Autopilot design for small fixed wing aerial vehicles

Randy Beard
Brigham Young University
Outline

• Control architecture
• Low level autopilot loops
• Path following
• Dubins airplane paths and path management
Control Architecture

- Path planner
  - Input: destination, obstacles
  - Output: Dubins path or waypoints
- Path manager
  - Input: straight-line or spiral path
  - Output: path following
- Path following
  - Input: airspeed, altitude, heading, commands
  - Output: autopilot
- Autopilot
  - Input: servo commands, wind
  - Output: unmanned aircraft
- Unmanned aircraft
  - Input: state estimator, on-board sensors
  - Output: map, status, tracking error, position error
  - State estimator
    - Input: survival error, status
    - Output: control architecture
  - Control architecture
    - Input: destination, obstacles
      - Path planner
        - Input: Dubins path or waypoints
          - Path manager
            - Input: straight-line or spiral path
              - Path following
                - Input: airspeed, altitude, heading, commands
                  - Autopilot
                    - Input: servo commands, wind
                      - Unmanned aircraft
                        - Input: state estimator, on-board sensors
Control Architecture (2)

- User input
- Dubins path or waypoints
- straight-line or spiral path
- airspeed, altitude, heading, commands
- servo commands
- wind

vision-based guidance

path manager

path following

autopilot

unmanned aircraft

state estimator

camera

on-board sensors

status

tracking error

position error

\( \hat{x}(t) \)
Outline

• Control architecture

• Low level autopilot loops

• Path following

• Dubins airplane paths and path management
Dubins path or waypoints

straight-line or spiral path

airspeed, altitude, heading, commands

servo commands

wind

destination, obstacles

map

status

tracking error

position error

unmanned aircraft

Control Architecture

state estimator

on-board sensors

\( \hat{x}(t) \)
Lateral Control loop

PI control for outer loop

PD control for inner loop

Control course angle $\chi$. 
If the airframe has a rudder, then the rudder is used to zero the side slip angle $\beta$.

Requires direct measurement or estimate of $\beta$.
Lateral Autopilot – In Flight Tuning

If model is not known, and autopilot must be tuned in flight, then the following gains are tuned one at a time, in this specific order:

**Inner Loop (roll attitude hold)**

- $k_{d\phi}$ - Increase $k_{d\phi}$ until onset of instability, and then back off by 20%.
- $k_{p\phi}$ - Tune $k_{p\phi}$ to get acceptable step response.

**Outer Loop (course hold)**

- $k_{p\chi}$ - Tune $k_{p\chi}$ to get acceptable step response.
- $k_{i\chi}$ - Tune $k_{i\chi}$ to remove steady state error.

**Sideslip hold (if rudder is available)**

- $k_{p\beta}$ - Tune $k_{p\beta}$ to get acceptable step response.
- $k_{i\beta}$ - Tune $k_{i\beta}$ to remove steady state error.
Longitudinal Autopilot – Traditional

- **Descend zone**:  
  - Zero throttle
  - Regulate airspeed by commanding pitch attitude

- **Altitude hold zone**
  - Regulate altitude by commanding pitch attitude
  - Regulate airspeed by commanding throttle

- **Climb zone**
  - Full throttle
  - Regulate airspeed by commanding pitch attitude

- **Take-off zone**
  - Full throttle
  - Regulate pitch to a fixed $\theta^c$
- Pitch attitude hold used for most longitudinal modes.

- Note that DC gain is not unity.

- Adding an integrator on this inner loop is not a good strategy.
6.4.5 Altitude-control State Machine

The longitudinal autopilot deals with the control of the longitudinal motions in the body $i^b$-frame: pitch angle, altitude, and airspeed. Up to this point, we have described four different longitudinal autopilot modes: 1- pitch attitude hold, 2- altitude hold using commanded pitch, 3- airspeed hold using commanded pitch, and 4- airspeed hold using throttle. These longitudinal control modes can be combined to create the altitude control state machine shown in Figure 6.20.

In the climb zone, the throttle is set to its maximum value, $t_t = 1$, and the airspeed hold from commanded pitch mode is used to control the airspeed, thus ensuring that the aircraft avoids stall conditions. Similarly, in the descend zone, the throttle is set to its minimum value, $t_t = 0$, and the airspeed hold from commanded pitch mode is again used to control airspeed. In this way, the MAV climbs at its maximum possible climb rate until it is close to the altitude set point. Similarly, in the descend zone, the MAV descends at a steady rate until it reaches the altitude hold zone. In the altitude hold zone, the airspeed-from-throttle mode is used to regulate the airspeed.
Longitudinal Autopilot – In Flight Tuning

If model is not known, and autopilot must be tuned in flight, then the following gains are tuned one at a time, in this specific order:

**Inner Loop (pitch attitude hold)**

- $k_{d\theta}$ - Increase $k_{d\theta}$ until onset of instability, and then back off by 20%.
- $k_{p\theta}$ - Tune $k_{p\theta}$ to get acceptable step response.

**Altitude Hold Outer Loop**

- $k_{p_h}$ - Tune $k_{p_h}$ to get acceptable step response.
- $k_{i_h}$ - Tune $k_{i_h}$ to remove steady state error.

**Airspeed Hold Outer Loop**

- $k_{p_{V_2}}$ - Tune $k_{p_{V_2}}$ to get acceptable step response.
- $k_{i_{V_2}}$ - Tune $k_{i_{V_2}}$ to remove steady state error.

**Throttle hold (inner loop)**

- $k_{p_V}$ - Tune $k_{p_V}$ to get acceptable step response.
- $k_{i_V}$ - Tune $k_{i_V}$ to remove steady state error.
• Traditional approach assumes longitude and lateral dynamics are decoupled
  – Altitude and airspeed does not influence course and position
  – Valid assumption

• Airspeed and altitude are decoupled
  – Invalid Assumption
# Airspeed and Attitude Coupling

<table>
<thead>
<tr>
<th>Pitch</th>
<th>Thrust</th>
<th>Assumed Response</th>
<th>True Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Altitude</td>
<td>Airspeed</td>
</tr>
<tr>
<td>↑</td>
<td>-</td>
<td>↑</td>
<td>-</td>
</tr>
<tr>
<td>↓</td>
<td>-</td>
<td>↓</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>↑</td>
<td>-</td>
<td>↑</td>
</tr>
<tr>
<td>-</td>
<td>↓</td>
<td>-</td>
<td>↓</td>
</tr>
</tbody>
</table>
Total Energy Control System

- Developed in the 1980’s by Antonius Lambregts
- Based on energy manipulation techniques from the 1950’s
- Control the energy of the system instead of the altitude and airspeed

![Energy Control System Diagram]

Thrust

Kinetic Energy

Total Energy

Potential Energy

Elevator

Drag
Total Energy Control System

- Kinetic Energy: \( E_K \triangleq \frac{1}{2} mV_a^2 \)

- Potential Energy: \( E_P \triangleq mgh \)

- Total Energy: \( E_T \triangleq E_P + E_K \)

- Energy Difference: \( E_D \triangleq E_P - E_K \)
Original TECS proposed by Lambregts is based on energy rates:

- \( T^c = T_D + k_{p,t} \dot{E}_t + k_{i,t} \int_{t_0}^{t} \dot{E}_t \delta_T \)
  - \( T_D \) is thrust needed to counteract drag
  - PI controller based on total energy rate

- \( \theta^c = k_{p,\theta} \dot{E}_d + k_{i,\theta} \int_{t_0}^{t} \dot{E}_d \delta_T \)
  - PI controller based on energy distribution rate

- Stability shown for linear systems

We will show that the performance of this scheme is less than desirable.
Total Energy Control System

Nonlinear re-derivation:

- **Error Definitions**
  
  \[
  \tilde{E}_K = \frac{1}{2} m \left( (V_a^d)^2 - V_a^2 \right) \\
  \tilde{E}_P = mg (h^d - h)
  \]

- **Lyapunov Function**
  
  \[
  V = \frac{1}{2} \tilde{E}_T^2 + \frac{1}{2} \tilde{E}_D^2
  \]

- **Controller**
  
  \[
  T^c = D + \frac{\tilde{E}_T^d}{V_a} + k_T \frac{\tilde{E}_T}{V_a} \\
  \gamma^c = \sin^{-1} \left( \frac{\dot{h}^d}{V_a} + \frac{1}{2mgV_a} \left( -k_1 \tilde{E}_K + k_2 \tilde{E}_P \right) \right)
  \]
Total Energy Control System

- Original: \[ T^c = D + k_{p,t} \frac{\dot{E}_T}{mgV_a} + k_{i,t} \frac{\tilde{E}_T}{mgV_a} \]
- Nonlinear: \[ T^c = D + \frac{\dot{E}_d}{V_a} + k_T \frac{\bar{E}_T}{V_a} \]

Similar if \( k_{p,T} = mg \) and \( k_{i,T} = mgk_T \).

The nonlinear controller uses the desired energy rate.
Total Energy Control System

- Modified Original (Ardupilot):

\[ \theta^c = \frac{k_{p,\theta}}{V_a mg} \left((2 - k) \dot{E}_P - k \dot{E}_K\right) + \frac{k_{i,\theta}}{V_a mg} \tilde{E}_D \]

\[ k \in [0, 2] \]

- Nonlinear:

\[ \gamma^c = \sin^{-1} \left( \frac{\dot{h}^d}{V_a} + \frac{1}{2mgV_a} \left( -k_1 \tilde{E}_K + k_2 \tilde{E}_P \right) \right) \]

\[ k_1 \triangleq |k_T - k_D| \]

\[ k_2 \triangleq k_T + k_D \]

\[ 0 < k_T \leq k_D \]

Lyapunov derivation suggests potential energy error should be weighted more than kinetic energy
If the drag is unknown, then we can add an adaptive estimate:

\[
T^c = \hat{D} + \Phi^\top \hat{\Psi} + \frac{\dot{E}_T^d}{V_a} + k_T \frac{\tilde{E}_T}{V_a}
\]

\[
\gamma^c = \sin^{-1} \left( \frac{\dot{h}^d}{V_a} + \frac{1}{2mgV_a} \left( -k_1 \tilde{E}_K + k_2 \tilde{E}_P \right) \right)
\]

\[
\dot{\Psi} = \left( \Gamma_T \tilde{E}_T - \Gamma_D \tilde{E}_D \right) \Phi V_a
\]
Total Energy Control System

Step in Altitude, Constant Airspeed

Altitude (m)

Airspeed (m/s)

Time (s)

Original TECS  Nonlinear TECS  Adaptive TECS  SLC  TECS PID  ArduPilot PID
Step in Airspeed, Constant Altitude

- Total Energy Control System
- Original TECS
- Nonlinear TECS
- Adaptive TECS
- SLC
- TECS PID
- ArduPilot PID

Graphs showing altitude and airspeed over time for different control systems.
Total Energy Control System

Step in Altitude and Airspeed

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Altitude (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Airspeed (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>16</td>
</tr>
<tr>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>50</td>
<td>16</td>
</tr>
</tbody>
</table>

Legend:
- Original TECS
- Nonlinear TECS
- Adaptive TECS
- SLC
- TECS PID
- ArduPilot PID
Observations

- TECS seems to work better than successive loop closure.
- Still need separate mode for take-off
- Nonlinear TECS seems to work better, but the Ardupilot controller works very well.
Outline

• Control architecture

• Low level autopilot loops

• Path following

• Dubins airplane paths and path management
Focus on following straight lines and helices
Dubins airplane model:

\[ \dot{r}_n = V \cos \psi \cos \gamma^c \]
\[ \dot{r}_e = V \sin \psi \cos \gamma^c \]
\[ \dot{r}_d = -V \sin \gamma^c \]
\[ \dot{\psi} = \frac{g}{V} \tan \phi^c. \]

with the constraint that

\[ |\phi^c| \leq \bar{\phi} \]
\[ |\gamma^c| \leq \bar{\gamma}. \]
Path defined as intersection of two 2D manifolds in $\mathbb{R}^3$:

$$\begin{align*}
\alpha_1(r) &= 0 \\
\alpha_2(r) &= 0
\end{align*}$$

The path given by the intersection is connected and one-dimensional. Define the function

$$V(r) = \frac{1}{2} \alpha_1^2(r) + \frac{1}{2} \alpha_2^2(r).$$

Consider the velocity command

$$u' = -K_1 \frac{\partial V}{\partial r} + K_2 \frac{\partial \alpha_1}{\partial r} \times \frac{\partial \alpha_2}{\partial r},$$

where $K_1 > 0$ and $K_2 > 0$.

To ensure that the magnitude of the velocity vector is $V$, normalize $u'$ as

$$u = V \frac{u'}{\|u'\|}.$$

The commanded flight-path angle $\gamma^c$, and the desired heading angle $\psi^d$ are therefore given by

$$\gamma^c = -\text{sat}_{\bar{\gamma}} \left[ \sin^{-1} \left( \frac{u_3}{V} \right) \right]$$

$$\psi^d = \text{atan2}(u_2, u_1).$$

3D Vector Field Path Following

Straight-line Paths

\[
\alpha_{\text{lon}}(r) \triangleq n_{\text{lon}}(r - c_\ell) = 0
\]
\[
\alpha_{\text{lat}}(r) \triangleq n_{\text{lat}}(r - c_\ell) = 0.
\]

Parameterized by

- Start Position \( c_\ell \)

3D Vector Field Path Following

Helical Paths

\[ \alpha_{cyl}(r) = \left( \frac{r_n - c_n}{R_h} \right)^2 + \left( \frac{r_e - c_e}{R_h} \right)^2 - 1 \]

\[ \alpha_{pl}(r) = \left( \frac{r_d - c_d}{R_h} \right) + \frac{\tan \gamma_h}{\lambda_h} \left( \tan^{-1} \left( \frac{r_e - c_e}{r_n - c_n} \right) - \psi_h \right) . \]

Parameterized by
- Spiral Center \((c_n, c_e, c_d)\)
- Radius \(R_h\)
- Direction \(\lambda_h\)
- Initial heading \(\psi_h\)

Outline

• Control architecture
• Low level autopilot loops
• Path following
• Dubins airplane paths and path management
Dubins path or waypoints → path planer

straight-line or spiral path → path manager

airspeed, altitude, heading, commands → path following

servo commands → autopilot

wind → unmanned aircraft

destination, obstacles → map

status → tracking error

position error → state estimator

on-board sensors → \( \hat{x}(t) \)
Options for Dubins car

For Dubins airplane, also need to address altitude
Low altitude case: \(|z_{de} - z_{ds}| \leq L_{car}(R_{\text{min}}) \tan \bar{\gamma}\),
Can achieve altitude gain without modifying Dubins car path.
High altitude case: \[|z_{de} - z_{ds}| > [L_{car}(R_{\text{min}}) + 2\pi R_{\text{min}}] \tan \bar{\gamma}.\]
Can achieve altitude gain by spiraling at beginning or end of Dubins car path.
Medium altitude case: $L_{\text{car}}(R_{\text{min}}) \tan \bar{\gamma} < |z_{\text{de}} - \bar{z}_{ds}| \leq [L_{\text{car}}(R_{\text{min}}) + 2\pi R_{\text{min}}] \tan \bar{\gamma}$

To achieve altitude gain, must add path deviation.
Path Management

Follow start helix

Cross $\mathcal{H}_s$

Intermediate arc at beginning

yes → Follow intermediate arc

no → Cross $\mathcal{H}_i$

Follow straight line

Cross $\mathcal{H}_s$

Intermediate arc at end

yes → Follow intermediate arc

no → Cross $\mathcal{H}_i$

Follow end helix

Cross $\mathcal{H}_e$

END
Summary

• **Autopilot**
  – Successive Loop Closure for Lateral Control
  – Total Energy Control System for Longitudinal Control

• **Path Following**
  – 3D Vector field method

• **Path Management**
  – Dubins airplane paths

• Methods are computationally simple, and easily fit on embedded processors.

• Methods have flown extensively on small fixed wing UAS.
Questions?