Flight Testing Small UAVs for Aerodynamic Parameter Estimation

Adam T. Chase* and Robert A. McDonald †

California Polytechnic State University, San Luis Obispo, California, 93407

A flight data acquisition system was developed to aid unmanned vehicle designers in verifying the system's design performance. The system is reconfigurable and allows the designer to choose the correct combination of complexity, risk, and cost for a given flight test, as well as allowing the designer to reconfigure the system to meet packaging requirements. System functionality was validated by collecting data during flight of an radio-controlled aircraft, and future work includes further system validation, quantifying system accuracy, sensor fusion, and stability and control derivative estimation.

Nomenclature

σ	Standard deviation
AR	Aspect ratio
C_D	Drag coefficient
C_L	Lift coefficient
C_{D_0}	Parasite drag coefficient
F_A	Aerodynamic forces
F_G	Gravitational forces
R_n^m	Rotation matrix from m to n
CFD	Computational Fluid Dynamics
DOF	Degree of freedom
e	Wing lift distribution
FSO	Full scale output
G	Gravity
I^2C	Inter-integrated circuit
I/O	Input-output
IC	Integrated circuit
INS	Inertial navigation system
LCC	Leadless chip carrier
LGA	Land grid array
LiPo	Lithium Polymer
OLS	Ordinary least squares
PWM	Pulse width modulation
QFN	Quad-flats no-leads
R/C	Radio-controlled
RMS	root-mean-squared
RPM	Rotations per minute
SPI	Serial peripheral interface
UAS	Unmanned aerial system
UAV	Unmanned aerial vehicle

I. Introduction

Aircraft designers often use model fits and "rules of thumb" to successfully complete designs, many of which can be found in various design textbooks.[1,2,3] These models and practices have been established based on years of data analysis and validation that the designs perform as expected. However, the scope of these empirical design techniques is limited to the input of the regression model: an intelligent designer will not use a general aviation weight model for a transport category aircraft. To this end, there is a significant lack of small UAV-class guidelines for designing an aircraft. Accurate estimations of a small-scale vehicle's lift and drag characteristics are extremely critical to the aircraft designer, and affect both point performance (turn rates, climb rates, stall speeds, etc) and mission performance (range and endurance.) Much of the prediction tools available in the classic lift and drag textbooks from Hoerner[4,5] only apply to larger scale structures. In addition, drag prediction is extremely difficult on small vehicles, as some sources of drag are not easily modeled. For instance, actuator control horns and protruding screw heads are not typically

*Graduate Student, Aerospace Engineering, One Grand Avenue, achase90@gmail.com, Student Member AIAA

†Associate Professor, Aerospace Engineering, One Grand Avenue, Senior Member AIAA

included in CFD analysis of aircraft. However, for small vehicles, these sources of "crud" drag can be a significant portion of the total vehicle's drag value. One advantage of small UAVs is that they are fairly inexpensive, which makes building multiple fully-functioning prototypes a viable option. The authors chose to take advantage of this fact and develop a flight data acquisition system with a primary goal of measuring the lift and drag characteristics of a small UAV. This would enable vehicle designers to build a design-level prototype, and both develop and validate predictive, regression-based models. The system would also allow designers to make quantitative trade studies, such as the trade between drag reduction technologies (wheel pants, retractable landing gear, winglets, etc.) and the weight associated with them.

The authors also wanted additional sensors to aid designers with more than just lift and drag characteristics. Some areas of interest were estimating stability and control derivatives, as well as possible control algorithm testing, in-flight thrust measurement, and payload integration capabilities. To accomplish this, sensors not directly necessary to lift and drag estimation were included in the overall system. The system was also designed in a manner that made it reconfigurable. This allows future designers to decide what combination of accuracy, risk, and sensing capabilities they need on a given flight test, and to then package the sensors in a manner that meets vehicle integration requirements.

For this paper, the authors chose to integrate the necessary sensors to estimate lift and drag forces, as well as those necessary for a basic INS , ambient temperature measurement, and servo and motor signals. The system will be validated by measuring the as-built drag polar of an R/C aircraft, as well as all other available states the system is capable of measuring.

II. Method

Some basic assumptions will apply throughout the modeling of dynamics in this paper. They are as follows:

1. The vehicle is a fixed mass.
2. Coriolis effects are negligible.
3. Thrust will be assumed to be 0.

Note that a stationary atmosphere is not assumed. The fixed mass assumption is consistent with the electric aircraft used to test the system. At the altitude and speed at which the vehicles will be tested, Coriolis effects can be ignored.[6] The zero-thrust assumption is in place to minimize drag error. Any error in a measured state will decrease the accuracy of the drag measurement, and the accuracy of a given drag measurement can be no better than the worst state measurement error. In-flight thrust is difficult to measure accurately, so a folding propeller will be used, and the motor will be turned off during data acquisition. This will allow the propeller to fold back, eliminating most of the wind-mill drag associated with a stalled propeller.

Reference Frames

The reference frames used in this paper will follow standard convention,[6] but will be repeated for clarity. The NED frame is a vehicle carried frame, with the $+\hat{i}$ axis pointing due North, the $+\hat{j}$ axis pointing due East, and the $+\hat{k}$ axis pointing toward the center of the earth. The body axis system is defined by the body-mounted accelerometer, is a right handed coordinate system, with the $+\hat{i}$ axis roughly pointing out the nose of the aircraft, the $+\hat{j}$ axis roughly pointing out the right wing, and the $+\hat{k}$ axis roughly pointing out the bottom of the vehicle. The wind axes are a vehicle-carried coordinate system, with the origin at the same location as the Body reference frame. The $\hat{i}$ axis of the wind reference frame points in the direction of the free-stream velocity. The $\hat{k}$ direction lies in the x-z plane of the body reference frame. The $\hat{j}$ direction is then defined to be out the right side of the vehicle, in order to follow the right hand rule.

Equations of Motion

Newton's 2nd Law of Motion states

$$\vec{F} = \frac{d}{dt}(m\vec{V}) \tag{1}$$

where $\vec{F}$ is the sum of all applied forces, $\vec{m}$ is the mass of the vehicle, and $\vec{V}$ is the vehicle's velocity. Using the fixed mass assumption, this reduces to

$$\vec{F} = m\frac{d\vec{v}}{dt} \tag{2}$$

$$= m\vec{a} \tag{3}$$

The applied forces on the vehicle for drag polar estimation are

$$\vec{F} = \vec{F_A} + \vec{F_G} + \vec{F_T} \tag{4}$$

where $\vec{F_A}$ accounts for all aerodynamic forces acting on the vehicle, $\vec{F_G}$ is the force due to gravity, and $\vec{F_T}$ accounts for forces from the propulsion system, which are assumed to be zero.
Aerodynamic forces can be described in the wind reference frame. In general, they are defined as

$$\vec{F}_{A_w} = D\hat{i}_w + Y\hat{j}_w + L\hat{k}_w \tag{5}$$

where D is drag force, Y is side force, and L is lift force.
The gravitational force on the vehicle acts in the $+z_{ned}$ direction and is equal in magnitude to the vehicle's weight W, leading to

$$\vec{F}_{G_{ned}} = 0\hat{i}_{ned} + 0\hat{j}_{ned} + W\hat{k}_{ned} \tag{6}$$

The aerodynamic and gravity forces can be combined and expressed in the body frame

$$m\vec{a} = \vec{F}_{G_b} + \vec{F}_{A_b} \tag{7}$$

which can then be expressed as

$$m\vec{a}_b - m\vec{g} = \vec{F}_{A_b} \tag{8}$$

$$m(\vec{a}_b - \vec{g}) = \vec{F}_{A_b} \tag{9}$$

Then, it is noted that a body mounted accelerometer will not measure $\vec{a}_b$, but will instead measure the quantity $\vec{a}_b - \vec{g}$. This means Equation 9 is really

$$\vec{F}_{A_b} = -m\vec{r}_b \tag{10}$$

where $\vec{r}_b$ is the reading from a body mounted accelerometer. The aerodynamic forces in wind axes, which is the goal, can then be calculated using a rotation matrix

$$\vec{F}_{A_w} = R_w^b \vec{F}_{A_b} \tag{11}$$

Equations 10 and 11 are important, and show that the only sensors necessary for drag polar estimation during gliding flight are a body-mounted accelerometer and an air data system.

Kalman Filter Usage

This paper utilizes multiple Kalman filters to estimate both regression coefficients and improved states. As a brief overview, the Kalman filter combines the measured state with a predicted state to give an optimal[7] estimate of the actual system state.

Linear Kalman Filter

A linear Kalman filter can be applied[8] where the system in question can be described in the form

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{12}$$

where A is the state transition matrix, x_{k-1} is the previous state, B is the input matrix, u_{k-1} is the input vector, and w_{k-1} is random process noise.

The measured state is then

$$z_k = Hx_k + v_k \tag{13}$$

where H is the output matrix and v_k is measurement noise.

The Kalman filter operates in a predictor-corrector manner, where the predictor step is often called the *a priori* estimate, and the corrector step is often called the *a posteriori* estimate. The *a priori* state estimate is calculated using prior states and inputs, while assuming no process noise

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{14}$$

The *a priori* estimate of the covariance matrix is projected in a similar manner

$$P_k^- = AP_{k-1}A^T + Q \tag{15}$$

where P is the covariance matrix and Q is the process noise matrix.

The Kalman gain is calculated by combining the predicted, *a priori* covariance matrix with the measurement noise covariance matrix R

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{16}$$

This optimal Kalman gain is then used to estimate the *a posteriori* estimate of the state and covariance matrix

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - y_k) \tag{17}$$

$$P_k = (I - K_k H)P_k^- \tag{18}$$

where y_k is the predicted value of z_k found using the output matrix H and the *a priori* state estimate

$$y_k = H\hat{x}_k^- \tag{19}$$

Note that Equation 17 is essentially a weighted average of a measured state and an expected state. The weighting is the Kalman gain, which is related to the ratio of confidence in the measured state and the expected state. For a 1-D case with equal confidence between the measured state and the expected state, the Kalman gain $K_k = 0.5$, and the Kalman Filter becomes a simple mean.

Extended Kalman Filter

The Extended Kalman filter is used for a non-linear system and is essentially a linearization of a nonlinear plant. A non-linear system can be described as[8]

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{20}$$

$$z_k = h(x_k, v_k) \tag{21}$$

The process noise w_{k-1} and measurement noise v_k are not known (or the Kalman filter would not be necessary), so the states are approximated assuming both noise sources are 0

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{22}$$
$$\tilde{z}_k = h(\tilde{x}_k, 0) \tag{23}$$

The actual states are related to the approximate states by

$$x_k \approx \tilde{x}_k + A(x_k - \hat{x}_{k-1}) + Ww_{k-1} \tag{24}$$
$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_{k-1}) + Vv_k \tag{25}$$

where the matrices A, W, H, and V represent the different Jacobians matrices:

$$A = \frac{\partial f_i}{\partial x_j}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{26}$$
$$W = \frac{\partial f_i}{\partial w_j}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{27}$$
$$H = \frac{\partial h_i}{\partial x_j}(\hat{x}_k, 0) \tag{28}$$
$$V = \frac{\partial h_i}{\partial v_j}(\hat{x}_k, 0) \tag{29}$$

The Extended Kalman filter uses these linearized equations to perform the same process as the linear Kalman filter. Again, the first step is to calculate the *a priori* estimate of the state and the covariance matrix

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{30}$$
$$P_k^- = A_k P_{k-1} A_{k-1}^T + W_k Q_{k-1} W_k^T \tag{31}$$

Next, the Kalman gain is calculated

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{32}$$

The Kalman gain is then used to calculate the *a posteriori* estimate of the state and covariance matrix

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - y_k) \tag{33}$$
$$P_k = (I - K_k H_k) P_k^- \tag{34}$$

where y_k is, as in the linear case, the predicted value of z_k, but calculated using the nonlinear output function and the *a priori* state estimate

$$y_k = h(\hat{x}_k^-, 0) \tag{35}$$

Unlike the linear Kalman filter, the Extended Kalman filter is not proven to be optimal. However, it has been utilized for a wide range of applications with excellent results.

III. Error Analysis

Without estimates of the error in data, the data itself is fairly meaningless. There are two main types of error in the system: model regression error and the error of a single data point. The error of a single data point comes from random noise in sensors, and can be decreased by improving the accuracy of the sensor or filtering the results. The model regression error comes from the fact that every aspect of the dynamics of the system are not precisely modeled. This type of error can be reduced by improving the accuracy of the dynamics being modeled, choosing a better regression model, or by increasing the number of data points collected, as discussed in Section III.

Random Error

The equations of motion can be used to propagate uncertainty in a signal, and they allow the uncertainty in the coefficients to be estimated based on sensor noise. The error can be either systematic or random. Systematic error can be caused by many things, including poorly modeled physics and a steady-state offset of a sensor. Random error is due to the inherent inaccuracy of the sensors, and is generally assumed to be normally distributed.

The error of a single point is assumed to be random and normally distributed. The first step in error propagation is to define the y_i to be the i-th entry of the true function vector, $\hat{y}_i$ to be the i-th entry of the measured function vector, and to then do a Taylor series expansion about the operating point.

$$\hat{y}_i = y_i + \frac{\partial y_i}{\partial x_j} dx_j \tag{36}$$

where x_j is the j-th element of the state vector. Note that the term $\frac{\partial y_i}{\partial x_j}$ is the Jacobian ($\bar{J_{ij}}$) matrix of the state transition function. The error can then be defined as

$$dy_i = \hat{y}_i - y_i = \bar{J_{ij}} dx_j \tag{37}$$

If the error is then interpreted as a discrete difference instead of a continuous difference, Equation 37 becomes

$$\Delta y_i = \bar{J_{ij}} \Delta x_j \tag{38}$$

If the Δ values are further assumed to represent standard deviations of normally distributed error, Equation 38 becomes

$$\sigma_{y_i} = \bar{J_{ij}} \sigma_{x_j} \tag{39}$$

For the purposes of this research, σ_{y_i} is a vector of the standard deviations of the aerodynamic force coefficients,

$$\sigma_{y_i} = \begin{bmatrix} \sigma_{C_{D_i}} & \sigma_{C_{Y_i}} & \sigma_{C_{L_i}} \end{bmatrix}^T \tag{40}$$

and σ_{x_j} is a vector of the standard deviations of the state values,

$$\sigma_{x_j} = \begin{bmatrix} \sigma_{r_{b_j}} & \sigma_{\alpha_j} & \sigma_{\beta_j} \end{bmatrix}^T \tag{41}$$

For initial error estimation, the noise levels reported by instrument manufacturers was assumed to be one standard deviation of a normal distribution with mean equal to zero. The Jacobian matrix was calculated at each observed data point and combined with the sensors' standard deviation to produce estimates of the standard deviations of the aerodynamic coefficients.

One of the key findings of this section is that the error of the coefficient depends on the coefficient itself. This means that the error varies from data point to data point, which is called *heteroskedasticity* (as opposed to homoskedasticity, which means the error is independent of the state itself). To account for this, an error estimate function was created in MATLAB, which used the error propogation outlined in this section. This function was used whenever an estimate of point error was required, such as for the variance matrix P_k used in the Kalman filters utilized for state estimation. A plot of simulated flight data is shown in Figure III, where the error bounds shown were calculated using the error estimate function. Note that this figure also shows the heteroskedastic nature of the error.

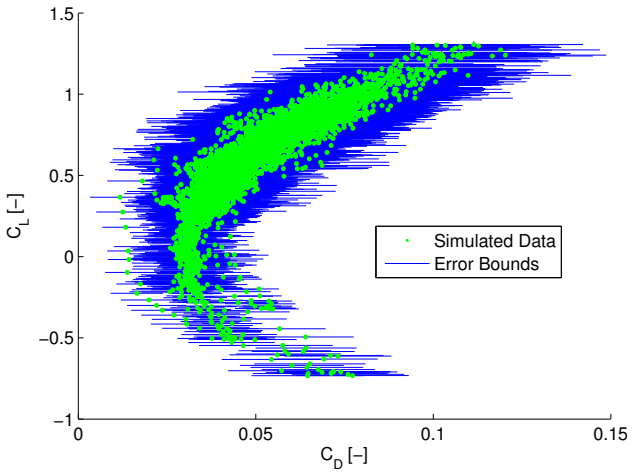


Figure 1. Heteroskedastic Error from Simulated Flight

While heteroskedasticity does not color the estimate of $\hat{\beta}_i$, it does color the confidence intervals. The method of dealing with this problem is discussed in Section III.

Least Squares Model Error

The error of a single data point is not the main driving factor in the accuracy of a regression model. The important factors in the model are how accurate the coefficients are known, which is a function of the accuracy of each point, as well as the number of points sampled. The main parameter that describes the accuracy of the regression coefficients are the confidence intervals. A confidence interval is a range of values such that, if the experiment were repeated, the parameter calculated would be within the range some percentage of the time. A parameter can be represented as an estimated value, with a confidence bound

$$\beta = \beta_{EST} \pm t\frac{\sigma}{\sqrt{n}} \tag{42}$$

where β is the parameter in question, β_{EST} is the estimated value of the parameter, t is the Student's t value based on the number of samples and the desired confidence interval, σ is the standard deviation of the sample, and n is the number of data points collected. Since the number of data points collected during flight will be large ($n > 100$), the t value will be taken as 1.96 for a 95% confidence interval.
As previously mentioned in Section III, one of the assumptions made in a Least Squares regression is homoskedasticity. However, the error using standard uncertainty propagation is heteroskedastic. This becomes a problem in estimating confidence intervals, because the standard error can be driven by outliers. If each data point had the same error, these outliers could be valid. However, if the data is heteroskedastic, the outlier may have a larger error bound, meaning the data point is not as likely as it first appears. This fact can drive the standard error estimate to be larger than is appropriate, which leads to a larger confidence interval and possibly a false lack of rejection of the confidence interval's hypothesis test. To account for this, the `robustfit` function in MATLAB was used to estimate both the coefficients and robust standard error estimates. The `robustfit` function calculates heteroskedastically-robust standard error estimates by doing a weighted average, where the weighting is based on a radial basis function. This means that the farther away a data point is from the estimated regression model, the less impact it has on the standard error of the model. The default weighting function used by `robustfit` is bisquare, and it was used for this research.

Kalman Filter Error

Kalman filters are often used to propagate states. The filter does this by combining the system dynamics with a measured state. The variance is propagated using Equation 34. When estimating coefficients using the Kalman filter, the $\bar{A}$ state transition matrix is an identity matrix, which is due to the fact that the coefficients stay constant. When propagated through the filter, this means the state estimate x_k is essentially a variance-weighted-average of the coefficient estimates. The matrix P_k contains the variance of the coefficient estimates.

Equation 42 calculates the confidence interval of regression coefficients and needs the standard deviation of the mean, also called the standard error. The matrix P_k can be used to calculate the confidence intervals by noting

$$P_k = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_i^2 \end{bmatrix} \tag{43}$$

The confidence interval for coefficient β_i can be calculated using σ_i in Equation 42.

IV. Simulation

A 6-DOF flight simulator was used to validate the drag prediction method before hardware was purchased. The main utility of the simulator was to provide simulated flight test data with signals that contained no errors. The actual sensors used for flight testing contain noise, and this noise can be added onto the pure simulator signals to test the sensitivity of the drag polar regression to sensor accuracy.

Simulation Environment

The flight simulator used was a model of the de Haviland Beaver that comes as a demo in the Aerospace Toolbox of Simulink. The Simulink model was modified to output required signals to the workspace, which essentially created a sensor with zero noise. The mass, moments of inertia, and reference lengths were then scaled to those of a Zagi R/C aircraft.[9] The original Simulink model was already connected to a FlightGear Flight Sim, used as a visualization engine. This model was slightly altered to make flight gauges function properly.

Simulation Inputs

The engine forces and moments were set to zero in the simulator, to match the assumption of a folding propeller. The drag force calculation built into the Beaver Simulink model was replaced with a parabolic drag polar of the form

$$C_D = C_{D_0} + K_1(C_L(\alpha) - C_{L_{min}})^2 + \frac{(C_L(\alpha))^2}{\pi e AR} \tag{44}$$

Airfoil data, including K_1, $C_{L_{min}}$, and $C_L(\alpha)$, was taken from nonlinear aerodynamic data of a NACA 4412. While this approximation to a drag polar does not capture the nonlinear section of profile drag rise due to stall, it does represent the limited lifting capability of a real wing, making it more realistic than assuming the wing does not stall.

Simulation Results

The first goal of the simulation testing was a to verify the drag polar equations were correct, and that the data analysis routines developed in MATLAB did in fact match inputs to outputs. The simulation was initialized with various initial states to ensure there was no dependency on initial conditions. The vehicle was then flown by an R/C aircraft pilot using a joystick attached to the simulation. It was noted early in the simulation testing that flying a sweep of speeds was beneficial, as a wider range of the drag polar was flown. This result was included in much of the flight test planning. After adequate data had been taken, the data was analyzed without adding simulated sensor noise. The results are shown in Figure IV.

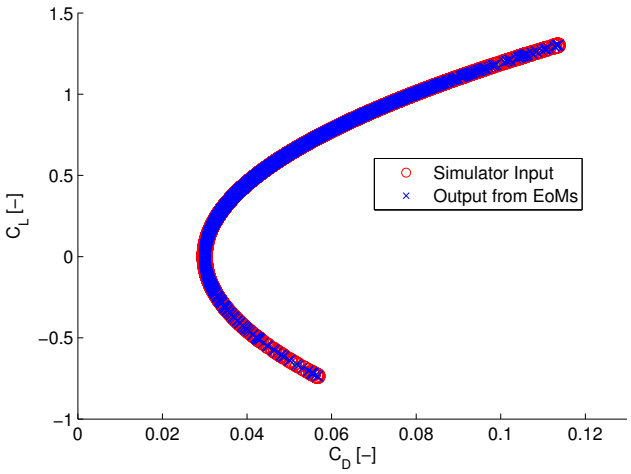


Figure 2. Data Analysis Verification (No Noise)

Figure IV shows that the equations of motion used in the data analysis functions properly calculate the coefficients being passed into the system. With this result, noise was added to the system to see how sensitive coefficient estimation was to noise in each sensor. This process was a balancing act between available sensor accuracy and the desired accuracy of the final solution. The final result guided sensor selection to those discussed in Section V. To check if the final sensors chosen were acceptable, Gaussian noise was added to each state, with a mean of zero and a standard deviation equal to the root-mean-squared error listed in the manufacturer's data sheet for each sensor.

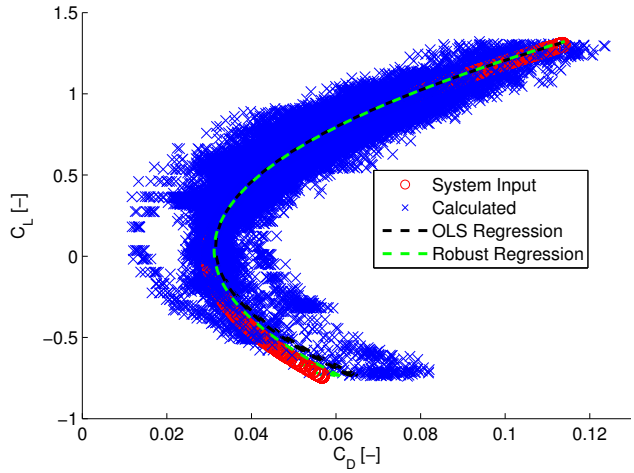


Figure 3. Drag Polar Prediction of Simulated Test Flight

For the particular simulated test flight shown in Figure 3, the estimated drag polar coefficients had error coefficients outlined in Table 1.

Table 1. Nonlinear Model Results

	C_{D_0}	K_1	K_2
System Inputs	0.0493	0	0.03
OLS Estimate	0.0355	-0.0136	0.02927
Robust LS Estimate	0.0460	-0.0037	0.0292

The results of this simulated flight test showed that the measurement system outlined in Section V

predicted the simulated drag polar with a reasonable error. It also demonstrates the necessity of the heteroskedasticity correction, as the OLS regression has a 38% error on K_2 and a 9% error on C_{D_0}, while the robust regression has a 7% error on K_2 and a 3% error on C_{D_0}.

V. Hardware

One of the main goals of this research was to give the designer flexibility in choosing the appropriate sensors for a given flight test. To this end, hardware that is available in breakout boards was given preference, since it gives the aircraft designer more flexibility. The aircraft designer could utilize surface mount components if vehicle integration space is extremely limited, or can use the available breakout boards to make circuit-level integration easier if vehicle space is not a driving flight test concern.

Flight Computer

The flight computer chosen was an Arduino Due. This board has a 32-bit ARM processor, 54 digital I/O pins, 12 analog input pins, and 2 analog output pins. The main driver in the decision to use an Arduino-based platform was the vast support community, which allowed quicker code development. The Arduino also offers a package that integrates well into most of the available airframes, and the stackable header pins allowed for easy integration with other boards. The Due in particular was chosen as it is (at the time of writing) the most advanced Arduino available. The main advantages it has over the comparable Arduino Mega is its increased clock speed (84 MHz for the Due[10] vs 16 MHz for the Arduino Mega[11]) and its 32-bit architecture (vs. 8-bit for the Arduino Mega).

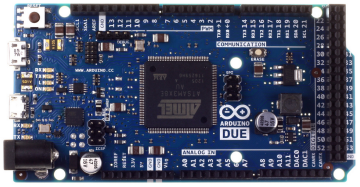


Figure 4. Arduino Due Flight Computer

The Arduino Due uses a 3.3V architecture instead of the usual Arduino architecture, which uses a 5V operating voltage. This was mainly beneficial, since most of the selected sensors used 3.3V as both supply and logic voltage. Logic level circuits, shown in Figure 5, were used to translate to 5V signals where required. The board is powered through the 3.5mm barrel jack, using a 3-cell LiPo battery, with a nominal voltage of 11.1V.

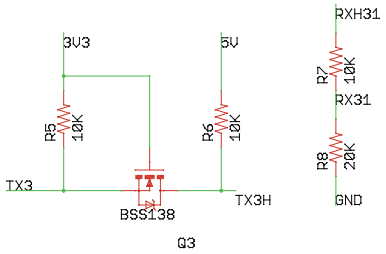


Figure 5. Logic Level Converter Circuit

Accelerometer

The accelerometer chosen for the data acquisiton system was the ADXL-362 from Analog Devices. It has a noise error of $175\mu G/\sqrt{Hz}$ and uses a 3.3V digital SPI interface.[12] The accelerometer is in a LGA package and was surface mounted to the main PCB, with a decoupling capacitor between power and ground. The circuit was modeled after Sparkfun's ADXL-362 Breakout Board.[13]

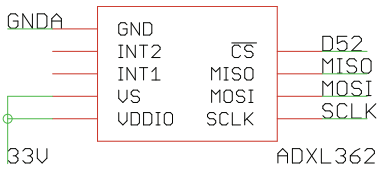


Figure 6. ADXL-362 Schematic

The accelerometer is calibrated in the field through an optimization routine using MATLAB's `fmincon` nonlinear constrained optimization function. For any given orientation, the accelerometer's reading can be expressed as

$$\begin{bmatrix} r_x & r_y & r_z & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \\ b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \tag{45}$$

where the r terms are the bit readings from the accelerometer for each axis, the m terms are the slope of a linear fit for each axis, the b terms are the zero offset of a linear fit for each axis, and the a terms are the actual accelerations.

The slope and offset terms can be found through field calibration and a nonlinear optimization routine. The algorithm uses the fact that, while in a static orientation, the magnitude of the measured vector should be exactly 1g. The minimization problem is then

$$\underset{x}{\text{minimize}}\ f(x) = \sqrt{(1G - |(\vec{a})|)^2} \tag{46}$$

where $\vec{a}$ is calculated according to Equation 45, and the variable of interest x is the vector of slopes and offsets in Equation 45. The slope terms in x are constrained to be positive. To be a deterministic system of equations, at least six static orientations are required. To avoid the noise of a single reading, the problem was expanded to take 100 data points in six different static orientations, and Equation 45 was expanded to become a least squares problem. The main benefit of this technique is that field calibration can be accomplished without needing precise knowledge of the orientation of gravity with respect to the sensor during the calibration routine. When tested, the results of the calibration were consistent between this algorithm and using known orientations. For error propagation, the noise during calibration was taken as the sensor's noise level.

Vehicle Mass

All test vehicles were weighed using a U-Line H-1650 counting scale. The scale has an accuracy of 0.001 lbs and a maximum capacity of 30 lbs. The minimum capacity of the scale is 10 grams.[14]

Magnetometers

Two separate magnetometers were used for separate purposes. A Honeywell HMR-2300 3-D magnetometer, shown in Figure 7, is the main magnetometer. It is used when extremely accurate heading information is needed, or when GPS course is unavailable, such as during extremely slow or vertical flight.

Figure 7. Honeywell HMR-2300 3-d Magnetometer

This magnetometer provides a RMS error of 0.1 milliGauss for all axes, using the 1 Gauss full-scale setting.[15] The HMR-2300 can be supplied with power between 6V and 15V, so the 3-cell 11.1V nominal LiPo battery that powers the Arduino also passes through to power the magnetometer. Additionally, the HMR-2300 operates using an RS-232 serial interface. To properly interface with the Arduino Due, which uses 3.3V TTL logic levels, a Max-3232 IC was used. This IC, when combined with charge pump capacitors, translates TTL levels between 3V and 5.5V to RS-232 logic levels of ±6V.

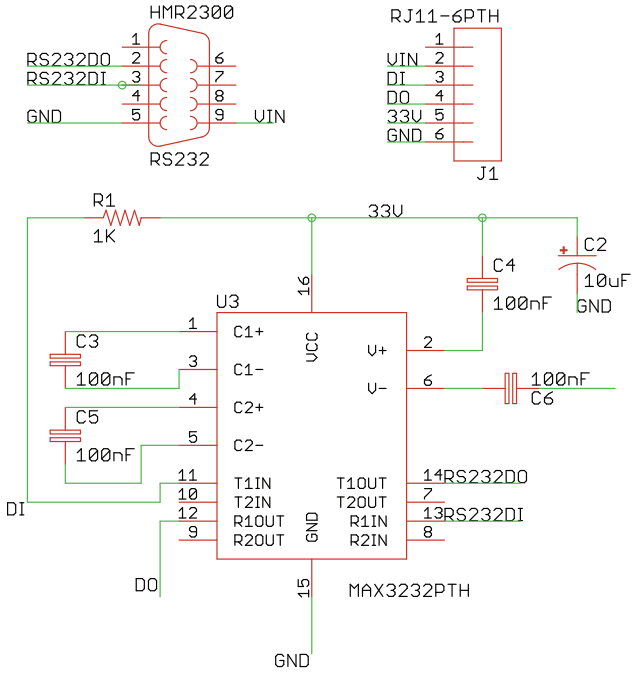


Figure 8. HMR-2300 Logic Level Circuit

The second magnetometer is a Honeywell HMC-5883L, which comes in an LCC package that was surface mounted to the main sensor board. It was added to the system for two main reasons: it is much smaller for applications where size is critical, and it is much less expensive for testing with unproven vehicles. It communicates with the Arduino using an I^2C interface and uses a 3.3V operating voltage.[16] The circuit used was similar to that used by Sparkfun on their HMC-5583L breakout board.[17] The HMC-5883L has an accuracy of 2 milliGauss on each axis.

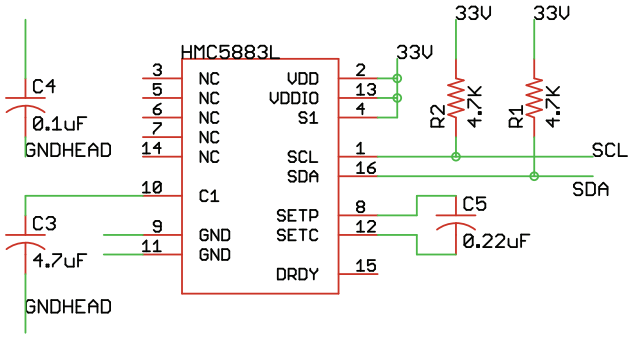


Figure 9. HMC-5883L Schematic

Both magnetometers were calibrated for both soft-iron and hard-iron effects.[18] To do this, data was acquired for 10 seconds with the magnetometer being swept through all directions. An ellipsoid was fit to the data using a ordinary least squares method available from the MATLAB file exchange.[19] The least squares fit estimates the center and radius of each axis. The center values for each axis was subtracted from the readings to remove hard-iron effects. Each i-th axis is then scaled by $\frac{1}{R_i}$ to reshape the ellipse into a circle, which removes soft-iron effects.

The surface-mounted HMC-5883L was assumed to be aligned with the surface-mounted accelerometer. Since the two magnetometers both measured the North vector, a rotation matrix that describes the difference in alignment between the two sensors can be calculated by

$$\vec{N}_{HMC5883} = \vec{R}_b^{M_1} \vec{N}_{HMR2300} \tag{47}$$

The rotation matrix $\vec{R}_b^{M_1}$ can then be used to align the HMR-2300's coordinate system with the body mounted accelerometers, using

$$\vec{N}_b = \vec{R}_b^{M_1} \vec{N}_{HMR2300} \tag{48}$$

Gyroscope

A three-axis gyroscope was also included in the system. The gyroscope chosen was the Invensense ITG-3200, which comes in a QFN package. This gyroscope has a total error of 0.38°/s-rms,[20] and uses a digital I^2C interface on a 3.3V operating voltage. The gyroscope has a full-scale span of ±2000°/s. The gyroscope was integrated into the main sensor board using a circuit based on that of Sparkfun's ITG-3200 breakout board.[21]

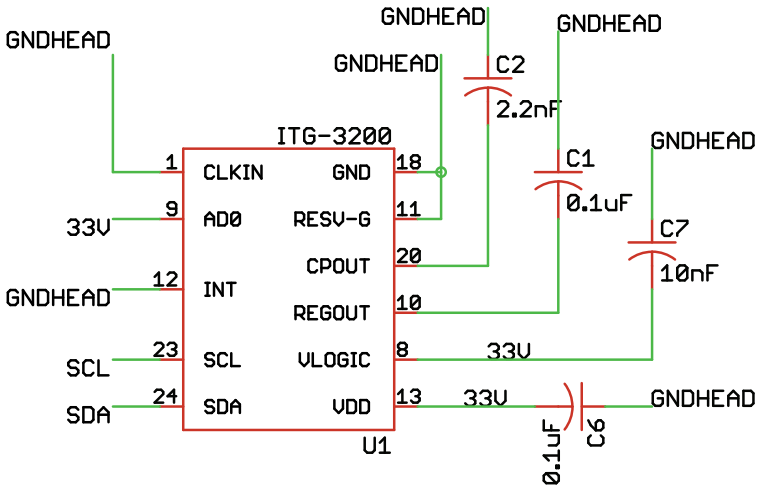


Figure 10. ITG-3200 Eagle Schematic

The gyroscope was calibrated in the same manner as the accelerometer. The device was placed in six orientations on a turn table which rotated at a constant 33 $\frac{1}{3}$ RPM. Slope and offset values for each axis were calculated using `fmincon`. Before each flight test, the offset values were re-calculated by taking 10 seconds of static readings.

Air Data System

A five-hole probe was chosen to measure aerodynamic angles as they do not contain moving parts and can provide very accurate, repeatable data. The five-hole probe selected was the Aeroprobe Air Data probe. It is 6 inches long, has a diameter of 1/8 inch, and uses a 0.25" hexagonal section as it's mounting section. The probe comes factory calibrated from angles to pressure readings, and was calibrated at and airspeed of 70 ft/s.
The probe was extended roughly one chord length in front of the leading edge of the wing by a 0.25" carbon fiber tube, which connected to the probe using set screws.

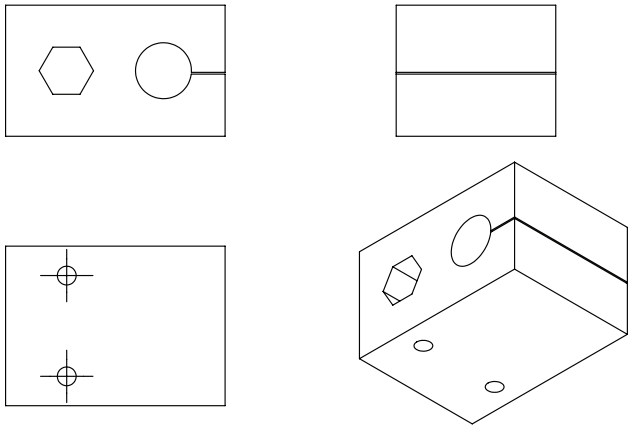


Figure 11. Five-Hole Probe Adapter

The adapter was manufactured to be square to itself and to have the reference flat of the probe's hexagonal section be parallel to one side of the adapter. An accelerometer was then glued to a side of the adapter, and the accelerometer was calibrated to the block using a level surface and the flat sides of the adapter. Before each flight test, three static readings are taken of the adapter's accelerometer and the main body-mounted accelerometer. Since the probe's orientation with respect to the adapter's accelerometer is known, this allows the wind angles measured by the probe to be calculated with respect to the body axes, and gives an accurate alignment of the wind reference frame to the body reference frame.

Each pair of lines of the air data probe is connected to an All Sensors digital differential pressure sensor with a full scale range of ± 5 in-H_2O.[22] The static port of the five-hole probe was connected to an All Sensors BARO-DO digital barometric pressure sensor,which has a range of 600 to 1100 mBar.[23] The barometric pressure sensor comes in the same package and uses the same communication protocol as the differential pressure sensors.

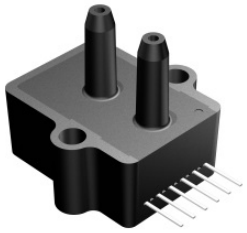


Figure 12. All Sensors 5-INCH-D-DO Pressure Sensor

The differential pressure sensors have a total error band of 0.25% FSO, and the barometric pressure sensor has a nominal error of 1 mBar. They use a UART serial interface that operates on a 5V logic level, so the logic levels were converted to the 3.3V levels of the Arduino Due. The serial interface includes addressable read commands, which allows multiples devices on a single bus, and ensures all devices record pressure at the same time. The sensor can output both a 14-bit pressure reading and a 12-bit temperature reading, which the device uses to correct its pressure measurement.

A digital temperature sensor was combined with the barometric pressure sensor to estimate the air density, which allowed air speed to be calculated.

Figure 13. Dallas Semiconductors' DS18B20 Digital Temperature Sensors

The DS18B20 from Dallas Semiconductors was chosen for its relatively simple One-Wire interface. The device can be powered with the communication line and has a $\pm 0.5°$C nominal accuracy.[24]

The differential pressure sensors were calibrated for slope using a . The zero offset of each differential pressure sensor was removed before each flight by taking 100 data samples in a no-wind condition. The barometric pressure transducer and the temperature sensor were calibrated for offset using a Paroscientific Model 745 Pressure Standard, capable of 0.008% FSO accuracy.[25]

Wind Angle Kalman Filter

To improve the accuracy of the wind angle estimation, a discrete Extended Kalman filter was used. The state transition functions are are

$$\dot{\alpha} = \frac{1}{V\cos\beta}(-a_x \sin\alpha + a_z \cos\alpha) + q - (p\cos\alpha + r\sin\alpha)\tan\beta \tag{49}$$

$$\dot{\beta} = \frac{1}{V}(-a_x \cos\alpha \sin\beta + a_y \cos\beta - a_z \sin\alpha \sin\beta) + p\sin\alpha - r\cos\alpha \tag{50}$$

These state transition equations come from solving for the vehicle forces in wind axes instead of body axes.[6] The equations for the Kalman filter for the wind angles are

$$\begin{bmatrix}\alpha_k \\ \beta_k\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha_{k-1} \\ \beta_{k-1}\end{bmatrix} + \begin{bmatrix}\Delta T & 0 \\ 0 & \Delta T\end{bmatrix}\begin{bmatrix}\dot{\alpha_k} \\ \dot{\beta_k}\end{bmatrix} + \hat{w}_{k-1} \tag{51}$$

$$z_k = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha_k \\ \beta_k\end{bmatrix} + \hat{v}_{k-1} \tag{52}$$

The process covariance matrix was calculated using the error propagation discussed in Section III and the standard deviation data from the zero offset calibration of each of the applicable sensors. The measurement noise covariance matrix was calculated based on the standard deviation data for the zero offset of each pressure transducer connected to the five-hole probe.

GPS Receiver

A uBlox LEA-6T GPS receiver was included in the data acquisition system. This model was selected for its ability to output raw timing data, which can be used to get an extremely accurate inertial velocity estimate.[26] The receiver itself was integrated onto a breakout board sold by CSG Shop and has UART, USB, and I^2C interface options.

Figure 14. CGS Shop Board for uBlox LEA-6T

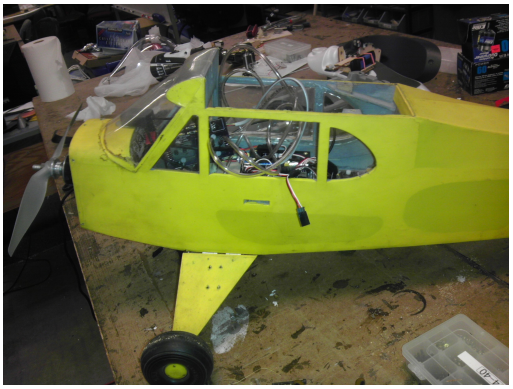
Data Acquisition System Integration

The sensors were packaged into a main shield for the Arduino. This shield plugs directly into the Arduino, eliminating the need to disconnect and reconnect wiring. Header pins capable of reading commanded PWM signals to servos were also added on the main board. Future work could include stability derivative estimation, and the header pins provide PWM measurement, which can map to servo angles, if it is assumed the servo is not stalled. The servo signal breakout pins also allowed the data to be easily split into sections with and without thrust for drag polar estimation.

A second board was developed to integrate the air data system with the main sensor board. This pressure board can be located near a wing tip and provides expandability should additional sensors be desired in the future. It also interfaces with the temperature sensor. Finally, all data was saved to a microSD card attached to the main sensor board. Data was saved in binary format for both increased speed and file size reductions. Once on the ground, the data is converted to meaningful values using a custom MATLAB data parser.

VI. Results and Future Work

The system was built and tested to prove functionality. After functionality testing was complete, the system was integrated into a 0.40-size Piper Cub R/C aircraft.



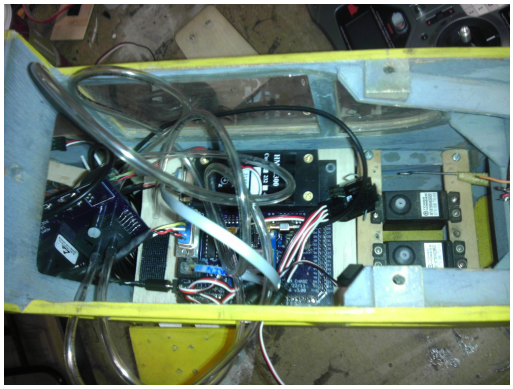


Figure 15. System Integration into 0.40-size R/C Piper Cub

After integration, an initial test flight was conducted. Unfortunately, an electrical short-circuit caused the vehicle to crash and corrupted some data, so no aerodynamic force estimation has been conducted. However, the micro-SD card did survive the crash, and provided enough data to show the system was collecting data as expected before the short-circuit.

Figure 16. Pre-Flight System Checks

Figure 17. Crashed Vehicle in Water

The small amount of data collected was analyzed using a user interface developed to rapidly process the data acquisition systems files. This user interface will allow designers to analyze data immediately following a test flight and make corrections to the test flight program while still at the test flight location.

Due to the vehicle crash, the accuracy of the system developed still needs to be extensively validated. The immediate future work following this paper will be quantifying the accuracy of the system, as well as verifying reliability and safe integration into multiple unmanned systems. Most of the accuracy estimation will be accomplished by measuring changes to a vehicle's drag polar. The ability to measure parasite drag will be proven by adding payloads with a known drag coefficient to "dirty" the aircraft, and then measuring the change in the parasite drag coefficient of the test vehicle. To quantify the accuracy of drag-due-to-lift measurement, wings with different aspect ratios will be used on the vehicle, which should combine into a single equivalent curve.[27] Following the accuracy estimation, a sensor fusion algorithm will be developed to combine inertial sensors with the air data system and other available sensors in a manner similar to other current research,[28,29] in order to give full situational awareness to the UAS. This situational awareness could allow stability and control derivative estimation, which the aircraft designer could use to size tail and control surfaces.

References

[1]Daniel P Raymer et al. *Aircraft design: a conceptual approach*, volume 3. American Institute of Aeronautics and Astronautics, 1999.

[2]Leland Malcolm Nicolai and Grant Carichner. *Fundamentals of aircraft and airship design*, volume 1. Amer Inst of Aeronautics &, 2010.

[3]Jan Roskam. *Airplane design*. DARcorporation, 1985.

[4]Sighard F Hoerner. *Fluid-dynamic drag: practical information on aerodynamic drag and hydrodynamic resistance*. Hoerner Fluid Dynamics, 1965.

[5]Sighard F Hoerner and Henry V Borst. Fluid-dynamic lift: Practical information on aerodynamic and hydrodynamic lift. *NASA STI/Recon Technical Report A*, 76:32167, 1975.

[6]Vladislav Klein and Eugene A Morelli. *Aircraft system identification: theory and practice*. American Institute of Aeronautics and Astronautics Reston, VA, USA, 2006.

[7]Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[8]Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[9]Brian L Stevens and Frank L Lewis. Aircraft control and simulation. 2003.

[10]Atmel. Sam3x-sam3a series summary. Data Sheet 11057BSATARM13-Jul-12, Atmel, 2012.

[11]Atmel. 8-bit atmel microcontroller with 64k/128k/256k bytes in-systemprogrammable flash. Data Sheet 2549PAVR10/2012, Atmel, 2012.

[12]Analog Devices. Micropower, 3-axis,2 g/4 g/8 g digital output mems accelerometer. Technical report, 2012.

[13]Sparkfun. Adxl362 bob v01. Technical report, Sparkfun, 2012.

[14]U-Line. Easy-count counting scale data sheet. Technical report, U-Line.

[15]Honeywell Magnetic Sensors. Smart digital magnetometer hmr2300. Technical report, Honeywell, 2006.

[16]Honeywell. 3-axis digital compass ic hmc5883l. Technical report, Honeywell.

[17]Sparkfun. Hmc5883l breakout v11. Technical report, Sparkfun, 2011.

[18]Talat Ozyagcilar. *Calibrating an eCompass in the Presence of Hard and Soft-Iron Interference*. Freescale Semiconductors, rev 3.0 edition, 04 2013.

[19]Yury Petrov. Ellipsoid fit. Matlab File Exchange, 08 2013.

[20]InvenSense. Itg-3200 product specification revision 1.4. Technical report, 2010.

[21]Sparkfun. Itg 3200 v10. Schematic, Sparkfun, 2010.

[22]All Sensors. Ds-0012 rev a. Technical report, All Sensors.

[23]All Sensors. Ds-0010. Technical report, All Sensors.

[24]Maxim Integrated. Ds18b20 programmable resolution 1-wire digital thermometer. Technical report, Maxim Integrated, 2008.

[25]Paroscientific. Model 745 high accuracy portable pressure standard data sheet. Technical report, Paroscientific.

[26]Doug Weibel. Proof of concept test - extremely accurate 3d velocity measurement with a ublox 6t module., December 2012.

[27]Ludwig Prandtl. *Applications of modern hydrodynamics to aeronautics*. National Advisory Committee for Aeronautics, 1923.

[28]Matthew B. Rhudy; Trenton Larrabee; Haiyang Chao; Yu Gu; Marcello Napolitano. Uav attitude, heading, and wind estimation using gps/ins and an air data system. 2013.

[29]Seung-Min Oh and Eric N Johnson. Development of uav navigation system based on unscented kalman filter. In *AIAA Guidance, Navigation and Control Conference*, 2006.