2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Tailsitter Attitude Control Using Resolved Tilt-Twist

Jason M. Beach, Matthew E. Argyle, Timothy W. McLain,
Randal W. Beard and Stephen Morris

Abstract— **The tailsitter aircraft merges the endurance and speed of fixed-wing aircraft with the flexibility and VTOL abilities of rotorcraft. Because of the requirement to be functional at a full range of attitudes, quaternions are typically employed to calculate attitude error. Attitude control is then accomplished by using the vector component of the error quaternion to drive flight control surfaces. This paper demonstrates that this method of driving the flight control surfaces can be suboptimal for tailsitter type aircraft and can lead to undesired vehicle movement. An alternate method of calculating attitude error called resolved tilt-twist is improved and validated. The resolved-tilt twist method is implemented in hardware and hardware-in-loop simulation results are presented.**

Unmanned air vehicle (UAV) use has become widespread in many market segments, particularly the military where their use enhances the gathering of intelligence. One challenge that exists for current fixed-wing UAVs is the launch and recovery equipment required for their use. This challenge is somewhat overcome by rotorcraft; however, they require more power than their fixed-wing counterparts which limits their range and endurance. The tailsitter aircraft merges the efficiency of fixed-wing aircraft with the vertical takeoff and landing (VTOL) capabilities of rotorcraft. It can take off vertically with no special launch equipment, transition to level flight to perform its mission and then transition back to hover flight for landing, with no recovery equipment. To be viable, however, it must be robust to external disturbances during critical phases of flight, particularly landing, and must be safe to vehicle operators. A considerable amount of research has focused on the transition between the vertical and horizontal flight regimes and has been on vehicles with exposed propellers close to or on the nose of the vehicle [1]–[9]. The only published results of flight testing in known windy conditions are given by [10], although others such as [11] are likely to have been performed in wind.

The research presented in this paper utilizes an electric tailsitter vehicle produced by the MLB Company [12] called the eV-Bat and a scale model of it called the Y-Bat. Both vehicles are shown in Figure 1.

Fig. 1: eV-Bat and Y-Bat models.

Because a tailsitter operates through a full range of attitudes, attitude control cannot be accomplished with only a single set of Euler angles. If Euler angles are used, multiple sets are required as demonstrated by the University of Sydney on the T-Wing aircraft [1], [2]. Quaternions are an attractive alternative because they are not affected by gimbal lock like Euler angles are. Computing attitude error using quaternions form the basis for the research performed by [5], [7], [8], [11], [13].

When testing quaternion feedback attitude control as given in [13] on the Y-Bat, unexpected behavior was observed. In particular, when large heading error was present, error that should intuitively be pitch (y-axis) error showed up as yaw

Jason Beach is a graduate student with the Department of Mechanical Eng., Brigham Young University, Provo, Utah, 84602 jason.m.beach@gmail.com

Matthew Argyle is a PhD candidate with the Department of Electrical Eng., Brigham Young University, Provo, Utah, 84602 matt.argyle@gmail.com

Tim McLain is with the Dept. of Mechanical Eng., Brigham Young University, Provo, Utah, 84602 mclain@byu.edu

Randy Beard is with the Dept. of Electrical Eng., Brigham Young University, Provo, Utah, 84602 beard@byu.edu

Stephen Morris is president of the MLB Company smorris@spyplanes.com

978-1-4799-2376-2/14/$31.00 ©2014 IEEE

(z-axis) error. Matsumoto et al. also observed this behavior and proposed a solution called resolved tilt-twist [9] in which attitude error is decomposed into tilt error and twist error. The research presented here validates their work and makes several improvements to their algorithm. In addition to the work presented in this paper, we have also developed an alternative attitude error parameterization called the resolved Euler angles, where the attitude error is resolved into a ZYX-Euler representation relative to the body frame [14]. The results are comparable to resolved tilt-twist in the sense that attitude error is mapped to the actuators in a natural way, with the added benefit that the resolved Euler angles appear to be more computationally efficient. Comparative work is in progress.

A brief overview of important principles of attitude representation is given in Section I. Section II covers the usage of quaternions in attitude control and discusses the observed behavior. The resolved tilt-twist algorithm is derived in Section III and test results are presented in Section IV. Conclusions and acknowledgments are given in Sections V and VI.

I. Attitude Representations

At the most fundamental level, attitude representation consists of expressing the relationship between different coordinate frames. With regard to UAVs, three coordinate frames are of particular interest: the inertial coordinate frame which is fixed at an arbitrary location, the vehicle frame [15], and the body frame. The vehicle frame is aligned with the inertial frame with its origin at the vehicle center of mass. Both the inertial and vehicle frames are oriented such that the x-axis points north, y-axis points east and z-axis points to the center of the earth. The body-fixed frame follows standard convention with the x-axis pointing out of the tailsitter nose, the y-axis out the right wing and the z-axis out the bottom of the fuselage.

A. Rotation Matrices

A rotation matrix is a 3×3 unitary matrix that can be used to either express a vector in a new coordinate frame or to rotate a vector within a coordinate frame. In the context of attitude representation, one item of primary interest is the ability to express vectors referenced to the inertial

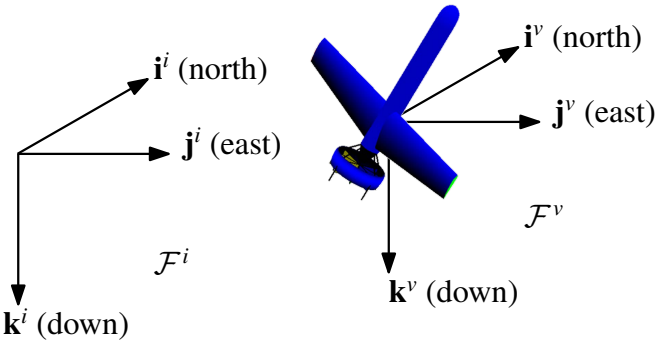


Fig. 2: The vehicle frame ($\mathcal{F}^v$) is a translated version of the inertial frame ($\mathcal{F}^i$).

frame in the body frame and vice versa–to express vectors referenced to the body frame in the inertial frame. As a useful example, the axes of the inertial and vehicle coordinate frames can be expressed in terms of the body frame by

$$\begin{bmatrix} \mathbf{i}_v^b & \mathbf{j}_v^b & \mathbf{k}_v^b \end{bmatrix} = \mathbf{R}_v^b \begin{bmatrix} \mathbf{i}^v & \mathbf{j}^v & \mathbf{k}^v \end{bmatrix}, \tag{1}$$

where $\mathbf{i}^v$, $\mathbf{j}^v$ and $\mathbf{k}^v$ are the unit vectors of the vehicle coordinate frame and $\mathbf{R}_v^b$ is the rotation matrix that rotates the vehicle frame to the body frame. Thus, $\mathbf{i}_v^b$, $\mathbf{j}_v^b$ and $\mathbf{k}_v^b$ are the axes of the vehicle coordinate frame expressed in the body coordinate frame.

Since $\begin{bmatrix} \mathbf{i}^v & \mathbf{j}^v & \mathbf{k}^v \end{bmatrix} = \mathbf{I_{3\times 3}}$, it can be observed from (1) that the columns of $\mathbf{R}_v^b$ are the axes of the vehicle or inertial frame expressed in the body frame.

Since a rotation matrix is unitary, the inverse of the matrix or the inverse rotation is simply its transpose, or

$$\mathbf{R}_b^v = \mathbf{R}_v^{b\top}. \tag{2}$$

This allows us to express the axes of the body frame in the vehicle coordinate frame by

$$\begin{bmatrix} \mathbf{i}_b^v & \mathbf{j}_b^v & \mathbf{k}_b^v \end{bmatrix} = \mathbf{R}_b^v \begin{bmatrix} \mathbf{i}^b & \mathbf{j}^b & \mathbf{k}^b \end{bmatrix}. \tag{3}$$

A similar observation can be made from (3): the columns of $\mathbf{R}_b^v$ are the axes of the body frame expressed in the vehicle frame. Noting from (2) that the columns of $\mathbf{R}_b^v$ are the rows of $\mathbf{R}_v^b$, the rows of $\mathbf{R}_v^b$ are also the body axes expressed in the vehicle frame. These observations are used in future sections of this paper.

B. Hover Euler Angles

To give physical intuition and aid in formulating a desired quaternion when the tailsitter is in hover, a new set of Euler angles similar to that given in [2] is developed. To do this, an additional 90° rotation must be inserted into the rotation sequence–the vehicle frame is first rotated by 90° about the $\mathbf{j}^v$ axis into what we call the hover coordinate frame. Similar to other intermediate frames, the hover frame is located at the tailsitter center of mass but has its x-axis pointing up, y-axis pointing east and z-axis pointing north as shown in Figure 3.

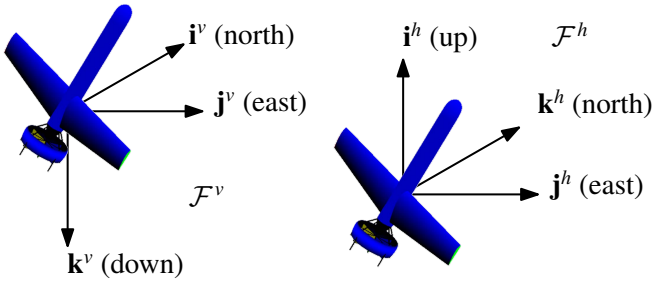


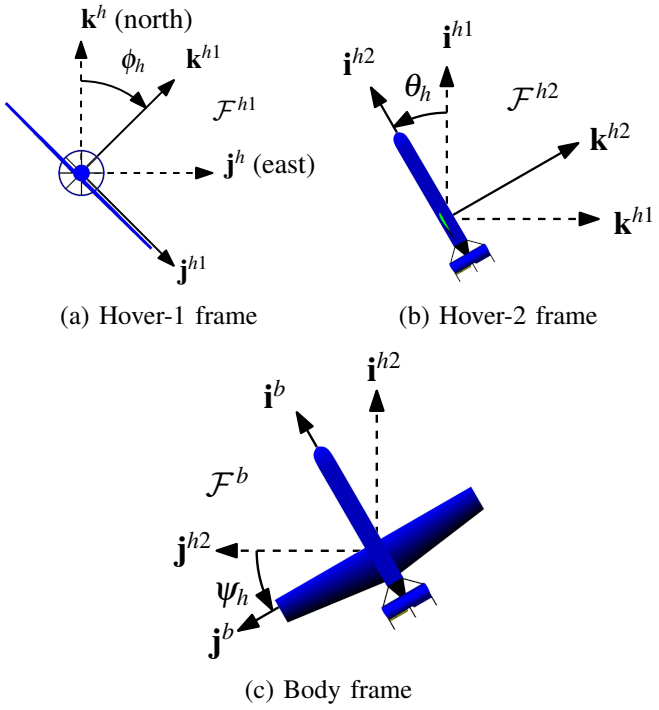
Fig. 3: The hover coordinate frame ($\mathcal{F}^h$) is simply the vehicle frame ($\mathcal{F}^v$) rotated by 90° about the $\mathbf{j}^v$ axis.

The complete rotation sequence from the vehicle frame to the body-fixed frame is

1) Rotate the vehicle frame 90° about the $\mathbf{j}^v$ axis into the hover frame, shown in Figure 3.
2) Rotate the hover frame about the negative $\mathbf{i}^h$ axis by the heading angle, ϕ_h, into the hover-1 frame as shown in Figure 4(a). Rotating about the negative $\mathbf{i}^h$ axis is necessary to maintain heading sense (i.e., so that $\phi_h = 90°$ still results in the belly pointing east).
3) Rotate the hover-1 frame about the $\mathbf{j}^{h1}$ axis by the elevation angle, θ_h, into the hover-2 frame as shown in Figure 4(b).
4) Rotate the hover-2 frame about the $\mathbf{k}^{v2}$ axis by the bank angle, ψ_h, into the body frame as shown in Figure 4(c).

Mathematically this is represented by

$$\mathbf{R}_v^h = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$



(a) Hover-1 frame (b) Hover-2 frame

(c) Body frame

Fig. 4: Hover flight Euler angle coordinate frames.

$$\mathbf{R}_h^{h1}(-\phi_h) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_{\phi_h} & -\mathrm{s}_{\phi_h} \\ 0 & \mathrm{s}_{\phi_h} & \mathrm{c}_{\phi_h} \end{bmatrix},$$

$$\mathbf{R}_{h1}^{h2}(\theta_h) = \begin{bmatrix} \mathrm{c}_{\theta_h} & 0 & -\mathrm{s}_{\theta_h} \\ 0 & 1 & 0 \\ \mathrm{s}_{\theta_h} & 0 & \mathrm{c}_{\theta_h} \end{bmatrix},$$

$$\mathbf{R}_{h2}^{b}(\psi_h) = \begin{bmatrix} \mathrm{c}_{\psi_h} & \mathrm{s}_{\psi_h} & 0 \\ -\mathrm{s}_{\psi_h} & \mathrm{c}_{\psi_h} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R}_v^b = \mathbf{R}_{h2}^b(\psi_h)\mathbf{R}_{h1}^{h2}(\theta_h)\mathbf{R}_h^{h1}(-\phi_h)\mathbf{R}_v^h, \quad (4)$$

where s_x and c_x are the sine and cosine of their respective angles. Like any other sequence of three rotations, this too is affected by gimbal lock when $\theta_h = \pm 90°$. To cover a full range of attitudes it is necessary to switch between standard aircraft Euler angles and these hover Euler angles depending on pitch. Also, it is important to note that in deriving the hover Euler angles, we have chosen to maintain ϕ, θ and ψ as rotations about the respective x, y and z axes. Doing so means that when using the level Euler angles, ψ_ℓ defines heading, but in hover mode ϕ_h does.

C. Quaternions

Instead of three separate rotations, the quaternion expresses attitude in terms of a single rotation. The *Euler-Rodrigues symmetric parameters* are related to a unit-length axis of rotation $\mathbf{e}$ and an angle of rotation Θ by

$$\eta = \begin{bmatrix} \eta_0 \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \cos\frac{\Theta}{2} \\ \mathbf{e}\sin\frac{\Theta}{2} \end{bmatrix}. \tag{5}$$

Here we have chosen to place the scalar element as the first component and to express the vector part in bold. A quaternion must be unit-norm to represent a rotation. The rotation matrix based on a unit-norm quaternion is

$$\mathbf{R}_v^b(\eta) = \tag{6}$$
$$\begin{bmatrix} \eta_0^2+\eta_x^2-\eta_y^2-\eta_z^2 & 2(\eta_x\eta_y+\eta_z\eta_0) & 2(\eta_x\eta_z-\eta_y\eta_0) \\ 2(\eta_x\eta_y-\eta_z\eta_0) & \eta_0^2-\eta_x^2+\eta_y^2-\eta_z^2 & 2(\eta_y\eta_z+\eta_x\eta_0) \\ 2(\eta_x\eta_z+\eta_y\eta_0) & 2(\eta_y\eta_z-\eta_x\eta_0) & \eta_0^2-\eta_x^2-\eta_y^2+\eta_z^2 \end{bmatrix}.$$

Successive rotations are easily expressed as a single quaternion through the composition operator

$$\eta'' = \eta' \otimes \eta = \begin{bmatrix} \eta_0'\eta_0 - \boldsymbol{\eta}'\cdot\boldsymbol{\eta} \\ \eta_0'\boldsymbol{\eta} + \eta_0\boldsymbol{\eta}' - \boldsymbol{\eta}'\times\boldsymbol{\eta} \end{bmatrix}. \tag{7}$$

As with most quaternion operations, composition is not commutative. In (7), η'' represents the overall rotation given by η followed by η'. This can also be written as

$$\eta'' = \eta' \otimes \eta = \left[\eta^{\otimes}\right]\eta', \tag{8}$$

where

$$\left[\eta^{\otimes}\right] \triangleq \begin{bmatrix} \eta_0 & -\eta_x & -\eta_y & -\eta_z \\ \eta_x & \eta_0 & -\eta_z & \eta_y \\ \eta_y & \eta_z & \eta_0 & -\eta_x \\ \eta_z & -\eta_y & \eta_x & \eta_0 \end{bmatrix}. \tag{9}$$

In this form, it can be seen that if $|\eta| = 1$, then $\left[\eta^{\otimes}\right]$ is unitary and $\left[\eta^{\otimes}\right]^{\top}\left[\eta^{\otimes}\right] = \mathbf{I}$. This is useful in cases where η' is the unknown quantity which can be solved for as

$$\begin{aligned} \eta'' &= \left[\eta^{\otimes}\right]\eta', \\ \eta' &= \left[\eta^{\otimes}\right]^{\top}\eta''. \end{aligned} \tag{10}$$

Quaternions have the advantage of being singularity free and computationally efficient; however, they are not without their challenges. Perhaps the greatest difficulty is a lack of simple physical intuition on the rotation a given quaternion represents. Another challenge is the unit-norm constraint. Since a quaternion that represents a rotation must be unit-norm, the operations of addition and subtraction are not meaningful. Lastly, any rotation can be represented by two quaternions, η and $-\eta$. This non-uniqueness is referred to as double coverage. Double coverage does not pose an issue in the strict mathematical operations of quaternion algebra, but does need to be accounted for in applications such as quaternion feedback attitude control. The typical way of dealing with double coverage is to simply keep the scalar element positive. If the scalar element becomes negative, negating all four elements of the quaternion maintains the same rotation, while protecting against double coverage.

D. Conversions

The expression for converting a quaternion to hover Euler angles is derived by setting (4) equal to (6). Elements of the third row and third column of each of these matrices are then used to determine the conversions. The resulting relationships are

$$\begin{bmatrix} \phi_h \\ \theta_h \\ \psi_h \end{bmatrix} = \begin{bmatrix} \text{atan2}\left[\eta_y\eta_z - \eta_0\eta_x, \eta_0\eta_y + \eta_x\eta_z\right] \\ \sin^{-1}\left[-\eta_0^2+\eta_x^2+\eta_y^2-\eta_z^2\right] \\ \text{atan2}\left[\eta_0\eta_x + \eta_y\eta_z, \eta_0\eta_y - \eta_x\eta_z\right] \end{bmatrix}. \tag{11}$$

To express the hover Euler angles as a single quaternion requires the use of the following intermediate quaternion definitions:

$$\begin{aligned} \eta_v &\triangleq \left[\tfrac{\sqrt{2}}{2} \quad 0 \quad \tfrac{\sqrt{2}}{2} \quad 0\right]^{\top}, \\ \eta_{\phi_h} &\triangleq \left[\cos\left(\tfrac{\phi_h}{2}\right) \quad \sin\left(-\tfrac{\phi_h}{2}\right) \quad 0 \quad 0\right]^{\top}, \\ \eta_{\theta_h} &\triangleq \left[\cos\left(\tfrac{\theta_h}{2}\right) \quad 0 \quad \sin\left(\tfrac{\theta_h}{2}\right) \quad 0\right]^{\top}, \\ \eta_{\psi_h} &\triangleq \left[\cos\left(\tfrac{\psi_h}{2}\right) \quad 0 \quad 0 \quad \sin\left(\tfrac{\psi_h}{2}\right)\right]^{\top}. \end{aligned}$$

Note that these are the quaternion equivalents of the four intermediate matrices used in (4). They are combined using composition by

$$\eta = \eta_{\psi_h} \otimes \left(\eta_{\theta_h} \otimes \left(\eta_{\phi_h} \otimes \eta_v\right)\right). \tag{12}$$

Simplifying the resulting vector gives

$$\eta = \frac{\sqrt{2}}{2}\begin{bmatrix} (\mathrm{c}\frac{\theta_h}{2} - \mathrm{s}\frac{\theta_h}{2})(\mathrm{c}\frac{\phi_h}{2}\,\mathrm{c}\frac{\psi_h}{2} - \mathrm{s}\frac{\phi_h}{2}\,\mathrm{s}\frac{\psi_h}{2}) \\ (\mathrm{c}\frac{\theta_h}{2} + \mathrm{s}\frac{\theta_h}{2})(\mathrm{c}\frac{\phi_h}{2}\,\mathrm{s}\frac{\psi_h}{2} - \mathrm{s}\frac{\phi_h}{2}\,\mathrm{c}\frac{\psi_h}{2}) \\ (\mathrm{c}\frac{\theta_h}{2} + \mathrm{s}\frac{\theta_h}{2})(\mathrm{c}\frac{\phi_h}{2}\,\mathrm{c}\frac{\psi_h}{2} + \mathrm{s}\frac{\phi_h}{2}\,\mathrm{s}\frac{\psi_h}{2}) \\ (\mathrm{c}\frac{\theta_h}{2} - \mathrm{s}\frac{\theta_h}{2})(\mathrm{c}\frac{\phi_h}{2}\,\mathrm{s}\frac{\psi_h}{2} + \mathrm{s}\frac{\phi_h}{2}\,\mathrm{c}\frac{\psi_h}{2}) \end{bmatrix}. \quad (13)$$

Internally the autopilot always uses quaternions to express vehicle attitude; however Equations (11) and (13) are used throughout this paper to intuitively express that attitude.

II. Quaternion Feedback Attitude Control

Quaternion feedback attitude control exploits the fact that the four elements of a quaternion are related to an axis and angle of rotation. To do this an error quaternion, $\tilde{\eta}$, is defined to be the rotation needed to get from the current estimated attitude, $\hat{\eta}$, to some desired attitude, η^d, or

$$\eta^d = \tilde{\eta} \otimes \hat{\eta}. \quad (14)$$

From (10), the quaternion error can be solved for by

$$\tilde{\eta} = \left[\hat{\eta}^{\otimes}\right]^{\top} \eta^d. \quad (15)$$

The vector component of $\tilde{\eta}$ defines the axis of rotation to get from the current attitude to the desired attitude, expressed in the body frame [5]. Because of this, each component of the vector can be used directly in a proportional-integral-derivative (PID) controller to drive a control surface: the x-component drives the aileron or roll effort, the y-component drives the elevator or pitch effort, and the z-component drives the rudder or yaw effort or

$$\delta_a = k_p\tilde{\eta}_x - k_d p + k_i \int \tilde{\eta}_x \,\mathrm{d}t, \quad (16)$$

$$\delta_e = k_p\tilde{\eta}_y - k_d q + k_i \int \tilde{\eta}_y \,\mathrm{d}t, \quad (17)$$

$$\delta_r = k_p\tilde{\eta}_z - k_d r + k_i \int \tilde{\eta}_z \,\mathrm{d}t. \quad (18)$$

where p, q, and r are the body angular rates given by rate-gyros and the control efforts δ_a, δ_e and δ_r are mapped to their respective control surfaces such that a positive control effort results in a positive moment about that axis.

Quaternion-based feedback control performed well in simulation in which MATLAB was used to simulate both vehicle dynamics and autopilot functions. Initial hardware testing then took place on the Y-Bat mounted in a 2-axis gimbal, shown in Figure 5. This gimbal allowed relatively free rotation in the body y- and z-axes while preventing rotation about the body x-axis.

Fig. 5: Y-Bat 2-axis gimbal.

Testing in the gimbal apparatus provided an opportunity to debug the control algorithms as implemented on the autopilot and perform initial gain tuning. Because testing was performed indoors where magnetometer measurements were unreliable and because rotation about the body x-axis was constrained, the x-axis gains in (16) were set to zero. It was assumed this was sufficient to remove the effects of heading error from the controller.

During the gimbal tests, a desired attitude of $\phi_h = 0°$, $\theta_h = 0°$, and $\psi_h = 0°$ was commanded. It was anticipated this would drive the Y-Bat to point its nose vertically; however, instead it continuously spiraled and wobbled around the vertical position. It was determined that even though the x-axis gains were set to zero, the error in the body x-axis still impacted the error in the other body axes.

Given the above desired attitude and a current estimated attitude pitched down slightly to $\theta_h = -10°$ with an arbitrary heading of ϕ_h, the intuitive control vane command in all cases would an elevator deflection to correct the pitch error and an

aileron-like deflection to correct the heading error. No rudder (z-axis) deflection is needed.

The impact of x-axis error is shown in Figure 6. Since the desired heading is zero, current heading and heading error have the same magnitude. As heading increases, so does the x-axis component of the error quaternion, as expected. When heading error is zero, pitch error appears fully in the y-axis as expected; however, as the heading error increases to 180°, the y-axis error shifts to the z-axis component until finally at $\phi_h = 180°$ the y-axis component is zero causing no correction in pitch to be made. This gradual shift was not anticipated.

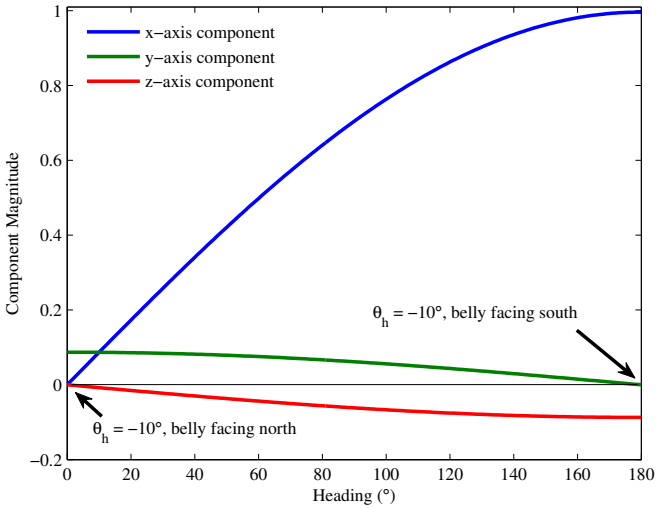


Fig. 6: Vector components of the error quaternion given a desired attitude of the nose pointing up and belly facing north and a estimated attitude pitched down to −10° and heading ranging from 0°-180°. Since the desired heading is zero, heading and heading error have the same magnitude.

III. Resolved Tilt-Twist Feedback Control

Matsumoto et al. describe this same phenomenon in [9]. They propose a solution called resolved tilt-twist angle control, or simply RTT, that calculates attitude error by decomposing it into a tilt error and a twist error. The work in this section validates RTT as a viable alternative to calculating attitude error directly with quaternions by incorporating the RTT control algorithm on the eV-Bat and Y-Bat. It also builds upon their work by making several algorithmic improvements and corrections. Deriving the required control effort makes heavy use of the fact that the columns of $\mathbf{R}_v^b$ are the vehicle axes expressed in the respective body frame and that the rows of $\mathbf{R}_v^b$ are the respective body axes expressed in the vehicle frame as explained in Section I-A.

A. Derive Tilt Error

To start, the estimated and desired attitude quaternions must be expressed as rotation matrices using (6). Alternatively, these rotation matrices can be generated directly from hover Euler angles using (4). Attitude error is then defined as the rotation required to rotate the estimated coordinate frame such that it is aligned with the desired coordinate frame or

$$\mathbf{R}_v^b(\eta^d) = \tilde{\mathbf{R}}_v^b \mathbf{R}_v^b(\hat{\eta}). \tag{19}$$

This definition is equivalent to that given by Matsumoto. Solving for the attitude error, $\tilde{\mathbf{R}}_v^b$, gives

$$\tilde{\mathbf{R}}_v^b = \mathbf{R}_v^b(\eta^d)\mathbf{R}_v^b(\hat{\eta})^\top = \begin{bmatrix} \tilde{r}_{11} & \tilde{r}_{12} & \tilde{r}_{13} \\ \tilde{r}_{21} & \tilde{r}_{22} & \tilde{r}_{23} \\ \tilde{r}_{31} & \tilde{r}_{32} & \tilde{r}_{33} \end{bmatrix}, \tag{20}$$

where $\tilde{r}_{mn}$ denotes the (m,n) element of the respective rotation matrix, which in this case is $\tilde{\mathbf{R}}_v^b$.

The tilt error consists of an error about the body y-axis and body z-axis. Two cases are considered to derive the error in these axes. The first case illustrates the y-axis (pitch) error and the second shows the z-axis (yaw) error. In each case, the desired attitude is the vehicle nose pointing straight up and the belly facing north (ϕ_h, θ_h and ψ_h all 0°).

If the estimated attitude was pitched up ($\phi_h = 0°$, $\theta_h = 10°$ and $\psi_h = 0°$) then the three rotation matrices in (19) are

$$\begin{aligned} \mathbf{R}_v^b(\eta^d) &= \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \\ \mathbf{R}_v^b(\hat{\eta}) &= \begin{bmatrix} -0.17 & 0 & -0.98 \\ 0 & 1 & 0 \\ 0.98 & 0 & -0.17 \end{bmatrix}, \\ \tilde{\mathbf{R}}_v^b &= \begin{bmatrix} 0.98 & 0 & 0.17 \\ 0 & 1 & 0 \\ -0.17 & 0 & 0.98 \end{bmatrix}. \end{aligned} \tag{21}$$

Although $\tilde{\mathbf{R}}_v^b$ is the rotation between the estimated and desired coordinate frames, it can also stand alone to rotate the vehicle frame into what is referred to here as the error frame. Since the columns of $\mathbf{R}_v^b(\eta^d)$ and $\mathbf{R}_v^b(\hat{\eta})$ represent the axes of the vehicle frame expressed in some other coordinate frame (namely, the desired body frame and estimated body frame, respectively), $\tilde{\mathbf{R}}_v^b$ also expresses the axes of the vehicle frame in another coordinate–the error frame. Comparing the columns of $\tilde{\mathbf{R}}_v^b$ in (21) with the frames shown in Figure 7, shows that this is indeed that case. Figure 7 also illustrates (19) graphically–if the rotation between the vehicle and estimated frame is combined with the rotation between the vehicle and error frame, the result is the rotation between the vehicle and desired frame.

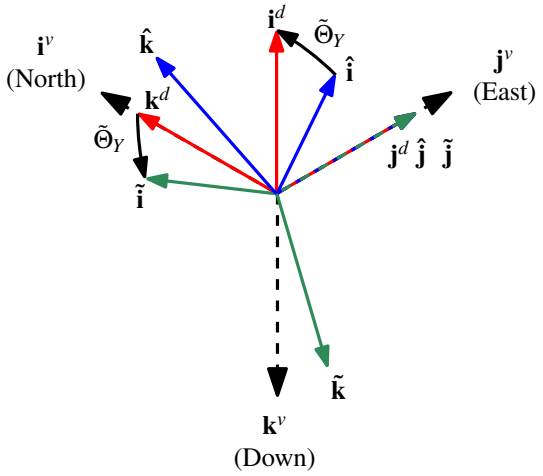


Fig. 7: The estimated attitude is $\phi_h = 0°$, $\theta_h = 10°$ and $\psi_h = 0°$. The error coordinate frame is the vehicle frame rotated 10° about negative $\mathbf{j}^v$ axis. Rotating the body frame by 10° about the negative $\hat{\mathbf{j}}$ will correct the attitude error. $\tilde{\Theta}_Y$ is defined as the angle between the projection of $\tilde{\mathbf{i}}$ onto the $\mathbf{i}^v$-$\mathbf{k}^v$ plane and $\mathbf{i}^v$.

Since the columns of $\tilde{\mathbf{R}}_v^b$ express the vehicle axes in the error frame, using the x-axis column ($\mathbf{i}^v$ expressed in the error frame) to determine the tilt error as originally done in [9] has an undesired side-effect. Heading error shows up in the error frame as a rotation about the error frame x-axis. This makes sense because in hover heading is changed by rotating the tailsitter about its x-axis; however, this changes $\tilde{\mathbf{j}}$ and $\tilde{\mathbf{k}}$ which in turn change how $\mathbf{i}^v$ is expressed in the error frame. Using $\mathbf{i}^v$ to define the tilt error would need to be accounted for as they do in the last step of their algorithm.

It is more convenient to instead use $\tilde{\mathbf{i}}$, the x-axis of the error frame expressed in the vehicle frame, or the top row in $\tilde{\mathbf{R}}_v^b$, to determine tilt error. Expressing the x-axis of the error frame in the vehicle frame has the property of being invariant with respect to heading error. This is because rotating a coordinate frame about an axis does not change that axis, and heading errors are rotations about the error x-axis. Given this, the y-axis error is given as

$$\tilde{\Theta}_Y = -\operatorname{atan2}\left[\tilde{r}_{13}, \tilde{r}_{11}\right]. \tag{22}$$

The negative sign accounts for the needed direction of rotation. From Figure 7 it can be seen that in this case without the negative sign, (22) would return a positive value; however, a negative rotation about the body y-axis is needed to correct the vehicle attitude.

To derive the z-axis tilt error consider the same desired attitude of the tailsitter nose pointing straight up with the belly facing north and a new estimated attitude with a bank angle of 10° ($\phi_h = 0°$, $\theta_h = 0°$ and $\psi_h = 10°$). In this case the rotation matrices are

$$\begin{aligned}
\mathbf{R}_v^b(\eta^d) &= \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \\
\mathbf{R}_v^b(\hat{\eta}) &= \begin{bmatrix} 0 & 0.17 & -0.98 \\ 0 & 0.98 & 0.17 \\ 1 & 0 & 0 \end{bmatrix}, \\
\tilde{\mathbf{R}}_v^b &= \begin{bmatrix} 0.98 & -0.17 & 0 \\ 0.17 & 0.98 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{23}$$

These coordinate frames are shown in Figure 8.

Using the x-axis row of $\tilde{\mathbf{R}}_v^b$ to determine the angle of rotation gives

$$\tilde{\Theta}_Z = \operatorname{atan2}\left[\tilde{r}_{12}, \tilde{r}_{11}\right]. \tag{24}$$

Similar to the previous case, this case requires a negative rotation about the body z-axis to correct the attitude; however, unlike the previous case, the sign of $\tilde{\Theta}_Z$ is already accounted for in the angle, which can be seen in Figure 8 and (23).

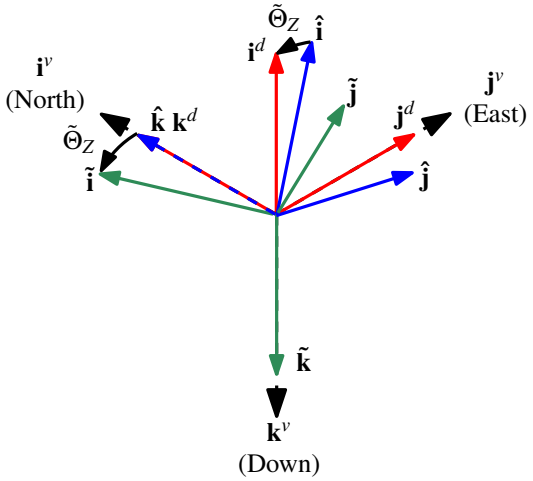


Fig. 8: The estimated attitude is $\phi_h = 0°$, $\theta_h = 0°$ and $\psi_h = 10°$. The error coordinate frame is the vehicle frame rotated 10° about negative $\mathbf{k}^v$ axis. Rotating the body frame by 10° about the negative $\hat{\mathbf{k}}$ will correct the attitude error. $\tilde{\Theta}_Z$ is defined as the angle between the projection of $\tilde{\mathbf{i}}$ onto the $\mathbf{i}^v$-$\mathbf{j}^v$ plane and $\mathbf{i}^v$.

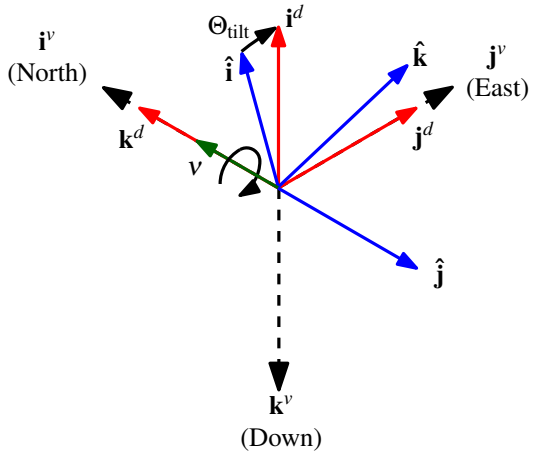


Fig. 9: Tilt error compensation by aligning the $\hat{\mathbf{i}}$ and $\mathbf{i}^d$ axes. In this case, the estimated body frame is the belly pointing east and pitched up slightly (ϕ_h, θ_h and ψ_h are 90°, 10°, and 0°). Rotating the estimated frame by 10° rotation about $\boldsymbol{\nu}$ would align $\hat{\mathbf{i}}$ and $\mathbf{i}^d$.

B. Derive Twist Error

The strategy for deriving the twist error is to find an intermediate coordinate frame with the same amount of twist error as the estimated frame, but whose x-axis is aligned with the x-axis of the desired body frame. The twist error is then the angle of rotation about the x-axis of this intermediate frame required to align the remaining axes.

To illustrate this case, we consider the scenario shown in Figure 9 where the estimated attitude is belly pointing east ($\phi_h = 90°$) and slightly pitched up to $\theta_h = 10°$. For this,

$$\begin{aligned}
\mathbf{R}_v^b(\eta^d) &= \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \\
\mathbf{R}_v^b(\hat{\eta}) &= \begin{bmatrix} 0 & -0.17 & -0.98 \\ -1 & 0 & 0 \\ 0 & 0.98 & -0.17 \end{bmatrix}, \\
\tilde{\mathbf{R}}_v^b &= \begin{bmatrix} 0.98 & 0 & 0.17 \\ -0.17 & 0 & 0.98 \\ 0 & -1 & 0 \end{bmatrix}.
\end{aligned} \tag{25}$$

To determine the rotation that would align the x-axes, the axes of both frames must be expressed in a common coordinate frame, the vehicle frame being the easiest to use. This is done using (6) and again noting that the rows of $\mathbf{R}_v^b$ represent the body axes expressed in the vehicle frame:

$$\begin{bmatrix} \mathbf{i}^{d\top} \\ \mathbf{j}^{d\top} \\ \mathbf{k}^{d\top} \end{bmatrix} \triangleq \mathbf{R}_v^b\left(\eta^d\right), \qquad \begin{bmatrix} \hat{\mathbf{i}}^\top \\ \hat{\mathbf{j}}^\top \\ \hat{\mathbf{k}}^\top \end{bmatrix} \triangleq \mathbf{R}_v^b\left(\hat{\eta}\right). \tag{26}$$

The angle of rotation needed to align the axes is determined from the dot product of the two x-axes or

$$\Theta_{\text{tilt}} = \cos^{-1}\left(\mathbf{i}^d \cdot \hat{\mathbf{i}}\right). \tag{27}$$

The unit-length axis of rotation is given by

$$\boldsymbol{\nu} = \frac{\hat{\mathbf{i}} \times \mathbf{i}^d}{|\hat{\mathbf{i}} \times \mathbf{i}^d|}, \tag{28}$$

which for this scenario, gives $\boldsymbol{\nu} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^\top$. Because the vectors used are expressed in the vehicle frame, $\boldsymbol{\nu}$ is also expressed in the vehicle frame. For the rotation to occur correctly, this axis must be expressed in the frame that is to be rotated, or in other words, the estimated body frame $\mathbf{R}_v^b(\hat{\eta})$. This is done by

$$\boldsymbol{\nu}^b = \mathbf{R}_v^b(\hat{\eta})\boldsymbol{\nu}. \tag{29}$$

For this case, the axis of rotation expressed prop-

erly is $\boldsymbol{\nu}^b = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^\top$, which is correctly the negative y-axis of the estimated body frame. Rodrigues' rotation formula [16] produces a rotation matrix that rotates a vector by a given angle about an arbitrary axis. Using $\boldsymbol{\nu}^b$ and Θ_{tilt} as the axis and angle of rotation gives

$$\mathbf{R}_{\nu^b} = \begin{cases} \mathbf{I} + [\boldsymbol{\nu}^{b\times}]\sin(\Theta_{\text{tilt}}) \\ \quad + [\boldsymbol{\nu}^{b\times}]^2[1-\cos(\Theta_{\text{tilt}})] & \Theta_{\text{tilt}} \neq 0 \\ \mathbf{I} & \Theta_{\text{tilt}} = 0 \end{cases}, \tag{30}$$

where

$$[\boldsymbol{\nu}^{b\times}] = \begin{bmatrix} 0 & -\nu_z^b & \nu_y^b \\ \nu_z^b & 0 & -\nu_x^b \\ -\nu_y^b & \nu_x^b & 0 \end{bmatrix}. \tag{31}$$

It is important to note that the rotation matrix obtained from (30) rotates a vector within a coordinate frame. We instead need to rotate the coordinate frame. This rotation matrix is obtained by transposing $\mathbf{R}_{\nu^b}$ or by simply changing the angle of rotation to $-\Theta_{\text{tilt}}$. The latter method has the advantage of being computationally faster and easier to implement. With this, (30) becomes

$$\mathbf{R}_{\nu^b}^\top = \begin{cases} \mathbf{I} - [\boldsymbol{\nu}^{b\times}]\sin(\Theta_{\text{tilt}}) \\ \quad + [\boldsymbol{\nu}^{b\times}]^2[1-\cos(\Theta_{\text{tilt}})] & \Theta_{\text{tilt}} \neq 0 \\ \mathbf{I} & \Theta_{\text{tilt}} = 0 \end{cases}. \tag{32}$$

The condition in (30) and (32) is required because the axis of rotation becomes ill-defined as the cross product of $\hat{\mathbf{i}}$ and $\mathbf{i}^d$ goes to zero. The cross product is zero whenever these two axes are aligned, or in other words, when $\Theta_{\text{tilt}} = 0$. This condition is more appropriate than that originally given in [9] since the presence (or absence) of twist error does not effect whether or not the x-axes are aligned. $\tilde{\mathbf{R}}_v^b$ is only equal to identity when there is zero tilt and zero twist error. The condition presented here has the added advantage of being easier to implement in programming languages such as C. The coordinate frame with the aligned x-axis is then given by

$$\mathbf{R}^a = \mathbf{R}_{\nu^b}^\top \mathbf{R}_v^b(\hat{\eta}). \tag{33}$$

The twist error is derived by decomposing this into row vectors, or

$$\begin{bmatrix} \mathbf{i}^{a\top} \\ \mathbf{j}^{a\top} \\ \mathbf{k}^{a\top} \end{bmatrix} \triangleq \mathbf{R}^a, \tag{34}$$

and then determining the angle of rotation by

$$\Theta_{\text{twist}} \triangleq \cos^{-1}\left(\mathbf{k}^a \cdot \mathbf{k}^d\right), \tag{35}$$

as shown in Figure 10. To determine the direction of rotation, Θ_{sign} is defined as

$$\Theta_{\text{sign}} \triangleq \cos^{-1}\left(\mathbf{j}^a \cdot \mathbf{k}^d\right), \tag{36}$$

making the twist error

$$\tilde{\Theta}_X = \begin{cases} -\Theta_{\text{twist}} & \Theta_{\text{sign}} \leq \frac{\pi}{2} \\ \Theta_{\text{twist}} & \Theta_{\text{sign}} > \frac{\pi}{2} \end{cases}. \tag{37}$$

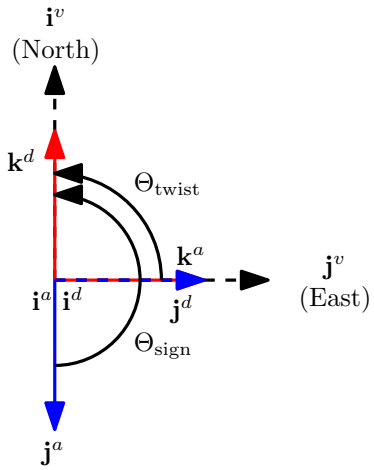


Fig. 10: Twist angle definition.

C. Feedback Control

Feedback control is accomplished using PID by

$$\delta_a = k_p\tilde{\Theta}_X - k_d p + k_i \int \tilde{\Theta}_X \, \mathrm{d}t, \tag{38}$$

$$\delta_e = k_p\tilde{\Theta}_Y - k_d q + k_i \int \tilde{\Theta}_Y \, \mathrm{d}t, \tag{39}$$

$$\delta_r = k_p\tilde{\Theta}_Z - k_d r + k_i \int \tilde{\Theta}_Z \, \mathrm{d}t, \tag{40}$$

where the control efforts δ_a, δ_e, and δ_r are again mapped to their respective control surfaces such that a positive control effort results in a positive moment about that axis. Figure 11 shows the error calculated based on the resolved tilt-twist method given the same scenario as in Figure 6. The desired

attitude is nose pointing up with belly facing north. The estimated attitude is pitched down slightly to $\theta_h = -10^\circ$ with a varying heading. As opposed to the quaternion attitude error computation, the RTT method produces a pitch error that is constant and a heading error that increases linearly, both as would be expected. RTT is summarized in Algorithm 1.

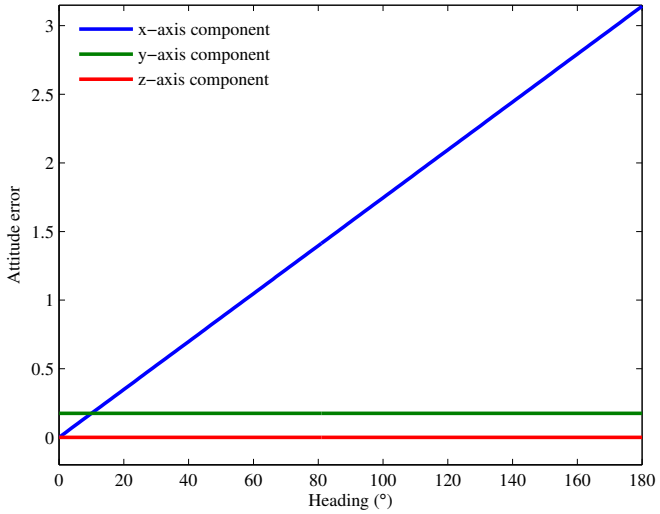


Fig. 11: RTT output given a similar scenario as the quaternion attitude control in Figure 6.

IV. Hardware in Loop Simulation

For debugging and algorithm validation, hardware-in-the-loop simulation tests were conducted. All autopilot estimation and control functions were implemented on and performed by a Lockheed-Martin Procerus Technologies Kestrel version 3.0 autopilot, shown in Figure 12. The estimator on the Kestrel was modified to allow operations around 90° pitch as described in [17]. Control algorithms were rewritten to implement quaternion feedback and RTT. A MATLAB simulator and custom interface were built to receive autopilot control surface commands and return simulated sensor data to the autopilot via the ground control station (GCS).

Initial attitude control performance was tested by modifying the autopilot guidance algorithms so that a remote control (RC) pilot could command a desired attitude quaternion via an RC transmitter. The desired attitude quaternion is generated using (13) where θ_h and ψ_h are determined from the position of the pitch and yaw RC sticks. ϕ_h was determined from current vehicle heading with the RC

Algorithm 1 Resolved Tilt-Twist Control

Input: The estimated vehicle attitude, expressed as a rotation matrix, $R_v^b(\hat{\eta})$ and the desired vehicle attitude, expressed as a rotation matrix, $R_v^b\left(\eta^d\right)$.

Calculate tilt error

$\tilde{\mathbf{R}}_v^b \triangleq \mathbf{R}_v^b\left(\eta^d\right)\mathbf{R}_v^b(\hat{\eta})^\top$

$\tilde{\Theta}_Y = -\operatorname{atan2}\left[\tilde{r}_{13}, \tilde{r}_{11}\right]$

$\tilde{\Theta}_Z = \operatorname{atan2}\left[\tilde{r}_{12}, \tilde{r}_{11}\right]$

Calculate twist error

$\hat{\mathbf{i}} \triangleq \begin{bmatrix}\hat{r}_{11} & \hat{r}_{12} & \hat{r}_{13}\end{bmatrix}^\top$

$\mathbf{i}^d \triangleq \begin{bmatrix}r_{11}^d & r_{12}^d & r_{13}^d\end{bmatrix}^\top$

$\mathbf{k}^d \triangleq \begin{bmatrix}r_{31}^d & r_{32}^d & r_{33}^d\end{bmatrix}^\top$

$\Theta_{\text{tilt}} = \cos^{-1}\left(\hat{\mathbf{i}} \cdot \mathbf{i}^d\right)$

if $\Theta_{\text{tilt}} = 0$ **then**

$\quad \mathbf{R}_{vb} = \mathbf{I}$

else

$\quad \boldsymbol{\nu} = \frac{\hat{\mathbf{i}} \times \mathbf{i}^d}{|\hat{\mathbf{i}} \times \mathbf{i}^d|}$

$\quad \boldsymbol{\nu}^b = \mathbf{R}_v^b(\hat{\eta})\boldsymbol{\nu}$

$\quad \mathbf{R}_{vb}^\top = \mathbf{I} - [\boldsymbol{\nu}^{b\times}]\sin(\Theta_{\text{tilt}}) + [\boldsymbol{\nu}^{b\times}]^2[1 - \cos(\Theta_{\text{tilt}})]$

end if

$\mathbf{R}^a = \mathbf{R}_{vb}^\top \mathbf{R}_v^b(\hat{\eta})$

$\mathbf{j}^a \triangleq \begin{bmatrix}r_{21}^a & r_{22}^a & r_{23}^a\end{bmatrix}^\top$

$\mathbf{k}^a \triangleq \begin{bmatrix}r_{31}^a & r_{32}^a & r_{33}^a\end{bmatrix}^\top$

$\Theta_{\text{twist}} \triangleq \cos^{-1}\left(\mathbf{k}^a \cdot \mathbf{k}^d\right)$

$\Theta_{\text{sign}} \triangleq \cos^{-1}\left(\mathbf{j}^a \cdot \mathbf{k}^d\right)$

$$\tilde{\Theta}_X = \begin{cases} -\Theta_{\text{twist}} & \Theta_{\text{sign}} \leq \frac{\pi}{2} \\ \Theta_{\text{twist}} & \Theta_{\text{sign}} > \frac{\pi}{2} \end{cases}$$

$\delta_a = k_p\tilde{\Theta}_X - k_d p + k_i \int \tilde{\Theta}_X \, \mathrm{d}t$

$\delta_e = k_p\tilde{\Theta}_Y - k_d q + k_i \int \tilde{\Theta}_Y \, \mathrm{d}t$

$\delta_r = k_p\tilde{\Theta}_Z - k_d r + k_i \int \tilde{\Theta}_Z \, \mathrm{d}t$

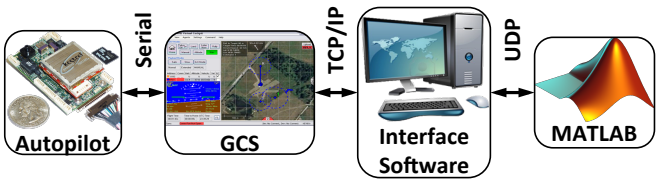


Fig. 12: HIL simulator block diagram. Photo of Kestel v3.0 autopilot courtesy of Lockheed-Martin Procerus Technologies.

stick command being summed onto the controller x-axis effort output. Figure 13 shows the result of commanding tilt steps. At t = 105s, position and heading feedback were enabled driving ϕ_h to zero and position to 0 m north and 0 m east.

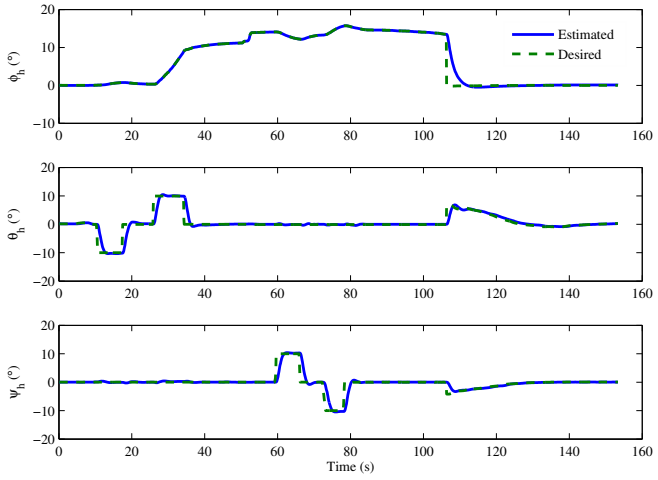


Fig. 13: Pitch and yaw steps using RTT.

To demonstrate roll performance a guidance mode in which translational velocity was controlled was used [17] since it allowed us to issue arbitrary heading commands. Figure 14 shows several 180° heading steps while translational velocity was commanded to be zero. This figure also shows the controller taking the shortest route to the commanded heading, particularly at t = 95s.

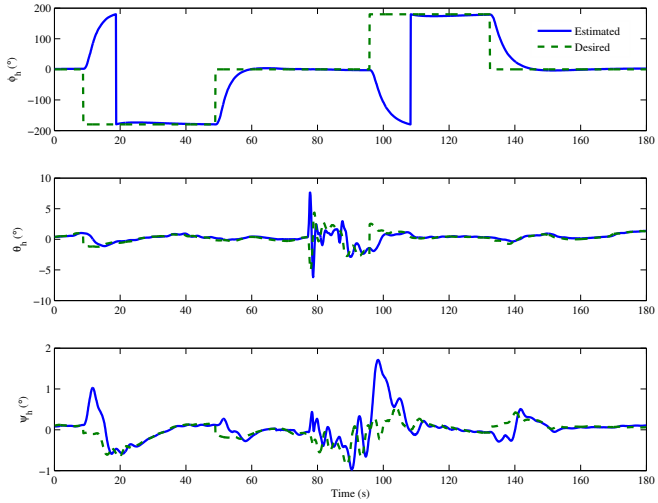
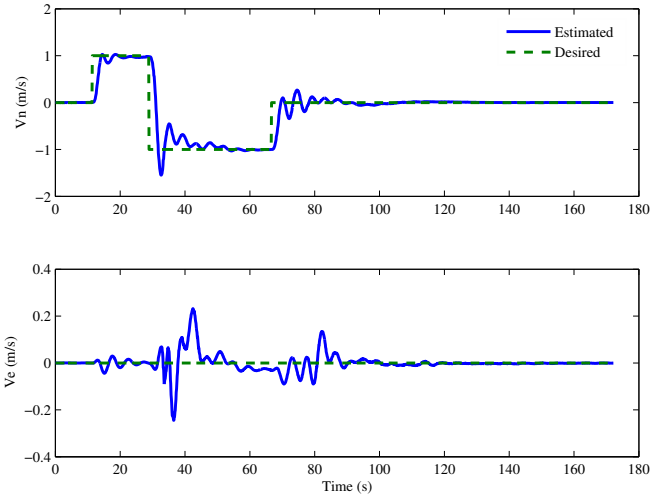
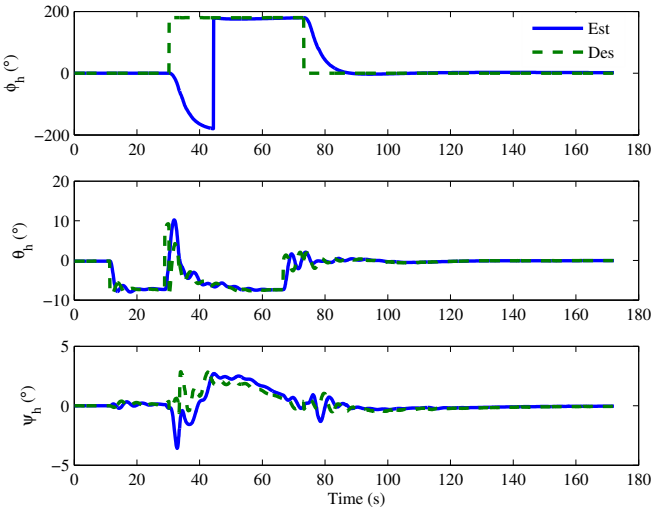


Fig. 14: Roll (Heading) steps with RTT.

Figure 15 shows RTT in use with outer-loop position control. In this scenario, the tailsitter was initially still with its belly facing north. It was commanded a north velocity of 1 m/s. Once that had stabilized, it was commanded a north velocity of -1 m/s and heading of 180°. The attitude loop performed adequately to allow the vehicle to complete the maneuver.



(a) Translational velocity



(b) Attitude

Fig. 15: Controlling translational velocity using RTT with higher level control.

V. Conclusions

This paper discussed how using the vector component of an attitude error quaternion to drive flight control surfaces can result in undesired movement on tailsitter aircraft, making it unsuitable for inner-loop attitude control. The resolved tilt-twist method provides commands that are physically

more intuitive and that result in more stable vehicle motion. Several corrections and improvements were made to RTT that improve its performance.

VI. Acknowledgments

This research was funded by the Air Force Research Lab through SBIR FA8650-13-C-2322 as a subcontract through the MLB Company.

References

[1] R. H. Stone and G. Clarke, "Optimization of Transition Maneuvers for a Tailsitter Unmanned Air Vehicle," in *Australian International Aerospace Congress*, Canberra, Mar. 2001.

[2] R. H. Stone, "Control Architecture for a Tail-sitter Unmanned Air Vehicle," *Control Conference, 2004. 5th Asian*, vol. 2, pp. 736–744, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1426743

[3] W. Green and P. Oh, "A MAV That Flies Like an Airplane and Hovers Like a Helicopter," in *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.* Monterey, CA: IEEE, 2005, pp. 693–698. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1511063

[4] A. Frank, J. S. McGrew, M. Valenti, D. Levine, and J. P. How, "Hover, Transition, and Level Flight Control Design for a Single-Propeller Indoor Airplane," in *AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, 2007.

[5] N. B. Knoebel, "Adaptive Quaternion Control of a Miniature Tailsitter UAV," Master's thesis, Brigham Young University, Dec. 2007. [Online]. Available: http://contentdm.lib.byu.edu/cdm/ref/collection/ETD/id/1197

[6] S. R. Osborne, "Transitions between Hover and Level Flight for a Tailsitter UAV," Master's thesis, Brigham Young University, Dec. 2007. [Online]. Available: http://contentdm.lib.byu.edu/cdm/ref/collection/ETD/id/1445

[7] E. N. Johnson, A. Wu, J. C. Neidhoefer, S. K. Kannan, and M. A. Turbe, "Flight-Test Results of Autonomous Airplane Transitions Between Steady-Level and Hovering Flight," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 2, pp. 358–370, Mar. 2008. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.29261

[8] K. Kita, A. Konno, and M. Uchiyama, "Transitions between Level Flight and Hovering of a Tailsitter Vertical Takeoff and Landing Aerial Robot," *Advanced Robotics*, vol. 24, no. 5/6, pp. 763–782, 2010.

[9] T. Matsumoto, K. Kita, R. Suzuki, A. Oosedo, K. Go, Y. Hoshino, A. Konno, and M. Uchiyama, "A Hovering Control Strategy for a Tail-sitter VTOL UAV that Increases Stability Against Large Disturbance," *2010 IEEE International Conference on Robotics and Automation*, pp. 54–59, May 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5509183

[10] R. H. Stone, P. Anderson, C. Hutchison, A. Tsai, P. Gibbens, and K. C. Wong, "Flight Testing of the T-Wing Tail-Sitter Unmanned Air Vehicle," *Journal of Aircraft*, vol. 45, no. 2, pp. 673–685, Mar. 2008. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.32750

[11] Y. Jung, S. Cho, and D. H. Shim, "A Comprehensive Flight Control Design and Experiment of a Tail-Sitter UAV," in *AIAA Guidance, Navigation, and Control (GNC) Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics, Aug. 2013, pp. 1–23. [Online]. Available: http://arc.aiaa.org/doi/pdf/10.2514/6.2013-4992

[12] MLB Company. [Online]. Available: http://spyplanes.com/products-v-bat/

[13] M. Argyle, R. Beard, and S. Morris, "The Vertical Bat Tailsitter: Dynamic Model and Control Architecture," *American Control Conference*, pp. 806–811, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6579935

[14] M. E. Argyle, J. M. Beach, R. Beard, T. W. McLain, and S. Morris, "Quaternion Based Attitude Error for a Tailsitter in Hover Flight," *American Control Conference*, 2014.

[15] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton, New Jersey: Princeton University Press, 2012. [Online]. Available: http://uavbook.byu.edu/

[16] S. Belongie. "Rodrigues' Rotation Formula." From *MathWorld*–A Wolfram Web Resource, created by E. W. Weisstein. [Online]. Available: http://mathworld.wolfram.com/RodriguesRotationFormula.html

[17] J. M. Beach, "Development of Tailsitter Hover Estimation and Control," Master's thesis, Brigham Young University, Feb. 2014.