

# VIMU: Effective Physics-based Realtime Detection and Recovery against Stealthy Attacks on UAVs

Yunbo Wang\*, Cong Sun\*<sup>✉</sup>, Qiaosen Liu\*, Bingnan Su\*, Zongxu Zhang\*, Michael Norris<sup>†</sup>, Gang Tan<sup>†</sup>, Jianfeng Ma\*

\* School of Cyber Engineering, Xidian University, China

<sup>†</sup> The Pennsylvania State University, University Park, PA, USA

Email: robertwang@stu.xidian.edu.cn, suncong@xidian.edu.cn, {man5336, gtan}@psu.edu

**Abstract**—Sensor attacks on robotic vehicles have become pervasive and manipulative. Their latest advancements exploit sensor and detector characteristics to bypass detection. Recent security efforts have leveraged the physics-based model to detect or mitigate sensor attacks. However, these approaches are only resilient to a few sensor attacks and still need improvement in detection effectiveness. We present VIMU, an efficient sensor attack detection and resilience system for unmanned aerial vehicles. We propose a detection algorithm, CS-EMA, that leverages low-pass filtering to identify stealthy gyroscope attacks while achieving an overall effective sensor attack detection. We develop a fine-grained nonlinear physical model with precise aerodynamic and propulsion wrench modeling. We also augment the state estimation with a FIFO buffer safeguard to mitigate the impact of high-rate IMU attacks. The proposed physical model and buffer safeguard provide an effective system state recovery toward maintaining flight stability. We implement VIMU on PX4 autopilot. The evaluation results demonstrate the effectiveness of VIMU in detecting and mitigating various realistic sensor attacks, especially stealthy attacks.

**Index Terms**—Cyber-Physical System, Unmanned Aerial Vehicle, Security, Sensor Attack, Attack Detection, Resilience

## 1. Introduction

Airborne drones rely on sensor measurements for navigation and flight control. The sensors continually transduce raw physical signals into digital forms that can be interpreted and operated by the controller software [9]. For instance, the autopilot controller uses gyroscope measurements to track changes in flight attitude. Autopilots use simple validation (e.g., majority voting) by default to isolate the faulty sensor. Although this proved efficient in handling hardware failures, this practice is being challenged by increasingly sophisticated sensor attacks.

Launching GPS spoofing attacks to take over robotic vehicles is straightforward [15], [28], and the evolved spoofing practices have been further facilitated by the widespread

low-cost software-defined radio platforms [21], [25]. Meanwhile, another primary sensor attack category targets the inertial measurement unit (IMU). An IMU consists of an accelerometer, a gyroscope, and an optional magnetometer. The accelerometer measures changes in velocity, whereas the gyroscope and magnetometer are critical to attitude control. If they report erroneous measurements, the drone will immediately lose control and crash [13], [27]. Hence, effective attacks have been demonstrated by tampering with the accelerometer or gyroscope readings using sound waves at a known resonant frequency of the target sensor [27], [30], [31] or remotely blocking the transmissions between IMU and flight controller through electromagnetic interference (EMI) [13].

Conventional security mechanisms, e.g., data encryption, network protection, or software-oriented mitigation, are inadequate to protect robotic vehicles against sensor attacks since such attacks are conducted on an orthogonal attack surface [9]. People have used the physical invariants intrinsic in cyber-physical systems to detect or mitigate sensor attacks [1], [2], [11], [18]. This physics-based approach also received growing attention in robotic vehicles due to its explainability and low overhead [5], [6], [22]. The physics-based approaches are categorized into *physics-based attack detection* (PBAD) and *physics-based attack resilience* (PBAR). Both rely on a physical model that makes iterative predictions on the expected system states. Generally, the PBAD approaches compare the predictions with sensor measurements to detect the sensor attacks. The PBAR approaches, on the other hand, strive to sustain system operation by using the physical model to recover or replace the effects of faulty sensors.

Pushing towards efficient PBAD and PBAR in robotic vehicles, recent efforts have used the linear state-space model to capture the physical invariants in the vehicles [5], [6]. Using a linear model to approximate the higher-order dynamic has been justified as analogous to the PID control [5]. However, PID control relies on accurate feedback from the functional sensors to correct the approximation error. In contrast, PBAD and PBAR are designed to reject the compromised sensors, and their physical model should continue the state prediction even without the correction from sensors. Without such sensor-based correction, the model performance will deteriorate as the approximation er-

The post-conference authors' version fixed several figure display issues (Figures 7, 8, 9, 16, 18, 19, 21) in the production of the conference-proceeding version.

✉ Corresponding author.

ror accumulates. We expect a more accurate physical model, e.g., a well-fitted and high-order nonlinear model, to delay this process and take effect as the basis of sensor-attack resilience. Nonlinear physical invariant has been deployed in PBAD [22] without considering recovery. This nonlinear physical model does not account for the propulsion losses caused by the electrical and mechanical factors, which impede its usage in attack resilience. Other approaches based on passive fault-tolerant control (FTC) [7], [8], [32] also apply nonlinear models to capture the high-order vehicle dynamic. However, these solutions' high computational overhead and processing latency make mitigating high-rate IMU attacks harder [13].

Besides the model's accuracy and efficiency, there are more security concerns in state-of-the-art approaches. First, existing PBAD and PBAR approaches [5], [6], [22] use statistic-based attack detection. However, they are limited in differentiating the measurement noise of each sensor (e.g., gyroscope) from the detection statistics, leaving a loose alarm threshold for advanced adversaries to conduct stealthy attacks. Second, high-rate IMU attacks can crash the drone within one sampling interval [13], [14]. Existing PBAD and PBAR approaches cannot identify such attacks before the faulty data enters the estimator and control loop. Third, passive FTC approaches [7], [8] accept all sensor measurements without validation, allowing the adversary to manipulate the flight control freely through vulnerable sensor data. Fourth, the state-of-the-art PBAR approach [5] requires a functional accelerometer for recovery when all gyroscopes are compromised. As both sensors belong to IMU, such recovery dependency is exposed to attacks that corrupt all transmissions between IMUs and autopilot [13].

Regarding the above limitations, we emphasize four criteria for an effective PBAR: 1) an accurate physical model that supports long recovery duration, 2) an effective detection algorithm with tightened thresholds to restrain stealthy sensor attacks, 3) a delay mechanism avoiding high-rate attack data reaching the state estimates before being detected, and 4) an efficient implementation meeting the timing and resources of real hardware. In this work, we present Virtual IMU (VIMU), a new PBAR framework to address sensor attacks, esp. stealthy attacks, against aerial vehicles. We tighten alarm thresholds with a new detection algorithm, CS-EMA. CS-EMA extends *Cumulative Sum* (CUSUM) with a residual-capped EMA detector, which also works as a low-pass filter to distinguish the persistent deviation from high-frequency measurement noises. Then, we use a nonlinear physical model that includes precise submodels on aerodynamic drag and propulsion wrench to achieve a recovery performance better than the state-of-the-art PBAR [5] and the nonlinear model of [22]. To address the high-rate IMU attacks, we design a FIFO buffer safeguard that prevents faulty data from reaching the state estimates immediately. We summarize our contributions as follows:

1. We found that specific measurement noise, e.g., the gyroscope's white noise, can be filtered out to improve detection effectiveness. Our detection algorithm, CS-EMA, outperforms the detection algorithms of the state-of-the-

art PBAD [6], [22] and PBAR [5] in effectiveness against the overt and stealthy gyroscope attacks of [10], [22], [31]. CS-EMA detector identifies more stealthy attacks than the detectors of [5], [6], [22].

2. We propose the buffer safeguard to protect the integrity of UAV's reference states from instant attack injection. With the fine-grained physical model and buffer safeguard, VIMU surpasses the existing PBAR [5] and the nonlinear physical model of [22] in recovery duration. Even when all IMUs are compromised, VIMU can sustain flight stability longer without relying on the accelerometer or external system, as opposed to [5].
3. We implement VIMU in PX4 autopilot. We demonstrate that our implementation meets the timing and resources on real hardware (CUAV v5+). Our implementation is available at <https://github.com/wangwwno1/Project-VIMU>.

## 2. Adversary Model

We target a similar adversary model to [5], [22]. The adversary can inject false signals into multiple sensors and obtain the following knowledge to carry out the attack:

1. Vehicle's hardware specifications and physical properties.
2. The parameter settings of the autopilot program and its anomaly detector, if the anomaly detector is available.
3. The maneuver commands from the autopilot or human operator.

Our detection approach identifies sensor attacks based on the deviation of sensor measurements from the predicted state of the physical model, which requires an accurate system state estimation at the initialization of VIMU. Since VIMU is designed to launch simultaneously with the autopilot controller, we assume the autopilot controller can acquire the drone's accurate initial system state, including position, velocity, and attitude. We assume the actuators' correct functionality to ensure our physical model can predict the system states with the actuator commands from the autopilot controller. Thus, handling the faulty actuators is out of our scope. The faulty actuators can be addressed by the fault-tolerant control approaches [7], [8], [37], [38], but such approaches require further efforts to reduce computational delay and meet the real-time constraints. Although our approach can tolerate considerable wind disturbances on GPS and gyroscope, detecting and mitigating more violent external disturbances, e.g., physical collisions or the proximity to obstacles, are out of our scope. These disturbances can be discriminated through existing solutions, e.g., contact detection [29] or ground effect model [3]. Moreover, the attack surface of software and cyber attacks is orthogonal to the sensor attacks addressed in this work. Mitigating these threats is also beyond our scope.

## 3. Design of VIMU

VIMU is a physics-based attack resilience solution. It attains attack resistance by providing the autopilot with

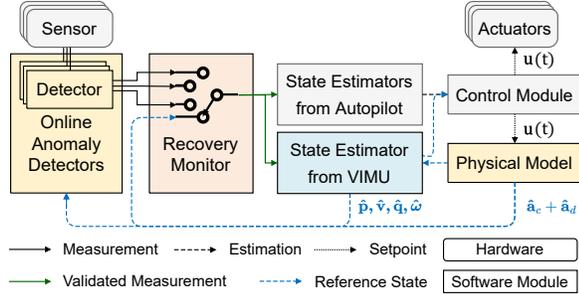


Figure 1. High-level Overview of VIMU Modules

uninterrupted state estimates even when the sensors are subjected to attack. VIMU has four modules: *anomaly detectors*, a *recovery monitor*, a *state estimator*, and a *physical model*. Fig. 1 shows the block diagram of VIMU. The anomaly detectors (VIMU-AD) identify compromised sensors. The recovery monitor (VIMU-RM) selects dependable data sources based on the detection results. The state estimator (VIMU-SE) combines predictions from the nonlinear physical model (VIMU-PM) with available sensor measurements to provide a reliable *estimate* of system states. We denote the estimates from VIMU-SE and VIMU-PM as *reference states*, in contrast to the estimates provided by state estimators from the autopilot. When the system is not under attack, VIMU ensures the estimates from the vanilla estimators converge to the reference states, as both estimated states are subject to the same physical laws. VIMU carries out the following functionalities to protect flight safety:

1. *Anomaly detection and recovery*. The first step towards attack resilience is to detect and isolate compromised sensors. The anomaly detector compares the reference states with sensor measurements to determine whether the sensor instance is compromised (Section 3.2). The recovery monitor enforces the lattice-based recovery policies, filters out compromised sensor data, and delivers the remaining measurements to all state estimators (Section 3.3). As compromised sensors get isolated, the state estimators with available inputs, including VIMU-SE, accurately estimate the system state, ensuring the safe control of autopilot.
2. *Uninterrupted estimation for flight control*. Vanilla state estimators in autopilot rely on IMU measurements to update their estimates. This design fails when all IMU instances are compromised. Accepting false sensor data disturbs the estimated state. On the other hand, isolating the compromised sensor instances will interrupt the vanilla estimators updating estimates. In both cases, the flight controller no longer tracks the system state, leading to an instant control loss. The reference state serves as a backup of existing estimates (Section 3.3). The physical model (Section 3.1) takes the last setpoint of actuator output issued by the controller as the control input  $\mathbf{u}(t)$  to update its reference states. VIMU-SE takes these reference states and the measurements of uncompromised IMUs to update the estimates of VIMU-SE (Section 3.4). This update process is iterative and works even without an operational IMU. When all IMUs get compromised,

TABLE 1. PHYSICAL PARAMETERS OF NONLINEAR PHYSICAL MODEL

Param	Definition
$\mathcal{M}$	Mass of the drone
$l$	Distance from the motor to the airframe CoG
$\mathbf{I}$	$3 \times 3$ symmetric matrix of airframe's moment of inertia. It comprises 3 diagonal terms ( $I_{xx}, I_{yy}, I_{zz}$ ) and 3 non-diagonal terms ( $I_{xy}, I_{xz}, I_{yz}$ )
$C_T$	thrust coefficient
$C_Q$	torque coefficient
$C_{Q,r}$	coefficient of rotor's gyroscopic moment
$t_{DC}$	time constant for the rotor speed to rise or fall
$V_{ref}$	nominal voltage when the battery module has charged to full capacity
$\bar{T}_{Min}$	the minimum command to spin up the rotor
$\bar{T}_{Range}$	the range of the motor command
$R_{int}$	battery internal resistance
$C_m$	momentum drag coefficient
$\mathbf{C}_{bxy}$	horizontal ballistic coefficient vector $[C_{bx}, C_{by}, 0]^T$ , s.t. $C_{bx}$ and $C_{by}$ are the forward and right direction of airframe, respectively

the control module switches to the reference states for flight control, thus ensuring continuous operation under attack.

### 3.1. Nonlinear Physical Model

The drone's airframe type impacts the concrete design of the physical model. However, different physical models generally follow the same development process. This work uses a quadcopter as the demonstration platform. Without loss of generality, we follow the common assumptions of prior works on physical model design [17], [22], [29], [34]: 1) The airframe is rigid and symmetrical. 2) All motors and propellers are rigid and produce the same thrust and torque. Each motor has the same distance to the airframe's *center of gravity* (CoG), leading to a coincidence of CoG with the center of thrust. 3) The Coriolis force and the aerodynamic torque are negligible due to the quadcopter's low flight speed and limited attitude changes.

Our nonlinear physical model is instantiated by deciding the following physical parameters:

$$\mathcal{P} := \{\mathcal{M}, l, \mathbf{I}, C_T, C_Q, C_{Q,r}, t_{DC}, \bar{T}_{Min}, \bar{T}_{Range}, V_{ref}, R_{int}, C_m, \mathbf{C}_{bxy}\} \quad (1)$$

Table 1 presents their definitions in detail. Specifically,  $C_T$ ,  $C_Q$ ,  $C_{Q,r}$ ,  $t_{DC}$ ,  $\bar{T}_{Min}$ ,  $\bar{T}_{Range}$ ,  $V_{ref}$ , and  $R_{int}$  are related to the propulsion system.  $C_m$  and  $\mathbf{C}_{bxy}$  are related to the aerodynamic drag. The parameters for each drone type are determined by the measuring and learning procedure in Appendix A.

The nonlinear physical model  $\mathcal{F}$  specifies the relations between the physical parameters  $\mathcal{P}$ , the system state  $\mathbf{x}(t)$ , and the control input  $\mathbf{u}(t)$ . With the relation

$$\mathbf{x}(t+1) = \mathcal{F}_{\mathcal{P}}(\mathbf{x}(t), \mathbf{u}(t)) \quad (2)$$

we can predict the runtime system states  $\mathbf{x}(t)$ . A precise physical model can accurately predict the drone's motion, improving the effectiveness of attack detection and recovery. Compared with SAVIOR's physical model [22], our physical model specifies the thrust estimation and the aerodynamic drag on the airframe more accurately. To illustrate our phys-

ical model, we concretize  $\mathbf{x}(t)$  and  $\mathbf{u}(t)$  into the following states and control inputs:

$$\mathbf{x} = [\mathbf{p}, \mathbf{v}, \mathbf{a}, \mathbf{R}, \omega, V_{load}, i_{load}, \mathbf{v}_w, \rho]^T \quad (3)$$

$$\mathbf{u} = [\bar{T}_1, \bar{T}_2, \bar{T}_3, \bar{T}_4]^T \quad (4)$$

All the states in  $\mathbf{x}$  and inputs in  $\mathbf{u}$  are timestamped. The system position  $\mathbf{p}$  is the Cartesian coordinates of the CoG.  $\mathbf{v}$  is the linear velocity.  $\mathbf{a}$  is the linear acceleration. The system attitude  $\mathbf{R}$  is a 3-axis rotation matrix, and  $\omega = [\omega_\phi, \omega_\theta, \omega_\psi]^T$  is the angular velocity specifying how fast the attitude changes along the roll( $\phi$ ), pitch( $\theta$ ), and yaw( $\psi$ ) axes.

The relationships between  $\mathbf{p}$ ,  $\mathbf{v}$ ,  $\mathbf{a}$ ,  $\mathbf{R}$ , and  $\omega$  are well-defined in the *rigid body model* [29]. To bridge the relationship between  $\mathbf{x}(t)$  and  $\mathbf{u}(t)$ , we summarize the rigid body equations as follows:

$$\mathbf{a} = g\mathbf{e}_3 + \mathbf{R}(\mathbf{a}_c + \mathbf{a}_d) \quad (5)$$

$$\dot{\omega} = \mathbf{I}^{-1}([\mathbf{I}\omega]_\times \omega + \mathbf{m}_c) \quad (6)$$

In (5), the linear acceleration  $\mathbf{a}$  is composed of the gravitational acceleration  $g$ , the acceleration from the control wrench ( $\mathbf{a}_c$ ), and the aerodynamic drag ( $\mathbf{a}_d$ ). The unit vector  $\mathbf{e}_3 = [0, 0, 1]^T$  at the z-axis combines  $g$  into  $\mathbf{a}$ , which is located in the North-East-Down (NED) inertial frame. However, both  $\mathbf{a}_c$  and  $\mathbf{a}_d$  are located in a different reference frame, the Forward-Right-Down (FRD) body frame. Therefore, they are multiplied by  $\mathbf{R}$  to align with the NED frame. In (6), the angular acceleration  $\dot{\omega} = [\dot{\omega}_\phi, \dot{\omega}_\theta, \dot{\omega}_\psi]^T$  is decided by the angular momentum from the airframe ( $[\mathbf{I}\omega]_\times \omega$ ) and the control torque from the actuators ( $\mathbf{m}_c$ ).  $[\cdot]_\times$  is the skew-symmetric matrix operator.  $\mathbf{a}_c$ ,  $\mathbf{a}_d$ , and  $\mathbf{m}_c$  capture the non-gravitational wrenches applied on the airframe.  $\mathbf{a}_c$  and  $\mathbf{m}_c$  are not measurable during the flight, but they are correlated with the measurable rotor speed  $\varpi_i$  and rotor acceleration  $\dot{\varpi}_i$ ,  $i \in [1..4]$ , as shown in [29]. Specifically, we estimate  $\mathbf{a}_c$  and  $\mathbf{m}_c$  with the following equations:

$$\mathbf{a}_c = \frac{C_T}{\mathcal{M}} (\varpi_1^2 + \varpi_2^2 + \varpi_3^2 + \varpi_4^2) \mathbf{e}_3 \quad (7)$$

$$\mathbf{m}_c = \begin{bmatrix} C_T l (\varpi_2^2 + \varpi_3^2 - \varpi_1^2 - \varpi_4^2), \\ C_T l (\varpi_1^2 + \varpi_3^2 - \varpi_2^2 - \varpi_4^2), \\ C_Q (\varpi_1^2 + \varpi_2^2 - \varpi_3^2 - \varpi_4^2) + \\ C_{Q,r} (\dot{\varpi}_1 + \dot{\varpi}_2 - \dot{\varpi}_3 - \dot{\varpi}_4) \end{bmatrix} \quad (8)$$

Measuring  $\varpi_i$  requires extra sensors. Therefore, we use the estimated relative thrust  $\hat{T}_i \geq 0$  as the approximation of  $\varpi_i$ . Given the actuator setpoint  $\bar{T}_i$  of  $\mathbf{u}(t)$ , we adjust  $\bar{T}_i$  by the battery voltage [3] and update  $\hat{T}_i$  [19]:

$$\bar{T}'_i(t) = \frac{\bar{T}_i(t) - \bar{T}_{Min}}{\bar{T}_{Range}} \cdot \frac{V_{load}(t) + R_{int} \cdot i_{load}(t)}{V_{ref}} \quad (9)$$

$$\hat{T}_i(t) = \alpha \hat{T}_i(t-1) + (1 - \alpha) \bar{T}'_i(t) \quad (10)$$

where  $\hat{T}_i(0) = \bar{T}'_i(0)$ .  $\alpha = \exp(-\frac{\Delta t}{t_{DC}})$ .  $\Delta t$  is the system control interval.  $V_{load}$  and  $i_{load}$  are the voltage and current measured by the battery power module. Equations (9) and (10) provide a more accurate estimation on actual thrust by compensating the voltage drop, mechanical frictions, and the time delay in rotor speed changes.

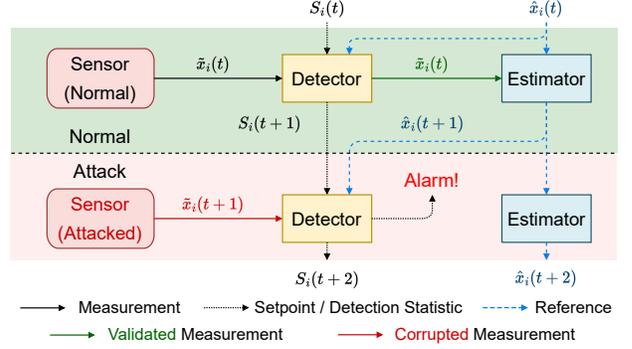


Figure 2. Workflow of Anomaly Detection

We estimate the drag-induced acceleration  $\mathbf{a}_d$  based on [29]:

$$\mathbf{a}_d = C_m \mathbf{v}_r + \frac{1}{2} \rho C_{bxy} \|\mathbf{v}_r\| \mathbf{v}_r \quad (11)$$

where  $\mathbf{v}_r = \mathbf{R}^T(\mathbf{v} - \mathbf{v}_w)$  is the relative airspeed.  $\mathbf{v}_w$  is the estimated wind velocity in the NED frame and is obtainable through multi-sensor fusion.  $\rho$  is the air density provided by the barometer.  $\|\cdot\|$  is the vector norm.

### 3.2. Online Anomaly Detection

Let  $\hat{\mathbf{x}}(t)$  and  $\tilde{\mathbf{x}}(t)$  be the reference states and sensor measurements, respectively. Given a sensor instance  $i$  of type  $j$  that measures  $\tilde{x}_i^j(t)$  on a subset of system states  $x^j(t) \subseteq \mathbf{x}(t)$ , the residual  $r_i^j(t)$  is defined as

$$r_i^j(t) = \tilde{x}_i^j(t) - \hat{x}^j(t) \quad (12)$$

If measurements from this instance significantly deviate from the reference state, the detector raises the alarm to block such a sensor instance from participating in sensor fusion. State-of-the-art detection approaches [5], [6], [22] take residuals as input and compute a detection statistic  $S_i(t)$  to quantify the deviation. For example, the statistic is defined as the average of the squared residuals [6] or the cumulative absolute residuals [5] within a fixed time window. CUSUM [35] used by [22] tracks the historical change in residuals and performs better than the time-window detectors [5], [6] in detecting persistent threats [33]. For each residual  $r_i(t)$ , CUSUM calculates the detection statistic  $S_i(t)$  iteratively based on the following equation:

$$S_i(t) = \max(0, S_i(t-1) + |r_i(t)| - b_i) \quad (13)$$

where  $S_i(0) = 0$ , and  $b_i > 0$  is the mean shift to suppress the increment of  $S_i(t)$  when there is no attack. The detector raises the alarm upon  $S_i(t)$  exceeding the predefined threshold.

However, the specificity of CUSUM would deteriorate when the monitored sensor exhibits intense measurement noise (e.g., white noise, random walk). The measurement noise affects the sensor data, and a significant presence of noise expands the discrepancy between sensor measurements and reference states (Equation (12)), leading to an increment of the absolute residual  $|r_i(t)|$  used in the CUSUM

algorithm. As a result, CUSUM requires a higher mean shift to suppress the false alarm, allowing the adversary to bypass the detection. For example, the gyroscope measurements are prone to frequent fluctuations caused by airframe vibrations. To avoid false alarms, the CUSUM detector on gyroscopes has to be configured with a relatively loose mean shift  $b_i$ , allowing us to inject a deviation of only 0.04 rad/s to crash the drone while maintaining  $|r_i(t)| < b_i$ . According to (13), this deviation keeps the  $S_i(t)$  below the detection threshold, and the CUSUM detector will raise no alarm.

To achieve a more robust detection, we propose *Cumulative Sum-Exponential Moving Average (CS-EMA)* as our detection algorithm. CS-EMA extends CUSUM with a residual-capped EMA detector, which filters white noise from the input to exhibit the persistent component of deviations:

$$S_i(t) = |MA_i(t)| \quad (14)$$

$$MA_i(t) = \lambda \cdot r'_i(t) + (1 - \lambda) \cdot MA_i(t - 1) \quad (15)$$

$$r'_i(t) = \max(\min(r_i(t), +R), -R) \quad (16)$$

where  $0 < \lambda \leq 1$  scales the impact of newest deviation on the moving average  $MA_i(t)$ . Smaller  $\lambda$  provides stronger noise filtering ability but requires a longer time to detect the attack. The *cap*  $R$  is a positive value greater than the detection threshold, which defines a hard limit on how much each deviation can impact on the moving average. Smaller  $R$  boosts the relative importance of minor deviations, making the detector focuses on deviations falling into the  $[-R, +R]$  interval. CS-EMA raises the alarm if the  $S_i(t)$  produced by the CUSUM component (Equation (13)) or the EMA component (Equation (14)) exceeds the alarm threshold.

Our detection approach (VIMU-AD) applies the CS-EMA algorithm on each sensor instance. An instance may measure more than one system state. For example, GPS measures position and velocity ( $\tilde{x}^{\text{GPS}} = \{\tilde{\mathbf{p}}, \tilde{\mathbf{v}}\}$ ). In that case, each measured state will employ an independent detector. Any alarm from these detectors indicates a compromised sensor instance, and the recovery monitor (VIMU-RM) will discharge this instance from the sensor fusion. Fig. 2 depicts the relationships between the detector, state estimator, and the monitored sensor instance. This detection workflow is neutral to sensor types. Each detector responds to every update of its corresponding sensor measurements. Thus, the update rate of  $S_i(t)$  follows the sampling rate of the monitored sensor instance.

### 3.3. Sensor Isolation and State Recovery

The recovery monitor (VIMU-RM) handles the compromised sensor instance identified by the anomaly detectors (VIMU-AD). The monitoring procedure comprises two steps in each iteration: *sensor isolation (SI)* and *system state recovery (SSR)*. When any anomaly detector updates its detection result, the SI step formulates a validated sensor list by filtering out the compromised sensor instances. Then, based on a predefined policy, the SSR step picks the best sources from the validated sensor measurements and the

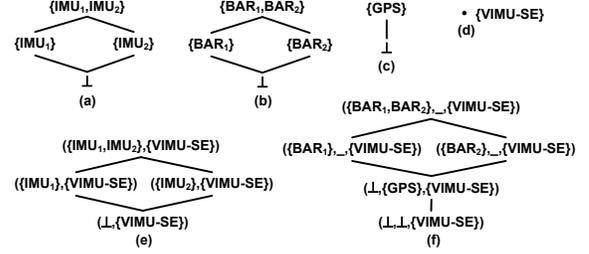


Figure 3. Lattice-based Recovery Policy for Flight Altitude ( $p_z$ ) and Angular Velocity ( $\omega$ ). “ $\_$ ” Stands for *Any Possible Cases* and “ $\perp$ ” Stands for *Sensor Instances are Compromised*

reference state. After that, the recovery monitor sends the selected data to the control module and state estimators.

For each system state, the validated sensor measurements determined in the SI step combine with the reference state from VIMU-SE to form an *available data source list*. The reference state is always treated as an available data source to prevent the interruption of state estimation and flight control. For example, on a drone with two redundant IMUs, the default available source list of angular velocity  $\omega$  is  $\{\tilde{\omega}_1^{\text{IMU}}, \tilde{\omega}_2^{\text{IMU}}, \tilde{\omega}^{\text{VIMU}}\}$  with  $\tilde{\omega}^{\text{VIMU}}$  as the reference state from VIMU-SE. When VIMU-AD detects the instance  $\text{IMU}_1$  is under attack, VIMU-RM will remove the source  $\tilde{\omega}_1^{\text{IMU}}$  in the SI step, and the available altitude source list becomes  $\{\tilde{\omega}_2^{\text{IMU}}, \tilde{\omega}^{\text{VIMU}}\}$ .

The SSR step determines which data source in the available data source list is the most reliable and accurate. Since the quality of data sources varies by the sensor type and the physical model, we propose a *lattice-based recovery policy* for recovering each system state. For each system state  $x(t)$ , we formulate a partial order  $(R_{x(t)}, <)$  to define the sensor type priority. For example,  $(R_{p_z}, <)$  for the flight altitude  $p_z$  is  $\text{VIMU-SE} < \text{GPS} < \text{BAR}$ , and  $(R_{\omega}, <)$  for the angular velocity is  $\text{VIMU-SE} < \text{IMU}$ . Thus,  $(\tilde{p}_z)^{\text{BAR}}$  and  $\tilde{\omega}^{\text{IMU}}$  have the highest priority in respective case. We define the compromise lattice of each sensor type, e.g., Fig. 3(a)-(d). Following the priority order relation, we make the ordered multiplications over the compromise lattices to build our lattice-based recovery policy.

The recovery policy lattices for  $\omega$  and  $p_z$  are abstracted in Fig. 3(e) and (f), respectively. Each element in Fig. 3(e) and (f) represents a *data source list status* under potential sensor attacks. In each status, VIMU-RM chooses the sensor instance from the leftmost valid sensor type (i.e., in best quality). The recovery monitor is initialized at the uppermost data source list status and will do its best to stay within an upper status in the lattice as long as possible unless the attack identified by VIMU-AD forces VIMU-RM change to a less qualified status in the lattice. Finally, VIMU-RM could reach a status with only VIMU-SE’s reference state as the valid data source. For example, when all IMUs are compromised, VIMU-RM follows the recovery policy of  $\omega$  to reach the status  $(\perp, \{\text{VIMU-SE}\})$  of Fig. 3(e), which means the control module receives the uninterrupted reference angular velocity  $(\tilde{\omega})^{\text{VIMU}}$  for flight control.

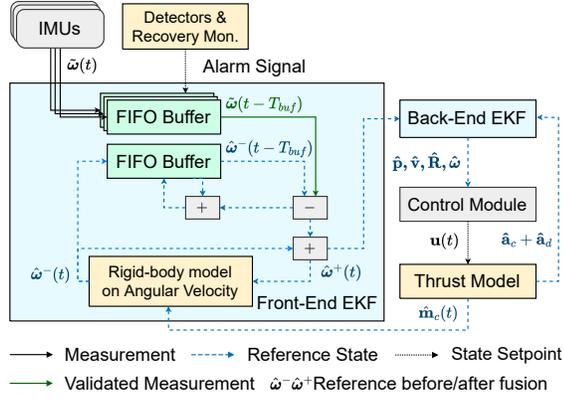


Figure 4. Workflow of Front-End EKF of VIMU-SE

### 3.4. State Estimator and Buffer Safeguard

Our fine-grained physical model can accurately capture the transition of the system states. However, due to the model bias and external perturbation, the model prediction could still slightly deviate from the actual system state. Such deviation can accumulate over time during the flight. Therefore, VIMU’s state estimator (VIMU-SE) uses a pair of *Extended Kalman Filters* (EKF) in a cascaded pattern to periodically rectify the deviation with benign measurements from the operational sensors. The *front-end EKF* fuses the current angular velocity  $\hat{\omega}^-$  (predicted by (6)) with IMU measurements  $\hat{\omega}$  to produce the corrected estimate  $\hat{\omega}^+$ . Then, the *back-end EKF* updates the reference states with  $\hat{\omega}^+$  and available measurements of position, speed, and attitude. This process further rectifies  $\hat{\omega}^+$  with other sensor data to derive the reference angular velocity  $\hat{\omega}$  given to the anomaly detectors and recovery monitor.

The back-end EKF uses a structure similar to the vanilla state estimator in the autopilot. However, to address IMU attacks, the design of the front-end EKF is nontrivial (Fig. 4). An advanced adversary can delay the alarm to several hundred milliseconds after the attack. Considering the high sampling rate of IMU sensors (from a hundred to thousands hertz), such a time delay allows the attacker to inject a remarkable bias into the reference state, rendering detection and recovery inoperable [10], [13]. To address this threat, we develop first-in-first-out (FIFO) buffers in the front-end EKF as the safeguard between the recent measurements and angular velocity estimates: 1) *Measurement buffers*: The front-end EKF holds one *measurement buffer* for each IMU instance to introduce a time delay to the sensor measurements. 2) *Estimate buffer*: The *estimate buffer* raises a similar time delay to align the timing of estimates with the delayed sensor measurements for a correct sensor fusion.

Even if the alarm has been delayed, such buffers can prevent the faulty data from being immediately fused into the estimation. At each iteration of state estimation, the front-end EKF first checks the detection report from VIMU-RM. If VIMU-RM reports a compromised IMU instance, the EKF will remove all IMU measurements stored in the corresponding buffer to prevent the potential attack. Then,

the EKF updates the estimation with (6), pushes the newest estimate into, and pops the oldest estimate from the estimate buffer. After that, the EKF iterates over the measurement buffers to find a validated and time-aligned IMU measurement. Such a measurement can be unavailable due to the sensor attack. In that case, the front-end EKF will output  $\hat{\omega}^-$  to the back-end EKF and let it correct the estimate with other sensors. Otherwise, it pops the IMU measurement out from the buffer to perform sensor fusion. The sensor fusion in each iteration proceeds as follows: the EKF calculates a residual between the delayed estimate and measurement, adds the residual to the newest estimate  $\hat{\omega}^-(t)$ , and outputs the corrected estimate  $\hat{\omega}^+(t)$ . The same residuals are also added to the estimates in the estimate buffer to cumulate the corrections over time. The buffer size is determined by:

$$size_{\text{buffer}} := 1 + \lceil T_{\text{buf}} \cdot H \rceil \quad (17)$$

where  $T_{\text{buf}}$  is the desired time to hold the received measurements, and  $H$  is the output rate of the stored data.

With the FIFO buffer safeguard, we can further strengthen the integrity of reference states even under stealthy IMU attacks. However, using our buffer safeguard could increase the bias in reference angular velocity because of the delayed sensor fusion. Such bias causes a modest reduction in the detector specificity, especially for CUSUM. Our CS-EMA algorithm does not rely on the CUSUM component to identify stealthy attacks. Therefore, we add an extra reference angular velocity for the CS-EMA detectors that monitor the gyroscope. This reference state is received by the CUSUM component of CS-EMA and is corrected by the newest IMU measurements to shorten the detection delay towards overt attacks. As both improvements work independently, the extra reference improves the overall performance of the CS-EMA detector while maintaining its robustness towards stealthy attacks.

## 4. Implementation

### 4.1. Security-Enhanced Autopilot

We implement VIMU on PX4 v1.13.3. We add new message formats to the PX4 architecture for reference state and sensor health status. Then, we implement the detection algorithms as a library, including CS-EMA, CUSUM, and the time-window detections [5], [6]. We insert VIMU-AD into the sensor middlewares and modules to ensure the detection covers all received measurements. The detector location depends on the sensor type. For example, we insert the detectors for the gyroscope and accelerometer into the middleware `PX4Gyroscope` and `PX4Accelerometer`. For other sensors, we implement detectors in their data processing modules (e.g., `vehicle_gps_position`). We implement VIMU-RM in the state estimation (EKF2) and estimator selector (EKF2Selector) modules. The recovery monitor receives the runtime detection results. Its IMU-related component in `EKF2Selector` applies the IMU-related recovery policies (e.g., Fig. 3(e)) to isolate com-

TABLE 2. SENSOR ATTACKS USED IN OUR EVALUATION

Category	Notation	Description
Overt	OA <sub>GPS</sub>	Injecting spoofed GPS positions [5], [6], [22]
	OA <sub>GPS-PV</sub>	Combines position and velocity spoofing to deviate a hovering drone with faked GPS signal [23]
	OA <sub>Gyro</sub> <sup>2/3</sup>	Modulated acoustic attacks injecting controlled fixed deviation to target gyroscopes or accelerometers [10]. The gyroscope attacks follow the scenarios in [5].
	OA <sub>Gyro</sub> <sup>3/3</sup>	
	OA <sub>Accel</sub> <sup>3/3</sup>	
	OA <sub>ICM20602</sub> <sup>3/3</sup>	Unmodulated acoustic attacks injecting sinusoidal attack signal into gyroscope measurements of two IMU models: ICM20602 and ICM20689 [14], [31]
	OA <sub>ICM20689</sub> <sup>3/3</sup>	
OA <sub>Baro</sub> <sup>2/2</sup>	Injecting fixed deviations to all barometers [5]	
OA <sub>Mag</sub> <sup>2/2</sup>	Injecting fixed deviations to all magnetometers [20]	
Stealthy	SA <sub>GPS</sub>	Stealthy position spoofing [22]
	SA <sub>Gyro</sub> <sup>3/3</sup>	Stealthy gyroscope attack [10] following the scenario of [22]
Multi-Type	MA*	A combination of multiple overt attacks [5], [16], [20] for testing recovery effectiveness. Especially, attack MA <sub>Mag Accel Gyro</sub> is equivalent in impact to the EMI attack of [13] according to the implementation of [16]

promised IMUs by orderly responding to the angular velocity, acceleration, and attitude errors. Similarly, VIMU-RM’s IMU-unrelated component in EKF2 applies the IMU-unrelated policies (e.g., Fig. 3(f)) to isolate other sensors by orderly responding to the horizontal position, altitude, and velocity errors. We incorporate the physical model into the two cascaded EKFs of VIMU-SE. For angular velocity estimation, we add a new module (`VirtualIMU`) to the autopilot. This module integrates the front-end EKF and the nonlinear physical model. It uses actuator setpoints from the `actuator_outputs` message to update current acceleration and angular velocity. For the back-end EKF, we reuse the existing EKF2 module as it already provides estimation and sensor fusion in the position, velocity, and attitude states. When all hardware IMUs are compromised, `EKF2Selector` will switch to the VIMU-SE for state estimation, thus preventing the in-flight control loss leading to drone crashes.

## 4.2. Sensor Attack Simulation

Following [5], [6], [22], we simulate sensor attacks by instrumenting attack code into the sensor middleware and modules of the autopilot. These middlewares and modules convert raw sensor readings (e.g., pressure, currents, and magnetic field strength) into measurements of system states (e.g., position, acceleration, and attitude) before delivering them to the state estimators and control module. Modifying those measurements within the autopilot to imitate the impact of injected attack signals has also been adopted in [8], [37], [38]. These software-based attacks also allow us to efficiently apply attacks on various sensors with diverse attack amplitudes.

The number and kinds of victim sensors vary by the attack type. Table 2 presents the sensor attacks used in our evaluations, including all sensor attacks from baseline approaches [5], [6], [22] and several recent popular attacks [10], [13], [20], [23], [31]. We denote the attack in the form of  $attack\_category_{sensor\_type}^{\#compr/\#avail}(deviation)$ . Specifically,

we implement overt attacks (OA), stealthy attacks (SA), and multi-type attacks (MA) on different sensor types (GPS, Gyro, Accel, Baro, or in combination).  $\#compr$  and  $\#avail$  respectively represent the number of compromised and available sensor instances, which are omitted if only one instance exists in the drone. The *deviation* specifies the amplitude of injected deviation to the compromised instances, which could be constant or time-dependent. More specifically, the adversary of the modulated acoustic attacks spoofs the majority (OA<sub>Gyro</sub><sup>2/3</sup>) or all (OA<sub>Gyro</sub><sup>3/3</sup>) of the gyroscopes, misleading the drone’s majority voting to discharge the operational gyroscope or even disabling all redundant IMUs. The unmodulated acoustic attacks (OA<sub>ICM20602</sub><sup>3/3</sup>, OA<sub>ICM20689</sub><sup>3/3</sup>) follow the principle of [31] to compromise our quadcopter’s gyroscopes ICM20602 and ICM20689. The time-dependent deviation is defined as  $A_i \cdot \cos(2\pi F_i \cdot t)$ , where  $t$  is the time elapsed since the attack,  $A_i \in (0, A_{max}]$  is the induced amplitude ( $A_{max}$  is the maximum induced amplitude), and  $F_i$  is the induced acoustic frequency.  $A_{max}$  and  $F_i$  are decided by the targeted sensor model. According to [14], we have  $A_{max} = 0.927$  rad/s and  $F_i = 19.7$  Hz for OA<sub>ICM20602</sub><sup>3/3</sup>;  $A_{max} = 1.899$  rad/s and  $F_i = 205.9$  Hz for OA<sub>ICM20689</sub><sup>3/3</sup>, respectively.

The overt attacks represent the common attack scenarios. However, the most threatening attacks against PBAD are the stealthy attacks [22], [33]. Stealthy attackers know the victim drone’s characteristics, e.g., mission plan, reference states, detection algorithm and parameters. Since VIMU and the related PBAR approach [5] require a prior detection alarm to initialize the recovery procedure, stealthy attacks can prevent or delay the recovery effects, causing more significant damage. The stealthy attacks SA<sub>GPS</sub> and SA<sub>Gyro</sub><sup>3/3</sup> in Table 2 adaptively inject a minor deviation to all target sensor instances, maximizing the attack effect without triggering the alarm. Specifically, the attacker tries injecting a series of attack signals  $\tilde{x}_i^*(t)$  into the targeted sensor instance  $i$  to maximize the deviation misleading the system behaviors while holding the detection statistic  $S_i(t)$  under the alarm threshold  $\tau$ :

$$r_i^*(t) = \tilde{x}_i(t) + \tilde{x}_i^*(t) - \hat{x}_i(t) \quad (18)$$

$$\operatorname{argmax}_{\tilde{x}_i^*(t)} |r_i^*(t)| \text{ s.t. } S_i(t) \leq \tau_i \quad (19)$$

where  $r_i^*(t)$  is the residual that includes  $\tilde{x}_i^*(t)$ . The most effective  $\tilde{x}_i^*(t)$  depends on the detector parameters and the difference between measurement  $\tilde{x}_i(t)$  and reference state  $\hat{x}_i(t)$ . Finally, the multi-type attacks in Table 2 apply all-instance overt attacks on multiple sensor types simultaneously. Following the setup of [16], we use the attack MA<sub>Mag|Accel|Gyro</sub> to simulate the impact of the EMI attack [13] on the simulator.

## 5. Evaluation

### 5.1. Experimental Setup

**5.1.1. Testbed.** We evaluate VIMU with quadcopters in simulation and real-world flight. We use the default simula-

TABLE 3. LIST OF SENSORS OF REAL WORLD DRONE

Sensor Type	Product Type	Number	Measurement
GPS Module	Neo V2	1	Position and Velocity
	ICM20602	1	Acceleration
IMU	ICM20689	1	and
	BMI055	1	Angular Velocity
Magnetometer	IST8310	2	Attitude
Barometer	MS5611	2	Vertical Position

tor of PX4 autopilot, i.e., jMAVSIM, and its default quadrotor as the simulation testbed. Wind is a common external disturbance in flight. Thus, we evaluate the robustness of VIMU with the wind simulation enabled (more results in Appendix C.3). We also activate all the detectors throughout experiments rather than only those of the sensors under attack. This setting can reveal how the recovery performance is affected by detection delay and false alarms. For the real-world evaluations, we assemble a quadcopter with a ZD550 airframe. The autopilot board is a CUAV V5+, with a 216MHz Arm Cortex-M7 CPU and 512 kB RAM for runtime memory. We calibrate all sensors before real-flight tests, and Table 3 lists the sensors used by this drone. Throughout the evaluation, we use QGroundControl as the ground control station and communicate with the drone through the Mavlink v2.0 protocol.

**5.1.2. Baselines.** We compare VIMU with the following approaches, including the variants of VIMU and the state-of-the-art PBAD [6], [22] and PBAR [5] approaches.

- *Control Invariant*<sup>1</sup> (CI) [6] uses a linear state-space model to obtain the expected system state and applies a squared error time-window detector (denoted as L2TW).
- *Software-based Real-time Recovery*<sup>2</sup> (SRR) [5] reuses the linear model for recovery and applies a time-window detector based on the absolute error (denoted as L1TW).
- *SAVIOR*<sup>3</sup> [22] predicts system states with a nonlinear physical model and applies CUSUM for detection.
- *SAVIOR-Buffer*<sup>3</sup> is a PBAR approach that extends SAVIOR with the recovery monitor (Section 3.3) and FIFO-buffer guarded state estimation (Section 3.4). It also delivers the estimated state as an available data source for the SSR step in Section 3.3.
- *VIMU-CS*<sup>4</sup> is a variant of VIMU that replaces CS-EMA with the CUSUM detector to demonstrate the contribution of our detector.
- *VIMU-NoBuffer*<sup>5</sup> is a variant of VIMU that discharges FIFO buffers to demonstrate the contribution of buffer safeguard. This variant’s reference state is corrected with the newest IMU measurements.

The original implementations of [5], [6], [22] are on different autopilots and simulators. Therefore, we reimplement these baselines on PX4 autopilot v1.13.3 to ensure a fair

<sup>1</sup><https://github.com/wangwno1/Project-VIMU/tree/baseline/CI>

<sup>2</sup><https://github.com/wangwno1/Project-VIMU/tree/baseline/SRR>

<sup>3</sup><https://github.com/wangwno1/Project-VIMU/tree/baseline/SAVIOR>

<sup>4</sup><https://github.com/wangwno1/Project-VIMU/tree/baseline/Virtual-IMU+CUSUM-Detector>

<sup>5</sup><https://github.com/wangwno1/Project-VIMU>

comparison. We obtain CS-EMA detector parameters and follow the procedures described in [5], [6], [22] to obtain the model and detector parameters (Appendix B). Comparing VIMU with machine-learning-based approaches, i.e., [7], [8], [14], is out of our scope due to their high computational overheads and the difficulty of deployment over low-end drones.

**5.1.3. Data Collection.** We design three missions to represent the most common flight states of drones. 1) *Hovering*: Taking off to an altitude of 15m before flying horizontally to a preset waypoint located 10m north and 10m east of the launch point. Then, hovering for 300 seconds before returning to the home. 2) *Moving*: Taking off to an altitude of 50m before flying horizontally to the same preset waypoint. Then, flying horizontally to the northeast for a distance of 1,000m before returning to the home. 3) *Maneuver*: Taking off to an altitude of 15m. Then, flying horizontally in an equilateral-triangle route with an edge length of 2.5m. On this route, the drone changes its direction between clockwise and counter-clockwise for ten times whenever reaching the initial hovering point. In the flight, we randomly pick one waypoint from the flight direction changing point or the triangle vertices, as the preset waypoint. In Section 5, we mainly present VIMU’s effectiveness with the *Hovering* and *Moving* scenarios. We discuss the impact of drastic maneuvers (e.g., rapid acceleration and sharp turns in the *Maneuver* scenario) on VIMU’s effectiveness in Appendix C.2. We combine the mission scenarios with the attacks listed in Table 2, in terms of *Mission[Attack]*, to represent the attack test cases. For example, *Hovering[OA<sub>GPS</sub>(10.0)]* is an overt GPS attack test case with the attack deviation set to 10.0 meters in the *Hovering* scenario.

We compare VIMU with the baselines on various attack test cases. For each candidate approach, we collect 50 flight records per attack test case. In the test case, the drone initializes the system, takes off, and activates the detectors before reaching the preset waypoint. After the drone reaches the waypoint, we activate the attack and record the attack-beginning timestamp  $t_{\text{Atk}}$ . The anomaly detectors will examine received measurements to decide whether the monitored sensor instance  $s(i)$  is under attack. If the detector of  $s(i)$  alarms (including false alarms), we record the first-alarm timestamp on instance  $s(i)$ , i.e.,  $t_{\text{Alarm}}^{s(i)}$ . We define the earliest  $t_{\text{Alarm}}^{s(i)}$  of all sensor instances as the first-alarm timestamp of this flight, i.e.,  $t_{\text{Alarm}}$ . Regardless of the detection results, the drone continues the flight until it completes its mission, crashes, or the position deviation between the estimate  $\hat{\mathbf{p}}(t)$  and the actual position  $\mathbf{p}(t)$  has reached 5 meters. We group the collected records by sensor type after the flight ends.

**5.1.4. Metrics.** We evaluate detection effectiveness with *true positive rate* (TPR), *false positive rate* (FPR), and *time to detect* (TTD). TPR and FPR are classical metrics in anomaly detection. They measure the sensitivity and specificity of the detectors. The attacks targeting IMUs can cause irreparable damage within tens to hundreds of milliseconds [13], [14], and GPS spoofing requires several

seconds to become effective [23], [25]. Therefore, we define an upper time-bound  $T_{\text{Alarm}}$  on the detection delay, e.g.,  $T_{\text{Alarm}}^{\text{GPS}} = 20$  s and  $T_{\text{Alarm}}^{\text{Gyro}} = 1$  s. When deciding the detection metrics, we consider the alarm *effective* in attack mitigation if and only if  $t_{\text{Atk}} \leq t_{\text{Alarm}}^{s(i)} \leq t_{\text{Atk}} + T_{\text{Alarm}}$ . Given a sensor instance participated in a flight mission, TPR and FPR are decided on the following detection result classification:

- *True positive* (TP): The detector raises an effective alarm when the monitored instance is compromised.
- *False positive* (FP): The detector raises the alarm despite the instance being operational.
- *True Negative* (TN): The detector correctly raises no alarm on an operational instance.
- *False Negative* (FN): The detector fails to raise an effective alarm on a compromised instance, i.e., either raising no alarm or raising an ineffective alarm later than the upper time-bound  $T_{\text{Alarm}}$ .

Note that each detection result and its classification are defined on the flight mission. Then, the *time to detect* is the time a detector takes to identify the presence of the attack. A shorter TTD implies a faster response in attack detection and isolation. As the recovery procedure starts with attack isolation, the TTD estimates the attack damage before detection and affects the recovery performance. For a sensor instance  $s(i)$  under attack, if the detection result is TP, we calculate TTD with  $(t_{\text{Alarm}}^{s(i)} - t_{\text{Atk}})$ . Otherwise, we label TTD as  $\geq T_{\text{Alarm}}$  to indicate that the detector failed to report the attack within a reasonable time.

Following SRR [5], we use the *effective recovery duration* as the metric of recovery performance. We regard the recovery procedure as *in effect* if any detector raises the alarm and, since then, the distance  $d(t)$  between the current position estimate  $\hat{\mathbf{p}}(t)$  and the actual position  $\mathbf{p}(t)$  never exceeds the error threshold  $\epsilon$ .

$$d(t) = \|\mathbf{p}(t) - \hat{\mathbf{p}}(t)\| \leq \epsilon, t \in [1, \dots, k] \quad (20)$$

where  $t$  is the timestamp of each position state,  $k$  is the first timestamp of  $d(t) > \epsilon$  or the end of the flight. Thus, we define *effective recovery duration* as the time from the first alarm ( $t_{\text{Alarm}}$ ) to the above timestamp  $k$ . We set  $\epsilon$  as 3 meters throughout the evaluations. Due to the flight time constraint, we cannot always measure the maximum recovery duration since in several attack categories, the recovery approaches can sustain the flight for a long time. We set an upper time bound  $T_{\text{rec}} = 300$  s for the effective recovery duration, and label the result as  $\geq T_{\text{rec}}$  if the approach has an effective recovery duration longer than  $T_{\text{rec}}$ .

## 5.2. Detection Effectiveness

We first evaluate the detection effectiveness of the candidate approaches using overt attacks, including GPS spoofings ( $OA_{\text{GPS}}$ ,  $OA_{\text{GPS-PV}}$ ) and gyroscope attacks ( $OA_{\text{Gyro}}^{2/3}$ ,  $OA_{\text{Gyro}}^{3/3}$ ,  $OA_{\text{ICM20602}}^{3/3}$ ,  $OA_{\text{ICM20689}}^{3/3}$ ). Then, we evaluate the resilience of different approaches toward stealthy attacks ( $SA_{\text{GPS}}$ ,  $SA_{\text{Gyro}}^{3/3}$ ). The measurements affected by GPS spoofing are the north-axis position (in meters) and velocity (in

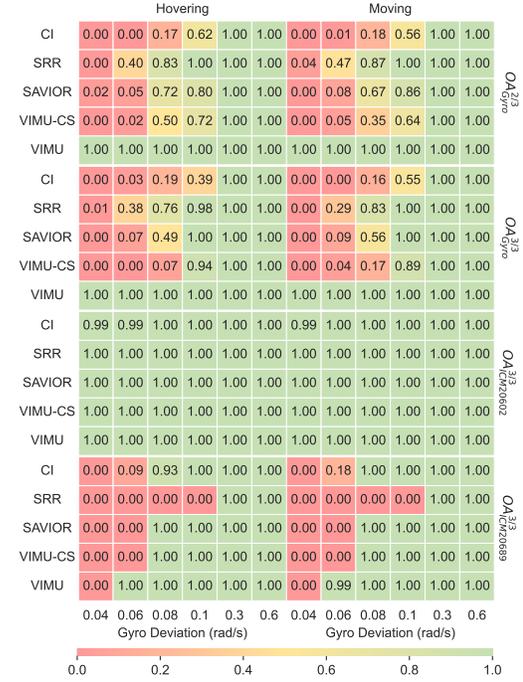


Figure 5. TPR Heatmap on Overt Gyro Attacks

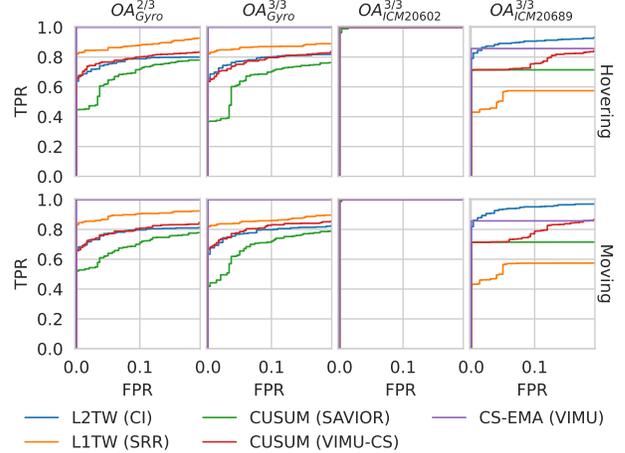


Figure 6. ROC Curves on Overt Gyro Attacks

m/s, only for  $OA_{\text{GPS-PV}}$ ). For gyroscope attacks, the injected signal applies to the three-axis angular velocity (in rad/s).

**5.2.1. Effectiveness on Overt Attacks.** Fig. 5 shows that the TPR positively correlates with the attack deviation of overt gyroscope attacks. All the approaches achieve 100% TPR when the attack deviation reaches 0.3 rad/s. In almost all attack test cases, VIMU has the highest TPR and the largest area under the ROC curves (Fig. 6). Our CS-EMA detector successfully detects most attacks with minor deviations (0.04 rad/s and 0.06 rad/s), whereas the CUSUM detectors of SAVIOR and VIMU-CS achieve lower TPRs. Such attacks have deviations considerably smaller than the angular velocity encountered in quadcopter's flight maneuvers (up to 2.0 rad/s [5]), but can lead to control loss within hundreds of milliseconds. Therefore, our CS-EMA detector

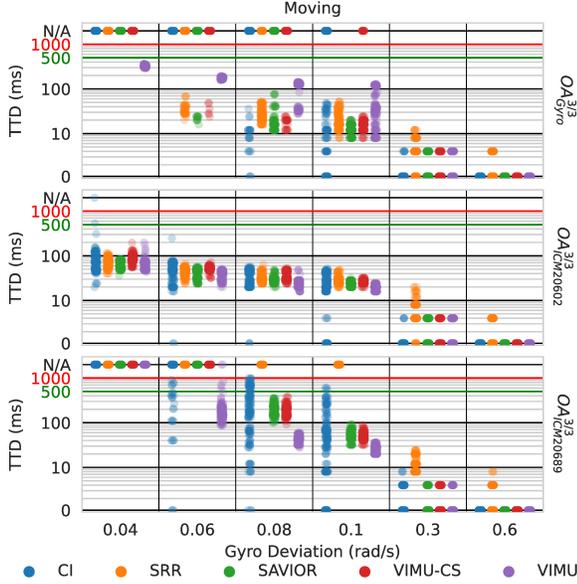


Figure 7. Time to Detect on Overt Gyro Attacks in *Moving* Mission (Red Line:  $T_{Alarm}^{Gyro}$ , Green Line:  $T_{buf}$ )

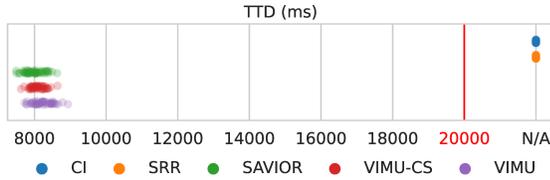


Figure 8. TTD on GPS Spoofing  $OA_{GPS-pv}$

is more suitable for identifying threats targeting the gyroscope. Compared with  $OA_{ICM20602}^{3/3}$ , detecting  $OA_{ICM20689}^{3/3}$  is more challenging because its attack signal operates at a frequency ( $F_i = 205.9$  Hz) closer to the angular velocity control (250 Hz in our evaluation), which makes it more similar to the measurement noise described in Section 3.2. We further compare the AUC of VIMU-CS and SAVIOR in Fig. 6 to determine the contribution of the physical model to detection effectiveness. As the FPR is calculated on attack-free flight records, it will not be affected by the FIFO buffer (Section 3.4). Meanwhile, both solutions use the same CUSUM detector. Thus, the difference in AUC mainly comes from the fine-grained physical model (Equations (8) to (11)), indicating that the fine-grained physical model has contributed to attack detection. GPS data is less affected by the airframe vibration than the gyroscope data. Therefore, our CS-EMA detector performs similarly to the CUSUM of SAVIOR and VIMU-CS in detecting  $OA_{GPS}$ . The implementation of  $OA_{GPS-pv}$  does not have a constant deviation setting to elaborate on the TPRs on various attack deviations. We observed that VIMU, SAVIOR, and VIMU-CS can detect  $OA_{GPS-pv}$  in 100% TPR with an AUC=0.98~1.00, while CI and SRR can only achieve an AUC less than 0.15.

Fig. 7 presents the log-scaled TTD of different approaches against overt gyroscope attacks. We omit the TTD on  $OA_{Gyro}^{2/3}$  because they are similar to  $OA_{Gyro}^{3/3}$ . VIMU has a TTD similar to SAVIOR and VIMU-CS in most attack

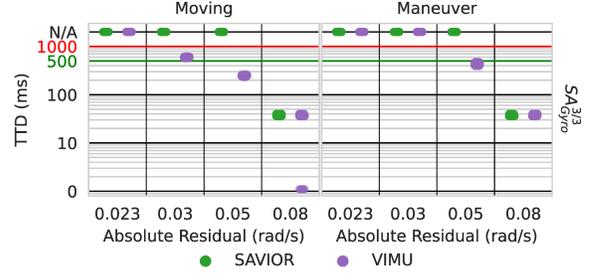


Figure 9. TTD on Stealthy Attacks. N/A = No Alarm

cases and maintains a steady TTD in small attack deviation, which is favorable in the FIFO buffer design. Besides, we test with gyroscope attacks at extremely large deviations, as they can disrupt the system state within one sampling interval [14]. Based on our quadcopter’s sensor specification, we measure the TTD of CS-EMA on  $OA_{ICM20602}^{3/3}$  and  $OA_{ICM20689}^{3/3}$  executed at their maximum induced amplitude  $A_{max}$  (Section 4.2). Results show that our CS-EMA detector detects the attacks within one sampling interval (4 ms), thus preventing damage to the flight control. In the hovering flights, we observed the same comparative advantages of the CS-EMA detector in TTD on the overt gyroscope attacks. Fig. 8 shows the TTD under the real-world GPS spoofing  $OA_{GPS-pv}$ . VIMU, SAVIOR, and VIMU-CS have similar TTDs (7~9 seconds), indicating that  $OA_{GPS-pv}$  is more complex to detect than the constant-deviation spoofing  $OA_{GPS}$  (with  $TTD \leq 1$  second, shown in Fig. 16).

**5.2.2. Resilience towards Stealthy Attacks.** We evaluate the robustness of detectors by analyzing the maximum  $|r_i^*(t)|$  of (19) that the adversary can achieve while remaining stealthy. A larger  $|r_i^*(t)|$  means the adversary can inject more deviation  $\hat{x}_i^*(t)$  in (18) with the same measurement and reference state. Fig. 9 shows three  $SA_{Gyro}^{3/3}$  and four  $SA_{Gyro}^{3/3}$  attacks. These attacks are stealthy because at least one detector under evaluation raises no alarm. We use TTD to indicate the effectiveness of the alarm. In Fig. 9, our CS-EMA detector identifies more attacks within the time bound than other detectors. Moreover, the CS-EMA detector has the tightest bound on stealthy attack deviation. Both results show that the CS-EMA detector is more robust against stealthy attacks targeting GPS or gyroscope.

### 5.3. Recovery Effectiveness

**5.3.1. Comparison with SRR.** We compare VIMU’s recovery performance with the state-of-the-art PBAR approach SRR [5] in the attacks  $OA_{GPS}$ ,  $OA_{Baro}^{2/3}$ ,  $OA_{Gyro}^{3/3}$ , and  $MA^*$ . Both approaches require a detection alarm to initialize the recovery procedure, which means the TTD affects their recovery performance. For a fair comparison, we use the attacks that ensure the immediate initialization of the recovery procedure at the attack-activation moment, e.g., the attack cases of  $OA_{Gyro}^{3/3}$  (0.60) with  $TTD \approx 0$  in Fig. 7.

Fig. 10 shows the effective recovery duration of VIMU and SRR. The durations depend on the type of compromised

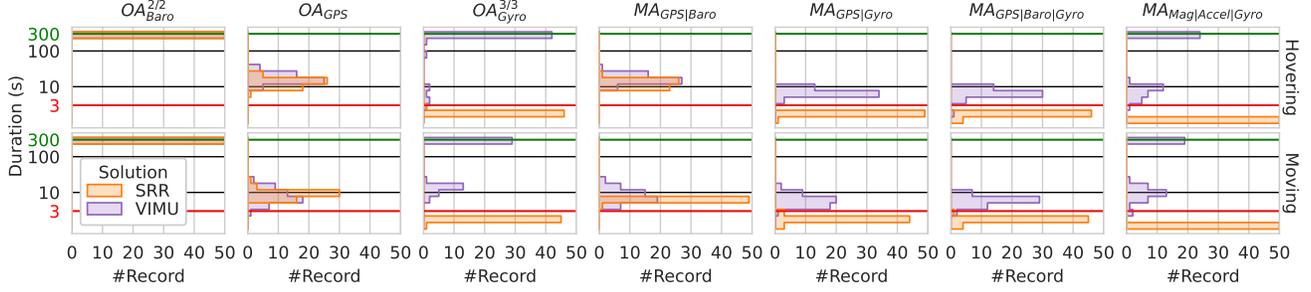


Figure 10. Recovery Duration Histogram on Flight Records

sensors. If only the barometers are unavailable ( $OA_{Baro}^{2/2}$ ), the autopilot replaces the altitude source with GPS, and both approaches can achieve a recovery duration close to the upper bound  $T_{rec}$ . In contrast, the autopilot of the testbed drone has to rely on the reference states provided by the physical model (or SRR’s linear model) in the attack  $OA_{GPS}$  and  $OA_{Gyro}^{3/3}$  because no other replacement is available for these compromised sensors. Therefore, the recovery durations in  $OA_{GPS}$  and  $OA_{Gyro}^{3/3}$  depend heavily on the precision of the physical model. Specifically, VIMU and SRR have similar recovery durations in  $OA_{GPS}$ . The GPS primarily measures the position and velocity states. Both our physical model and SRR’s linear model are sufficient to predict their changes over time. However, the recovery durations diverge in  $OA_{Gyro}^{3/3}$  and multi-type sensor attacks involving gyroscopes, because the attitude changes measured by the gyroscope are nonlinear and cannot be accurately predicted by a linear model. To reduce the error in attitude and angular velocity, SRR employs a *supplementary compensation* (SC) mechanism that derives the system attitude from the measurements of the accelerometer and magnetometer [5]. SRR activates SC when all gyroscopes are under attack ( $OA_{Gyro}^{3/3}$ ). Despite this effort, its effective recovery duration in  $OA_{Gyro}^{3/3}$  remains less than 3 seconds. In contrast, VIMU succeeds in much longer recovery durations without the SC mechanism, even surpassing the upper bound  $T_{rec}$  in many flight records.

To further investigate the effect of SRR’s SC mechanism, we analyze the attitude estimation and its ground truth before and after the activation of  $OA_{Gyro}^{3/3}$  attack. Fig. 11 presents the roll and pitch angle estimated by SRR (with and without SC) and VIMU. The roll estimate of SRR without SC diverges immediately since the attack starts (red area), indicating that the linear model of SRR is insufficient to capture the non-linearity in attitude changes. The estimations given by SC also deviate from the ground truth to a certain extent. Such deviation could be due to the acceleration-based estimation requiring several seconds to approach a steady estimate [17]. In contrast, the attitude estimated by VIMU is closer to the ground truth, even without the supplementary compensation.

**5.3.2. Comparison with SAVIOR on Physical Model.** Our physical model retrofits SAVIOR’s nonlinear physical model [22] with a more accurate specification in estimating motor thrust and aerodynamic drag on the airframe. Fig. 6 has depicted the physical model’s contribution to the detec-

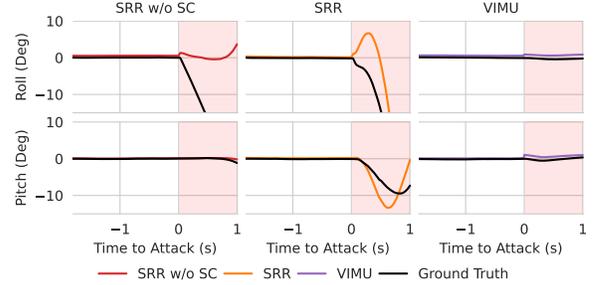


Figure 11. Effect of Supplementary Compensation in Attitude Recovery

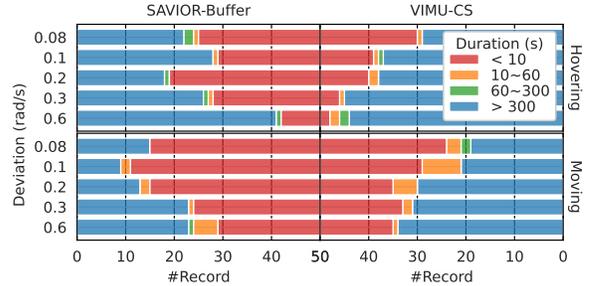


Figure 12. Recovery Durations (VIMU-CS vs. SAVIOR-Buffer)

tion effectiveness. Here, we further investigate its contribution to recovery effectiveness. Considering the effect of the FIFO buffer, we compare SAVIOR-Buffer with VIMU-CS regarding the recovery duration in attack  $OA_{Gyro}^{3/3}$ . Fig. 12 presents the results. We exclude the deviation settings at 0.04 and 0.06 rad/s because neither approach can detect these attacks to launch the recovery. For the rest deviation settings, we observed that VIMU-CS outperforms SAVIOR-Buffer in the average recovery duration by 8%~123%, w.r.t. different attack deviations. Since SAVIOR-Buffer and VIMU-CS have the same detector (and parameters) and FIFO buffers, the differences in recovery duration reflect the discrepancy between the physical models.

**5.3.3. Real World Test.** We demonstrate VIMU’s recovery effect on the real-world quadcopter by injecting an  $OA_{Gyro}^{3/3}$  attack. In the test, the drone took off from home and hovered at the preset altitude. Then, we launched the  $OA_{Gyro}^{3/3}$  attack on all three gyroscopes by injecting constant deviation (0.60 rad/s) to disrupt the gyroscopes’ roll rate measurements. The drone with VIMU’s detection and recovery prevented the immediate crash and maintained the flight attitude for

TABLE 4. RUNTIME OVERHEADS WITH ONBOARD CPU

		Armed		Hover		Moving	
		PX4	VIMU	PX4	VIMU	PX4	VIMU
CPU (%)	Mean	51.34	65.24	50.94	69.77	51.61	69.10
	Peak	57.00	68.54	52.46	70.67	52.65	69.84
RAM (%)	Peak	57.73	67.68	71.96	82.66	71.96	82.66

around 26.1 seconds<sup>6</sup>. As a comparison, we conducted another flight with VIMU but disabled VIMU’s recovery. Without the recovery, the drone immediately deviated from its hovering position and crashed<sup>7</sup>.

*Comparison on TTD.* The real-world IMU data quality causes longer TTD in attack detection as the detector needs more sampling periods to distinguish the attack from sensor noises. To investigate CS-EMA’s advantage over other detectors, e.g., CUSUM, we compare VIMU with VIMU-CS in TTD with the attack  $OA_{Gyro}^{3/3}$  to real-world flights. We did not use SAVIOR because we cannot properly tune SAVIOR’s physical model on our quadcopter. The results show that the CS-EMA detector takes 19.4~20.7 ms to raise the alarm, while the CUSUM detector fails to identify the attack in 1 second, i.e.,  $T_{Alarm}^{Gyro}$ . Moreover, in the same mission flight without applying attack, CS-EMA reports no false positive.

*Energy Consumption and Runtime Overhead.* We perform real flights with our quadcopter. We use the `battery_status` message to evaluate the power consumption during the flight. The gross power of our drone is around 350 watts, mostly (>340 watts) consumed by the actuators. The onboard ARM CPU consumes 0.85 watts at 100% usage. We then evaluate the performance overhead of VIMU with onboard CPU and memory usage. We use the `cpuload` message to record the runtime overhead at the armed, hover, and moving conditions. Table 4 presents the runtime overhead of the autopilot with and without the VIMU deployment. On average of the flight maneuvers, the CPU usage has increased by 15.65% (peak) and 16.74% (mean), and the overall memory has increased by 54 KB. As a result, the energy overhead of VIMU is around 0.15 watts (0.043%), which is negligible compared with the total power consumption.

## 6. Discussion

Our CS-EMA detector alerts sensor attacks faster and has a tight TTD. Although our evaluation mainly focuses on spoofing attacks, these results can generalize to sensor attacks that aim for a denial of service, e.g., [13]. These attacks will introduce large deviations to the sensor measurements, which makes them easily identified by our detector (Section 5.2.1). Besides, such an advantage also benefits the development of the FIFO buffer (Section 3.4), as TTD positively correlates with  $T_{buf}$  in (17). By the definition of  $size_{buffer}$  in (17), lower TTD results in a reduced memory cost to implement the buffers. Given the results in Fig. 7, we

<sup>6</sup><https://www.youtube.com/watch?v=IVZsC0wPPA8>

<sup>7</sup>[https://www.youtube.com/watch?v=WaUyQXz\\_tIE](https://www.youtube.com/watch?v=WaUyQXz_tIE)

set  $T_{buf} = 500$  ms for VIMU, which is sufficient to cover the TTD in most cases. For the related PBAR approach SRR [5], a much larger FIFO buffer is needed (4x to 10x per IMU compared with VIMU) because SRR’s detector encounters numerous ineffective alarms ( $TTD \geq T_{Alarm}$ ). For SAVIOR-Buffer used in  $OA_{Gyro}^{3/3}$  ( $\geq 0.08$ ), its FIFO buffer size can generally equal to VIMU’s buffer size, since the detectors of SAVIOR and VIMU have their respective merit on these attack cases in Fig. 7. We have examined the FIFO buffer’s contribution to the recovery duration. After using the buffer safeguard, the average recovery duration increases by 66%~746%, w.r.t. different TTDs (Appendix C.1). Moreover, the effectiveness of EMA as a low-pass filter offers the potential for applying other forms of low-pass filter, e.g., wavelet transform [4], to our detection scenario to distinguish persistent deviation from the high-frequency measurement noise.

## 7. Conclusion

As an effective and efficient PBAR approach against sensor attacks, VIMU uses an accurate nonlinear physical model to estimate the system states for the sensor-compromised autopilot. VIMU’s anomaly detector deploys a CS-EMA detector to improve detection effectiveness and reduce the detection time delay on high-rate gyroscope attacks. With a properly learned physical model and selected detector parameters, VIMU outperforms the state-of-the-art PBAD and PBAR approaches regarding the effectiveness and efficiency of detection and system-state recovery.

## References

- [1] E. Aggarwal, M. Karimibiuki, K. Pattabiraman, and A. Ivanov, “CORIGIDS: A correlation-based generic intrusion detection system,” in *CPS-SPC@CCS ’18*. ACM, 2018, pp. 24–35.
- [2] W. Aoudi, M. Iturbe, and M. Almgren, “Truth will out: Departure-based process-level detection of stealthy attacks on control systems,” in *CCS ’18*. ACM, 2018, pp. 817–831.
- [3] M. Bangura and R. E. Mahony, “Nonlinear dynamic modeling for high performance control of a quadrotor,” in *Australasian Conference on Robotics and Automation*, 2012.
- [4] C. Callegari, S. Giordano, M. Pagano, and T. Pepe, “WAVE-CUSUM: improving CUSUM performance in network anomaly detection by means of wavelet analysis,” *Comput. Secur.*, vol. 31, no. 5, pp. 727–735, 2012.
- [5] H. Choi, S. Kate, Y. Aafer, X. Zhang, and D. Xu, “Software-based realtime recovery from sensor attacks on robotic vehicles,” in *RAID’20*. USENIX Association, 2020, pp. 349–364.
- [6] H. Choi, W. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, “Detecting attacks against robotic vehicles: A control invariant approach,” in *CCS’18*. ACM, 2018, pp. 801–816.
- [7] P. Dash, G. Li, Z. Chen, M. Karimibiuki, and K. Pattabiraman, “Pid-piper: Recovering robotic vehicles from physical attacks,” in *DSN’21*. IEEE, 2021, pp. 26–38.
- [8] F. Fei, Z. Tu, D. Xu, and X. Deng, “Learn-to-recover: Retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks,” in *ICRA’20*. IEEE, 2020, pp. 7358–7364.
- [9] K. Fu and W. Xu, “Risks of trusting the physics of sensors,” *Commun. ACM*, vol. 61, no. 2, pp. 20–23, 2018.

- [10] M. Gao, L. Zhang, L. Shen, X. Zou, J. Han, F. Lin, and K. Ren, "Exploring practical acoustic transduction attacks on inertial sensors in mdof systems," *IEEE Transactions on Mobile Computing*, 2023.
- [11] J. Giraldo, D. I. Urbina, A. A. Cárdenas, J. Valente, M. A. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 76:1–76:36, 2018.
- [12] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization 2nd Edition*. SIAM, 2009.
- [13] J. Jang, M. Cho, J. Kim, D. Kim, and Y. Kim, "Paralyzing drones via EMI signal injection on sensory communication channels," in *NDSS'23*. The Internet Society, 2023.
- [14] J. Jeong, D. Kim, J. Jang, J. Noh, C. Song, and Y. Kim, "Un-rocking drones: Foundations of acoustic injection attacks and recovery thereof," in *NDSS'23*. The Internet Society, 2023.
- [15] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via GPS spoofing," *J. Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.
- [16] H. Kim, R. Bandyopadhyay, M. Ozmen, Z. Celik, A. Bianchi, Y. Kim, and D. Xu, "A systematic study of physical sensor attack hardness," in *IEEE S&P'24*. IEEE, 2024, pp. 142–142.
- [17] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors and accelerometers: State estimation with an improved dynamic model," *IEEE Control Systems Magazine*, vol. 34, no. 1, pp. 28–41, 2014.
- [18] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 13:1–13:33, 2011.
- [19] K.-Y. Nam, W.-T. Lee, C.-M. Lee, and J.-P. Hong, "Reducing torque ripple of brushless dc motor by varying input voltage," *IEEE Transactions on Magnetics*, vol. 42, no. 4, pp. 1307–1310, 2006.
- [20] S. Nashimoto, D. Suzuki, T. Sugawara, and K. Sakiyama, "Sensor con-fusion: Defeating kalman filter in signal injection attack," in *AsiaCCS'18*. ACM, 2018, pp. 511–524.
- [21] J. Noh, Y. Kwon, Y. Son, H. Shin, D. Kim, J. Choi, and Y. Kim, "Tractor beam: Safe-hijacking of consumer drones with adaptive GPS spoofing," *ACM Trans. Priv. Secur.*, vol. 22, no. 2, pp. 12:1–12:26, 2019.
- [22] R. Quinonez, J. Giraldo, L. E. Salazar, E. Bauman, A. A. Cárdenas, and Z. Lin, "SAVIOR: securing autonomous vehicles with robust physical invariants," in *29th USENIX Security Symposium*. USENIX Association, 2020, pp. 895–912.
- [23] H. Sathaye, M. Strohmeier, V. Lenders, and A. Ranganathan, "An experimental study of GPS spoofing and takeover attacks on uavs," in *31st USENIX Security Symposium*. USENIX Association, 2022, pp. 3503–3520.
- [24] T. Setati, N. Botha, and J. M. Roux, "Experimental approach to calculate the moments of inertia of a hexacopter unmanned aerial vehicle," in *MATEC Web of Conferences*, vol. 370. EDP Sciences, 2022, p. 05001.
- [25] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under GPS spoofing," in *29th USENIX Security Symposium*. USENIX Association, 2020, pp. 931–948.
- [26] M. D. Shuster, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.
- [27] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security Symposium*. USENIX Association, 2015, pp. 881–896.
- [28] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *CCS'11*. ACM, 2011, pp. 75–86.
- [29] T. Tomic, P. Lutz, K. Schmid, A. Mathers, and S. Haddadin, "Simultaneous contact and aerodynamic force estimation (s-cafe) for aerial robots," *Int. J. Robotics Res.*, vol. 39, no. 6, 2020.
- [30] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "WALNUT: waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks," in *EuroS&P'17*. IEEE, 2017, pp. 3–18.
- [31] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors," in *27th USENIX Security Symposium*. USENIX Association, 2018, pp. 1545–1562.
- [32] Z. Tu, F. Fei, M. Eagon, D. Xu, and X. Deng, "Flight recovery of mavs with compromised IMU," in *IROS'19*. IEEE, 2019, pp. 3638–3644.
- [33] D. I. Urbina, J. A. Giraldo, A. A. Cárdenas, N. O. Tippenhauer, J. Valente, M. A. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *CCS'16*. ACM, 2016, pp. 1092–1105.
- [34] C. Wang, B. Song, P. Huang, and C. Tang, "Trajectory tracking control for quadrotor robot subject to payload variation and wind gust disturbance," *J. Intell. Robot. Syst.*, vol. 83, no. 2, pp. 315–333, 2016.
- [35] W. H. Woodall and B. M. Adams, "The statistical design of cusum charts," *Quality Engineering*, vol. 5, no. 4, pp. 559–570, 1993.
- [36] M. H. Wright, "Direct search methods: Once scorned, now respectable," *Pitman Research Notes in Mathematics Series*, pp. 191–208, 1996.
- [37] L. Zhang, X. Chen, F. Kong, and A. A. Cárdenas, "Real-time attack-recovery for cyber-physical systems using linear approximations," in *RTSS'20*. IEEE, 2020, pp. 205–217.
- [38] L. Zhang, P. Lu, F. Kong, X. Chen, O. Sokolsky, and I. Lee, "Real-time attack-recovery for cyber-physical systems using linear-quadratic regulator," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, pp. 79:1–79:24, 2021.

## Appendix A. Physical Parameters Determination

The physical model described in Section 3.1 applies to any commercial quadcopter. Although the parameters of the physical model may vary by the airframe specification, the learning process usually only needs once for each airframe type, and the specific parameters will be deployed as default drone product settings. This section determines the physical parameters  $\mathcal{P}$  in (1). Specifically, we divide  $\mathcal{P}$  into  $\mathcal{P}_{\text{measure}} = \{\mathcal{M}, l, \mathbf{I}, C_T, C_Q, \bar{T}_{\text{Min}}, T_{\text{Range}}, V_{\text{ref}}, R_{\text{int}}\}$  and  $\mathcal{P}_{\text{learn}} = \{C_{Q,r}, t_{DC}, C_m, C_{bxy}\}$ .

**Determination of  $\mathcal{P}_{\text{measure}}$ .** The drone mass  $\mathcal{M}$  and the distance  $l$  between the motor and the airframe's CoG are directly measured on the specific drone airframe. We determine the diagonal inertia terms  $I_{xx}$ ,  $I_{yy}$ , and  $I_{zz}$  of the inertia matrix  $\mathbf{I}$  through torsional pendulum tests [24]. We omit the non-diagonal elements in  $\mathbf{I}$  by assuming the airframe is symmetrical [22], [29]. To ensure the model precision, we measure  $C_T$ ,  $C_Q$ , and  $\bar{T}_{\text{Min}}$  with a static test stand<sup>8</sup>. Our drone uses PWM-based motor controllers, which give us  $\bar{T}_{\text{Range}} = 1000$  us. We read  $R_{\text{int}}$  from a RadioLink CB86-PLUS balance charger. For the 6-cell LiPo battery used by our drone, the full capacity voltage of each cell is 4.05 V, and we have  $V_{\text{ref}} = 24.3$  V.

<sup>8</sup><https://www.youtube.com/watch?v=O6upmU9ubPQ>

TABLE 5. PHYSICAL PARAMETERS IN SIMULATION AND REAL-WORLD

Param	VIMU (Simulation)	VIMU (Real-world)	Unit
$\mathcal{M}$	0.80	2.64	kg
$l$	0.165	0.288	m
$\mathbf{I}$	$\text{diag}\{5.0, 5.0, 9.0\} \cdot 10^{-3}$	$\text{diag}\{5.17, 5.50, 7.62\} \cdot 10^{-2}$	kg·m <sup>2</sup>
$C_T$	4.0	23.0	N
$C_Q$	0.05	0.44	N·m
$C_{Q,r}$	0.0	0.061	N·m/s
$t_{DC}$	0.005	0.035	s
$\bar{T}_{Min}$	1000	840	us
$\bar{T}_{Range}$	1000	1000	us
$V_{ref}$	N/A	24.3	V
$R_{int}$	N/A	0.072	$\Omega$
$C_m$	0.001	0.031	1/s
$C_{bxy}$	[0.022, 0.022, 0]	[0.161, 0.145, 0]	m <sup>2</sup> /kg

*Determination of  $\mathcal{P}_{learn}$ .* Based on (2), we fit the physical parameters with known system states  $\mathbf{x}(t)$  and control inputs  $\mathbf{u}(t)$  from the normal flight records. We learn  $C_{Q,r}$ ,  $t_{DC}$ ,  $C_m$ ,  $C_{bx}$  and  $C_{by}$  of  $C_{bxy}$  through the Nelder-Mead [36] searching over the nonlinear least squares data fitting problem [12].

Given the input data  $X$ , the fitting target  $Y$ , the differential equations  $F(\cdot)$  describing the physical model, and the parameter set  $\mathcal{P}$ , the nonlinear least squares method decides the optimal parameters by minimizing the squared error:

$$\min_{\mathcal{P}} \sum_{t=1}^T (F_t(X_t; \mathcal{P}) - Y_t)^2 \quad (21)$$

where  $F_t(X_t; \mathcal{P})$  is the state at time  $t$  predicted by  $F(\cdot)$ .  $Y_t$  is the estimated state or sensor measurement from the flight record w.r.t. the prediction result  $F_t(X_t; \mathcal{P})$ .

Before deciding the specific parameters of  $\mathcal{P}$ , we first follow the guides<sup>9,10</sup> to obtain the following data: 1) sensor measurements  $\tilde{\rho}$ ,  $\tilde{\omega}$ , and body acceleration  $\tilde{\mathbf{a}}^B$ , 2) estimated states  $\hat{\mathbf{q}}$ ,  $\hat{\mathbf{v}}$ , and  $\hat{\mathbf{v}}_w$ , 3)  $\mathbf{u}(t)$  and 4)  $V_{load}$  and  $i_{load}$ . This data collection process is conducted for one time on the standard instance of the airframe model. Here, we used the quaternion form of system attitude, i.e.,  $\mathbf{q} = [q_w, q_i, q_j, q_k]^T$ , as the substitute of the rotation matrix  $\mathbf{R}$  in Section 3.1. The conversion between the quaternion and  $\mathbf{R}$  is straightforward [26]. Using  $\mathbf{q}$  instead of  $\mathbf{R}$  can reduce the computational cost of learning because it is commonly used by the autopilots, e.g., PX4 and ArduPilot.

According to [17], the accelerometer measures the specific acceleration  $\tilde{\mathbf{a}}^B = [\tilde{a}_x, \tilde{a}_y, \tilde{a}_z]^T$ . Indeed,  $\tilde{\mathbf{a}}^B$  stands for the difference between the vehicle's acceleration ( $\mathbf{a}$  in (5)) and gravitational acceleration ( $g\mathbf{e}_3$ ). This measurement is already located in the FRD frame and thus requires no rotation to align with  $\mathbf{a}_c$  and  $\mathbf{a}_d$ . From that, we have  $\tilde{\mathbf{a}}^B = \mathbf{a}_c + \mathbf{a}_d$ . Since  $\mathbf{a}_c$  is generally vertical to the x-axis and y-axis,  $\tilde{a}_x$  and  $\tilde{a}_y$  are dominated by the drag acceleration  $\mathbf{a}_d$ . Therefore, we use  $\tilde{a}_x$  and  $\tilde{a}_y$  to solve  $C_m$ ,  $C_{bx}$ , and  $C_{by}$  with (11). We first derive the relative airspeed  $\mathbf{v}_r$  from the estimated  $\hat{\mathbf{q}}$ ,  $\hat{\mathbf{v}}$ , and  $\hat{\mathbf{v}}_w$ . Then, we calculate the vector norm  $\|\mathbf{v}_r\|$  in (11). After that, we replace  $\mathbf{a}_d$  with  $[\tilde{a}_x, \tilde{a}_y]$  of  $\tilde{\mathbf{a}}^B$  and use  $\|\mathbf{v}_r\|$ ,  $\tilde{\rho}$ , and the x-axis and y-axis components of  $\mathbf{v}_r$  to solve the

<sup>9</sup>[https://docs.px4.io/main/en/advanced\\_config/tuning\\_the\\_ecl\\_ekf.html](https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html)

<sup>10</sup><https://docs.px4.io/main/en/config/autotune.html>

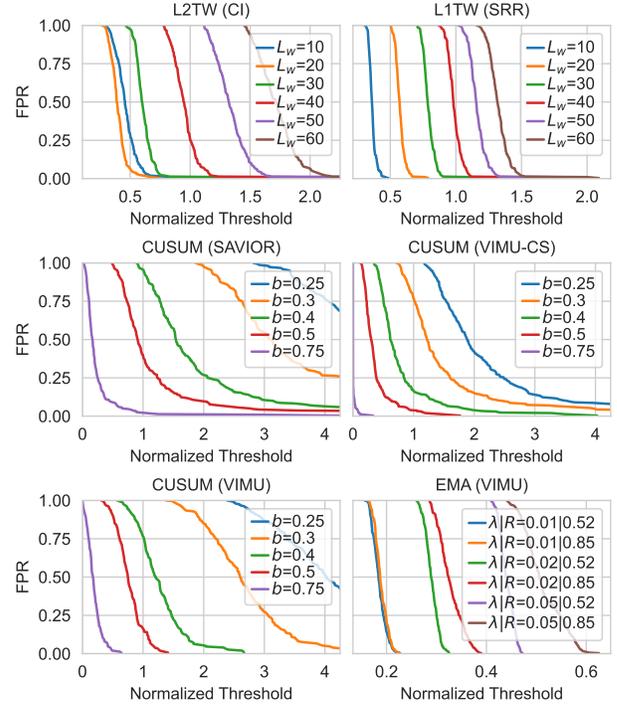


Figure 13. FPR-Threshold Curves of Gyroscope Detectors at Different Parameters

equation. In short, we consider  $X_1 = \{\hat{\mathbf{q}}, \hat{\mathbf{v}}, \hat{\mathbf{v}}_w, \tilde{\rho}\}$  as input data and  $Y_1 = \{\tilde{a}_x, \tilde{a}_y\}$  as the fitting target in (11) to resolve  $C_m$ ,  $C_{bx}$ , and  $C_{by}$ . We compute the estimate of  $\mathbf{a}_d$  and use it to derive  $\mathbf{a}_c = \tilde{\mathbf{a}}^B - \mathbf{a}_d$ . Then, we solve (7), (9), and (10) to obtain  $t_{DC}$  with  $X_2 = \{\mathbf{u}(t), V_{load}, i_{load}\}$  as input data and  $Y_2 = \mathbf{a}_c$  as the fitting target. Finally, we solve  $C_{Q,r}$  in (8) with the input data  $X_3 = \{\mathbf{u}(t), \tilde{\omega}, V_{load}, i_{load}\}$  and the fitting target  $Y_3 = \dot{\omega}(t)$ , where  $\dot{\omega}(t)$  is estimated from the angular velocity  $\tilde{\omega}$  at timestamp  $t$  and  $t - 1$ .

## Appendix B. Detector Parameter Selection

We select the most suitable detector parameters based on two criteria: 1) the FPR in the survey mission of QGround-Control or real flight, and 2) the theoretical TTD at possible attack deviations. The survey mission contains multiple waypoints in its task route. It covers flight maneuvers that any drone will encounter in common flight scenarios, e.g., takeoff, hover, cruise, and turn. The diversity in flight maneuvers qualifies the survey mission for parameter selection, as the detector performance in attack-free flight is mainly affected by the measurement noise and modeling error.

We first collect the flight data for parameter selection. For each approach, we use the survey mission to collect 100 attack-free flight records. During this process, we turn off the isolation and recovery functionality to ensure the autopilot can complete the mission. For each sensor instance, we calculate the in-flight residuals to construct the validation dataset. Then, we determine the candidate detector parameter settings (e.g., different value pairs of  $(\lambda, R)$  for VIMU's

TABLE 6. DETECTOR PARAMETERS

Detector	LITW		L2TW		CUSUM		CS-EMA				
	$\tau$	$L_w$	$\tau$	$L_w$	$\tau$	$b_i$	$\tau_{cs}$	$b_i$	$\tau_{ema}$	$\lambda$	$R$
Measurement	1.15	10	1.402	10	3.0	0.50	3.0	0.50	0.45	0.01	0.85
GPS Position	3.10	10	4.42	10	3.5	1.00	3.5	1.00	0.50	0.01	1.10
GPS Velocity	0.10	10	0.02	10	3.0	0.25	3.0	0.25	0.15	0.05	0.52
Barometer	0.35	10	0.2	10	3.0	0.25	3.0	0.25	0.30	0.01	0.52
Magnetometer	0.464	10	0.65	20	3.0	0.50	3.0	0.50	0.25	0.01	0.85
Gyroscope	5.20	10	30.25	10	3.0	1.00	3.0	1.00	0.95	0.01	1.10
Accelerometer											

TABLE 7. IMPACT OF FPR STANDARD ON PARAMETER SELECTION

$\tau$ (Actual FPR under $\tau$ )	CUSUM (SAVIOR)			CUSUM (VIMU)		
	1%	3%	5%	1%	3%	5%
$b = 0.25$	N/A	N/A	N/A	N/A	N/A	N/A
$b = 0.3$	N/A	N/A	N/A	4.5 (2%)	4.5 (2%)	4.5 (2%)
$b = 0.4$	N/A	N/A	5.5 (5%)	3.0 (0%)	3.0 (0%)	3.0 (0%)
$b = 0.5$	N/A	5.5 (3%)	3.0 (4%)	3.0 (0%)	3.0 (0%)	3.0 (0%)
$b = 0.75$	3.0 (1%)	3.0 (1%)	3.0 (1%)	3.0 (0%)	3.0 (0%)	3.0 (0%)

EMA detector) and use the validation dataset to decide their corresponding detection threshold (e.g.,  $\tau_{ema}$  for VIMU’s EMA detector). The specific steps are as follows.

- 1) For each setting of candidate detector parameters, we use the validation set of each flight to figure out the *minimum false-alarm-avoidance threshold* ( $\tau_{avoid}$ ) of the flight. We rank the  $\tau_{avoid}$  of the 100 flights in ascending order to obtain one curve in a subfigure of Fig. 13. Due to the ascending order of  $\tau_{avoid}$ , the vertical axes of Fig. 13 represent the FPR of the gyroscope detector configured with one flight’s  $\tau_{avoid}$  and the specific candidate detector parameters when applied on other flights. We call this curve *FPR-Threshold curve* w.r.t. specific candidate detector parameters.
- 2) We define the fifth largest  $\tau_{avoid}$  on each FPR-Threshold curve as the *reference detection threshold* ( $\tau_{ref}$ ) of each candidate detector parameter setting. In other words, the detector equipped with  $\tau_{ref}$  should cause no more than five false alarms in the 100 missions.
- 3) We take different candidate detector parameter settings and the corresponding  $\tau_{ref}$  to configure the gyroscope detector and figure out the TTDs under these detector configurations at various attack deviations. Then, we confirm the most appropriate detector parameter setting (e.g.,  $(\lambda, R) = (0.01, 0.85)$  for the EMA detector) and the corresponding  $\tau_{ref}$ . We add a *safety margin* to this  $\tau_{ref}$  to obtain the final detection threshold  $\tau$ . Specifically,  $\tau = 1.05 \cdot \tau_{ref}$ .

Following the above procedure, Table 6 presents the detector parameters used in the experiments in Section 5. Note that, to conform with the autopilot implementation, we scale the residual defined in (12), detection thresholds  $\tau$ , mean shifts  $b_i$ , and  $R$  to the standard deviation by dividing them by the noise parameters.

*Rationality of 5%-FPR standard to decide  $\tau_{ref}$ .* The safety margin added onto the reference threshold  $\tau_{ref}$  ensures that the final detection threshold  $\tau$  has an FPR much lower than  $\tau_{ref}$ . Since the detector’s sensitivity could be affected by this FPR standard used to decide  $\tau_{ref}$ , we discuss how the variation of this FPR standard can affect the final  $\tau$ . Taking the CUSUM gyroscope detectors as an example, Table 7 presents the final detection thresholds  $\tau$  decided by different FPR standards (1%, 3%, and 5%). N/A means the obtained threshold is ineffective in identifying overt attacks, thus not

applicable for attack detection. Compared with SAVIOR, most thresholds of VIMU’s CUSUM component can achieve an actual FPR of 0.

*Deciding Detector Parameters for Maneuver Scenario and Real-World Drone.* The *Maneuver* scenario involves several drastic attitude changes. Therefore, we estimate the impact on detector parameter selection. We confirmed that only the parameters of gyroscope detectors need to be reconfigured. Compared with Fig. 13, the FPR-Threshold curves in Fig. 14 show higher FPR at the same threshold, indicating that drastic maneuvers cause more false alarms during the attack-free flight. Meanwhile, Fig. 14 shows VIMU’s CUSUM component keeps an FPR lower than SAVIOR at the same detector parameters ( $\tau_{avoid}, b_i$ ), which means our solution has fewer false alarms than the SAVIOR. Following the same selection procedure, we adjust the  $\tau_{ema}$  of gyroscope detectors from 0.25 to 0.32. The detection effectiveness in this scenario is presented in Appendix C.2. On the other hand, we take this procedure to decide the detector parameters for the real-world quadcopter. We fly the quadcopter to collect 100 attack-free flight logs and use these logs to decide the detector parameter. For the gyroscope detector, the specified parameters are  $\tau_{cs} = 3.0$ ,  $b_i = 0.75$ ,  $\tau_{ema} = 0.22$ ,  $R = 0.52$ , and  $\lambda = 0.075$ . Such parameters of the CS-EMA and CUSUM detectors are deployed on our quadcopter to obtain the real-world TTD evaluation results in Section 5.3.3.

*Sensitivity of  $\lambda$ .* The value of  $\lambda$  in (15) affects the detector performance. We discuss the sensitivity of  $\lambda$  with real-world flight. With fixed  $\tau_{ema}$  and  $R$ , Fig. 15 presents the real-world impact of  $\lambda$  selection on the FPR and the TTD of  $OA_{Gyro}^{3/3}$  (0.60). It indicates that the  $\lambda$  selection is a trade-off between TTD and FPR. We also observed that a larger  $\tau_{ema}$  results in a lower FPR and a longer TTD. Note that in Fig. 15, both the FPR and TTD remain relatively stable when  $\lambda \in [0.05, 0.10]$ , which means in this range of  $\lambda$ , the detector performance is less sensitive to  $\lambda$ , justifying our choice of  $\lambda = 0.075$ .

## Appendix C. Evaluation Results Supplementary

### C.1. Detection and Recovery Effectiveness

Fig. 16 presents the TTD under  $OA_{GPS}$ . VIMU, SAVIOR, and VIMU-CS have similar TTD performance on this attack. The TTDs are generally less than one second; thus, detecting  $OA_{GPS}$  is much simpler than detecting  $OA_{GPS-PV}$ , as can be compared with Fig. 8.

We also evaluate the impact of TTD and FIFO buffer on recovery duration with attack *Hovering* [ $OA_{Gyro}^{3/3}$  (0.60)]. In Fig. 17, the recovery duration of VIMU without the FIFO buffer generally reduces as the TTD increases, indicating that the adversary can cause more damage to the system state in the detections with longer TTD. When evaluating the complete VIMU (with the FIFO buffer), we observed that the average recovery duration increases by 66%~746%,

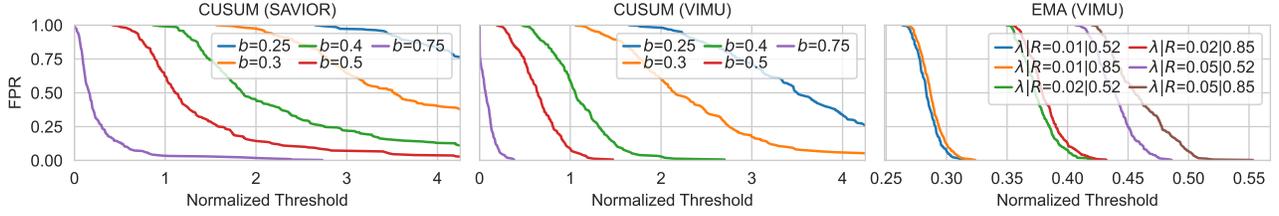


Figure 14. FPR-Threshold Curves of Gyroscope Detectors at Different Parameters in *Maneuver* Mission

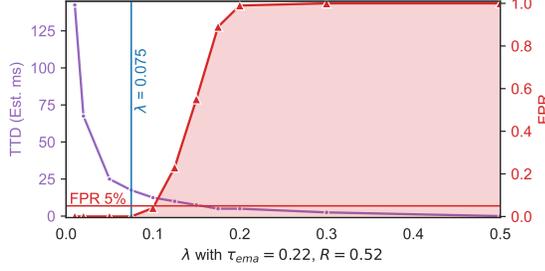


Figure 15.  $\lambda$ 's Sensitivity in Real-World

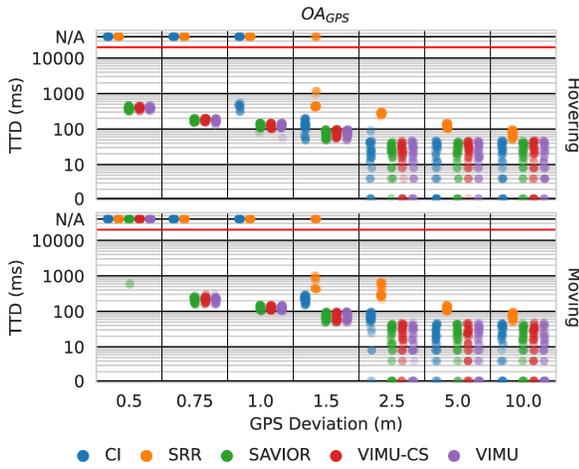


Figure 16. Time to Detect on  $OAGPS$  (Red Line:  $T_{Alarm}^{GPS}$ )

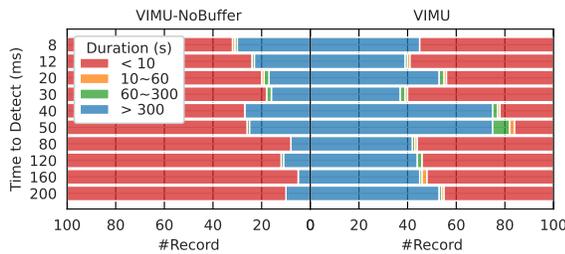


Figure 17. Recovery Duration of VIMU in Different TTDs

w.r.t. different TTDs. The distribution of recovery duration becomes insensitive to TTD with the buffer deployed. These results showed that the FIFO buffer effectively mitigates high-rate sensor attacks.

## C.2. Impact of Drastic Flight Maneuvers

We investigate VIMU's detection effectiveness in the *Maneuver* scenario by comparing it with SAVIOR under the gyroscope attacks. Fig. 18 shows the log-scaled TTDs

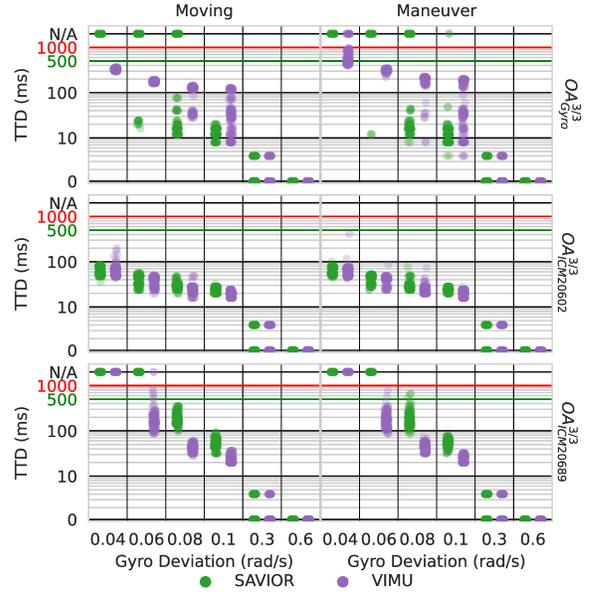


Figure 18. TTD on Overt Gyro Attacks in *Maneuver* Mission

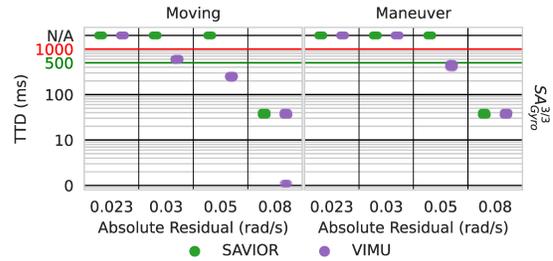


Figure 19. TTD on Stealthy Gyro Attack in *Maneuver* Mission

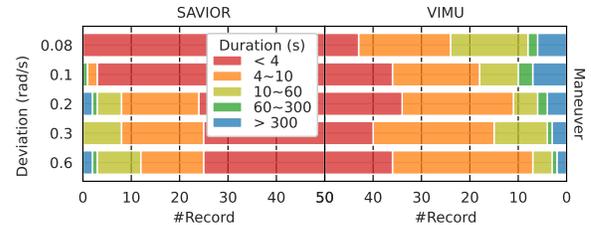


Figure 20. Recovery Duration in *Maneuver* Mission

against overt attacks, while Fig. 19 presents the results against stealthy attacks. In Fig. 18, both approaches exhibit similar TTDs in the *Maneuver* scenario as in the *Moving* scenario. Due to the relaxed detection threshold  $\tau_{ema}$ , VIMU's TTD in the *Maneuver* scenario is slightly longer than in the *Moving* scenario. Such threshold relaxation also makes VIMU hard to detect the stealthy attack  $SA_{Gyro}^{3/3}(0.03)$

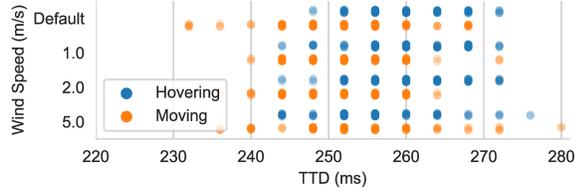


Figure 21. TTD on  $SA_{Gyro}^{3/3}$  at Different Wind Speeds

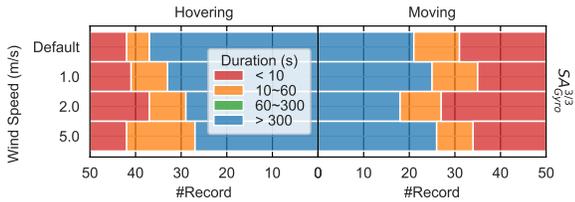


Figure 22. Recovery Duration at Different Wind Speeds

in the *Maneuver* scenario, as shown in Fig. 19. In general, VIMU’s advantage in detection effectiveness remains unaffected despite the maneuvers. We also observed similar comparative advantages in recovery duration. The original SAVIOR does not contain a recovery component, so we extend SAVIOR with the recovery monitor described in Section 3.3. As presented in Fig. 20, even with the maneuvers, VIMU outperforms SAVIOR in recovery duration, which validates the contribution of our fine-grained physical model and proposed buffer safeguard.

### C.3. External Disturbance

External disturbances, e.g., wind gusts, disturb the drone in different ways [29], and they can raise false alarms that will disrupt the recovery of the system states. Although the autopilot can compensate on these disturbances, it relies upon a precise state estimate, which validates our usage of the wind velocity term in our aerodynamic drag model ( $v_w$  in (11)). The wind speed data required to calculate this term is measurable by an anemometer [22] or estimated by the drag acceleration [17]. However, we encounter difficulties in equipping the quadcopter with a wind-speed sensor and integrating it with the autopilot board to measure the wind in flight. Therefore, we use simulation to evaluate VIMU’s performance in the presence of wind disturbances.

We configure jMAVSIM with different wind speeds and compare the results with the default wind setting. In jMAVSIM, the wind velocity starts from a base vector  $\mu$  and follows a random walk process subjected to  $\mathcal{N}(0, \sigma^2)$ . By default,  $\mu = [0.0, 0.0, 0.0]$  m/s and  $\sigma = [6.0, 8.0, 0.0]$  m/s in the NED frame. We inject the north-axis wind with  $\mu' = [1.0/2.0/5.0, 0.0, 0.0]$  m/s for the wind speeds of 1 m/s, 2 m/s, and 5 m/s, respectively. Then, we evaluate how the wind disturbance impacts the TTD and the effective recovery duration in stealthy attack  $SA_{Gyro}^{3/3}$ . Fig. 21 shows that the TTD in  $SA_{Gyro}^{3/3}$  remains steady at different wind speeds. In Fig. 22, we only observed a slight decrease in average recovery duration (177.8 s at default setting, 178.0 s at 1.0 m/s, 145.8 s at 2.0 m/s, and 163.4 s at 5.0 m/s). In *Moving*

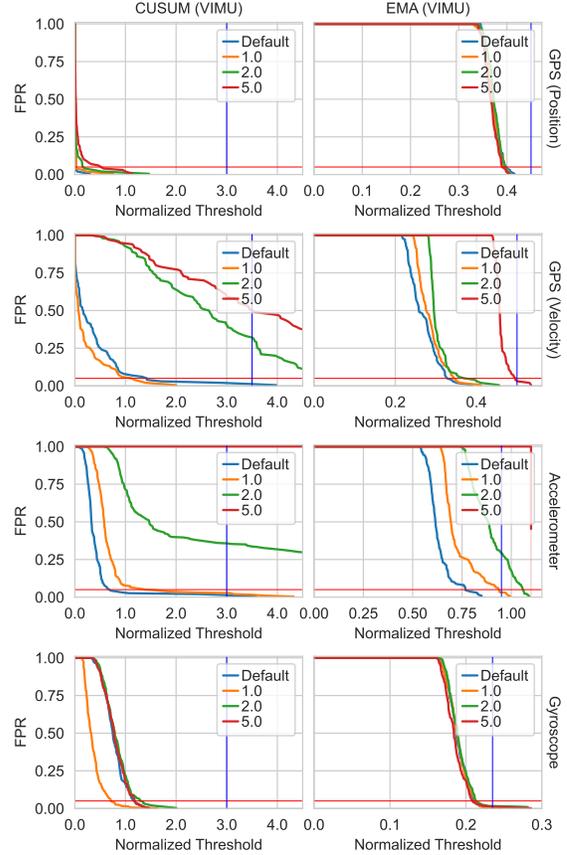


Figure 23. FPR-Threshold Curves at Different Wind Speeds (Red Line: 5% FPR ; Blue Line: Optimal Threshold w.r.t. Optimal Detector Parameters)

missions, VIMU even achieves a longer recovery duration in the higher wind speeds, indicating wind disturbance is not a dominant factor in the recovery of these tasks.

Because adapting different detector parameters to various wind speeds is challenging, we examine the generalizability of the optimal detector parameters obtained at the default wind setting to the wind disturbance environments. Specifically, we use these parameters to plot the FPR-Threshold curves at different wind speeds (following Appendix B). Fig. 23 presents the FPR-Threshold curves for GPS and IMU. The FPRs on the gyroscope and GPS position detectors demonstrate that for both CUSUM and EMA detectors, the optimal detector parameters at the default wind setting can tolerate the windy environments, i.e., all curves meet the 5% FPR requirements specified in the parameter selection (Appendix B). In contrast, the FPRs on the accelerometer and GPS velocity detectors show that the false positives increase drastically with wind disturbance. At 2.0 m/s and 5.0 m/s wind speeds, only the EMA detector for GPS velocity can tolerate the wind. The detection parameters of two CUSUM detectors and the EMA detector for the accelerometer need to be reconfigured for wind disturbance.