UAV Crosswind Tail Stabilizer

Dan Binyamin[1], Dov Liudmirsky[1], Danniel Stoller[1], Shabtay Bilu[1]
[1]Department of Mechanical Engineering, SCE, Shamoon College of Engineering, Beer Sheva, Israel
danbinyamin13@gmail.com, dov11liud@gmail.com, dannist@ac.sce.ac.il, shaykebilu@gmail.com

***Abstract*—Landing UAVs in crosswinds is a high-risk phase; sudden yaw moments often cause runway excursions or structural failure. This paper develops an autonomous active tail stabilization system to maintain the aircraft's heading during crosswind landings. Using an MPU6050 IMU and an ESP32 dual-core microcontroller, the system detects rapid yaw disturbances and applies immediate proportional rudder corrections with sub-20ms response latency. The dual-core architecture separates real-time flight control from continuous data logging, preventing delays that compromise stabilization. This bridges the gap between high-frequency wind dynamics and the slower human reaction time. A hardware fail-safe switcher ensures manual override capability. Results demonstrate that autonomous tail control effectively dampens yaw oscillations, significantly enhancing landing safety and reducing accident risk during adverse meteorological conditions.**

I. Introduction

The landing phase is the most critical stage of any flight mission, requiring precise maintenance of approach path and runway centerline while managing kinetic and potential energy. Crosswind—the component of wind perpendicular to the flight path—is among the most significant disturbances: between 1994 and 2003, wind-related factors contributed to 2,726 of 4,159 weather-related aviation accidents [1], [2]. The problem is compounded by the limitations of existing automation; for example, the Airbus A320 autoland system is certified only to 20 knots crosswind, whereas a skilled manual pilot can handle 35 knots [3]. Wind shear has been identified as the single greatest cause of air crashes in the United States [4], a risk further characterized by NASA studies linking turbulence and wind phenomena to loss of control [5].

For large transport aircraft, mechanical solutions such as steerable main landing gear [6], [7] can assist crosswind operations, but their mass and complexity make them infeasible for small unmanned aerial vehicles (UAVs). Prior work on autonomous fixed-wing crosswind landing [8] demonstrates that closed-loop correction at the acceleration level achieves accurate touchdowns, but does not address the embedded real-time implementation challenges on a low-cost platform. This work differs from prior crosswind landing research by focusing not on high-level control synthesis, but on a fully embedded, low-cost real-time stabilization system with hardware-level redundancy, validated in real flight under strong gust conditions. A pilot/meteorologist survey conducted as part of this research (n = 40, mean crosswind criticality score 4.15/5) and preliminary manual flight tests confirm that human reaction latency—typically 300–500 ms—is the primary bottleneck: in controlled trials at 13 km/h crosswind, late corrections caused touchdown with residual lateral drift, imposing dangerous side loads on the landing gear.

The central physical mechanism driving these disturbances is the weathercock yawing moment

$$N = \tfrac{1}{2}\rho V^2 S b\, C_{n_\beta}\, \beta, \tag{1}$$

where $C_{n_\beta} > 0$ for positive static stability. While beneficial in cruise, this tendency opposes heading alignment with the runway during landing and must be actively countered by the rudder.

Contributions

This paper makes the following contributions to embedded UAV control:

- A **low-cost autonomous yaw stabilization system** (<$15 bill-of-materials) for small UAVs, validated in real crosswind flight.
- A **dual-core FreeRTOS control architecture** that provably separates real-time rudder actuation (Core 1, 50 Hz) from SD-card logging (Core 0), achieving sub-20 ms response without scheduler conflict.
- A **hardware-level fail-safe** using a solid-state PWM switcher, providing pilot override completely independent of processor state.
- **Quantitative flight validation** under realistic gust conditions, including a direct manual-vs-autonomous statistical comparison of yaw deviation metrics.

II. System Design and Methodology

A. Design Requirements

Three primary requirements drove the system architecture. First, the control loop must respond in under 20 ms to capture gust dynamics before significant heading deviation accumulates, as characterized by the F-factor wind-shear hazard index [9]. This immediately rules out a single-core architecture where SD-card writes introduce deterministic 20 ms blocking delays. Second, the system must integrate into the critical rudder-actuator path, which demands hardware-level redundancy: a software-only override is unusable if the processor freezes [1]. Third, total added mass must remain negligible (<30 g) so as not to shift the center of gravity of a small UAV platform.

B. Test Platform

Flight tests used the Javelin (Old School Model Works), a high-wing RC aircraft with 1.52 m wingspan, 1.24 m fuselage length, and 2.6 kg all-up weight, powered by an O.S. 55 AX

two-stroke glow engine (9 cm^3) driving a 12-inch propeller. The high-wing configuration was selected because locating the CG below the center of lift produces a passive pendulum restoring moment in roll [10], allowing the active controller to focus exclusively on the yaw axis. The high thrust-to-weight ratio of the nitro engine enables immediate energy recovery under wind shear [9]. A large top-access service bay permits installation of the stabilization module at the CG, minimizing IMU lever-arm errors as specified in [8].

Fig. 1: Selected test aircraft (Javelin high-wing RC model).

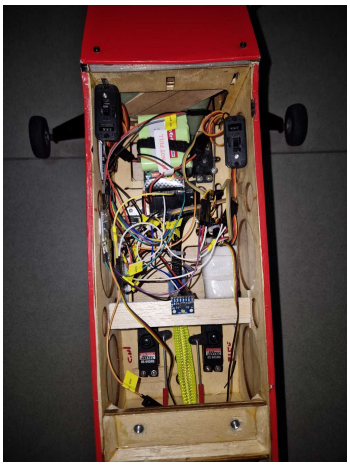

Fig. 2: Stabilization module installed at the CG inside the fuselage.

C. Hardware Architecture

Table I summarizes the microcontroller trade study. The ESP32 DevKit V1 (dual-core, 240 MHz) was chosen because it is the only candidate that can execute SD-card writes on a dedicated core without stalling the control loop; the Raspberry Pi Zero, though faster, offers non-deterministic OS scheduling unsuitable for hard real-time control.

The IMU is an MPU6050 (6-DOF gyro + accelerometer, I^2C), selected over the BNO055 and ADXRS610 for its raw data access and low cost ($3). Internal fusion on the BNO055 introduces a latency penalty incompatible with the 20 ms budget; the MPU6050's digital motion processor (DMP) allows custom filtering without that penalty. For fail-safe switching, a solid-state PWM video switcher (Table II) replaces both software-only override and mechanical relay options. It is fully processor-independent, vibration-resistant, and adds only $\sim$2 g.

The electrical architecture uses two isolated power rails to prevent servo back-EMF from causing brownout on the logic bus: Battery 1 (4.8 V NiMH) powers the ESP32 and MPU6050 exclusively, while Battery 2 (6.6 V LiFe) powers the Futaba receiver, PWM switcher, and Hitec HS-645MG tail servo. The parallel (bypass) architecture is shown in Fig. 3: even under processor failure the switcher maintains an independent path returning control to the pilot.

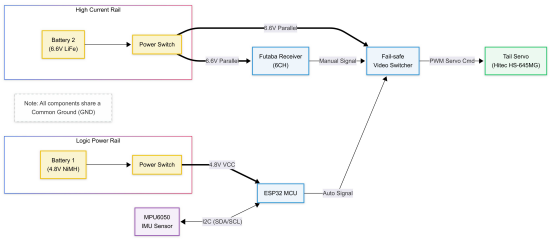

Fig. 3: Control system in parallel (bypass) configuration. The PWM switcher routes either the autonomous or the manual signal to the servo, independent of processor state.

D. Control Law and Gain Tuning

The controller is a proportional rate-damping law acting on the filtered yaw rate:

$$u = K_D \cdot \dot{\psi}_{\text{filtered}}, \tag{2}$$

where u is the rudder servo command increment (PWM units) and $\dot{\psi}_{\text{filtered}}$ is the yaw rate after exponential low-pass filtering with an exponential smoothing coefficient $\alpha_f = 0.3$. A deadband of $\pm 4.0\,^\circ$/s removes engine-vibration noise below the threshold of aerodynamically meaningful disturbances.

Unlike classical heading control problems, which require proportional feedback to eliminate steady-state error, crosswind disturbances during landing are predominantly transient and oscillatory. Stabilization is therefore achieved through active damping of yaw rate rather than angle regulation, making a pure rate-damping law both sufficient and physically appropriate. Furthermore, adding a proportional heading term would conflict with the pilot's intentional crab-angle commands and with the airframe's own weathercock restoring moment ($C_{n_\beta} > 0$, Eq. 1), which already provides passive proportional yaw stability [11, Ch. 5,6].

Gain selection. K_D was determined empirically using a two-stage process. In ground tests, the rudder servo was commanded at known deflections while yaw rate was measured under static engine run-up, establishing the actuator gain (PWM units per degree of deflection). In flight, K_D was swept from an initial conservative estimate upward in increments

TABLE I: Comparison of candidate microcontrollers

Criterion	Arduino Nano	Raspberry Pi Zero	ESP32 (chosen)
Architecture	8-bit, single core	32-bit, single core	32-bit, dual core
Clock speed	16 MHz	1 GHz	240 MHz
SD-card writing	Blocks processor (∼20 ms)	Fast but non-deterministic	Background (separate core)
Operating voltage	5 V	3.3 V	3.3 V
Estimated cost	∼$5	∼$25	∼$6

TABLE II: Comparison of fail-safe mechanisms

Criterion	Software override	Mech. relay	Video Switcher (chosen)
Operation	Code-specific command	Electromechanical	Solid-state transistor
Processor dependence	Full (freezes on failure)	Depends on processor	Completely independent
Vibration resistance	Excellent	Low (contacts shift)	Excellent (no moving parts)
Weight	–	Heavy (∼20–50 g)	Minimal (∼2 g)

of 10 % while monitoring for overshoot in the logged servo-vs-yaw-rate traces (Fig. 8). The final value $K_D = 1.2$ PWM-units·s/° was the highest gain that produced zero overshoot across all five test approaches; the linear fit in Fig. 8 (Pearson $r = 1.00$) confirms that the operating point remains well within the proportional regime with no saturation. A formal root-locus or LQR analysis is reserved for future work on higher-order controllers.

The software runs under FreeRTOS with Core 1 pinned to the control task (50 Hz, hard deadline) and Core 0 pinned to the logging task. Inter-core communication uses a lock-free ring buffer to prevent priority inversion. Flight data (timestamp, receiver commands, IMU readings, servo commands) are stored as CSV for MATLAB post-processing.

III. Experimental Results

A. Baseline: Manual Flight Characterization (Test 1)

Test 1 characterized manual landings at ∼13 km/h crosswind (crosswind component 9.2 km/h). Video analysis revealed three distinct failure modes of manual control. First, a **phase lag** of 0.3–0.5 s was observed between each gust onset and the pilot's corrective stick input—the physiological delay required to visually detect the deviation and move the stick. Second, this lag caused **pilot-induced oscillations (PIO)**: by the time a correction arrived, the original gust had already passed, so the late rudder input pushed the aircraft in the opposite direction, creating a secondary disturbance that required a further correction. The pilot was effectively fighting the aircraft. Third, **frequency mismatch** was apparent: the pilot's corrections were broad, slow, low-frequency inputs, while the wind environment contained sharp high-frequency gusts. The human operator acts as a physiological low-pass filter and cannot track or cancel fast micro-disturbances. In the third landing, this combination caused touchdown with residual lateral drift, imposing dangerous side loads on the landing gear. The key characteristic of manual control under crosswind is therefore **low correction frequency at high amplitude**: few large late inputs, each arriving after the disturbance has already grown to a threatening magnitude and risking PIO with every attempt to recover.

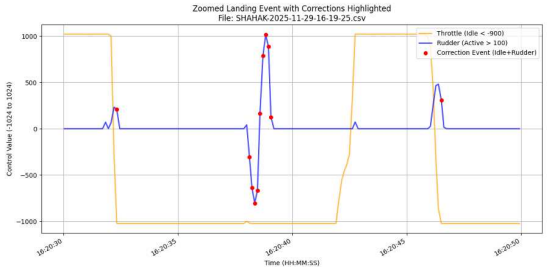

Fig. 4: Throttle versus rudder during manual landing (Test 1). High-frequency rudder activity persists throughout the idle/glide phase.

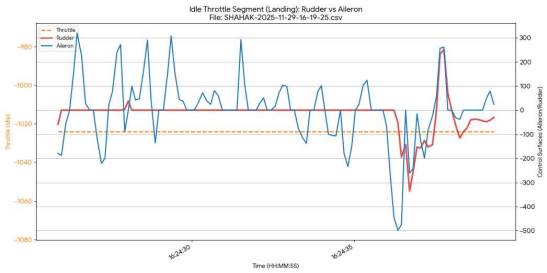

Fig. 5: Rudder versus aileron at engine idle, confirming yaw-roll cross-coupling during manual correction.

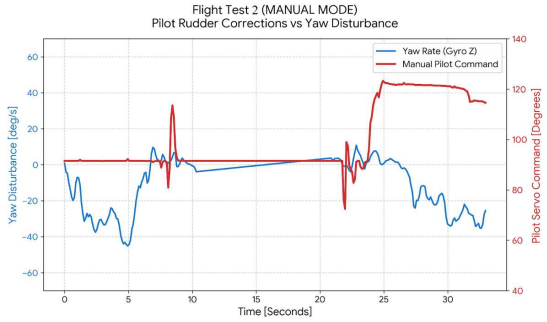

Fig. 6: Manual landing (Test 1): yaw-rate disturbance (blue) vs. pilot rudder command (red), showing the 0.3–0.5 s phase lag and resulting PIO cycles.

B. Autonomous Flight Performance (Test 2)

Five autonomous approaches were performed in Auto mode (average wind 18 km/h, gusts to 35.3 km/h, prevailing azimuth 229.7° SW relative to north-south runway at Mevo Horon). The effective crosswind component was

$$V_{\text{cw}} = V_{\text{wind}} \sin(\alpha), \quad \alpha = 130.3^\circ,$$

yielding a steady crosswind of 13.7 km/h and peak gust component of 26.9 km/h. Wind conditions are shown in Fig. 7.

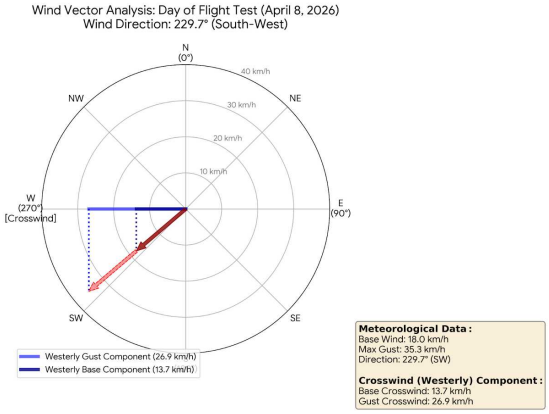

Fig. 7: Wind data (Visual Crossing Weather, 8 April 2026). Prevailing direction 229.7° (SW); gusts to 35.3 km/h.

The system detected and corrected 214 yaw-rate events exceeding the 10°/s threshold, with a peak measured yaw rate of 59.1°/s. Every disturbance was addressed within a single 20 ms control cycle. This represents the inverse control signature of manual flight: **high correction frequency at low amplitude**. Rather than waiting for yaw error to grow until it demands a large aggressive input, the system intervenes at the onset of each disturbance with a small proportional correction, preventing error accumulation entirely.

Fig. 14 illustrates a further benefit observed during autonomous operation: **flight-axis decoupling**. The red trace shows the pilot's aileron (roll) activity, which remained high throughout the approach as the pilot managed wing levelling against the crosswind. The blue trace shows the autonomous rudder commands generated by the ESP32. The two signals are independent and simultaneous, demonstrating that the system transparently managed the yaw axis while the pilot focused exclusively on roll and pitch. Without the autonomous yaw channel, the pilot would have been required to coordinate all three axes simultaneously under turbulence—a task saturation scenario that is a primary cause of crosswind accidents. The system also prevented **adverse yaw**: sharp aileron inputs without matched rudder correction generate a yawing moment that can cause asymmetric tip stall at low altitude; the continuous 50 Hz rudder corrections neutralised this effect in real time.

C. Manual vs. Autonomous Comparison

Table III reveals a counter-intuitive but physically important contrast. The autonomous system performed *more* corrections than the manual pilot (214 detected events vs. infrequent

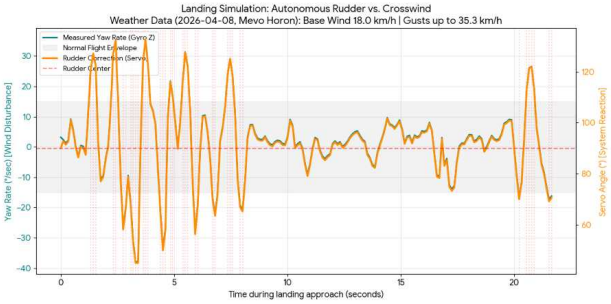

Fig. 8: Yaw-rate disturbance (teal) vs. autonomous rudder correction (orange) over a full landing approach. The servo mirrors every gust immediately, confirming proportional rate-damping with no phase lag, no overshoot, and $K_D = 1.2$ PWM-units·s/°.

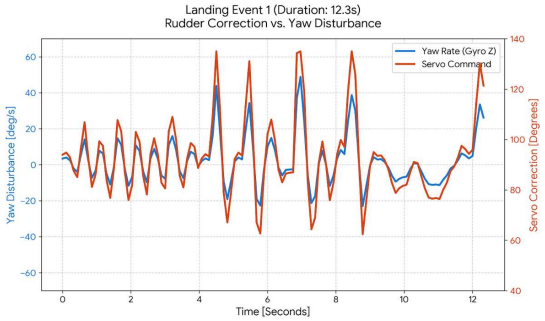

Fig. 9: Autonomous landing 1 (duration 12.3 s): yaw rate vs. servo command.

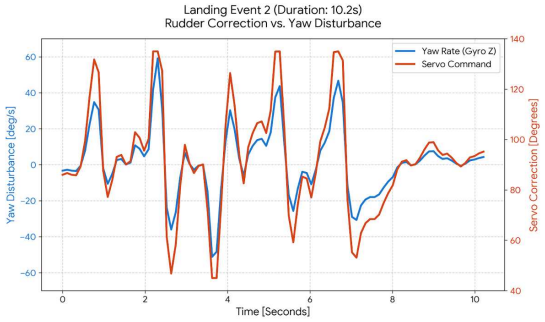

Fig. 10: Autonomous landing 2 (duration 10.2 s).

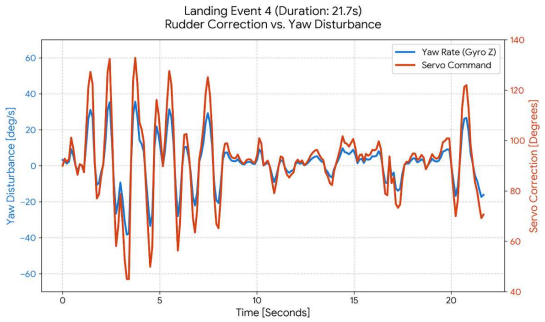

Fig. 11: Autonomous landing 3 (duration 13.2 s).

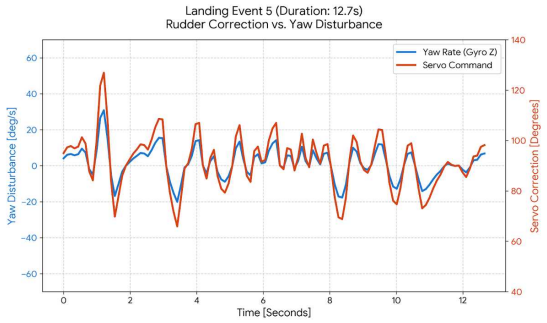

Fig. 12: Autonomous landing 4 (duration 21.7 s).

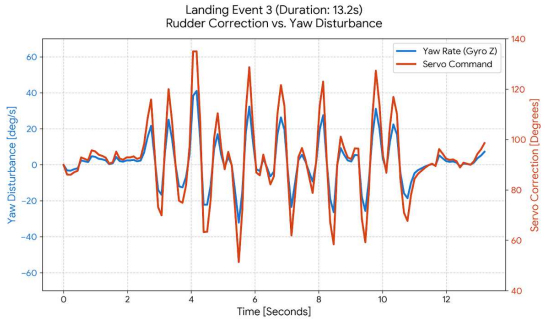

Fig. 13: Autonomous landing 5 (duration 12.7 s).

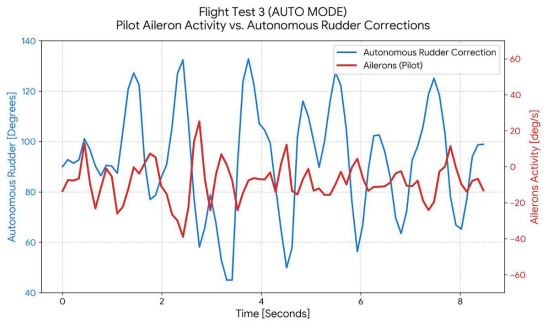

Fig. 14: Rudder versus aileron at engine idle, confirming yaw-roll cross-coupling during auto correction.

manual inputs), yet each correction was small, early, and proportional. Manual corrections were few but large—arriving only after the yaw disturbance had already grown to a threatening magnitude. **High correction frequency at low amplitude** is the signature of effective active damping; **low frequency at high amplitude** is the signature of task saturation and impending loss of control. This behavior indicates operation in a fundamentally different control regime, rather than a simple increase in response speed.

Beyond raw yaw metrics, the data demonstrate a qualitative safety benefit: by autonomously managing the yaw axis at 50 Hz, the system reduces the pilot's simultaneous control burden from three axes to two during the most critical seconds of landing. This **pilot workload reduction** directly addresses the task saturation mechanism responsible for the PIO events observed in Test 1.

TABLE III: Manual vs. autonomous yaw performance (landing window)

Metric	Manual	Auto	Improv.
Yaw-rate variance $(^\circ/s)^2$	500	<20	$-96\% \downarrow$
Peak yaw rate ($^\circ$/s)	<70	59.1	$-15\% \downarrow$
Correction count	low	214	$\uparrow$ freq.
Max single correction	large	proportional	amplitude $\downarrow$
Response latency (ms)	300–500	≈ 20	$>15\times$
Side-load events	1/3	0/5	—
Control axes (pilot)	3	2	1 offloaded

The autonomous controller eliminated all lateral drift events: no side-load touchdowns were observed across five autonomous landings, with a response latency more than 15 times shorter than the measured human threshold. Battery isolation and the hardware fail-safe completely eliminated servo jitter under high aerodynamic load. The chosen gain delivered smooth proportional resistance to disturbances without overshoot, as confirmed by the near-perfect linear correlation ($r = 1.00$ within measurement resolution) in Fig. 8.

IV. Conclusions

This paper demonstrated the feasibility and effectiveness of a low-cost autonomous yaw stabilization system for small UAVs landing in crosswind. By combining an MPU6050 IMU with a dual-core ESP32 and a hardware PWM fail-safe, the system achieves sub-20 ms proportional rate-damping control, over $15\times$ faster than human reaction time. Flight Test 2 (18 km/h mean wind, 35 km/h gusts, peak crosswind component 26.9 km/h) validated 214 corrected yaw events with a peak rate of 59.1°/s and zero overshoot. Direct comparison with manual flight confirms a consistent and substantial reduction in yaw-rate variance and elimination of side-load touchdown events. Beyond yaw variance metrics, the system achieves flight-axis decoupling: by handling the yaw channel autonomously, it reduces the pilot's simultaneous control workload from three axes to two, eliminating the pilot-induced oscillations and adverse yaw events observed in manual flight.

Future work will pursue formal gain scheduling, LQR/H_∞ synthesis for improved robustness at higher wind speeds, and full 6-DOF autonomous landing validated on larger UAV platforms.

Acknowledgment

The authors extend sincere gratitude to Sami Shamoon College of Engineering (SCE) for institutional support and computational resources. *AI use declaration: an AI language tool (Claude) was used to assist with language refinement and structural editing of this manuscript. All research design, data collection, analysis, interpretations, and conclusions are solely the work of the authors. The authors retain full responsibility for the accuracy, originality, and integrity of all content.*

REFERENCES

[1] C.-L. Lee and J.-G. Juang, "Aircraft landing control in wind shear condition," in *2011 International Conference on Machine Learning and Cybernetics*, vol. 3. IEEE, 2011, pp. 1180–1185.

[2] J. Theis, D. Ossmann, and H. Pfifer, "Robust autopilot design for crosswind landing," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 318–13 323, 2017.

[3] A. O'Connor, "Demonstration of a novel 3-d wind sensor for improved wind shear detection," Master's thesis, Technological University Dublin, 2018.

[4] IEEE Spectrum, "The menacing microburst: Wind shear, the single greatest cause of air crashes," *IEEE Spectrum*, June 1986.

[5] J. K. Evans, "An examination of aviation accidents associated with turbulence, wind shear and thunderstorm," NASA Langley Research Center, Tech. Rep. NASA/CR-2013-217989, 2013.

[6] D. Vechtel, "How future aircraft can benefit from a steerable main landing gear for crosswind operations," *CEAS Aeronautical Journal*, vol. 11, no. 2, pp. 417–429, 2020.

[7] D. Vechtel, U. M. Meissner, and K. U. Hahn, "On the use of a steerable main landing gear for crosswind landing assistance," *CEAS Aeronautical Journal*, vol. 5, no. 2, pp. 213–225, 2014.

[8] A. de Bruin and T. Jones, "Accurate autonomous landing of a fixed-wing unmanned aircraft under crosswind conditions," *IFAC-PapersOnLine*, vol. 49, no. 17, pp. 170–175, 2016.

[9] F. H. Proctor, D. A. Hinton, and R. L. Bowles, "A windshear hazard index," in *9th Conference on Aviation, Range, and Aerospace Meteorology*. American Meteorological Society, 2000.

[10] J. Lynch and et al., "Creating MATLAB simulations to model aircraft dynamics," University of Bristol, UK, 2019.

[11] R. C. Nelson, *Flight Stability and Automatic Control*, 2nd ed. McGraw-Hill, 1998.