Real-Time Moving Horizon Estimation of Air Data Parameters and Wind Velocities for fixed-wing UAVs

Andreas Wenz *, Tor Arne Johansen*
*Centre for Autonomous Marine Operations and Systems
Department of Engineering Cybernetics
Norwegian University of Science and Technology, Trondheim, Norway

***Abstract*—We present a real-time implementation of an estimation algorithm for angle of attack, airspeed and wind velocities estimation on a single board computer. The estimator uses only sensor data from a standard fixed-wing UAV autopilot, which consists of a Global Navigation Satellite System receiver, an inertial measurement unit and a pitot-static tube. This sensor data is fused with a combination of kinematic, aerodynamic and stochstic wind models in a nonlinear moving horizon estimator using numerical optimization. An algorithmic differentiation toolbox and automatic code generation is used to create a real-time capable estimator which is able to run within a UAV on an on-board computer. Hardware in the Loop simulation results show that the latency of the estimator is significantly below the expected wind gust period and gives low root-mean-square estimation errors for angle of attack** ($0.29°$) **, airspeed** ($0.21m/s$) **and wind velocities** ($0.44m/s$)**.**

I. Introduction

Knowledge of the air data parameters, angle of attack (AoA) α, angle of sideslip β and airspeed V_a, is critical to assess a fixed-wing aircraft's performance and safety. If the angle of attack is increased over a certain value the flow over the airfoil separates and the aircraft stalls, which results in a sudden loss of lift. This can cause the aircraft to reach an unstable flight state which it might be difficult to recover from. A recent study on UAV mishaps [1] found that a majority of accidents can be attributed to *Loss of Control* events which includes stall events.

In addition to the air data parameters, the prevailing wind velocities are also quantities of interest. Knowing the wind velocities allows the operator to use an optimized path for the UAV, which minimizes flight time or energy consumption [2], [3]. If the wind velocities are known, the air data parameters can be calculated using kinematic relationships [4], [5].

It is common to equip larger aircraft with dedicated sensors to measure the airdata parameters. For smaller fixed wing aircraft, especially small unmanned aerial vehicles (UAVs), no cost-effective, lightweight sensor solution exists that can measure all air data parameters reliably. Recent research has therefore focused on estimating the air data parameters using kinematic and aerodynamic models as well as sensor data from the UAVs autopilot system. A typical autopilot sensor suite for small UAVs consists of a Global Navigation Satellite System (GNSS) receiver, measuring position and velocity, three axis inertial measurement units (IMU), measuring specific forces and angular velocities, a heading reference system (i.e. a magnetometer) and a pitot static tube, which measures the airspeed. A navigation filter within the autopilot uses this input data to estimate attitude, position and velocity of the aircraft at a high rate.

Approaches to air data estimation can mainly be categorized into two main groups: Model based approaches, which utilize an aerodynamic model, and model-free approaches which are based on exact kinematic relationships. The first group includes [6] where a detailed aircraft model and a nonlinear observer is used for wind estimation. A hybrid system approach using Bayesian estimation is presented in [7], achieving promising results. Air data estimation methods using simplified aerodynamic and kinematic models have been presented in [7], [4] and [6]. In [8] and [9] approaches to air data estimation for jet aircraft using a full model of the aerodynamics are presented. The disadvantage of these methods is that they require a large set of aerodynamic parameters which typically have to be identified using wind tunnel experiments or computational fluid dynamics analysis. In most cases this data is unavailable for specific UAV airframes, since they are expensive to obtain.

Into the second group falls the work presented in [10] where kinematic relationships are used to estimate air data parameters and wind velocities. Four different wind velocity estimation methods, using kinematic models combined with different sensor sets in a UKF are studied in [11]. Popular estimation methods to combine kinematic models with sensor data are the Extended Kalman Filter (EKF), which has been used in [12]–[14], and the Unscented Kalman Filter (UKF) which has been applied to the problem in [15], [16]. Wind velocity estimation methods based on kinematic models are easier to apply to the scenario of small UAVs. However, their performance is limited by the need for excitation since not all wind directions can be observed at all times [10], and the assumption that the wind field is static which causes estimation errors in the case of turbulence.

This paper is based on previous work published in [17]. There we presented a Moving Horizon Estimator (MHE) that is able to estimate the air data parameters as well as wind

velocities and lift coefficients based on above described sensor set and the combination of kinematic, aerodynamic and stochastic wind models. Data from extensive flight tests with two different air frames were presented in [17] to validate this approach. The results show that the proposed method is able to estimate the angle of attack and airspeed with low estimation errors which are within the same accuracy as the measurement errors of the reference measurement used in those experiments. Accuracy of the sideslip angle estimate is limited by the lack of a lateral model. Furthermore, the estimator provides estimates of the lift coefficients which can be used for icing detection as shown in [18].

So far the estimator has only been tested offline on previous recorded flight data. The computational complexity is high since a nonlinear optimization problem must be solved repeatedly. In this paper we show how the estimator can be implemented within a UAV avionics system and that it is able to run in real-time on a single-board computer. In order to verify this setup hardware in the loop (HIL) tests with a high-fidelity aircraft simulation model and the autopilot software is used. The assessment includes both the real-time capabilities and the accuracy of the estimator.

II. Methodology

In this section we will first summarize the modeling of the problem and the proposed estimator. For a more detailed discussion we refer to [17]. We will then show how that estimator has been implemented in the DUNE navigation environment [19], which is run on an Odroid XU4 single board computer that is interfaced to a Pixhawk autopilot. Afterwards the simulation setup will be presented.

A. Modeling

The modeling process is based on [17]. Variables to be estimated are the static and turbulent wind velocities decomposed in inertial frame $\boldsymbol{v}_s^n$ and $\boldsymbol{v}_t^n$, the scaled constant and linear lift coefficients $KC_{L,0}$ and $KC_{L,\alpha}$ as well as the pitot-tube scaling factor γ. The lift coefficient scaling factor K is given by $K = \frac{\rho S}{2m}$, where m is the aircraft's mass, S is the surface area of the airfoil and ρ is the air density. Since only the turbulent wind velocities are assumed to be dynamically changing we only consider them in the state vector $\boldsymbol{x}$, and treat the other variables as parameters in the vector $\boldsymbol{p}$.

$$\boldsymbol{x} = \begin{bmatrix} u_t^n & v_t^n & w_t^n \end{bmatrix}^T \tag{1}$$

$$\boldsymbol{p} = \begin{bmatrix} u_s^n & v_s^n & w_s^n & KC_{L_0} & KC_{L_\alpha} & \gamma \end{bmatrix}^T \tag{2}$$

The inputs to the system are the velocity over ground decomposed in body frame

$$\boldsymbol{v}^b = \begin{bmatrix} u^b & v^b & w^b \end{bmatrix}^T = \boldsymbol{R}_n^b \boldsymbol{v}_{GNSS}^n \tag{3}$$

where $\boldsymbol{v}_{GNSS}^n$ is the velocity over ground decomposed in inertial frame measured by a GNSS receiver, altitude above ground h, and attitude given by a rotation matrix $\boldsymbol{R}_n^b$. The inputs are gathered in an input vector $\boldsymbol{u}$.

The continuous-time state transition function is based on the discrete Dryden wind model [20] and is given by:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = -\| \boldsymbol{v}^b - \boldsymbol{R}_n^b (\boldsymbol{v}_s^n + \boldsymbol{v}_t^n) \| \cdot \boldsymbol{L} \boldsymbol{v}_t^n \tag{4}$$

where the spatial wavelengths $\boldsymbol{L} = \operatorname{diag}\left(\begin{bmatrix} L_u^{-1} & L_v^{-1} & L_w^{-1} \end{bmatrix}\right)$ are given by:

$$L_u = L_v = \frac{h}{(0.177 + 0.0027 \cdot h)^{1.2}} \tag{5}$$

$$L_w = h \tag{6}$$

The random noise transition function due to turbulence is given by

$$\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu_x}, \boldsymbol{p}) = \begin{pmatrix} \sigma_u \sqrt{2\Delta T \frac{V_a}{L_u}} \nu_{u_t} \\ \sigma_v \sqrt{2\Delta T \frac{V_a}{L_v}} \nu_{v_t} \\ \sigma_w \sqrt{2\Delta T \frac{V_a}{L_w}} \nu_{w_t} \end{pmatrix} \Bigg|_{\substack{\boldsymbol{u} \\ \boldsymbol{x} \\ \boldsymbol{p}}} \tag{7}$$

with unknown inputs $\boldsymbol{\nu_x} = \begin{bmatrix} \nu_{u_t} & \nu_{v_t} & \nu_{w_t} \end{bmatrix}^T$, which are modeled as random Gaussian white noise processes, and V_a defined as [4], [17], [21]:

$$V_a = \| \boldsymbol{v}_r^b \| \tag{8}$$

where $\boldsymbol{v}_r^b = \begin{bmatrix} u_r & v_r & w_r \end{bmatrix}^T = \boldsymbol{v}^b - \boldsymbol{R}_n^b (\boldsymbol{v}_s^n + \boldsymbol{v}_t^n)$ is called the relative air velocity vector which is defined as the difference between velocity over ground and wind velocity both decomposed in body frame.

The turbulence amplitudes are given by:

$$\sigma_u = \sigma_v = V_{w,G} \frac{1}{(0.177 + 0.0027 \cdot h)^{0.4}} \tag{9}$$

$$\sigma_w = 0.1 \cdot V_{w,G} \tag{10}$$

where $V_{w,G}$ denotes the wind velocity measured 6 meters above ground.

The predicted state at a discrete time step k is then given by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta T \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}_k) + \boldsymbol{w}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\nu}_{\boldsymbol{x},k}, \boldsymbol{p}_k) \tag{11}$$

where ΔT is the discretization time step.

The parameters $\boldsymbol{p}$ are assumed to be slowly time varying

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{\nu}_{\boldsymbol{p},k} \tag{12}$$

where $\boldsymbol{\nu_p}$ is Gaussian distributed white process noise. The covariance matrix of the combined process noise $\boldsymbol{\nu_q} = \begin{bmatrix} \boldsymbol{\nu_x} & \boldsymbol{\nu_p} \end{bmatrix}$ is given by $\boldsymbol{Q}$.

In order to obtain a measurement model we use a simplified aerodynamic lift model and a kinematic model. The first measurement equation uses the z-accelerometer measurement together with a linearized aerodynamic lift model to relate the aerodynamic coefficients and the wind velocities via the indicated airspeed V_a^m and angle of attack α (cf. [17], [21], [22]).

$$f_z = -(V_a^m)^2 K (C_{L,0} + \alpha C_{L,\alpha}) + \nu_a \tag{13}$$

f_z is the specific force in vertical direction measured by an accelerometer and ν_a is the modeling error which we assume to be a white Gaussian noise process. Note that if

the measurement of the indicated airspeed $\tilde{V}_a^m$ is affected by Gaussian white measurement noise η_{V_a}, then $(\tilde{V}_a^m)^2$ will be biased:

$$E\left[\left(\tilde{V}_a^m\right)^2\right] = E\left[\left(V_a^m + \eta_{V_a}\right)^2\right] = E\left[V_a^m\right]^2 + E\left[\eta_{V_a}^2\right] \quad (14)$$

$$= \left(V_a^m\right)^2 + \sigma_{V_a}^2 \quad (15)$$

Therefore, we subtract the noise covariance $\sigma_{V_a}^2$ from the squared measurement.

The second measurement equation $V_a^m = \gamma V_a$ uses the definition of the airspeed (8) and the calibration factor γ to relate the measured airspeed to the wind velocities.

The third and fourth measurement equations can be obtained by utilizing the wind triangle

$$\boldsymbol{v}_r^b = \boldsymbol{v}^b - \boldsymbol{R}_n^b\left(\boldsymbol{v}_s^n + \boldsymbol{v}_t^n\right) = V_a \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix} \quad (16)$$

and the definition of the angle of attack α and sideslip angle β,

$$\alpha = \tan^{-1}\left(\frac{w_r}{u_r}\right) \quad (17)$$

$$\beta = \sin^{-1}\left(\frac{v_r}{V_a}\right) \quad (18)$$

which under the assumption $\cos\beta \approx 1$ can be rearranged to:

$$\begin{bmatrix} u^b \\ w^b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_n^b \boldsymbol{v}_w^n + V_a \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix} + \boldsymbol{\nu}_{ki} \quad (19)$$

where ν_{ki} denotes the approximation error. This relates the GNSS measurements of the velocity over ground $\boldsymbol{v}^b$, given by (3), to the wind velocities and the relative airspeed V_a given by second measurement equation as well as the angle of attack α. This is similar to the method described in [10].

This results in the following measurement model:

$$\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) = \begin{bmatrix} -K\left(V_a^m\right)^2\left(C_{L_0} + C_{L_\alpha}\alpha\right) \\ V_a\gamma \\ \boldsymbol{d}_1\boldsymbol{R}_n^b(\boldsymbol{v}_s^n + \boldsymbol{v}_t^n) + V_a\cos(\alpha) \\ \boldsymbol{d}_3\boldsymbol{R}_n^b(\boldsymbol{v}_s^n + \boldsymbol{v}_t^n) + V_a\sin(\alpha) \end{bmatrix} + \boldsymbol{\nu}_y \quad (20)$$

with

$$\boldsymbol{d}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$
$$\boldsymbol{d}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

where V_a and α are functions of $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{p}$.

The measurement vector is given by

$$\tilde{\boldsymbol{z}}_k = \begin{bmatrix} \tilde{f}_z \\ \tilde{V}_a^m \\ \tilde{u}^b \\ \tilde{w}^b \end{bmatrix} = \begin{bmatrix} f_z + \eta_{f_z} \\ V_a^m + \eta_{V_a} \\ u^b + \eta_{u^b} \\ w^b + \eta_{w^b} \end{bmatrix} = \boldsymbol{z}_k + \boldsymbol{\eta}_z \quad (21)$$

where $\boldsymbol{\eta}_z = \begin{bmatrix} \eta_{f_z} & \eta_{V_a} & \eta_{u^b} & \eta_{w^b} \end{bmatrix}^T$ is the output measurement noise. Since the velocity over ground is both an input and an output to the system, input and output noise variables are combined in a measurement noise vector $\boldsymbol{\eta}_m = \begin{bmatrix} \boldsymbol{\eta}_{v^b} & \eta_{V_a} & \eta_{f_z} \end{bmatrix}^T$ with a covariance matrix $\boldsymbol{W}_m$. The approximation errors within the models used in the measurement function are summarized in the noise variable $\boldsymbol{\nu}_{\boldsymbol{y}} = \begin{bmatrix} \nu_a & \nu_v & \boldsymbol{\nu}_{ki} \end{bmatrix}$ which is assumed to be white Gaussian noise with a covariance matrix $\boldsymbol{R}$.

We have chosen to separate noise variables for the measurement noise of input and output variables ($\boldsymbol{\eta}_m$) and the process noise resulting from model mismatches in the state transition ($\boldsymbol{\nu}_q$) and output functions ($\boldsymbol{\nu}_y$). The main motivation for this noise modeling is ease of tuning of the estimator since measurement noise covariances can be chosen according to sensor specifications, whereas the covariance matrices of the process noise can be chosen according to the magnitude of the errors in the aerodynamic (13) and kinematic (16) model.

B. Moving Horizon Estimation

In order to estimate the desired states and parameters we use a Moving Horizon Estimator (MHE). A MHE minimizes an objective function J_k in each iteration, which includes the deviations of the model output from the measurements $\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p})$ and the quadratic norm of the noise variables $\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\nu}_q & \boldsymbol{\nu}_y & \eta_m \end{bmatrix}$ at each sampling point $k-L,\ldots,k$, where k is the current sampling point and L is the window length.

$$J_k = \sum_{j=k-L}^{k} \left\| z - \boldsymbol{h}(\boldsymbol{x_j},\boldsymbol{u_j},\boldsymbol{p_j}) \right\|_{\boldsymbol{R}^{-1}}^2 + \sum_{j=k-L}^{k-1} \left\| \boldsymbol{\eta}_j \right\|_{\boldsymbol{W}^{-1}}^2 \quad (22)$$

where $\boldsymbol{W}$ is the covariance matrix of the noise variables in $\boldsymbol{\eta}$, adjusted for the sampling time, given by:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{0}_{[9x5]} \\ \boldsymbol{0}_{[5x9]} & \boldsymbol{W}_m \end{bmatrix} \Delta T \quad (23)$$

Since the window length can not be arbitrarily long, we add an arrival cost approximation to the objective function which summarizes the information obtained before the current filtering interval and get the new objective function

$$\Theta_k = \left\| \begin{matrix} \boldsymbol{x}_{k-L} - \hat{\boldsymbol{x}}_{k-L} \\ \boldsymbol{p}_{k-L} - \hat{\boldsymbol{p}}_{k-L} \end{matrix} \right\|_{\hat{\boldsymbol{P}}_{k-L}^{-1}}^2 + J_k \quad (24)$$

$\hat{\boldsymbol{x}}_{k-L}$, $\hat{\boldsymbol{p}}_{k-L}$ and $\hat{\boldsymbol{P}}_{k-L}^{-1}$ are estimated using a Unscented Kalman Filter.

The nonlinear program that has to be solved in each iteration is then

$$\min_{\boldsymbol{\theta}_k} \Theta_k(\boldsymbol{\theta}_k, \boldsymbol{u}_k, \boldsymbol{z}_k) \quad (25)$$

where the vector $\boldsymbol{\theta}_k$ includes the vectors $\boldsymbol{x}_j, \boldsymbol{p}_j, \boldsymbol{\eta}_j$ at each time step $j = k-L,\ldots,k$ inside the current window.

The solution of this nonlinear program is subject to both equality and inequality constraints. To ensure that the solution fulfills the state transition function (11) we add the following equality constraints:

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \Delta T\, \boldsymbol{f}(\boldsymbol{x}_j,\boldsymbol{u}_j,\boldsymbol{p}_j) + \boldsymbol{w}(\boldsymbol{x}_j,\boldsymbol{u}_j,\boldsymbol{\nu}_{\boldsymbol{x},j},\boldsymbol{p}_j) \quad (26)$$

$$\boldsymbol{p}_{j+1} = \boldsymbol{p}_j + \boldsymbol{\nu}_{\boldsymbol{p},j} \quad (27)$$

$$\text{for } j = k-L,\cdots,k-1$$

Furthermore, we use inequality constraints to bound the angle of attack in order to force a unique solution of the

trigonometric functions used in the model and to limit the lift coefficients to a region around a-priori known values.

Observability of this system has been analyzed and discussed in [17]. Assessing observability of nonlinear systems analytically is challenging and often does not yield results that are of practical use. Therefore, a numerical analysis of the local observability was done by checking the eigenvalues of the estimated covariance matrix $\hat{\boldsymbol{P}}_{k-L}$ for singularity. During several test flights no occurance of singularity was recorded and we consequently assume the system to be locally observable.

For a more detailed discussion of the MHE and the arrival cost approximation we refer to [17].

C. Implementation

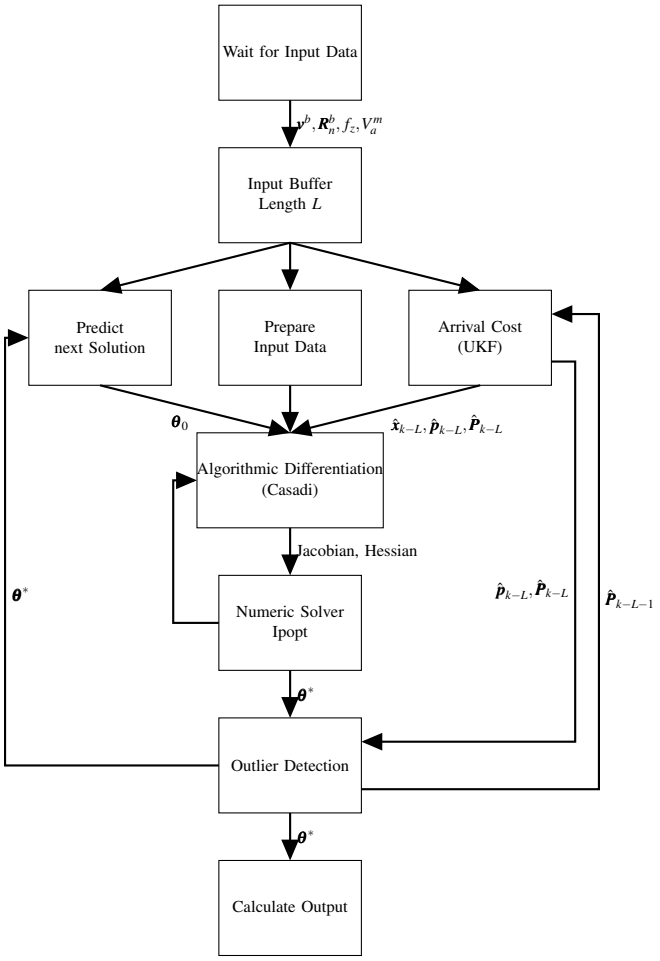

Fig. 1: Program Structure

Figure 1 shows a flow-chart to illustrate the structure of the estimation task. We have chosen an asynchronous implementation where a new estimation iteration is only performed when the previous one is completed. Each iteration begins with waiting to receive messages from all input data sources. The accelerometer data is filtered with a FIR lowpass filter in order to reduce noise from the sensor and vibrations.

The samples are then saved in a buffer $\boldsymbol{B}$ which has the same length as the MHE horizon. It saves the sensor data of the system inputs $\boldsymbol{u}_j$ and outputs $\boldsymbol{z}_j$. At startup, we wait until the buffer is full before we start any estimation. In addition, an altitude threshold is imposed below which samples are skipped in order to differentiate whether the aircraft is flying or on ground. In order to avoid oversampling, which would cause the timespan in the current estimation window to be too short to hold meaningful information, a minimal sampling time is specified. If the measured sampling period is below that threshold the sample is skipped. The minimal sampling period ΔT_{min} should be chosen to be half of the desired output. Note that when using an accelerometer, the sensor noise can be reduced by using a delta velocity output that averages over several samples. The specific force measurement is then given by $f_z = \frac{\Delta v_z}{\Delta T}$.

Next we calculate the initial guess for the optimization algorithm, prepare the input data and compute the arrival cost. We use *warm starting* to determine the initial guess θ_0 [23]:

1) Shift previous solution vector $\boldsymbol{\theta}^*$ by l elements, and remove the oldest l elements
2) Predict the value of x_k using x_{k-1}, the new input $\boldsymbol{u}$ and the state transition function (11)
3) Parameter values are kept constant and noise variables set to zero

where $l = \text{length}(\boldsymbol{x}) + \text{length}(\boldsymbol{p}) + \text{length}(\boldsymbol{\eta})$. The input data to the optimization algorithm includes the buffered sensor data $\boldsymbol{B}$ as well as the noise matrices $\boldsymbol{R}$ and $\boldsymbol{W}$.

Next, Casadi computes the Hessian and Jacobian of the objective function Θ at θ_0 and supplies them to the numeric solver, which in our case is Ipopt. The solver then searches for a new θ with $\Theta(\boldsymbol{\theta}, \boldsymbol{B}) < \Theta(\boldsymbol{\theta}_0, \boldsymbol{B})$. This is repeated until a convergence criterion is reached or the maximum number of solver iterations is reached. $\boldsymbol{\theta}^*$ denotes the so found solution.

Disturbances in the signal transmission or in the sensors themselves can create outliers in the input data. In order to detect those outliers, we compare the deviation of the parameters $\hat{\boldsymbol{p}}_k^*$ from the current solution $\boldsymbol{\theta}^*$ to the parameters predicted by the UKF $\hat{\boldsymbol{p}_{k-L}}$ with the estimated covariance $\hat{\boldsymbol{P}}_{k-L}$, using

$$\|\hat{p}_{j,k-L} - p_{j,k}^*\| > 3\sqrt{\hat{P}_{k-L,jj}} \tag{28}$$

as a criterion for outlier rejection. In that case the previous solution is used, the estimated covariance is not updated and all measurements of that timestep are removed from the buffer $\boldsymbol{B}$. In order to avoid falsely rejecting samples, a maximum number of consecutive outliers can be specified.

Using the estimated $\boldsymbol{x}_k^*$ and $\boldsymbol{p}_k^*$, as well as the inputs $\boldsymbol{u}_k$ we can now calculate the desired quantities, angle of attack, sideslip angle and airspeed susing equations (8),(17) and (18).

All elements in Figure 1, but the *Algorirthmic Differentiation* and *Numeric Solver* blocks were first implemented in Matlab and then converted to C code using Matlab's code-generation functionality to create a set of library functions. The system was specified as Casadi [24] code and then the NLP constructed as described in section II-B. Casadi is then able to perform the needed differentiations symbolically and

produce C++ code which can be compiled into a standalone executable.

The estimator was then implemented in C++ as a task in the DUNE unified navigation environment [19]. DUNE provides a Linux based framework to coordinate a set of different processes, called tasks, which can interact through inter-module communication (IMC) messages. On launch a solver object is created which links the Casadi executable to the Task and the solver IPOPT [25]. Within IPOPT we use the solver MA27 out of the HSL package [26] as a linear solver.

D. Simulation Setup

The setup of the Hardware in the Loop (HiL) tests is depicted in Figure 2. As shown the setup consists of four main parts:

1) A model of a Skywalker X8 flying wing implemented in Simulink. The simulation model is based on previous work shown in [27]. The model parameters are identified from CFD data as shown in [28]. The Simulink model has UDP interfaces in order to receive control signals, and send simulated inertial measurement unit, position, velocity and airspeed data to an autopilot. Noise can be added to this simulated sensor data.
2) As autopilot we use Ardupilot version 3.9. This autopilot software has an inbuilt software in the loop functionality, which is used in this test. The autopilot uses an Extended Kalman Filter to estimate position, velocity and attitude from the input data and uses these estimates to keep the simulated UAV on a desired path.
3) Using the Mavlink protocol via a TCP link [29] the autopilot sends the estimated vehicle state to DUNE which is running on an Odroid X4U single board computer. TCP is chosen since it ensures a reliable delivery of the Mavlink messages. We use BalenaOS [30] as a base operating system on the Odroid and then run DUNE inside a Debian Linux based Docker container. This has the benefit that all needed libraries and packages as well as DUNE itself can be easily cross-compiled on another PC and then deployed to the Odroid.
4) Neptus is a ground control station, which in this case is used as a to send flight plans to DUNE, record the estimated quantities and synchronize them with their reference values. These are sent from the Simulink model via IMC. Neptus also provides a graphical user interface for mission control.

III. Results

A. Simulation Scenario

The flight path, shown in Figure 3, consists of straight line segments and loiters as well as airspeed and altitude changes. This is to assess the performance of the estimator in various conditions. This flight plan was flown two times for a total flying time of $1116s$. Afterwards continuous loitering for $700s$ was performed to assess parameter drift in those conditions. The wind is set to $3m/s$ from north direction and Dryden wind gusts with gust amplitudes of $3m/s$. Initial values for $\boldsymbol{x}$, $\boldsymbol{p}$ and $\boldsymbol{P}$ as well as parameters used in the estimator can be found in Table I. The tuning of the estimator is done as described in [17].

TABLE I: MHE Parameters

Parameter	Value
L	6
ΔT_{min}	$0.1s$
Max. Solver Iterations	20
$\boldsymbol{x}_0$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
$\boldsymbol{p}_0$	$\begin{bmatrix}0 & 0 & 0 & 0 & 0.4 & 1\end{bmatrix}^T$
$\boldsymbol{P}_0$	$\mathrm{diag}\left(\begin{bmatrix}10^{-6} & 10^{-6} & 10^{-6} & 10^{-2} & \cdots \\ 10^{-2} & 10^{-6} & 10^{-5} & 10^{-5} & 10^{-5}\end{bmatrix}\right)$
$\boldsymbol{Q}$	$\mathrm{diag}\left(\begin{bmatrix}10^{-1} & 10^{-1} & 1 & 10^{-4} & \cdots \\ 10^{-4} & 10^{-6} & 10^{-15} & 10^{-30} & 10^{-15}\end{bmatrix}\right)$
$\boldsymbol{R}$	$\mathrm{diag}\left(\begin{bmatrix}1 & 10^{-4} & 1 & 1\end{bmatrix}\right)$
$\boldsymbol{W}_m$	$\mathrm{diag}\left(\begin{bmatrix}10^{-2} & 10^{-2} & 10^{-2} & 1 & 10^{-2}\end{bmatrix}\right)$

B. Latency

Figure 4 shows a statistic of the time elapsed between the reception of the input data and the output of the estimates. The maximal latency is $0.515s$ and the mean latency is $0.21s$. If we use the Dryden wind turbulence model [20] to calculate the cut off frequency f_{lim} at which $-3dB$ attenuation occurs for an assumed altitude of $100m$ and airspeed of $20m/s$, we get $f_{lim} = 0.0761Hz$. This frequency is considerably lower than the inverse of the maximal latency (which is about $2Hz$), and we can therefore assume that the latency does not influence the quality of the estimates. This shows that the presented algorithm is able to produce estimates in real-time.

C. Angle of Attack and Airspeed Estimates

Figure 5 shows the estimated AoA and its reference. It shows that the estimate convergences quickly to the reference with small errors overall. Equally, so does the airspeed estimate shown in Figure 6. Root mean square errors (RMSE) and maximum errors can be found in Table II. Note that data from the first $100s$ has been excluded from these values, to keep larger errors which only occur in the convergence phase out of these statistics.

For completeness, estimation errors for the sideslip angle are also included in Table II. The errors of the sideslip angle estimates are significantly larger then for the AoA estimates, which as already discussed in [17] is due to the lack of information in the lateral axis.

D. Wind Velocity Estimates

The wind velocity estimates in the north and east directions shown in Figures 7 and 8 show mainly low estimation errors with some larger errors occurring occasionally. As explained in more detail in [17], this is due to a lack

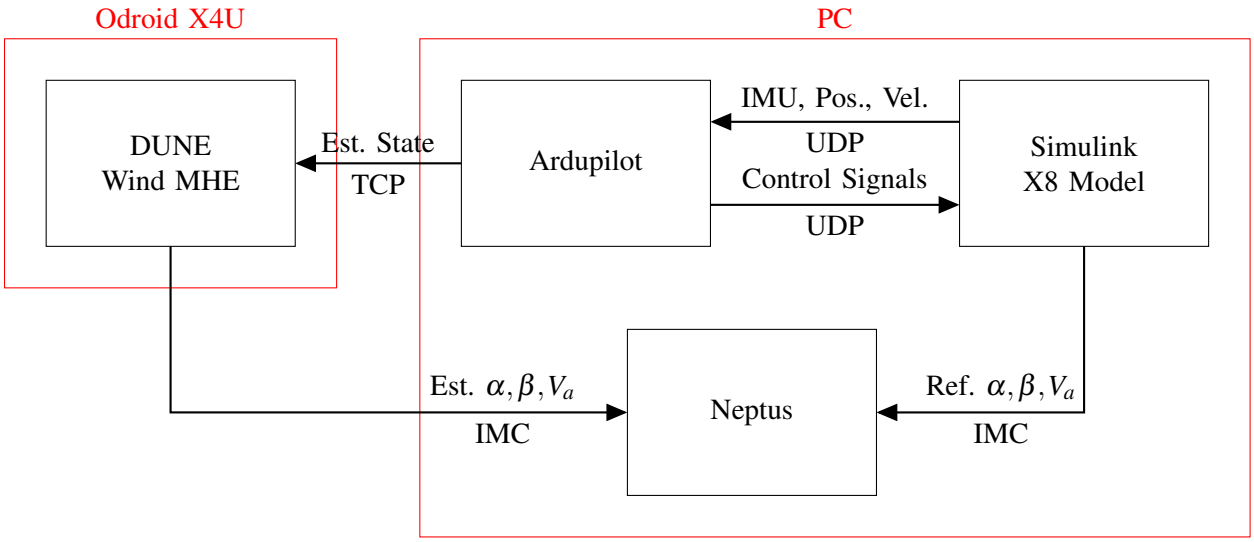

Fig. 2: Hardware in the Loop test setup

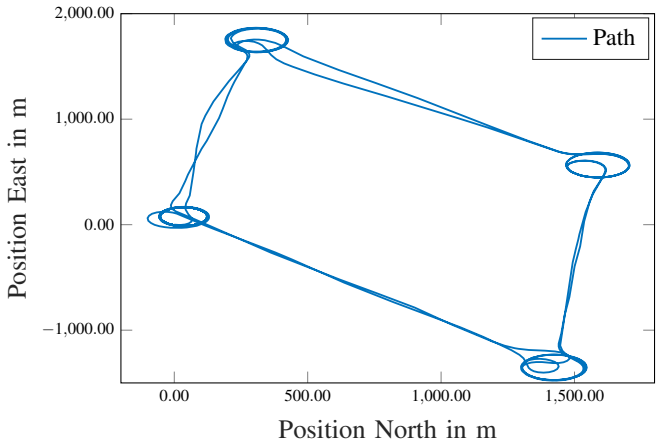

Fig. 3: Flight Path

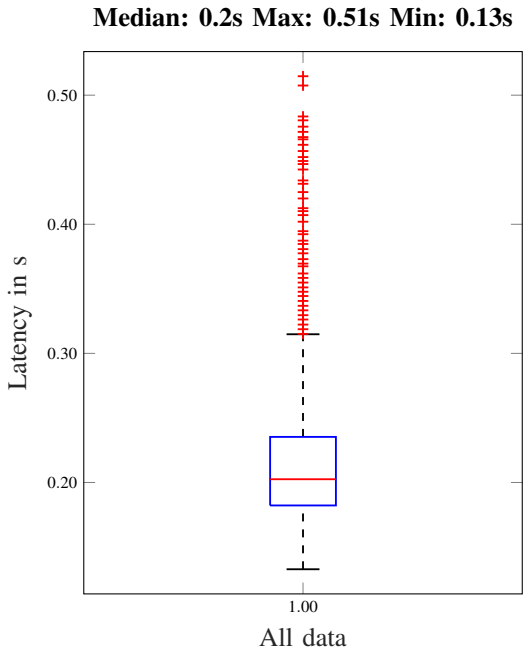

Fig. 4: Latency between data received and estimation output

of a model of the lateral dynamics and also no lateral relative velocity measurements. Since the chosen scenario includes quite intensive turbulence, these measurements are only representative for a short time and the error grows with increasing time between turns. The estimation error in down direction is quite small since the z-accelerometer measurement points in that direction most of the time. Some errors do occur when the aircraft is banking in turns, which is again due to the lack of information in the lateral axis. RMSE and maximum errors can be found in Table II.

E. Coefficient Estimates

In Figure 10 the estimate of the constant lift coefficient and its reference are shown. The estimate is slightly varying around the reference value over time which is mainly caused by the unmodeled lift force created by elevator deflections. Note that there is an offset during loitering where a constant elevator deflection occurs. The linear lift coefficient estimate shown in Figure 11 converges quite slowly towards its reference value. As discussed in [17] this is mainly due to

TABLE II: Estimation Errors excluding Convergence Phase

Variable	RMSE	Max Error
RMSE $v_{w,n}$	$0.44 m/s$	$3.24 m/s$
RMSE $v_{w,e}$	$0.44 m/s$	$2.26 m/s$
RMSE $v_{w,d}$	$0.2 m/s$	$0.65 m/s$
RMSE V_a	$0.21 m/s$	$1.54 m/s$
RMSE α	$0.29°$	$2.17°$
RMSE β	$3.2°$	$8.99°$
$V_{w,G}$	$3 m/s$	$3 m/s$

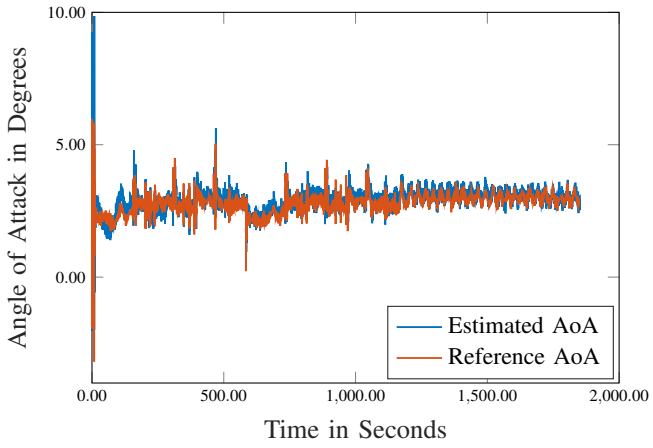

Fig. 5: Angle of Attack estimation

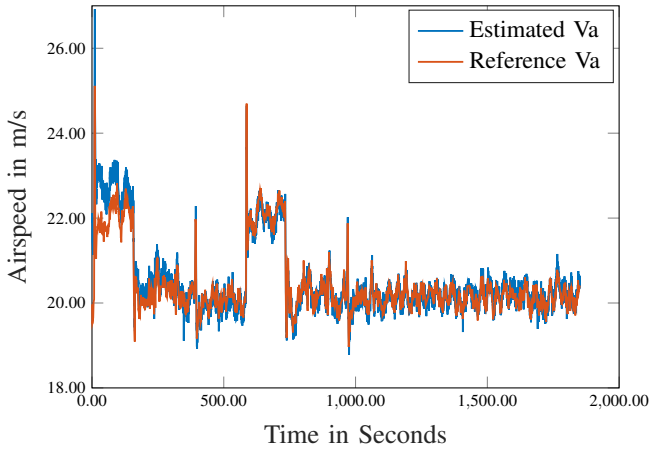

Fig. 6: Aispeed estimation

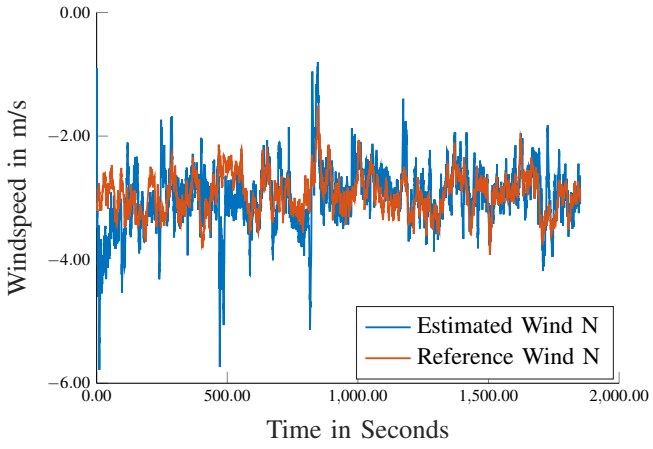

Fig. 7: Wind estimation north direction

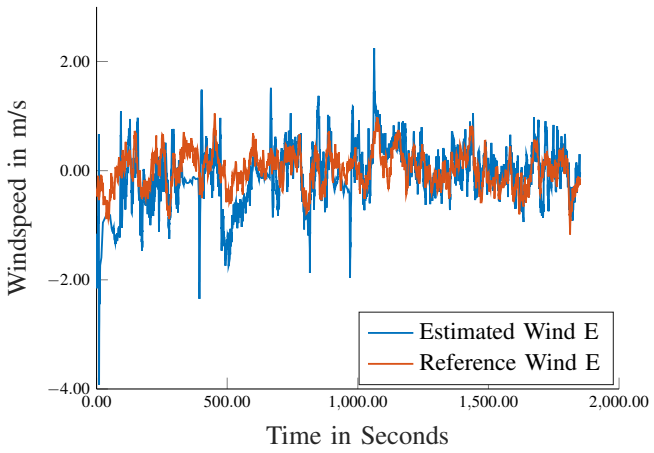

Fig. 8: Wind estimation east direction

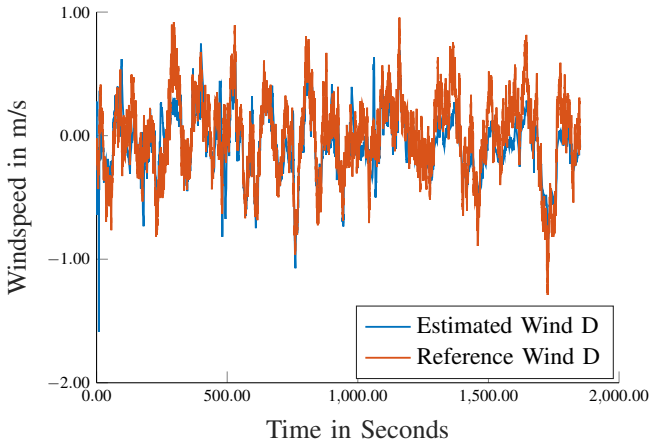

Fig. 9: Wind estimation down direction

the high process noise in the lift force model. Note that the convergence speed is faster in continuous loitering, due to the frequent angle of attack changes as shown in Figure 5. The estimate of the pitot tube scaling factor is shown in Figure 12. It shows that the scaling factor quickly converges to a neighborhood of 1. During longer loitering segments some drift occurs which is due to a non-zero sideslip angle over a longer time period which violates the assumption in equation (13) and can causes small biases.

IV. Conclusions

We have presented a real-time implementation of a Moving Horizon Estimator which is able to estimate angle of attack, airspeed and wind velocities with a high level of accuracy using standard autopilot sensors. Furthermore, estimates of lift coefficients and the pitot-tube scaling factors are available. The latency induced by the estimator is small compared to the occurring frequency components of typical wind spectra encountered by small UAVs.

V. Acknowledgments

The authors would like to thank Adrian Winter and Kristoffer Gryte for helping to build the simulation environment. This research was funded by the Research Council of Norway (RCN) through the Centres of Excellence funding scheme, grant number 223254 – NTNU AMOS, by RCN, Maritime Robotics and Equator Aircraft through the FlightSmart

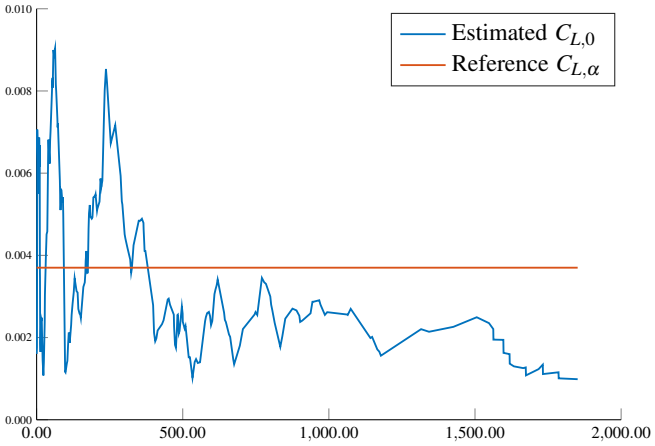

Fig. 10: Constant lift coefficient estimate

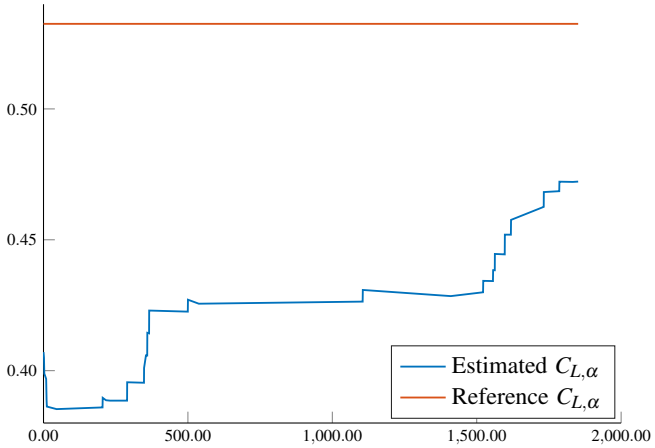

Fig. 11: Linear lift coefficient estimate

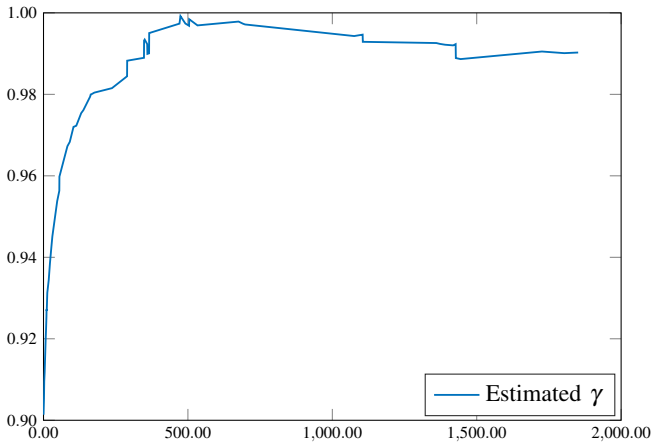

Fig. 12: Pitot tube scaling factor estimate

Project grant 282004, and a innovation scholarship provided by the Norwegian University of Science and Technology.

REFERENCES

[1] C. M. Belcastro, R. L. Newman, J. K. Evans, D. H. Klyde, L. C. Barr, and E. Ancel, "Hazards identification and analysis for unmanned aircraft system operations," *17th AIAA Aviat. Technol. Integr. Oper. Conf. 2017*, no. June, 2017.

[2] S. Benders, A. Wenz, and T. A. Johansen, "Adaptive path planning for unmanned aircrafts using inflight wind estimation," in *Int. Conf. Unmanned Aircr. Syst.* Dallas, TX: IEEE, 2018.

[3] A. R. Hovenburg, F. Augusto, D. A. Andrade, C. D. Rodin, T. A. Johansen, and R. Storvold, "Inclusion of Horizontal Windmaps in Path Planning Optimization of small UAS," in *Int. Conf. Unmanned Aircr. Syst.*, Dallas, 2018.

[4] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

[5] J. W. Langelaan, N. Alley, and J. Neidhoefer, "Wind Field Estimation for Small Unmanned Aerial Vehicles," *J. Guid. Control Dyn.*, vol. 34, pp. 1016–1030, 2011. [Online]. Available: http://www.aero.psu.edu/jack/pubs/LanAll10.pdf

[6] K. T. Borup, T. I. Fossen, and T. A. Johansen, "A Nonlinear Model-Based Wind Velocity Observer for Unmanned Aerial Vehicles," in *IFAC-PapersOnLine*, vol. 49, Monterrey, CA, USA, 2016, pp. 276–283.

[7] M. Shaqura and C. Claudel, "A hybrid system approach to airspeed, angle of attack and sideslip estimation in Unmanned Aerial Vehicles," in *Int. Conf. Unmanned Aircr. Syst.*, 2015, pp. 723–732.

[8] E. A. Morelli, "Real-Time Aerodynamic Parameter Estimation Without Air Flow Angle Measurements," pp. 1064–1074, 2012. [Online]. Available: http://arc.aiaa.org/doi/pdf/10.2514/1.C031568

[9] J. Zeis, jr., H. Lambert, R. Calcio, and D. Gleason, "Angle of attack estimation using an inertial reference platform," in *15th Atmos. Flight Mech. Conf.* Reston, Virigina: American Institute of Aeronautics and Astronautics, aug 1988. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.1988-4351

[10] T. A. Johansen, A. Cristofaro, K. Sørensen, J. M. Hansen, and T. I. Fossen, "On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors," in *Int. Conf. Unmanned Aircr. Syst.*, Denver, 2015.

[11] M. B. Rhudy, Y. Gu, J. Gross, and H. Chao, "Onboard Wind Velocity Estimation Comparison for Unmanned Aircraft Systems," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 1, pp. 55–66, 2017.

[12] A. Cho, J. Kim, S. Lee, and C. Kee, "Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, no. 1, pp. 109–117, 2011.

[13] M. Kumon, I. Mizumoto, Z. Iwai, and M. Nagata, "Wind Estimation by Unmanned Air Vehicle with Delta Wing," in *Proc. - IEEE Int. Conf. Robot. Autom.* IEEE, 2005, pp. 1896–1901. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=1570390

[14] H. Long and S. Song, "Method of Estimating Angle-of-Attack and Sideslip Angel Based on Data Fusion," in *2009 Second Int. Conf. Intell. Comput. Technol. Autom.*, vol. 1. IEEE, 2009, pp. 641–644. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5287571

[15] M. B. Rhudy, T. Larrabee, H. Chao, Y. Gu, and M. Napolitano, "UAV Attitude, Heading, and Wind Estimation Using GPS/INS and an Air Data System," in *AIAA Guid. Navig. Control Conf.*, Boston, 2013, pp. 1–11. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/6.2013-5201

[16] J.-P. Condomines, M. Bronz, and G. Hattenberger, "Experimental Wind Field Estimation and Aircraft Identification," in *IMAV 2015 Int. Micro Air Veh. Conf. Flight Compet.*, 2015.

[17] A. Wenz and T. A. Johansen, "Moving Horizon Estimation of Air Data Parameters for UAVs," *IEEE Trans. Aerosp. Electron. Syst.*, 2019.

[18] A. Wenz and T. A. Johansen, "Icing detection for small fixed wing UAVs using inflight aerodynamic coefficient estimation," *IEEE Aerosp. Conf. Proc.*, 2019.

[19] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The LSTS toolchain for networked vehicle systems," *Ocean. 2013 MTS/IEEE Bergen Challenges North. Dimens.*, 2013.

[20] "MIL-STD-1797A: Flying Qualities of Piloted Aircraft," U.S. Department of Defence, Tech. Rep., 2004.

[21] A. Wenz and T. A. Johansen, "Estimation of Wind Velocities and Aerodynamic Coefficients for UAVs using standard Autopilot Sensors and a Moving Horizon Estimator," in *Int. Conf. Unmanned Aircr. Syst.*, 2017, pp. 1267–1276.

[22] A. Wenz, T. A. Johansen, and A. Cristofaro, "Combining model-free and model-based angle of attack estimation for small fixed-wing UAVs using a standard sensor suite," in *Int. Conf. Unmanned Aircr. Syst.*, Arlington, VA, 2016, pp. 624–632.

[23] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, and H. G. Bock, "A real-time algorithm for moving horizon state and parameter estimation," *Comput. Chem. Eng.*, vol. 35, no. 1, pp. 71–83, 2011.

[24] J. Andersson, "A General Purpose Software Framework for Dynamic Optimization," PhD Thesis, KU Leuven, 2013.
[25] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
[26] Research Councils UK, "HSL. A collection of Fortran codes for large scale scientific computation." [Online]. Available: http://www.hsl.rl.ac.uk/
[27] K. Gryte, R. Hann, M. Alam, J. Rohac, T. A. Johansen, and T. I. Fossen, "Aerodynamic modeling of the Skywalker X8 Fixed-Wing Unmanned Aerial Vehicle," *2018 Int. Conf. Unmanned Aircr. Syst. ICUAS 2018*, pp. 826–835, 2018.
[28] A. Winter, R. Hann, A. Wenz, and T. A. Johansen, "Stability of a Flying Wing UAV in Icing Conditions," in *8th Eur. Conf. Aeronaut. Sp. Sci.*, 2019.
[29] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," *IEEE Access*, vol. 7, no. 8, pp. 87 658–87 680, 2019.
[30] "Balena." [Online]. Available: www.balena.io