Development and Application of a Dynamic Obstacle Avoidance Algorithm for Small Fixed-Wing Aircraft with Safety Guarantees

APPENDIX

Dennis J. Marquis & Mazen Farhood

Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061, USA

Appendix A. Testbed and System Identification

The CZ-150 is equipped with the Here3 high precision GNSS module, a TFRPM01 tachometer sensor to measure motor rotation, and a MATEKSYS digital airspeed sensor. The Cube Orange is the primary autopilot responsible for input/output tasks, navigation and control logic, and redundancy management. The more computationally demanding navigation and control algorithms are stored on the onboard Raspberry Pi. These algorithms are implemented in Robotic Operating System (ROS), which communicates with PX4 over a serial connection using the MAVROS interface. Custom modular ROS nodes have been written in C++, corresponding to the 3D motion planner, trajectory-tracking controllers, virtual obstacles, and steady wind estimator. The CZ-150 communicates with a ground station running QGroundControl. Additionally, the ground station includes a Real-Time Kinematic positioning (RTK) base station for improved aircraft position estimates.

An aerodynamic model for the CZ-150 has been developed using the Multivariate Orthogonal Function (MOF) technique and the Output Error (OE) method [1], based on flight test data capturing the aircraft's response to multisine actuator excitation. To ensure the model's validity for the aggressive maneuvers expected within the motion planner's intended flight envelope, the training dataset includes approximately 300 seconds of flight data, capturing a wide range of actuator inputs from test flights conducted in calm wind conditions. Table A.1 summarizes the range of key aircraft states within the training dataset. The aerodynamic model structure is determined to be the structure presented in Eq. (2) of the main manuscript. To determine static actuator mappings for the elevator, ailerons, and rudder of the CZ-150, we apply constant PWM inputs spaced between the upper and lower saturation limits to each control surface and measure the resulting deflection using a protractor. For throttle, we compare the measured motor rotation from the tachometer sensor to the recorded input while the sUAS is subjected to constant throttle inputs spanning the idle to maximum stick positions. The time constants τ_i for $i = E, A, R, T$ in Eq. (3) of the main manuscript are identified using the OE method with flight test data collected during system identification.

The accuracy of the aerodynamic model in the desired operational envelope is tested by performing both manual maneuvers (e.g., multistep inputs, coordinated turns) and automated maneuvers (e.g., multisine actuator inputs), then comparing the measured aircraft response to the response predicted by the aerodynamic model. Figure A.1 depicts a comparison between the measurements and the aerodynamic model prediction for a manual multistep maneuver consisting of pitch inputs and aggressive roll inputs. The angular rates, airspeed, and wind angles predicted by the model track the measured values. The accuracy of the model over the full set of validation maneuvers can be quantitatively summarized in terms of the Theil inequality coefficient (TIC). For a vector of N_m state measurements $\mathbf{z}$ and a vector of N_m predicted aerodynamic model outputs $\mathbf{y}$, TIC is defined as

$$\text{TIC} = \frac{\sqrt{\frac{1}{N_m}(\mathbf{z}-\mathbf{y})^T(\mathbf{z}-\mathbf{y})}}{\sqrt{\frac{1}{N_m}\mathbf{z}^T\mathbf{z}} + \sqrt{\frac{1}{N_m}\mathbf{y}^T\mathbf{y}}},$$

where a value of 0 indicates a perfect fit. Table A.2 displays the TIC values obtained over the set of validation maneuvers. A TIC value below 0.3 for each output is considered good agreement between the measurements and aerodynamic model [2], and the TIC results meet this criteria.

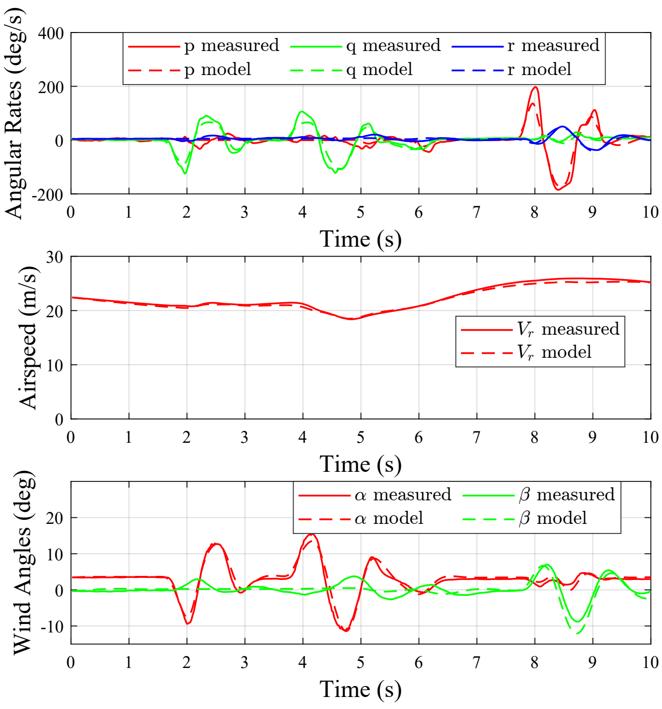

Figure A.1: An aerodynamic model validation maneuver.

Table A.1: CZ-150 Training Dataset Flight Envelope

State	Minimum	Maximum
p (deg/s)	-196	217
q (deg/s)	-117	106
r (deg/s)	-55	54
V_r (m/s)	15.6	29.6
α (deg)	-9.75	13.17
β (deg)	-11.84	10.92
ϕ (deg)	-76.28	70.15
θ (deg)	-27.06	37.29

Table A.2: TIC Values for CZ-150 System Identification

Output	TIC	Output	TIC	Output	TIC
p	0.193	q	0.170	r	0.213
V_r	0.007	α	0.097	β	0.278
a_x	0.205	a_y	0.181	a_z	0.122

Appendix B. UIC Controller Synthesis

This subsection provides the semidefinite programs (SDPs) that are solved during the UIC controller synthesis procedure, summarized from [3]:

1. Find $\xi^{\text{uic}}_{\min}$, the scalars s_1, s_2, s_3, s_4, and s_5 and the matrix sequences R_k and S_k for $k = 0, 1, \ldots, N$ by solving the SDP

$$\begin{aligned}
&\text{minimize} \quad \xi^{\text{uic}} \\
&\text{subject to} \quad s_1 + s_2 + s_3 < 2\xi^{\text{uic}}, \quad \Lambda^T S_0 \Lambda \prec s_2 \mathbf{I}, \\
&\begin{bmatrix} s_4 & 1 \\ 1 & s_3 \end{bmatrix} \geq \mathbf{0}, \begin{bmatrix} s_5 & 1 \\ 1 & s_1 \end{bmatrix} \geq \mathbf{0}, \begin{bmatrix} R_k & \mathbf{I} \\ \mathbf{I} & S_k \end{bmatrix} \geq \mathbf{0}, \\
&\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}^T \left(M \begin{bmatrix} R_k & \mathbf{0} \\ \mathbf{0} & s_4 \mathbf{I} \end{bmatrix} M^T - \begin{bmatrix} R_{k+1} & \mathbf{0} \\ \mathbf{0} & s_1 \mathbf{I} \end{bmatrix} \right) \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \prec 0, \\
&\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^T \left(M^T \begin{bmatrix} S_{k+1} & \mathbf{0} \\ \mathbf{0} & s_5 \mathbf{I} \end{bmatrix} M - \begin{bmatrix} S_k & \mathbf{0} \\ \mathbf{0} & s_3 \mathbf{I} \end{bmatrix} \right) \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \prec 0,
\end{aligned}$$

for $k = 0, 1, \ldots, N$, where

$$R_{N+1} = R_N, \quad S_{N+1} = S_N, \quad M = \begin{bmatrix} A & B_1 \\ C_1 & D_{11} \end{bmatrix},$$

$$\text{Im} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \text{Ker} \begin{bmatrix} B_2^T & D_{12}^T \end{bmatrix}, \quad \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}^T \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \mathbf{I},$$

$$\text{Im} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \text{Ker} \begin{bmatrix} C_2 & D_{21} \end{bmatrix}, \quad \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^T \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \mathbf{I},$$

and Im Q and Ker Q denote the image space and kernel of a linear mapping Q, respectively.

2. Find $L_k = \begin{bmatrix} A_k^K & B_k^K \\ C_k^K & D_k^K \end{bmatrix}$ for $k = 0, 1, \ldots, N$ by solving the SDP

$$H_k + \tilde{Q}^T L_k^T \tilde{P} + \tilde{P}^T L_k \tilde{Q} \prec 0,$$

for $k = 0, 1, \ldots, N$, where

$$H_k = \begin{bmatrix} H_k^{(1)} & H_k^{(2)} \\ \left(H_k^{(2)}\right)^T & H_k^{(3)} \end{bmatrix},$$

$$H_k^{(1)} = \begin{bmatrix} -R_{k+1} & -E_{k+1} & A \\ -E_{k+1}^T & -\mathbf{I} & \mathbf{0} \\ A^T & \mathbf{0} & -S_k \end{bmatrix}, \quad E_k = \left(R_k - S_k^{-1}\right)^{\frac{1}{2}},$$

$$H_k^{(2)} = \begin{bmatrix} \mathbf{0} & s_3^{-1/2} B_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ S_k E_k & \mathbf{0} & s_1^{-1/2} C_1^T \end{bmatrix},$$

$$H_k^{(3)} = \begin{bmatrix} -\mathbf{I} - E_k^T S_k E_k & 0 & 0 \\ 0 & -\mathbf{I} & (s_1 s_3)^{-1/2} D_{11}^T \\ 0 & (s_1 s_3)^{-1/2} D_{11} & -\mathbf{I} \end{bmatrix},$$

$$\tilde{P} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ B_2^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & s_1^{-\frac{1}{2}} D_{12}^T \end{bmatrix},$$

$$\tilde{Q} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_2 & \mathbf{0} & s_3^{-\frac{1}{2}} D_{21} & \mathbf{0} \end{bmatrix}.$$

Solving this feasibility problem yields a realization of the eventually time-invariant controller, represented by the sequence $(L_0, L_1, \ldots, L_N, L_N, \ldots)$. Well-conditioned controller matrices can be obtained by minimizing the spectral norm $\|L_k\|_2$ for $k = 0, 1, \ldots, N$.

Appendix C. Alternative LPV Formulation

This subsection discusses the linear parameter-varying (LPV) control scheme, an alternative that is investigated alongside the switched UIC control scheme. In this work, the LPV model of the nonlinear 6DOF dynamics is parameterized by both the curvature κ and FPA γ, extending prior formulations that used only κ as the scheduling parameter. This is a logical control strategy for the given motion primitive library, as each primitive is associated with a specific trim point corresponding to a value of (κ, γ).

To apply the synthesis approach from [4], the LPV model must be expressed as a linear fractional representation (LFR), which requires that the state-space matrix-valued functions have rational dependence on the scheduling parameters (κ, γ). The type of this dependence, whether linear, polynomial, or, more generally, rational, affects both the size of the LFR and the computational complexity of the synthesis problem. A polynomial model with terms up to second order is found to offer a reasonable trade-off between model accuracy and tractability. The following trim states are selected to be parameter-dependent: $p, q, r, \phi, \theta\, \delta_E, \delta_T$. Their trim fits are

$$\begin{aligned}
p_{\text{trim}}(\kappa, \gamma) &= -1.1476\,\kappa - 32.5405\,\kappa^2 - 20.3307\,\kappa\gamma, \\
q_{\text{trim}}(\kappa, \gamma) &= 726.0481\,\kappa^2, \\
r_{\text{trim}}(\kappa, \gamma) &= 16.7388\,\kappa, \\
\phi_{\text{trim}}(\kappa, \gamma) &= 38.6736\,\kappa, \\
\theta_{\text{trim}}(\kappa, \gamma) &= 0.0569 + 0.9844\,\gamma, \\
\delta_{\text{E,trim}}(\kappa, \gamma) &= 0.0654 - 128.0343\,\kappa^2, \\
\delta_{\text{T,trim}}(\kappa, \gamma) &= 0.5349 + 1.2584\,\gamma - 2.0742\,\gamma^2,
\end{aligned}$$

and the performance vector is chosen as

$$\mathbf{z}^{\text{lpv}} = \left[0.01\bar{\omega}^T \; 0.1\bar{V}_r \; 5\bar{\theta} \; 1\bar{\psi} \; 5\bar{\mathbf{p}}^T \; 50\bar{\delta}_E \; 10\bar{\delta}_A \; 10\bar{\delta}_R \; 10\bar{\delta}_T\right]^T .$$

Including γ as an additional scheduling variable significantly increases the dimension of the LFR. A minimal LFR realization is obtained using `mingss` from the SMAC [5] toolbox in MATLAB, which implements a variant of the Kalman decomposition. Then, an LPV $\mathcal{H}_\infty$ controller is synthesized by minimizing the ℓ_2-gain performance level $\xi^{\text{lpv}}_{\text{min}}$, which is an upper bound on the induced gain

$$||\mathbf{d} \to \mathbf{z}||_{\ell_2 \to \ell_2} = \sup_{||\mathbf{d}||_{\ell_2} \leq 1} ||\mathbf{z}||_{\ell_2},$$

for all permissible parameter trajectories. The synthesis procedure is summarized in Section 4.2 of [6].

With the current parameterization and performance output, the resulting LPV controller is found to under-perform in simulation relative to the switched UIC controller. For instance, when tracking a trim point corresponding to $(\kappa, \gamma) = (-0.02, 0.21)$ while subjected to measurement noise and light wind, without aerodynamic model uncertainty, the mean position error is approximately 10 m, substantially larger than the error observed with the switched UIC controller under identical conditions, which is less than 2 m. While adjustments to the set of parameter-dependent variables, the structure of the trim fit, or the tuning of performance output weights could potentially yield acceptable tracking performance, the effectiveness of the switched UIC controller has eliminated the need to further pursue the LPV approach.

Appendix D. ROS and PX4 Implementation Considerations

Each component of the system is implemented as a separate ROS node. To enable communication between these nodes, custom ROS messages are created and published by each node, allowing other nodes to subscribe to relevant information. For instance, the ROS Obstacle Node publishes the obstacle state, which is accessible to the Motion Planner Node, while the Motion Planner Node broadcasts a reference trajectory for the Controller Node to track. All nodes requiring real-time access to the sUAS state subscribe to standard MAVROS topics, as PX4 estimates the state variables $(\mathbf{p}, \mathbf{v}, \boldsymbol{\theta}, \omega)$ using its default sensor fusion techniques. Several factors contribute to delays within the system, including the time required to serialize and broadcast ROS messages and the processing time of computationally intensive algorithms. When significant delays occur, the controller may experience overshoot, resulting in unstable tracking behavior. To mitigate these delays and meet the expected end-to-end latency of the system (approximately 40 ms, or one sampling period), the following actions are taken:

1. **Baud Rate:** The baud rate for the serial connection between the Cube Orange and Raspberry Pi is increased from the default 115200 to 921600, enabling higher-frequency message transmission.
2. **Message Optimization:** Several unnecessary MAVROS message topics are disabled by editing the `plugin.yaml` file. For those topics that can not be completely disabled, the publish rate is reduced by modifying the MAVROS `px4.launch` file.
3. **Custom Message Size:** For custom messages containing large data sets, such as the trajectory generated by the Motion Planner Node, the messages are broadcast at high frequency, one trajectory point at a time. The subscribing node then reassembles the trajectory, avoiding the computational overhead of transferring the entire trajectory at once.
4. **Jacobian Computations:** The ATEKF implementation requires the computation of Jacobians at each time step. While the finite difference method is quick to implement and flexible with respect to the dynamic model, symbolic differentiation offers a more computationally efficient solution at the cost of increased complexity in storing and loading the necessary formulas. Therefore, the finite difference method is ideal for model iteration, whereas symbolic differentiation is preferable for performance-critical applications once the final model is established.

In addition to reducing delay, several other practical considerations arise when implementing the ATEKF and UIC controller:

1. **Error Wrapping for ψ:** The error computation for ψ in the linear controllers and the estimate of ψ in the ATEKF must be wrapped to the interval $[-\pi, \pi]$. While the `fmod()` function in C++ can be used for this purpose, it wraps the value to the range $[0, 2\pi]$. The formula $\psi_{\text{wrap}} = \texttt{fmod}(\psi + \pi,\, 2\pi) - \pi$ can be used to ensure the correct interval $[-\pi, \pi]$ is maintained.
2. **Coordinate Frame Conventions:** MAVROS publishes local coordinate frames using the East-North-Up (ENU) convention, whereas PX4 and the literature typically use North-East-Down (NED). Additionally, body frame orientations are published in the Front-Left-Up (FLU) convention, which must be converted to Front-Right-Down (FRD) to align with the sUAS dynamic model.
3. **Servo Direction and Polarity:** Depending on the testbed, some servos may respond in the opposite direction when the PWM signal increases. This polarity can be inverted at the transmitter level, within PX4 by modifying the corresponding parameter (e.g., PWM_MAIN_REV1 for the Main 1 connection), or compensated for in the custom C++ code. The polarity must be addressed when designing the actuator model to ensure consistency with the model obtained during system identification.
4. **Matrix Inversion Stability:** In C++, the Eigen library is often used to handle matrices and vectors, similar to MATLAB. Although the library provides an `inverse()` method, we find that the computation $C = A \setminus B = A^{-1}B$ is more numerically stable when using the formula `C = A.colPivHouseholderQr().solve(B)`.
5. **Airspeed Conversion:** MAVROS publishes calibrated airspeed but does not automatically convert it to true airspeed, which accounts for altitude. This conversion must be done manually. While this has not been found to significantly

impact ATEKF stability, failing to perform this correction results in a lower estimate of airspeed.

6. **Error Handling:** Many MAVROS topics may not publish valid data when certain values are unavailable. For example, if there is no valid home position, the position variable will be 'NaN' in C++. Robust error handling must be implemented to ensure that nodes continue operating, even when some calculations cannot be performed in every iteration. Another important aspect is the divergence of the ATEKF; the algorithm should be capable of detecting divergence and resetting, similar to the behavior of standard PX4 estimators.
7. **Handling Noisy Measurements:** Some MAVROS messages, particularly those related to IMU data, are prone to noise, occasional message drops, or outliers. For system identification, these signals must be smoothed using a low-pass filter, and outliers should be eliminated. For real-time applications, the challenge is greater, but it has been observed that reducing the performance weights with respect to angular rate data can improve tracking, as this makes the controller less sensitive to high-frequency fluctuations.

References

[1] V. Klein and E. A. Morelli, *Aircraft System Identification: Theory and Practice*, ser. AIAA education series. AIAA, 2006.

[2] R. V. Jategaonkar, *Flight Vehicle System Identification: A Time-Domain Methodology, Second Edition*. AIAA, Inc., 2015.

[3] M. Farhood and G. E. Dullerud, "Control of systems with uncertain initial conditions," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2646–2651, 12 2008.

[4] A. Packard, "Gain scheduling via linear fractional transformations," *Systems & Control Letters*, vol. 22, no. 2, pp. 79–92, 1994.

[5] C. Roos, "Systems modeling, analysis and control (SMAC) toolbox: An insight into the robustness analysis library," in *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*, 2013, pp. 176–181.

[6] D. Muniraj, M. C. Palframan, K. T. Guthrie, and M. Farhood, "Path-following control of small fixed-wing unmanned aircraft systems with H_∞ type performance," *Control Engineering Practice*, vol. 67, no. 2, pp. 76–91, 10 2017.