hat{Q}_k) \tag{12}$$

The bounding of the new noise term is accomplished using interval analysis, as shown in [20]. The linearization error is driven by uncertainty in the state, which is about the mean. Therefore, the use of a zero mean bound on the new noise term, as derived in [20], is slightly conservative. The sensor output equation (8) can be treated in a similar fashion.

The real time implementation of the ESMF requires a numerically stable square root implementation. The approach here is to use a factorization of the ESMF. Define the square root $R_{a,k+1}$ as

$$R_{a,k+1} R_{a,k+1}^T = C_{k+1} \frac{\Sigma_{k+1,k}}{1 - \rho_{k+1}} C_{k+1}^T + \frac{R_{k+1}}{\rho_{k+1}} \tag{13}$$

where $R_{a,k+1}$ is upper triangular when a Cholesky factorization is used, and hence the inverses are simpler computationally than the original full matrix inverse. Other matrix square roots are denoted as usual, that is, $Y^{1/2}$ denotes the square root of the matrix Y. Appendix B summarizes the SR-ESMF algorithm. Note that Θ and Φ are unitary matrices that can be obtained using approaches such as a QR factorization.

The actual state $\mathbf{x}_{k+1}$ is bounded by

$$[\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}]^T \Sigma_{k+1,k+1}^{-1} [\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}] \leq 1 \tag{14}$$

where the matrix $\Sigma_{k+1,k+1}$ is a function of the internal parameter δ_{k+1}, as given in Eqs. (B14) and (B15). The internal δ_{k+1} parameter is a good check for the algorithm in real time. If $\delta_{k+1} \geq 1$, the state uncertainty ellipsoid is not defined (i.e., positive definiteness is not guaranteed). This will only occur when the base algorithm assumptions have not been satisfied, namely, the initial state and noise ellipsoids are not valid. Thus, the noise bounds can be appropriately increased in size if $\delta_{k+1} \geq 1$. The prediction step is physically the addition of two ellipsoids, the augmented noise ellipsoid $\mathcal{E}(0, \hat{Q}_k)$ and the state uncertainty ellipsoid $\mathcal{E}(\hat{x}_{k,k}, \Sigma_{k,k})$ rotated and scaled by the dynamics A_k. The update step is the intersection of two sets, the predicted state ellipsoid and the set described by the output equation $\mathcal{E}(\mathbf{y}_{k+1}, \hat{R}_k)$.

The outer bounding ellipsoids for both the prediction and update step each depend on a bounded, free parameter, $\beta_k \in (0, 1)$ and $\rho_{k+1} \in (0, 1)$, respectively. Considering the prediction step first, two options exist for finding an optimal solution for β_k. The first is to minimize the sum of the squared semimajor axes of the ellipsoid,

$$\min_{\beta_k} \{\text{trace}(\Sigma_{k+1,k}(\beta_k))\} \tag{15}$$

which leads to a closed form solution for β_k. The second minimizes the volume of the ellipsoid

$$\min_{\beta_k} \{-\log\det(\Sigma_{k+1,k}(\beta_k))\} \tag{16}$$

which requires an optimization routine. Equation (15) is faster with its closed form solution, whereas Eq. (16) is typically a "smaller" ellipsoid. Similar definitions can be made for the update step and ρ_{k+1}.

As described in [20], the ESMF algorithm is asymptotically stable with and without bounded noise. The SR-ESMF implementation has identical properties, which are then used to prove stability when combined with the model predictive control scheme.

Robust Nonlinear Model Predictive Control (NMPC)

Current research in autonomous control focuses on the application of online optimization techniques for high-performance vehicles

such as aircraft [6,7]; these methods can be referred to as model predictive control (MPC) or receding horizon control. The advantages of these methods are the direct inclusion of constraints on states and inputs in the optimization as well as the ability to work with general nonlinear systems. An excellent, recent discussion of several nonlinear MPC theory, applications, and implementations is found in [27]. Several different implementations exist of NMPC. One example is based on the control Lyapunov function (CLF) [6,28], where stability is guaranteed by means of a Lyapunov function. Most of these approaches are based upon nominal models and operating conditions, assuming full state feedback.

The approach here is unique for three reasons. First, the SR-ESMF is integrated into the NMPC algorithm to provide constraints even in the presence of noise and uncertainties. Second, a contractive formulation of the MPC problem [29,30] (CMPC) is used, which allows the stability of the NMPC + SR-ESMF algorithm to be guaranteed by enforcing a terminal constraint. Third, a fast sequential quadratic programming (SQP) method is used which enables the development of an initial feasible solution, then continues to optimize the final solution to improve performance. This allows the computation to be cut off in the middle and still yield a productive, albeit suboptimal solution. This section summarizes each of these attributes.

Consider the same general discrete, nonlinear state-space system as in the SR-ESMF:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{17}$$

$$\mathbf{y}_{k+1} = h(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) + \mathbf{v}_{k+1} \tag{18}$$

The general nonlinear MPC problem is formulated as the following optimization problem:

$$\mathbf{u}_{i:i+N-1} = \operatorname{argmin} \sum_{k=i}^{i+N-1} \left[\mathbf{x}_k^T Q_{x,k} \mathbf{x}_k + \mathbf{u}_k^T R_{x,k} \mathbf{u}_k \right] + V_F(\mathbf{x}_{i+N}) \tag{19}$$

subject to the dynamics given by Eqs. (17) and (18) and state and input constraints:

$$\mathbf{x}_k \in \mathcal{X}_k, \qquad \mathbf{u}_k \in \mathcal{U}_k \tag{20}$$

Here, $Q_{x,k}$ and $R_{x,k}$ are weighting matrices and V_F is the terminal cost.

Feasible Nonlinear SQP

Solving the nonlinear program in real time, even for state feedback with no noise, is a challenge. Recently, a variety of nonlinear solvers and solutions have been proposed for such problems. Previous research [7] has shown that a commercial solver such as NPSOL [31] based on SQP can be implemented in real time for NMPC [32]. In [33] it was shown that computation times can be decreased by exploiting a differential flatness structure of the system.

The approach adopted here is based on combining a fast MPC method that uses sparse matrices and an initial feasible solution, with the guarantees of the SR-ESMF. The NMPC approach [34,35] first solves the approximate quadratic subproblem:

$$\min m(\Delta\mathbf{z}) = \nabla f(\mathbf{z})^T \Delta\mathbf{z} + \tfrac{1}{2} \Delta\mathbf{z}^T H \Delta\mathbf{z} \tag{21}$$

such that

$$c(\mathbf{z}) + \nabla c(\mathbf{z})^T \Delta\mathbf{z} = 0 \tag{22}$$

$$d(\mathbf{z}) + \nabla d(\mathbf{z})^T \Delta\mathbf{z} \leq 0 \tag{23}$$

where $\mathbf{z}$ is the minimization variable which is the input sequence $\mathbf{u}_{i:i+N-1}$ to be found, $m(\mathbf{z})$ is the quadratic approximation of f at $\mathbf{z}$, H is the Hessian of the Lagrangian, and $\Delta\mathbf{z}$ is the optimization step. This general SQP is augmented with a trust region $\|D\Delta\mathbf{z}\|_2 \leq \Delta$. The primary difference with existing methods is that a perturbation $\widetilde{\Delta\mathbf{z}}$ from the current step $\Delta\mathbf{z}$ is chosen such that feasibility is maintained: $\mathbf{z} + \widetilde{\Delta\mathbf{z}} \in \mathcal{F}$. This allows the optimization to be cut off at any time, and the current, suboptimal solution can be used. This cutoff can be advantageous when computation times are limited. In addition, this MPC formulation also takes advantage of the sparseness of the Hessians [34,36] which makes computation much faster.

Contractive MPC

One approach to adding robustness to the current MPC formulations is adding another constraint to the nonlinear optimization problem, namely, a terminal constraint of the form:

$$\mathbf{x}_{i+N} \in \mathcal{X}_F \tag{24}$$

This method is known as contractive MPC [29,30]. In the CMPC formulation, the terminal constraint is of the form:

$$X_F = \{\mathbf{x}_{i+N} \mid \|\mathbf{x}_{i+N}\| \leq \alpha \|\mathbf{x}_i\|\}, \qquad 0 \leq \alpha < 1 \tag{25}$$

If this constraint is enforced at each optimization step, it is possible to show that the resulting controller will steer the states to a control invariant set when $k \to \infty$ [29,30]. This fact, along with the SR-ESMF guarantees [20], enables the combined NMPC + SR-ESMF algorithm to be provably stable [21]. Note that it is not necessary to enforce this constraint at each time step to guarantee stability. Doing so, however, will steer it faster to the set than if it is only enforced at the optimization horizon N.

Integration with the SR-ESMF

The approach here uniquely integrates the SR-ESMF estimator with the nonlinear MPC method described previously. This allows the NMPC formulation to address realistic limitations such as sensor and process noise, and model uncertainty. The primary advantage of using the SR-ESMF in this integration, as opposed to the SR-SPF or other stochastic estimators, is that the integration is straightforward: the nonlinear optimization of the NMPC problem requires bounded constraints, while the SR-ESMF produces a bounded set in which the true state must lie.

In addition to using the SR-ESMF to address realistic noises or uncertainties, several additional approaches exist to integrating the SR-ESMF to improve the speed of optimization. They are as follows:

1) Reduce the initial uncertainty set of the optimization horizon: The measurement update [Eqs. (B7–B15)] results in a set $\mathcal{E}(\hat{\mathbf{x}}_{k+1,k+1}, \Sigma_{k+1,k+1})$ which can be used to update the state constraint [Eq. (20)] at time $k+1$.

2) Reduce the MPC search space by predicting the reachable space: The prediction step of the SR-ESMF [Eqs. (B1–B6)] is used over a prediction horizon N to solve for a reachable space for the state; this space is used as a constraint in the NMPC optimization.

3) Bound coupled uncertainties with state constraints (collision avoidance): In the presence of multiple vehicles, obstacles, or stay out zones, constraints on relative states can be integrated.

Each of these are used in the NMPC formulation and demonstrated in the Results section.

Streamline Path Planning (SPP)

The best path planners adequately trade between accuracy and computation. Because path planning solves for a trajectory over a longer time horizon as compared to the short time horizon of low level control, a considerable advantage can be found if path planning can be done relatively fast. Recent research [2,22] introduced a method for obtaining smooth trajectories around (circular) obstacles with very low computational effort at the expense of optimality. The generated trajectories are based on aerodynamics theory, and therefore yield smooth point-to-point paths around obstacles that are well suited to aircraft-type vehicles (as opposed to start–stop vehicles like robots). The trajectories obtained are so-called stream functions, which are a subset of potential methods. This method does not

require a fully actuated vehicle, but still guarantees that the vehicle will reach the desired way point.

The complex potential for a 2-D flow is given as

$$\omega(z) = \phi(z) + i\psi(z) \tag{26}$$

where ϕ is the potential function, ψ the stream function, and $z = z_x + iz_y$. The theory enables the definition of UAV or robot paths by defining different flow types; some of the more useful definitions with closed form solutions are a sink or a vortex. For the case of a sink flow (such as attempting to move to a way point) with a stationary obstacle, the complex potential is

$$\omega(z) = -C\,\ell_n(z) - C\,\ell_n\left(\frac{a^2}{z-b} + b^*\right) \tag{27}$$

where a is the radius of the obstacle, $b = (b_x + ib_y)$ is the obstacle center location given in complex notation, and C is the magnitude of the attraction to the goal. The imaginary component of this complex potential then yields the stream function for the flow,

$$\psi(z_x, z_y) = -C\tan^{-1}\frac{z_y}{z_x} + C\tan^{-1}\frac{\frac{a^2(z_y-b_y)}{\|z-b\|^2} + b_y}{\frac{a^2(z_x-b_x)}{\|z-b\|^2} + b_x} \tag{28}$$

This closed form solution for the streamlines allows the generation of trajectories very fast. To generate the planning trajectories, the stream function must first be differentiated to find the stream velocities,

$$u_{\text{str}}(z_x, z_y) = \frac{\partial\psi}{\partial z_y}, \qquad v_{\text{str}}(z_x, z_y) = -\frac{\partial\psi}{\partial z_z} \tag{29}$$

Then, the velocity must be scaled from the normalized stream function to the UAV velocity, and used recursively to find the position of the vehicle over time, $\mathbf{x}_{\text{pla}}$. This recursion is written as

$$\mathbf{x}_{\text{pla},k+1} = \begin{bmatrix} x_{\text{pla},k+1} \\ y_{\text{pla},k+1} \end{bmatrix} = \begin{bmatrix} x_{\text{pla},k} \\ y_{\text{pla},k} \end{bmatrix} + \Delta t \begin{bmatrix} u_{\text{str},k}(z_x, z_y)|_{z_x = x_{\text{pla},k}, z_y = y_{\text{pla},k}} \dfrac{V_{uav,k}}{\sqrt{u_{\text{str},k} + v_{\text{str},k}}} \\ v_{\text{str},k}(z_x, z_y)|_{z_x = x_{\text{pla},k}, z_y = y_{\text{pla},k}} \dfrac{V_{uav,k}}{\sqrt{u_{\text{str},k} + v_{\text{str},k}}} \end{bmatrix} \tag{30}$$

where the recursion starts from the current UAV position. The parameter C can be varied to change the length of the path and closeness to the obstacle. Extensions to moving obstacles [22] and multiple obstacles [2] have been made.

The result is a smooth set of trajectories around obstacles with guarantees of reaching the destination, all very quickly. As a benchmark, 100 streamline path options are easily generated at a 30 Hz rate on a Pentium IV, 2 GHz [2]. This enables adaptations to the path plan very quickly, which is important in the case of a fast changing, uncertain environment.

For this work, the SPP is integrated into the midlevel controller of the autonomous control block diagram presented in Fig. 1. When no NMPC control is present, the SPP simply provides future way points to the UAV in the form of $\mathbf{x}_{\text{pla},k}$. In the presence of the NMPC controller, the SPP can help "guide" the optimization. The optimization is formulated as a small change to Eq. (19) as

$$\mathbf{u}_{i:i+N-1} = \text{argmin} \sum_{k=i}^{i+N-1} [(\mathbf{x}_k - \mathbf{x}_{\text{pla},k})^T Q_{x,k}(\mathbf{x}_k - \mathbf{x}_{\text{pla},k}) + \mathbf{u}_k^T R_{u,k}\mathbf{u}_k] + V_F(\mathbf{x}_{i+N}) \tag{31}$$

where $\mathbf{x}_{\text{pla},k}$ is the trajectory generated by the SPP midlevel control.

SeaScan UAV

The SeaScan unmanned aerial vehicle platform, developed by Insitu Group, was used in conjunction with a Cornell developed real time payload to test each of the estimation, planning, and control algorithms in real time. This section details the SeaScan hardware, software, simulators, and models used in this work.

Hardware

Figure 2 (top) shows the SeaScan platform being prepared for flight, while Fig. 2 (bottom) shows an expanded view of the SeaScan avionics and Payload. The dimension, weight, and performance of the SeaScan are shown in Table 1. The SeaScan launches at 50 kt using a car-top cradle or a low-pressure pneumatic catapult; the SeaScan retrieval/landing is accomplished using the Skyhook, an Insitu-patented technique, where the aircraft flies into a single suspended line and hooks on the wing tip. The SeaScan UAV also includes an electrical generator powered by the engine and is capable of generating up to 140 W.

Figure 2 (bottom) shows the internal physical layout of the electronics, where electronics boards for primary functions are plugged into a backplane with the same shape as the SeaScan fuselage. The architecture of the avionics is shown in Fig. 3, along with its integration with the Payload. The primary functions of the onboard avionics used in this work include the following:

Flight controller: The SE555 computer provides inner loop control by receiving data from aircraft sensors and sending commands to control surfaces. Flight path characteristics can be determined from preprogrammed or in-flight commands such as way points, orbits/boxes/other patterns, and control surface commands.

Air-to-ground communication: This 9600 baud wireless data link at 1.7 GHz is used to communicate aircraft status, control, and mission data, as well as to relay messages from payload modules.

Avionics-to-payload communication: This serial data link is used to send sensor reports to the Payload, and receive commands from the Payload.

GPS receiver: The SE555 communicates serially with an onboard global positioning system (GPS) receiver for inertial position and velocity information, and differential position and velocity information during landing.

Sensor reports: The onboard sensors include roll, pitch and yaw rate gyros, vertical, lateral and longitudinal accelerometers, external temperature sensors, relative pressures of pitot tube, alpha, beta, gamma, and absolute pressures of the barometer and manifold.

Payload

The Payload is an Embedded Planet 8260 board with a Motorola PPC8260 processor. The processor performs at 570MIPS, 300 MHz,

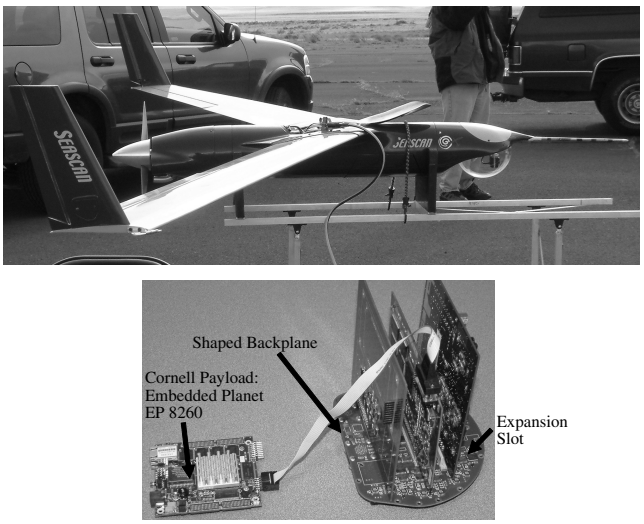


Fig. 2 Top: The SeaScan UAV prepped for launch. Bottom: Internal avionics, showing all boards and connections to the Payload. Parallel boards plug into a "shaped" backplane in the form of the fuselage (top to bottom): Cornell payload, CPU + GPS + modem, power.

Table 1 Specifications of the SeaScan UAV, developed by the Insitu Group

Dimensions	
Wing span	3.04 m
Wing area	0.72 m^2
Wing sweep	23 deg
Fuselage diameter	0.18 m
Overall length	1.2 m
Mass properties	
Airframe	3.7 kg
Avionics/Payload	3.2 kg
Power plant	2.4 kg
Empty weight	11.0 kg
Max fuel weight	4.3 kg
Max launch weight	15.4 kg
I_{xx}	3.7560 kg · m^2
I_{yy}	1.3447 kg · m^2
I_{zz}	4.6774 kg · m^2
I_{xz}	0.0225 kg · m^2
Performance	
Ambient temperature range	−10°C– + 40°C
Max level speed	68 kt
Cruise speed @ max wt	48 kt
Min speed max wt	42 kt
Max S/L climb max wt	2.5 m/s
Service ceiling @ max wt	5000 m
Still-air range, no reserves	1500 km
Navigation	GPS (inc. differential)
Communication	9600 baud half-duplex UHF U/L, D/L; 1.7 GHz

and includes 64 MB RAM, 32 MB Flash, two serial ports, and is VxWorks 5.4 compatible. The power specifications are 1.5 A, 5 V max. The physical interface between the Payload and the SeaScan avionics is shown in Fig. 3. The Payload module is designed to implement low-, mid-, and high-level control commands for the SeaScan UAV, similar to the blocks shown in Fig. 1. In the set of experiments here, way points are generated on the payload and sent to the flight computer for implementation. State/parameter estimation, fault detection, model predictive control, and path planning are realized within this module.

A special ground station software module was developed which allows full communication between the ground station and the Payload. Thus, onboard telemetry from the Payload during all experiments is passed through the primary UAV processor and onto the ground. Also, command functions, such as starting/stopping estimators and placing obstacles for avoidance, are realized by passing the commands from the ground station, to the primary UAV processor, and finally to the Payload.

Operating System and Middleware

The Payload uses the VxWorks operating system for precise timing in the estimation and control functions. A subset of the tests also included the OCP from Boeing [15], a middleware software framework which provides services for controlling the execution and scheduling of components, intercomponent communication, distribution, and deployment of application components onto a target system. The primary function used in both the OCP and VxWorks was the triggering each of the software modules was provided by custom coding and VxWorks. There was no notable performance difference between the two approaches.

Simulators

Two simulators exist for developing and testing algorithms before flight. The first is a software only simulator in C + +. Validation at this level tests interfaces to the flight code, as well as the relative speed of the implementation. It does not, however, verify the real time implementation of the coding. The hardware in the loop (HIL) simulation includes not only the full avionics suite (identical to that used in flight) running in real time, but also servomechanisms that mimic onboard actuators, radio frequency communication for air to ground and air to air, and a high fidelity aircraft dynamics simulator. Thus, the HIL is a true replica of the real time flight operations on the ground with the exception of the nonlinear dynamics simulation.

SeaScan Models

To develop and evaluate the proposed algorithms, two nonlinear models of the SeaScan UAV are used. The first is a complex nonlinear model developed by the Insitu Group for their HIL which includes all components of the system such as the nonlinear aerodynamics, engine, propeller, fuel, weather, and other modules. The model includes 12 global aircraft states:

$$\mathbf{x} = \begin{bmatrix} V & \alpha & q & \theta & p & \phi & r & \psi & \beta & x_p & y_p & h \end{bmatrix}^T \quad (32)$$

in order: vehicle velocity, angle of attack, pitch, yaw, and roll rates and angles, sideslip angle, x, y position, and altitude. There are seven control actuators: ailerons (2), flaps (2), rudders (2), and thrust (1). These are combined in hardware/software to give five control inputs

$$\mathbf{u} = \begin{bmatrix} T & \delta_e & \delta_a & \delta_r & \delta_T \end{bmatrix}^T \quad (33)$$

in order: thrust, effective elevator, effective aileron, effective rudder, and effective flap. This model is used in the software simulator, HIL simulator, and in the real time, online implementation of the SR-SPF. The model was implemented as an executable software function of the form:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \quad (34)$$

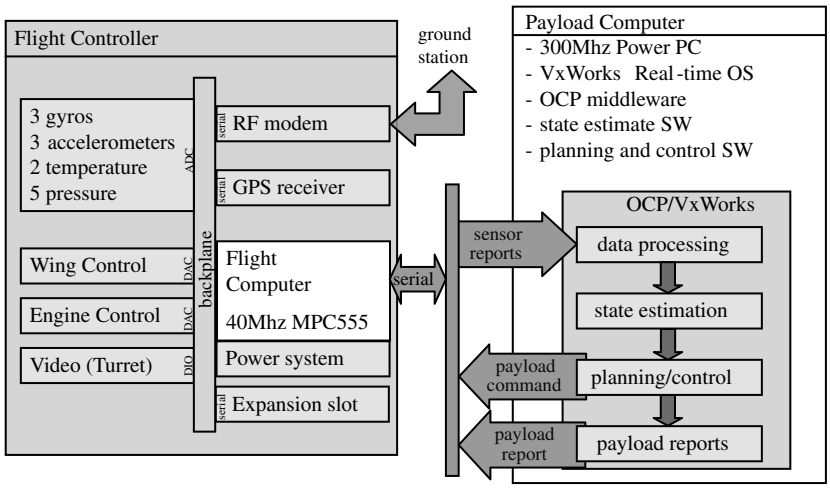


Fig. 3 Interface between the Cornell Payload and the SeaScan avionics.

where $\mathbf{w}_k$ is a three-dimensional wind vector. Because the SR-SPF was used, and no functional Jacobians were required, the model in Eq. (34) could be built into a black box, real time function by Insitu and delivered to Cornell, thus removing potential intellectual property issues.

The second model has the identical states as the first model, but includes only the inertia properties given in Table 1, full nonlinear dynamics and kinematics, and a spline fit to the aerodynamics table. This model was used in the SR-ESMF and NMPC implementations, where linearizations are required.

Software Implementations

Software modules coded in C++ were developed for versions of the SR-SPF, SR-ESMF, NMPC, and SPP. Each of these functions requires state-of-the-art tools in matrix operations. Current computer architectures rely heavily on the use of cache and memory structure for speed. To take advantage of these features a logical choice for matrix operations on an embedded system is the use of optimized libraries. A common library for matrix and vector operations is BLAS [37]; a C++ implementation of level 1 (vector–vector operations), 2 (vector–matrix operations), and 3 (matrix–matrix operations) BLAS is provided by the uBLASpackage, which enables the user to call all BLAS functions with the use of expression templates in C++. Often, BLAS is tuned specifically for a given platform to create efficient code. Another option is ATLAS [38], an automatically tuned version of BLAS which provides a high-performance library for most all platforms. In addition, ATLAS includes several LAPACK [39] functions for linear algebra. In this work, both BLAS and ATLAS were used in software simulations, but only BLAS was used in the HIL and flight tests.

Hardware in the Loop and Flight Test Results

HIL tests were performed regularly over a period of 3 years (2003–2005). A subset of these HIL experiments were then packaged into a series of four flight tests performed on 29 October 2003, 20 January 2004, and 28–29 August 2005. These experiments evaluated many implementations of estimators, fault detection, model predictive control, and path planning. Maturation of the experiments transitioned from software only simulations to real time HIL evaluations to flight tests. Data presented here are only real time HIL or flight test results. In the case of presented flight test results, the HIL results are typically not presented because they are nearly identical to the flight test results (by design). In the case of presented HIL results, flight tests were not run, typically due to the risk placed on the UAV hardware.

Real time HIL/flight test results are presented for the following cases:

1) Aeromodel estimation using the SR-SPF (flight): demonstrates online estimation of aerodynamic models using the SR-SPF in real time, and in flight.

2) Aeromodel estimation during failure using the SR-SPF (HIL): demonstrates online estimation of aerodynamic models during normal flight conditions and during failure using the SR-SPF in real time.

3) Pursuit of a moving target using SR-SPF + SPP (HIL): demonstrates the SPP tracking a moving target in real time.

4) Pop-up obstacle avoidance using NMPC + SR-ESMF + SPP (HIL): demonstrates full integration of the NMPC + SR-ESMF + SPP and inner loop control in the HIL.

5) Pop-up obstacle avoidance using NMPC + SR-ESMF + SPP (flight): demonstrates full integration of the NMPC + SR-ESMF + SPP in real time using way points in flight.

All algorithms were implemented on the EP8260 payload in real time with VxWorks.

Aerodynamic Model Estimation Using the SR-SPF + SPP (Flight)

The SR-SPF was implemented to develop aerodynamic model estimates online, and in real time. If a new model can be identified quickly, the new model can be passed to an online control customization procedure to generate new control signals that are appropriate for the system; the speed of the model generation is dependent on the control required. The Payload was fed raw sensor data at 20 Hz from the primary CPU while the SeaScan UAV was commanded to perform a variety of maneuvers. The ground speed of the UAV was controlled to be ≈ 25 m/s.

Figure 4 (left) shows a snippet of a time trace of the three-dimensional trajectory. Shown are a variety of trajectories including 1) trajectories around selected "orbits" (R, X, Y, Z), 2) box and bow tie patterns, and 3) way point planning using the SPP. The SR-SPF error bounds for the vehicle position ($\sigma \approx 1.5$ m) compare favorably to the output of the GPS receiver.

To estimate the aerodynamic forces, a small time snippet of the flight was used. This snippet was of a bow tie pattern with varying altitude. This enabled each of the control actuators in Eq. (33) to be activated. The aerodynamic model estimation assumed the traditional expansion of the coefficients into linear contributions from the actuators and states [40], written as

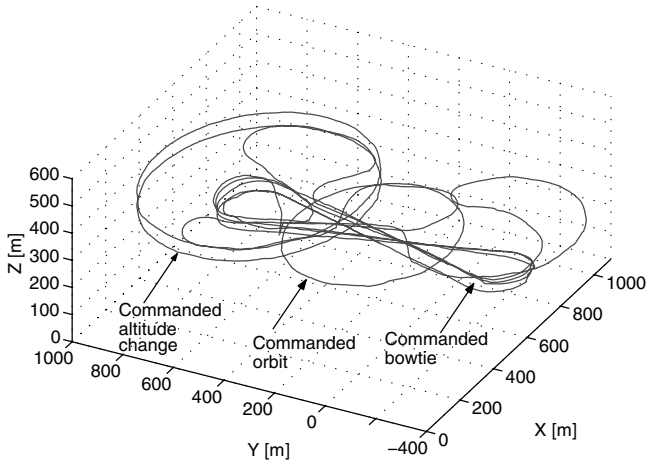
$$\begin{aligned}
C_D &= C_D(V,\alpha,\delta_e) = C_{D0} + \sum^{(\cdot)} C_{D(\cdot)}\Delta(\cdot) \\
C_Y &= C_Y(\beta,r,p,\delta_r) = C_{Y0} + \sum^{(\cdot)} C_{Y(\cdot)}\Delta(\cdot) \\
C_L &= C_L(V,\alpha,q,\delta_f) = C_{L0} + \sum^{(\cdot)} C_{L(\cdot)}\Delta(\cdot) \\
C_l &= C_l(\beta,r,p,\delta_r,\delta_a) = C_{l0} + \sum^{(\cdot)} C_{l(\cdot)}\Delta(\cdot) \\
C_m &= C_m(V,\alpha,q,\delta_e) = C_{m0} + \sum^{(\cdot)} C_{m(\cdot)}\Delta(\cdot) \\
C_n &= C_n(\beta,r,p,\delta_r,\delta_a) = C_{n0} + \sum^{(\cdot)} C_{n(\cdot)}\Delta(\cdot)
\end{aligned} \tag{35}$$

where $\Delta(\cdot)$ is the perturbed state/control from the nominal (typically trim or zero) condition.

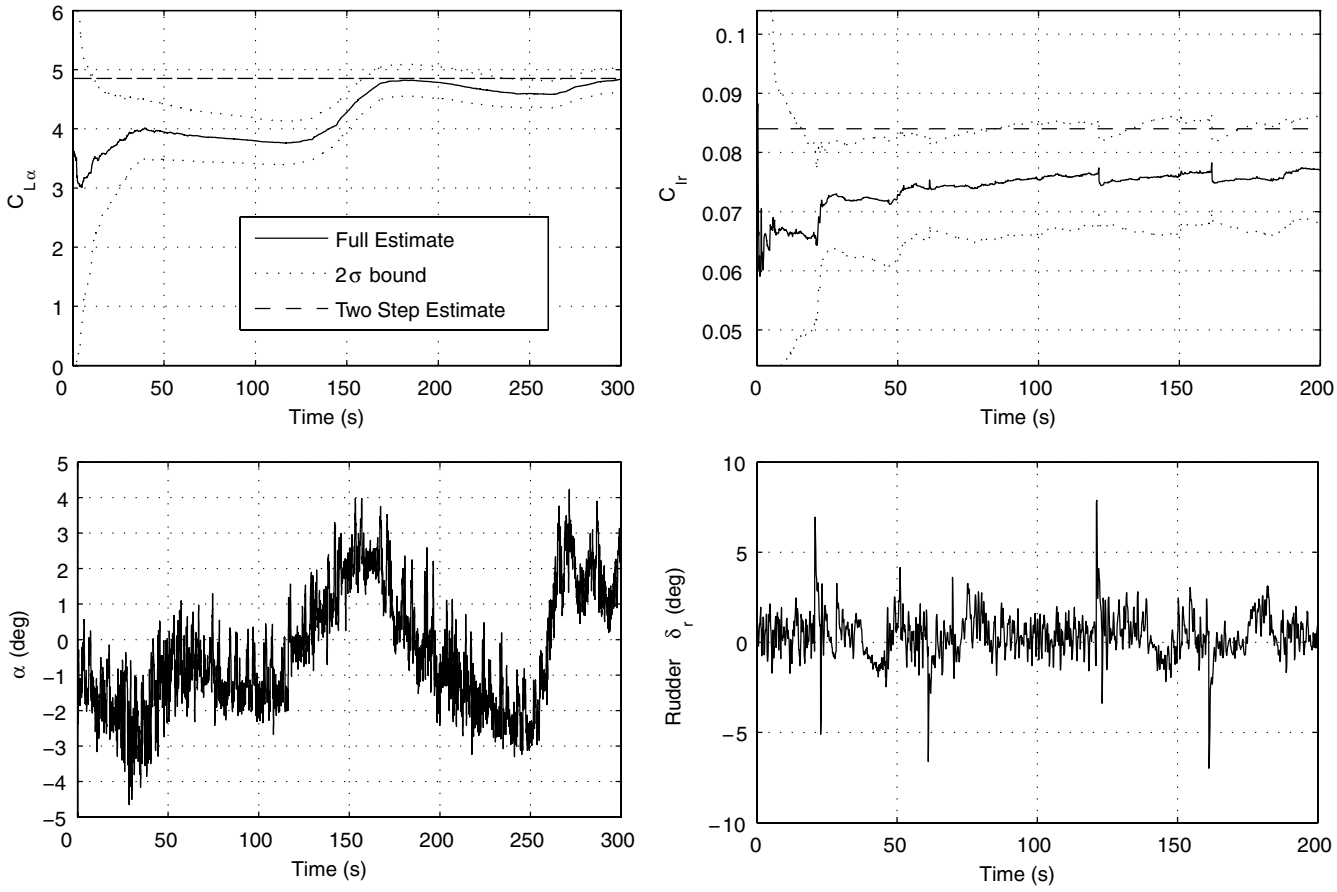
The primary approach to estimating the aerodynamics was to use the SR-SPF, along with an aircraft model and the linear expansion of the coefficients in Eq. (35), to estimate the control and stability derivatives $C_{(\cdot)(\cdot)}$ online and in real time. Each derivative was added to the state vector and given dynamics of a random walk, with tuned process noise. Tuning of the estimator process noise was accomplished using the two simulator models: 1) simulator within the HIL and 2) simulator with known aerodynamic tables fit to splines.

Figure 4 (right) shows flight results for two estimated aerodynamic coefficients and their respective bounds using the SR-SPF. Shown are the $C_{L\alpha}$ and C_{lr} derivatives. The $C_{L\alpha}$ derivative converges but does so slowly, based primarily on the tuned process noise of the random walk associated with the derivative, and the slow variation and noisy signal of α. The C_{lr} derivative has jumps in the estimate, corresponding to each time the roll actuator is actuated. For most of the derivatives, the estimates converged within several seconds with 2σ bounds that were within 25% of their final values. Notable exceptions were those derivatives in the pitching moment and slideslip directions, resulting in estimates that are more uncertain. These directions are typically difficult to estimate using any algorithm due to the lack of control and disturbances in these directions. This is especially true in the case of the SeaScan, where there is no tail section.

Although the "truth" is not known for the flight data, an additional approach was used to evaluate the estimation flight data. This approach, denoted as the two-step method, estimated the raw coefficients $C_{(\cdot)}$ online and in real time, followed by a linear least-squares fit to the derivatives offline. The online estimator in this case added only six parameters to the state vector to estimate, which typically reduces variation in the parameter estimates. The approach has been shown to be quite reliable in practice [41,42]. Table 2 shows the identified derivatives with each method, showing good correlation between the two methods across most estimates. It is



a) Position tracking



b) Aerodynamic derivative estimation

Fig. 4 Flight test results for using the SR-SPF for aerodynamic model estimation.

interesting to note that several of the derivatives are different than conventional aircraft, primarily because of the lack of tail section. This is especially true in the derivatives related to the moments, where moments that are a function of the tail section (C_{mq}, C_{nr}, for example) are relatively small.

Aerodynamic Model Estimation During Failure Using the SR-SPF + SPP (HIL)

A key benefit of using an online aerodynamic model estimator such as the SR-SPF is for fault detection and recovery (if coupled with an online control customization routine). If a fault occurs, and if the estimator can quickly evaluate the fault and generate a new model, the new model can be used to generate control signals that take the fault into account. Many fault detection methods attempt to model each potential failure in an estimation framework. But this requires many failure models and does not scale well with the estimation architecture. The objective here is to understand how well the SR-SPF estimator can estimate faults with no a priori fault models.

Two cases were run on the HIL to evaluate the ability to estimate a new aerodynamic model quickly in real time: 1) normal flight with a preprogrammed set of way points, and 2) the same flight as (1), but reporting a 50% failure of the aileron at time $t = 4775$ s to the estimator. The simulated failure was accomplished by giving the estimator an aileron signal that was 50% reduced from the nominal simulation. Because all other data delivered to the estimators were

Table 2 Summary of identified control and stability estimates and 2σ bounds for the SeaScan UAV using the two-step method and the square root SPF

Derivative	Two step	SPF
$C_{D\alpha}$	0.157	0.148 ± 0.310
$C_{D\delta_e}$	0.168	0.160 ± 0.011
$C_{Y\beta}$	−0.300	-0.264 ± 0.151
C_{Yr}	0.110	0.081 ± 0.101
C_{Yp}	−0.084	-0.084 ± 0.022
$C_{Y\delta_r}$	−0.148	-0.151 ± 0.051
$C_{L\alpha}$	4.95	4.54 ± 0.231
C_{Lq}	0.692	0.730 ± 0.159
$C_{L\delta_f}$	0.270	0.288 ± 0.121
$C_{l\beta}$	−0.121	-0.100 ± 0.119
C_{lr}	0.083	0.074 ± 0.010
C_{lp}	−0.601	-0.601 ± 0.146
$C_{l\delta_r}$	−0.010	-0.011 ± 0.010
$C_{l\delta_a}$	0.411	0.390 ± 0.095
$C_{m\alpha}$	−1.01	-0.612 ± 0.146
C_{mq}	−4.58	-4.618 ± 0.312
$C_{m\delta_e}$	1.05	1.01 ± 0.110
$C_{n\beta}$	0.034	0.0328 ± 0.026
C_{nr}	−0.018	-0.012 ± 0.005
C_{np}	−0.039	-0.040 ± 0.007
$C_{n\delta_r}$	0.026	0.029 ± 0.007
$C_{n\delta_a}$	−0.013	-0.013 ± 0.010

identical, this enabled the two estimators (with and without aileron failure) to have roughly similar observability and controllability conditions, which is important for comparison purposes. Therefore, by giving the estimator the same flight path, state, and control signals, and an aileron signal at a 50% reduction, the estimator with the failure should estimate *larger* control derivatives associated with the aileron.

Figure 5 shows HIL results for estimating a key aerodynamic derivative, the control derivative $C_{l\delta_a}$, with and without aileron failure and with failure. In the case of a failure, the control derivative $C_{l\delta_a}$ immediately spikes, and then quickly settles into steady state again (within 3 s). Generation of a new model for the control customization within 3 s is most likely sufficient for recovering from some, but not all, failures. Therefore, one proposed failure detection scheme is to use a hybrid SR-SPF which includes failure models for the most common failures, supplemented with the nominal SR-SPF to detect additional failures that are not modeled.

Pursuit of a Moving Target Using SR-SPF + SPP (HIL)

A challenging problem that integrates target tracking and path planning is using the SR-SPF to estimate vehicle states *and* track a moving target, and to pursue the target and rendezvous as quickly as possible. In the case presented here, the target is commanded to move along a straight line at a constant velocity ($V_{\text{tar}} = 15$ m/s, which is slower than the SeaScan UAV. For the simulation, pseudorange ($\sigma_R = 1$ m) and bearing ($\sigma_\theta = 1$ deg) measurements are created using the data from the simulator. An SPP path planner is then used to pursue the moving target in real time. The SPP planner, in this case, is cast with an obstacle radius equal to the minimum UAV turning radius, with placement to the side of the UAV. This forces the SPP to maintain UAV constraints if the initial command is to turn toward the target. The location of the sink is the time predicted (constant velocity) location of the target. This is written as

$$\omega(z) = -C\,\ell_n(z) - C\,\ell_n\left(\frac{R^2_{\text{min-turn}}}{z - b_{\text{side}}} + b^*_{\text{side}}\right) \tag{36}$$

The SR-SPF was run at 10 Hz, while the SPP was run at a rate of 1 Hz. Although not shown, the SR-SPF tracking estimator produced an estimation error of $\sigma \approx 10$ m. Figure 6 shows the trajectory of a target pursuit for a constant velocity target. The SeaScan UAV is initialized to a trajectory that is nearly opposite in direction to the target. During the 2 min simulation, the SeaScan UAV rendezvoused with the target once, with a relative range of $\Delta R = 1.05$ m ($t = 40.7$ s). And then the algorithm was restarted after the first pass, resulting in a second rendezvous at $\Delta R = 3.03$ m ($t = 70.8$ s). Both cases are well within the range of the target estimation error. In repeated trials, all rendezvous cases were within the 1σ target tracking estimation error.

Pop-Up Obstacle Avoidance Using NMPC + SR-ESMF + *SPP* (HIL)

An application of the integrated NMPC algorithm with estimator and planner algorithms is to estimate and command the SeaScan UAV when a threat pops up quickly. The SR-ESMF was used to estimate the UAV states and aerodynamic forces and moments. The operator on the ground then keyed an entry into the Cornell ground station, which uplinked a command to the Payload. This command, in turn, placed a "virtual" cylindrical obstacle (30 m in radius) 200 m ahead of the current UAV trajectory. The Payload then implemented the SPP to plan a reference trajectory around the obstacle. The Payload then implemented NMPC to optimize the control signals $\mathbf{u}_k$ for the UAV to follow the reference trajectory, in the presence of the aircraft and obstacle constraints (in the form of a collision/state constraint). Both the SPP and NMPC were then iterated at a 20 Hz rate throughout the maneuver until the original trajectory was recovered. The SR-ESMF was integrated into the NMPC algorithm as described previously: estimating reachable sets, providing uncertainty ellipsoids for constraints, and providing updated state uncertainty ellipsoids.

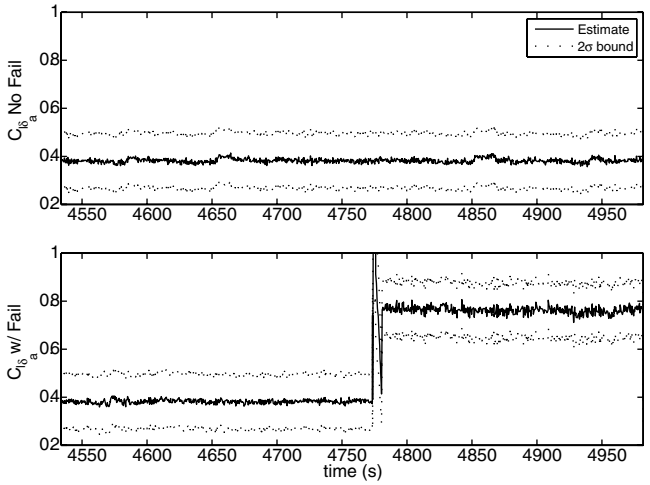


Fig. 5 Online estimation of an aileron control derivative in the linearized aerodynamic model using the SR-SPF for the cases of a SeaScan UAV with and without an aileron failure.

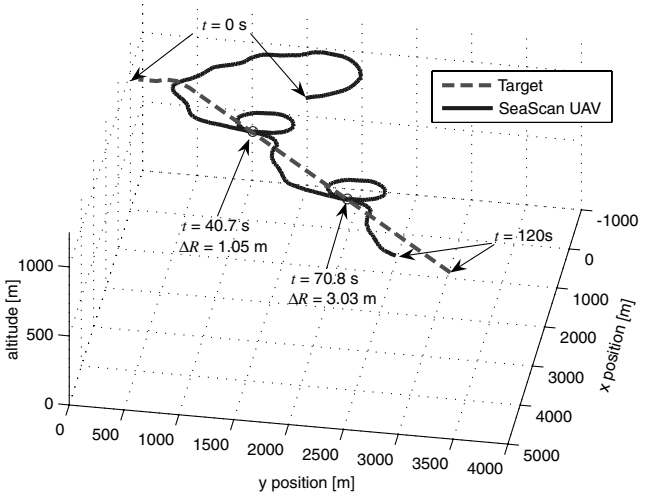
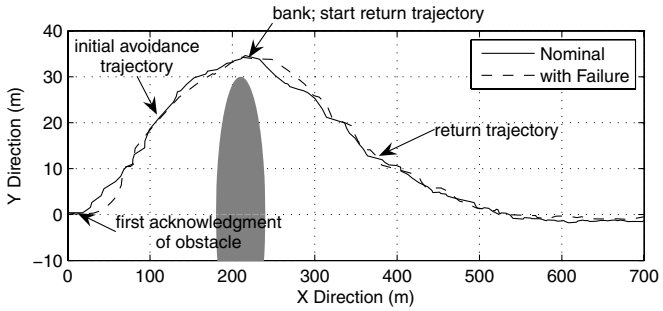


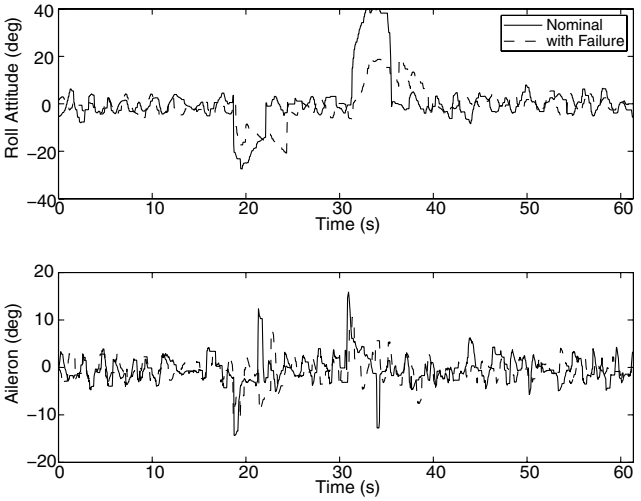
Fig. 6 SeaScan HIL results for target tracking and pursuit planning.

Figure 7 (left) shows the trajectory of a maneuver around the cylindrical pop-up threat. The SR-ESMF estimator provides hard bounds on the actual position to allow the vehicle to safely, and in the presence of uncertainties, steer around the threat. The roll attitude and aileron telemetry streams are shown in Fig. 7 (right). In both the initial avoidance and the return trajectory, the aileron control signals are nearly saturating (as in, the aileron constraints in the NMPC algorithm are active, but these constraints are smaller than the actual UAV capability). Because the input constraints become active, this implies that the maneuver is quite extreme for the given control constraints. The vehicle must use its actuators at their saturation levels to maneuver around the obstacle. In this example, no constraints were used on the input rates, which in practice also are limited.

A more complex example of the same application is the case when there is a 50% failure in the aileron, yet the vehicle still must maneuver about the pop-up obstacle. In this example, the failure occurs 60 s before the pop-up obstacle is placed. The SR-SPF was used to estimate the control and stability derivatives during the failure, while the integrated NMPC + SR-ESMF + SPP algorithm planned the trajectory and control signals around the pop-up obstacle. When the failure occurred, the SR-SPF estimated nearly a 50% reduction in the aileron control derivatives, which was then used as a change in the constraints (and model) in the integrated NMPC + SR-ESMF + SPP algorithm. The dashed lines in Fig. 7 denote the example with an aileron failure. When large maneuvers were considered, near the start of the avoidance trajectory and just passing the obstacle, the trajectory is rounder for the failed case. This is a result of the reduced control authority for the maneuver. It is noted that the aileron and roll signals are longer (in time), reduced in amplitude, and of higher variance. The higher variance is most likely due to aircraft stability and handling problems in roll using an aileron control surface that is reduced in authority.



a) Avoidance trajectory



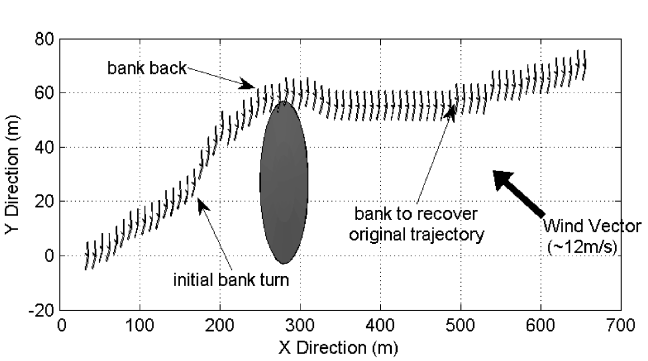
b) Attitude and control signals

Fig. 7 SeaScan UAV executing a hard avoidance maneuver around a pop-up threat (HIL simulation results). Control input constraints are noticeable.

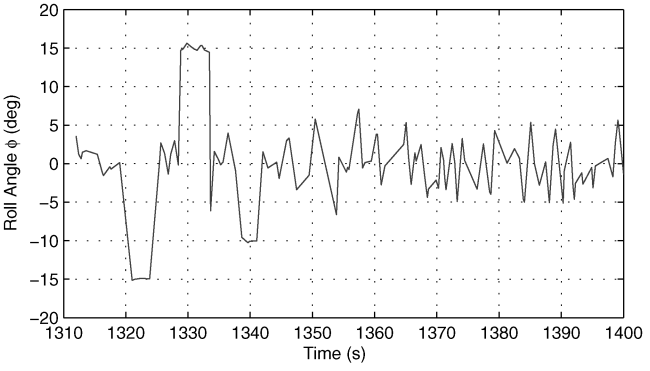
Pop-Up Obstacle Avoidance Using NMPC + SR-ESMF+ SPP (Flight)

It was desired to repeat the pop-up obstacle avoidance problem in flight. For the flight tests, it was decided that the risk of implementing the inner loop control loop was too high. Therefore, the control signals were transformed into way points which were fed back to the UAV primary flight control computer at a 20 Hz rate. Note that these are way points from the NMPC optimized controller, not the SPP planner (which were used simply to guide the NMPC optimization).

Figure 8 shows the results of the flight tests. Figure 8 (left) shows the trajectory about the obstacle along with time snapshots of the vehicle showing the UAV attitude. There is some conservatism due to wingspan of the SeaScan (note that the SeaScan figure is drawn 10× scale). The SeaScan UAV performs a hard bank roll, as it should. Although the control inputs do not actually saturate (because only way points are delivered to the flight control computer), it does appear from the roll angle in Fig. 8 (right) that the SeaScan UAV is performing at a limiting effort to achieve the given flight plan. Note that the maximum roll angle of the flight tests (≈15 deg) is much smaller than the roll angle of the HIL tests (≈50 deg) due to the conservative



a) Avoidance trajectory



b) Roll angle

Fig. 8 Flight tests results of the SeaScan UAV executing a hard avoidance maneuver around a pop-up threat; the SeaScan UAV is drawn 10× scale so that it can be examined more easily.

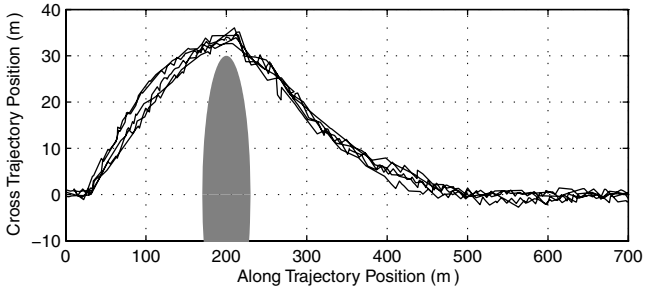


Fig. 9 Multiple obstacle avoidance trajectories from the SeaScan flights, overlaid on the same axes.

nature of the low level control inside the production SeaScan vehicles.

A big challenge in the flight tests was the presence of a strong wind during the tests. The SeaScan UAV was commanded to fly at 25 m/s, while the wind during the tests averaged 12 m/s with gusts up to 20 m/s. Intuitively, because the NMPC continues to reoptimize at a 20 Hz rate, the controller should be robust to some of the uncertainty due to the unmodeled wind. Figure 9 shows the avoidance trajectory results for five tests at different heading angles, overlaid on one plot. Here, the initial SeaScan UAV location, obstacle location, and vehicle velocity are normalized for all tests, thus allowing easier comparison on one plot. Although there is variation due to the high winds, the integrated algorithm does successfully navigate the pop-up obstacle robustly. Modeling the wind in the algorithm, either stochastically or deterministically, would improve the robustness of the results further.

Conclusions

Autonomous control provides a suite of algorithms that enable future complex single and multiple vehicle systems. This paper described an important, integrated set of these algorithms for real time single vehicle control, including stochastic and bounded estimators, nonlinear model predictive control, and streamline path planning. Key developments include real time, integrated implementations of these algorithms. Square root versions of the estimators were developed to maintain numerical accuracy. The model predictive control algorithm used a sequential quadratic programming approach with sparse matrices which enabled the fast development of an initial control solution, and the continued optimization of the solution as time allows. The model predictive control formulation also used a contractive terminal cost which integrated constraints from the bounded set estimator, and, on occasion, a streamline path planner, for robust implementation.

Real time hardware in the loop and flight tests results were demonstrated for all algorithms in a variety of applications. Experimental results include the following: 1) the square root sigma point filter was able to estimate the aerodynamic model very accurately, including during a failure; 2) the streamline path planner, coupled with a tracking estimator, was able to pursue and rendezvous with a target consistently and accurately; and 3) the model predictive control was able to avoid pop-up obstacles consistently and accurately, even during an example control surface failure. Test results showed that it is critical to model the wind (level and direction) in the control and planning for small UAVs in the class of the SeaScan.

Appendix A: Summary of the SR-SPF Algorithm

SR-SPF Prediction

$$\mathcal{X}^{x-}_{i,k+1} = f\left(\mathcal{X}^{x}_{i,k}, \mathbf{u}_k, \mathcal{X}^{w}_{i,k}\right), \qquad i = 1, \ldots, 2n_a + 1 \tag{A1}$$

$$\mathcal{Y}^{-}_{i,k+1} = h\left(\mathcal{X}^{x-}_{i,k+1}, \mathbf{u}_{k+1}, \mathcal{X}^{v}_{i,k+1}\right) \tag{A2}$$

$$\hat{\mathbf{x}}^{-}_{k+1} = \sum_{i=0}^{2n_a} W_i^m \mathcal{X}^{x-}_{i,k+1} \tag{A3}$$

$$\hat{\mathbf{y}}^{-}_{k+1} = \sum_{i=0}^{2n_a} W_i^m \mathcal{Y}^{-}_{i,k+1} \tag{A4}$$

$$\begin{aligned}[\mathcal{X}^{x-}_{c0,k+1} \mid \mathcal{X}^{x-}_{c,k+1}] = [&\mathcal{X}^{x-}_{0,k+1} - \hat{\mathbf{x}}^{-}_{k+1} \\ &\mid \mathcal{X}^{x-}_{1,k+1} - \hat{\mathbf{x}}^{-}_{k+1} \cdots \mathcal{X}^{x-}_{2n_a,k+1} - \hat{\mathbf{x}}^{-}_{k+1}]\end{aligned} \tag{A5}$$

$$\begin{aligned}[\mathcal{Y}^{-}_{c0,k+1} \mid \mathcal{Y}^{-}_{c,k+1}] = [&\mathcal{Y}^{-}_{0,k+1} - \hat{\mathbf{y}}^{-}_{k+1} \\ &\mid \mathcal{Y}^{-}_{1,k+1} - \hat{\mathbf{y}}^{-}_{k+1} \cdots \mathcal{Y}^{-}_{2n_a,k+1} - \hat{\mathbf{y}}^{-}_{k+1}]\end{aligned} \tag{A6}$$

SR-SPF Update

$$\begin{aligned}K_{k+1} = &\left[\left(\mathcal{X}^{x-}_{c,k+1}\right)\left(\mathcal{Y}^{-}_{c,k+1}\right)^T \right. \\ &+ \frac{W_0^c}{W}\left(\mathcal{X}^{x-}_{c0,k+1}\right)\left(\mathcal{Y}^{-}_{c0,k+1}\right)^T\Bigg]\Bigg[\left(\mathcal{Y}^{-}_{c,k+1}\right)\left(\mathcal{Y}^{-}_{c,k+1}\right)^T \\ &\left. + \frac{W_0^c}{W}\left(\mathcal{Y}^{-}_{c0,k+1}\right)\left(\mathcal{Y}^{-}_{c0,k+1}\right)^T\right]^{-1}\end{aligned} \tag{A7}$$

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}^{-}_{k+1} + K_{k+1}(\mathbf{y}_{k+1} - \hat{\mathbf{y}}^{-}_{k+1})$$

$$\begin{aligned}S_{xx,k+1} = &\,\mathrm{up}\left\{\mathrm{orth}\left\{\left(\mathcal{X}^{x-}_{c,k+1} - K_{k+1}\mathcal{Y}^{-}_{c,k+1}\right)\sqrt{W^c}\right\}\right. \\ &\left.\times\left(\mathcal{X}^{x-}_{c0,k+1} - K_{k+1}\mathcal{Y}^{-}_{c0,k+1}\right)\sqrt{|W_0^c|}, \mathrm{sgn}\left(W_0^c\right)\right\}\end{aligned} \tag{A8}$$

$$\hat{\mathbf{x}}^{a}_{k+1} = \begin{bmatrix}\hat{\mathbf{x}}_{k+1} \\ 0 \\ 0\end{bmatrix}$$

$$S^{a}_{xx,k+1} = \begin{bmatrix}S_{xx,k+1} & 0 & 0 \\ 0 & \sqrt{Q_{k+1}} & 0 \\ 0 & 0 & \sqrt{R_{k+1}}\end{bmatrix} \tag{A9}$$

$$\begin{aligned}\mathcal{X}^{a}_{k+1} &= \begin{bmatrix}\mathcal{X}^{x}_{k+1} \\ \mathcal{X}^{w}_{k+1} \\ \mathcal{X}^{v}_{k+1}\end{bmatrix} \\ &= \begin{bmatrix}\hat{\mathbf{x}}^{a}_{k+1} & \hat{\mathbf{x}}^{a}_{k+1}e_{n_a} + \sigma_f S^{a}_{xx,k+1} & \hat{\mathbf{x}}^{a}_{k+1}e_{n_a} - \sigma_f S^{a}_{xx,k+1}\end{bmatrix}\end{aligned} \tag{A10}$$

Appendix B: Summary of the SR-ESMF Algorithm

SR-ESMF Prediction

$$\hat{\mathbf{x}}_{k+1,k} = f(\hat{\mathbf{x}}_{k,k}, \mathbf{u}_k) \tag{B1}$$

$$X^{i}_{k,k} = [\hat{\mathbf{x}}^{i}_{k,k} - \sqrt{\Sigma^{i,i}_{k,k}}, \hat{\mathbf{x}}^{i}_{k,k} + \sqrt{\Sigma^{i,i}_{k,k}}] \qquad i = 1, \ldots, n \tag{B2}$$

$$X^{n}_{R_{k,k}} = \mathrm{diag}_n\left(X^{T}_{k,k}\right)\begin{bmatrix}H_1^f \\ \vdots \\ H_n^f\end{bmatrix}_{k,k} X_{k,k} \tag{B3}$$

$$[\overline{Q}_k]^{i,j}_{R_{k,k}} = \begin{cases}2\left(X^{i+}_{R_{k,k}} - X^{i-}_{R_{k,k}}\right)^2, & i = j \\ 0, & i \neq j\end{cases} \tag{B4}$$

$$\hat{Q}_k = \frac{\overline{Q}_k}{1-\beta_{Q,k}} + \frac{Q_k}{\beta_{Q,k}} \tag{B5}$$

$$\Sigma_{k+1,k}^{1/2} \rightarrow \begin{bmatrix} \Sigma_{k+1,k}^{1/2} & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{1-\beta_k}} A_k \Sigma_{k,k}^{1/2} & \frac{1}{\sqrt{\beta_k}} \hat{Q}_k^{1/2} \end{bmatrix} \Theta \tag{B6}$$

SR-ESMF Update

$$X_{k+1,k}^i = [\hat{\mathbf{x}}_{k+1,k}^i - \sqrt{\Sigma_{k+1,k}^{i,i}}, \hat{\mathbf{x}}_{k+1,k}^i + \sqrt{\Sigma_{k+1,k}^{i,i}}] \qquad i = 1, \ldots, n \tag{B7}$$

$$X_{R_{k+1,k}}^n = \mathrm{diag}_{n_y}\left(X_{k+1,k}^T\right) \begin{bmatrix} H_1^h \\ \vdots \\ H_n^h \end{bmatrix}_{k+1,k} X_{k+1,k} \tag{B8}$$

$$[\overline{R}_{k+1}]_{R_{k+1,k}}^{i,j} = \begin{cases} 2(X_{R_{k+1,k}}^{i+} - X_{R_{k+1,k}}^{i-})^2, & i = j \\ 0, & i \neq j \end{cases} \tag{B9}$$

$$\hat{R}_{k+1} = \frac{\overline{R}_{k+1}}{1-\beta_{R,k+1}} + \frac{R_{k+1}}{\beta_{R,k+1}} \tag{B10}$$

$$R_{a,k+1}, \bar{\Sigma}_{k+1,k+1}^{1/2} \rightarrow \begin{bmatrix} R_{a,k+1} & 0 \\ \frac{\Sigma_{k+1,k} C_{k+1}^T (R_{a,k+1}^T)^{-1}}{1-\rho_{k+1}} & \bar{\Sigma}_{k+1,k+1}^{1/2} \end{bmatrix} = \begin{bmatrix} \frac{\hat{R}_{k+1}^{1/2}}{\sqrt{\rho_{k+1}}} & C_{k+1} \frac{\Sigma_{k+1,k}^{1/2}}{\sqrt{1-\rho_{k+1}}} \\ 0 & \frac{\Sigma_{k+1,k}^{1/2}}{\sqrt{1-\rho_{k+1}}} \end{bmatrix} \Phi \tag{B11}$$

$$K_{a,k+1} = \Sigma_{k+1,k} C_{k+1}^T R_{a,k+1}^{-T} \tag{B12}$$

$$\hat{\mathbf{x}}_{k+1,k+1} = \hat{\mathbf{x}}_{k+1,k} + K_{a,k+1} R_{a,k+1} [\mathbf{y}_{k+1} - h(\hat{\mathbf{x}}_{k+1,k}, \mathbf{u}_{k+1})] \tag{B13}$$

$$\Sigma_{k+1,k+1}^{1/2} = (1-\delta_{k+1}) \bar{\Sigma}_{k+1,k+1}^{1/2} \tag{B14}$$

$$\delta_{k+1} = (1-\beta_k)(1-\rho_{k+1})\delta_k + [\mathbf{y}_{k+1} - h(\hat{\mathbf{x}}_{k+1,k}, \mathbf{u}_{k+1})]^T R_{a,k+1}^{-T} R_{a,k+1}^{-1} [\mathbf{y}_{k+1} - h(\hat{\mathbf{x}}_{k+1,k}, \mathbf{u}_{k+1})] \tag{B15}$$

where

$$A_k = \left.\frac{\partial f(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_{k,k}}, \qquad C_{k+1} = \left.\frac{\partial h(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})}{\partial \mathbf{x}_{k+1}}\right|_{\mathbf{x}_{k+1} = \hat{\mathbf{x}}_{k+1,k+1}}$$

$$[H_i^{(f,h)}]_{k,k} = \left.\frac{\partial (f,h)_i (\mathbf{x}_k, \mathbf{u}_k)^2}{\partial \mathbf{x}_k^2}\right|_{\mathbf{x}_k = \hat{\mathbf{x}}_{(k+1,k),k}}$$

Θ, Φ: unitary matrices, $\beta_k, \rho_{k+1}, \beta_Q, \beta_R \in (0, 1)$

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Software Enabled Control program (no. F33615-99-C-3612), with Helen Gill and John Bay as DARPA program monitors and Raymond Bortner and James McDowell from the Air Force Research Laboratory (AFRL) as contract monitors.

References

[1] Antsaklis, P., and Passino, K., "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues," *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 1, No. 5, 1989, pp. 315–342.

[2] Sullivan, J., Waydo, S., and Campbell, M., "Using Stream Functions for Complex Behavior and Path Generation," AIAA Paper 2003–5800, 2003.

[3] Frazzoli, E., Dahleh, M. A., and Feron, E., "A Maneuver-Based Hybrid Control Architecture for Autonomous Vehicle Motion Planning," *Software Enabled Control, Information Technology for Dynamical Systems*, edited by T. Samad and G. Balas, IEEE Press/Wiley, Piscataway, NJ, 2003, Chap. 15, pp. 299–324.

[4] Richards, A., and How, J., "A Decentralized Algorithm for Robust Constrained Model Predictive Control," *American Control Conference*, IEEE, Piscataway, NJ, 2004, pp. 4261–4266.

[5] Milam, M. B., Mushambi, K., and Murray, R. M., "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems," *IEEE Conference on Decision and Control*, IEEE, Piscataway, NJ, 2000, pp. 845–851.

[6] Murray, R. M., Hauser, J., Jadbabaie, A., Milam, M. B., Petit, N., Dunbar, W. B., and Franz, R., "Online Control Customization via Optimization Based Control," *Software Enabled Control, Information Technology for Dynamical Systems*, edited by T. Samad and G. Balas, IEEE Press/Wiley, Piscataway, NJ, 2003, Chap. 6, pp. 149–174.

[7] Bhattacharya, R., Balas, G. J., Kaya, M. A., and Packard, A., "Nonlinear Receding Horizon Control of an F-16 Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, 2002.

[8] Vachtsevanos, G., Rufus, F., Prasad, J. V. R., Yavrucuk, I., Schrage, D., Heck, B., and Willis, L., "An Intelligent Methodology for Real-Time Adaptive Mode Transitioning and Limit Avoidance of Unmanned Aerial Vehicles," *Software Enabled Control, Information Technology for Dynamical Systems*, edited by T. Samad and G. Balas, IEEE Press/Wiley, Piscataway, NJ, 2003, Chap. 12, pp. 225–252.

[9] Campbell, M. E., Scholte, E., and Brunke, S., "Active Model Estimation for Complex Autonomous Systems," *Software Enabled Control, Information Technology for Dynamical Systems*, edited by T. Samad and G. Balas, IEEE Press/Wiley, Piscataway, NJ, 2003, Chap. 8, pp. 201–224.

[10] Karsai, G., Biswas, G., Narasimhan, S., Pasternak, T., Abdelwahed, S., Szemethy, T., Peceli, G., Simon, G., and Kovacshazy, T., "Towards Fault-Adaptive Control of Complex Dynamical Systems," *Software Enabled Control, Information Technology for Dynamical Systems*, edited by T. Samad and G. Balas, IEEE Press/Wiley, Piscataway, NJ, 2003, Chap. 17, pp. 347–368.

[11] Murphey, R., "An Approximate Algorithm for a Weapon Target Assignment Stochastic Program," *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, Springer, New York, 1999.

[12] Samad, T., and Balas, G. (eds.), *Software Enabled Control, Information Technology for Dynamical Systems*, IEEE Press/Wiley, Piscataway, NJ, 2003.

[13] Murphey, R., and Pardalos, P. (eds.), *Cooperative Control and Optimization*, Springer, New York, 2002.

[14] Paunicka, J. L., Mendel, B. R., and Corman, D. E., "The OCP—An Open Middleware Solution for Embedded Systems," *American Control Conference*, IEEE, Piscataway, NJ, 2001, pp. 3445–3450.

[15] Anon., "Open Control Platform: A Software Platform Supporting Advances in UAV Control Technology," *Software Enabled Control, Information Technology for Dynamical Systems*, edited by T. Samad and G. Balas, IEEE Press/Wiley, Piscataway, NJ, 2003, Chap. 4, pp. 39–62.

[16] King, E., Kuwata, Y., Alighanbari, M., Bertuccelli, L., and How, J. P., "Coordination and Control Experiments on a Multi-Vehicle Testbed," *American Control Conference*, IEEE, Piscataway, NJ, 2004, pp. 5315–5320.

[17] McLain, T. W., and Beard, R. W., "Unmanned Air Vehicle Testbed for Cooperative Control Experiments," *American Control Conference*, IEEE, Piscataway, NJ, 2004, pp. 5327–5331.

[18] Teo, R., Jang, J. S., and Tomlin, C. J., "Automated Multiple UAV Flight the Stanford Dragonfly UAV Program," *Conference on Decision and Control*, IEEE, Piscataway, NJ, 2004, pp. 4268–4273.

[19] Brunke, S., and Campbell, M. E., "Square Root Sigma Point Filtering for Aerodynamic Model Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, 2004, pp. 314–317.
[20] Scholte, E., and Campbell, M., "A Nonlinear Set-Membership Filter for On-Line Applications," *International Journal of Robust and Nonlinear Control*, Vol. 13, No. 15, 2003, pp. 1337–1358.
[21] Anon., "Robust Nonlinear Model Predictive Control with Partial State Information," AIAA Paper 2003-5804, 2003.
[22] Waydo, S., and Murray, R., "Vehicle Motion Planning Using Stream Functions," *IEEE International Conference on Robotics and Automation*, IEEE, Piscataway, NJ, 2003, pp. 2484–2491.
[23] Campbell, M. E., Han, J., Lee, J., Scholte, E., and Ousingsawat, J., "Validation of Active State Model Based Control Using the Seascan UAV," AIAA Paper 2003-6540, 2003.
[24] Campbell, M., D'Andrea, R., Lee, J.-W., and Scholte, E., "Experimental Demonstrations of Semi-Autonomous Control," *American Control Conference*, IEEE, Piscataway, NJ, 2004, pp. 5338–5343.
[25] Julier, S., Uhlmann, J., and Durrant-Whyte, H. F., "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Transactions on Automatic Control*, Vol. 45, No. 3, 2000, pp. 477–482.
[26] Brunke, S., and Campbell, M., "Estimation Architecture for Future Autonomous Vehicles," *American Control Conference*, IEEE, Piscataway, NJ, 2002, pp. 1108–1114.
[27] Allgöwer, F., and Zheng, A. (eds.), *Nonlinear Model Predictive Control*, Progress in Systems and Control Theory, Birkhaüser, Cambridge, MA, 2000, Vol. 26.
[28] Jadbabaie, A., Yu, J., and Hauser, J., "Receding Horizon Control of the Caltech Ducted Fan: A Control Lyapunov Function Approach," *IEEE International Conference on Control Applications*, IEEE, Piscataway, NJ, 1999, pp. 51–56.
[29] Yang, T. H., and Polak, E., "Moving Horizon Control of Nonlinear Systems with Input Saturation, Disturbances and Plant Uncertainty," *International Journal of Control*, Vol. 58, No. 4, 1993, pp. 875–903.
[30] Oliveira, S. D., and Morari, M., "Contractive Model Predictive Control for Constrained Nonlinear Systems," *IEEE Transactions on Automatic Control*, Vol. 45, No. 6, 2000, pp. 1053–1071.
[31] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., *NPSOL—Nonlinear Programming Software*, Stanford Business Software, Inc., Mountain View, CA, 2000.
[32] Nocedal, J., and Wright, S. J., *Numerical Optimization*, Springer–Verlag, New York, 1999.
[33] Mahadevan, R., and Doyle, F. J., III, "Efficient Optimization Approaches to Nonlinear Model Predictive Control," *International Journal of Robust and Nonlinear Control*, Vol. 13, Nos. 3–4, 2003, pp. 309–329.
[34] Wright, S. J., and Tenny, M. J., "A Feasible Trust-Region Sequential Quadratic Programming Algorithm," University of Wisconsin, Optimization TR 02-05, 2002.
[35] Tenny, M. J., Wright, S. J., and Rawlings, J. B., "Nonlinear Model Predictive Control via Feasibility-Perturbed Sequential Quadratic Programming," University of Wisconsin, Optimization TR 02-06, 2002.
[36] Rao, C. V., Wright, S. J., and Rawlings, J. B., "On the Application of Interior Point Methods to Model Predictive Control," *Journal of Optimization Theory and Applications*, Vol. 99, No. 3, 1998, pp. 723–757.
[37] Lawson, C. L., Hanson, R. J., Kincaid, D., and Krogh, F. T., "Basic Linear Algebra Sub-Programs for Fortan Usage," *ACM Transactions on Mathematical Software*, Vol. 5, No. 3, 1979, pp. 308–323.
[38] Whaley, R. C., Petitet, A., and Dongarra, J. J., "Automated Empirical Optimization of Software and the Atlas Project," *Parallel Computing*, Vol. 27, Nos. 1–2, 2001, pp. 3–35, http://www.netlib.org/lapack/lawns/lawn147.ps.
[39] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D., *LAPACK Users Guide*, 3rd ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[40] Stevens, B. L., and Lewis, F. L., *Aircraft Control and Simulation*, Wiley, Hoboken, NJ, 2003.
[41] Mulder, J. A., Sridhar, J. K., and Breeman, J. H., *Part 2: Nonlinear Analysis and Manoevre Design*, Identification of Dynamic Systems—Applications to Aircraft, North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development, 1994, AGARD-AG-300-VOL-3-PT-2.
[42] Brunke, S., "Nonlinear Filtering and System Identification Algorithms for Complex Autonomous Systems," Ph.D. Dissertation, University of Washington, Seattle, WA, 2001.