Concurrent Error Detection in Embedded Digital Control of Nonlinear Autonomous Systems Using Adaptive State Space Checks

Md Imran Momtaz, Chandramouli N Amarnath and Abhijit Chatterjee
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta GA 30332
Email: momtaz@gatech.edu, chandamarnath@gatech.edu, abhijit.chatterjee@ece.gatech.edu

***Abstract*—The advent of pervasive autonomous systems such as self-driving cars and drones has raised questions about their safety and trustworthiness. This is particularly relevant in the event of on-board subsystem errors or failures. In this research, we show how encoded Extended Kalman Filter can be used to detect anomalous behaviors of critical components of nonlinear autonomous systems: sensors, actuators, state estimation algorithms and control software. As opposed to prior work that is limited to linear systems or requires the use of cumbersome machine learned checks with *fixed detection thresholds*, the proposed approach necessitates the use of *time-varying checks with dynamically adaptive thresholds*. The method is lightweight in comparison to existing methods (does not rely on machine learning paradigms) and achieves high coverage as well as low detection latency of errors. A quadcopter and an automotive steer-by-wire system are used as test vehicles for the research and simulation and hardware results indicate the overhead, coverage and error detection latency benefits of the proposed approach.**

I. Introduction

The increased deployment of autonomous vehicles and drones has raised questions regarding the safety and trustworthiness of such systems [1]. Risks in the real-time operation of such nonlinear systems can arise from errors and failures in sensors, actuators and runtime execution of control software [2], [3] due to field stress, part wearout and hostile operating conditions. Such risks, if manifested during real-time system operation can result in abnormal behavior of the system or that of its constituent subsystems (electrical, electro-mechanical, sensors, actuators and relevant controllers). Abnormal behaviors which are *different from nominal system function* are broadly called *anomalies*, and need to be detected with high coverage and low latency to enable quick recovery. We are particularly concerned with applications such as aerial drone maneuvers and subsystems of autonomous vehicles where the *latency of anomaly detection must be extremely small* (fractions of a second) to minimize the risk of accidents. In this context, the key contributions of this research are the following:

Key Contributions:

(a) A new methodology for checking anomalous behaviors in subsystems (sensors, actuators, state estimation algorithms and control program execution) of *nonlinear autonomous systems using encoded Extended Kalman Filter (EKF)*, is developed. The method is scalable to a variety of nonlinear systems, incurs minimal computation overhead and latency and requires no machine learning as compared to [3]. For the first time, state encoding based checks are introduced *both for checking the state transition as well the covariance matrix computation steps* of the Extended Kalman Filter.

(b) As opposed to prior checking schemes with *time-invariant coding schemes and fixed detection thresholds*, the state transition behavior changes from one time step to the next in generalized nonlinear systems (as given by the Jacobian of the state transition function). For this reason, prior encoding techniques *cannot be directly applied to Extended Kalman filter (EKF)*. This fundamental problem is solved in this research by employing a *time-varying encoding scheme with variable detection thresholds*. This further allows the checks to adapt to changing performance sensitivities to error/failure conditions under diverse maneuvers of autonomous systems (e.g. hovering vs. sharp turns for quadcopters), achieving higher error coverage than possible with prior techniques with significantly lower computational overhead. Results on two test cases are demonstrated: a quadcopter and a steer-by-wire system for autonomous vehicles.

II. Prior Research

With regard to *anomaly detection*, there has been work on statistical estimation algorithms for the detection of outliers (anomalies) [4]. There is a significant body of work revolving around prediction of the future observable states of a system and their statistics from the statistics of prior states and comparison with achieved future state (measured) values for anomaly detection [5]–[9]. Recently there has been work on sensor data fusion to identify sensor as well as actuator malfunction in robotic systems [10]. Of particular relevance is prior work on the use of neuromorphic networks for anomaly detection and correction [11], [12]. In [3], [13], [14], past observed sensor measurements and inputs are used to predict a linear encoding of all the system states using machine learning techniques that require extensive training. A hierarchical error detection and error localization scheme is presented in [14]. In [15], state space encoding of linear Kalman filters is proposed for detecting soft errors in Kalman filter computations. The resulting checks in all of the above use fixed (worst case across all input stimulus) detection thresholds. In contrast to the above, the proposed methods are applicable to generic nonlinear systems, do not rely on (expensive) machine learning paradigms and use adaptive checks for high error/failure

978-1-7281-9113-3/20/$31.00

coverage. The core techniques can be extended to Unscented Kalman as well as Particle filters [16].

The primary basis of this research rests on two decades of evolution in the domain of algorithm based fault tolerance first introduced by [17]–[19], whereby fault tolerance of matrix operation and FFT networks was investigated. Later this idea was applied to linear state variable systems with low hardware overhead by Chatterjee and d'Abreu in [20], [21]. Extensions to error detection and correction in linear and nonlinear control systems were further studied in [13], [22]–[26]. In [23], a state space encoding based error checking methodology was developed that could detect soft errors in processors running control software as well as malfunction of sensors and actuators of nonlinear systems. However, only a single centralized checking methodology was employed and diagnosis of errors was not addressed. In [27], the authors looked at concurrent detection of erroneous responses in linear analog circuits, however, the proposed approach cannot be applied directly to nonlinear system with limited controllability and observability. Additionally, the error-detection circuit operates as a low-order duplicate of the original circuit, which can be quite complicated for a regular nonlinear systems requiring higher computation and memory overhead.

III. Preliminaries: Extended Kalman Filter

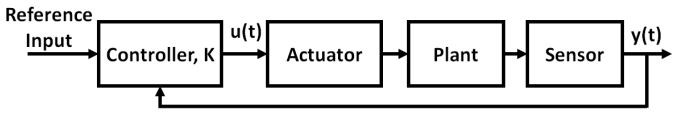

Fig. 1: A nonlinear state space control system

Figure 1 shows the block diagram of a nonlinear state variable system. The plant behavior is expressed in the form of an ordinary differential equation:

$$\dot{x}(t) = f(x(t), u(t)) + w(t) \tag{1}$$

where the vectors $x(t)$ and $u(t)$ represent the system states and inputs, respectively. The function $f(.)$ defines the system dynamics and $w(t)$ represents zero-mean process noise. The output equation of the plant is given by,

$$y(t) = h(x(t), u(t)) + v(t) \tag{2}$$

where $h(.)$ represents an output mapping and $v(t)$ represents zero-mean measurement noise. For both linear and nonlinear state variable systems, the input $u(t)$ is computed by an external controller K that strives to maintain system performance under dynamically changing plant conditions. In this work, we assume that the states and measurements follow Gaussian statistics. Check computation for states and measurements with non-Gaussian statistics is addressed in [16].

In general, due to limited observability of all the internal states of the plant, a recursive state estimator such as the Extended or Unscented Kalman filter is used to estimate the plant states in the presence of measurement noise. In partially observable nonlinear state variable systems, the Extended Kalman Filter (EKF) [28] is used to estimate the observable as well as the unobservable states of the system in the presence of measurement noise under the assumption that the system states have Gaussian distributions. The EKF assumes a system model of the form of Equations 1 and 2. The nonlinear functions $f(.)$ and $h(.)$ are linearized at discrete time instants k and are denoted by the Jacobian matrices F_k and H_k, respectively. The estimated state and covariance matrices are represented by $\hat{x}$ and P, respectively. Q_k and R_k represent covariance matrices for the process and observation noise, respectively. z_k is a vector of measurements performed on the system and corresponds to discretized values of the measured outputs of the system $y(t)$ of Figure 1. In the following, $\hat{x}_{k|k-1}$ and $P_{k|k-1}$ represent the state estimate and covariance matrix at time instant k based on observations up to time instant $k-1$. Similarly $\hat{x}_{k|k}$ and $P_{k|k}$ represent corresponding estimates based on observations up to time instant k.

The EKF works in two steps at every time instant k. During the first *prediction step*, the states and covariance matrix are predicted by $\hat{x}_{k|k-1} = F(\hat{x}_{k-1|k-1}, u_k)$, and $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$ on the basis of knowledge of $\hat{x}_{k-1|k-1}$, u_k, F_k, $P_{k-1|k-1}$ and Q_k. The predicted state and covariance matrices are denoted by x_k^{prior} and P_{prior}, respectively. In the second *update step*, the measurements z_k are used to determine the state residue as $\tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$, the covariance residue S_k is computed as $S_k = H_k P_{k|k-1} H_k^T + R_k$, and the Kalman gain K_k is determined by $K_k = P_{k|k-1} H_k^T S^{-1}$. Using these quantities, the state and covariance matrix are updated using the following equations: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$, $P_{k|k} = (I - K_k H_k) P_{k|k-1}$. For the rest of this work, we denote the state estimates $\hat{x}_{k|k-1}$ and $\hat{x}_{k|k}$ as x_k^{prior} and x_k^{post} and covariance matrices $P_{k|k-1}$ and $P_{k|k}$ as P_k^{prior} and P_k^{post}, respectively. These are used to deduce the proposed checks.

IV. Proposed Approach: State Encoding Based EKF Checks

For error detection, we propose to encode the Jacobian F_k by including an extra check state given by a time-varying linear sum of the system states. To this effect, we introduce a time-varying row vector called the coding vector CV_k, defined as $CV_k = [\alpha_{1k}, \alpha_{2k}, \ldots, \alpha_{nk}]$, where n is the number of states of the system (the mechanism for determining the coding vector CV_k is discussed later). We augment the Jacobian of the system F_k and the system state estimate $\hat{x}_k$ in the following manner:

$$F_{k,aug} = \left[\begin{array}{c|c} F_k & 0 \\ \hline CV_k F_k & 0 \end{array}\right], \; \hat{x}_{k,aug} = \left[\begin{array}{c} \hat{x}_k \\ \hline C_k \end{array}\right]$$

where, CV_k is the coding vector at time k and C_k is defined as $C_k = CV_k x_k$, which is the inner product of CV_k and x_k. As the state vector is augmented, the covariance matrix takes the following form:

$$P_{aug} = \begin{bmatrix} Var(x_1) & cov(x_1, x_2) & \ldots & cov(x_1, C_k) \\ cov(x_1, x_2) & Var(x_2) & \ldots & cov(x_2, C_k) \\ \vdots & \vdots & \vdots & \vdots \\ cov(x_1, C_k) & cov(x_2, C_k) & \ldots & Var(C_k) \end{bmatrix} \tag{3}$$

From these representations, it is clear that the augmented system contains all the computations for the canonical EKF, and hence, it can be considered as a filter augmented with a check state. We propose two checks, namely the state check

and the variance check (explained below) by reusing these EKF computations. The overhead can vary from $\mathcal{O}(1/k)$ to 100% where k is the number of system states, and this depends on the composition of the Jacobian.

A. State Check:

In the computation flow of the EKF, the states are computed during predict (x_k^{prior}) and update (x_k^{post}) steps along with corresponding check states (C_k^{prior} and C_k^{post}). The response of a state variable system is highly correlated in time in the presence of bounded inputs, and hence, for a nominal system, the quantities C_k^{prior} and C_k^{post} are highly correlated as well. As a result, the quantity $C_k^{prior} - C_k^{post}$ is very small (limited by modeling inaccuracies and noise). However, this argument is not true when the system experiences failure. Here, the quantity $C_k^{prior} - C_k^{post}$ exceeds a time-varying threshold (discussed later), which can be used in real-time to detect the presence of failures or errors in the system.

We define the state check $e_x(k)$ as $e_x(k) = C_k^{prior} - C_k^{post}$. $e_x(k)$ is compared against a time-varying threshold to determine whether the system is experiencing failure. The threshold needs to be selected carefully to ensure good 'precision' and 'recall'. For this reason, a statistical bound is computed to determine the threshold and a two-tail statistical test is performed on these bounds to determine whether the system is experiencing anomalies. To make this bound even tighter and to make the state check more responsive to different anomalies, the time varying coding vector CV_k is determined as follows.

1) Determination of Time-Varying Coding Vector

To allow the checks to adapt to changing performance sensitivities to error/failure conditions, the coding vector CV_k is computed as suggested by Lemma 1. Here, $CV_k = [\alpha_{1k}, \alpha_{2k}, \ldots, \alpha_{nk}]$ and $\{\alpha_{ik}\}$ is varied inversely to the variance of the state x_i at time instant k (the latter variance is obtained as a direct by-product of the EKF computations). This reduces the variance of the check quantity $e_x(k)$ allowing a tighter detection threshold at time instant k. Example state check plots for a quadcopter system with constant (as in prior research) and variable coding vectors are shown in Figure 2, where a small error is introduced in the sensor reading and escapes the constant coding vector based check (left), but is detected by the time-varying coding vector based check (right). Let x_k be the vector of states $x_k = [s_{1k}, s_{2k}, .., s_{ik}, .., s_{nk}]^T$, and the variance of the state s_{ik} be σ_{ik}, then we have,

Lemma 1. *Given N independent normally distributed states and elements of coding vectors α_{ik} such that the variance of each $s_{ik} > 1$, the variance of the resultant check state reduces when the coding vectors are produced by $\alpha_{ik} = 1/\sigma_{ik}$ at time step k.*

Proof. From the properties of a normal distribution, for independent and identically distributed states, the variance of $\sum_{i=1}^{N} \alpha_{ik} s_{ik}$ is the sum of the individual variances of each $\alpha_{ik} s_{ik}$, i.e. $Var(C_k) = Var(\sum_{i=1}^{N} \alpha_i s_{ik}) = \sum_{i=1}^{N} \alpha_{ik}^2 \sigma_{ik}^2$ for each $\alpha_{ik}\sigma_{ik}$ being the standard deviation of a given $\alpha_{ik} s_{ik}$. Since each $\alpha_{ik}^2 \sigma_{ik}^2 > 1$ we have $Var(C_k) > N$.

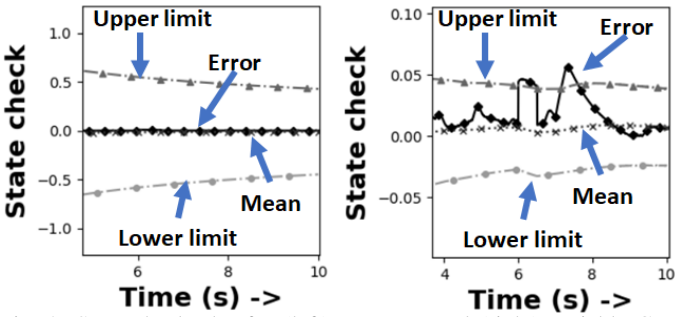

Fig. 2: State check plot for (left) constant, and (right) variable CV for a transient sensor failure in quadcopter

When we scale the states such that $\alpha_{ik} = 1/\sigma_{ik}$ we map each s_{ik} to the unit normal distribution with unity standard deviation, giving $Var(\alpha_{ik} s_{ik}) \approx 1$, and the overall check state variance as $Var(C_k) = \sum_{i=1}^{N} 1^2 \approx N$. From the previous, we see that the variance is reduced. □

Accordingly, the coding vector at time step k is selected to satisfy the constraint $\alpha_{1k}\sigma_{1k} = \alpha_{2k}\sigma_{2k} = \cdots = \alpha_{ik}\sigma_{ik}$, $1 \le i \le n$ with $\sum_{i=1}^{n} \alpha_{ik} \le R$, where R is selected to prevent overflow in the numerical arithmetic involved.

2) Computation of Time-Varying Check Threshold

Since C_k is a weighted sum of the states, a hypothesis test is performed to determine how the state check should be thresholded as a proxy for flagging errors in the system states. For a hypothesis test, we choose the simple two-tail test for normal distributions [29]. To determine the statistical bound (threshold) of the check, the trajectory of C_k^{prior} and C_k^{post} are recorded and mean and variance of the error $e_x(k)$ are determined for a certain time window. In general, the time window can be selected to be *adaptive* (smaller time-window when the system is in equilibrium and higher when performing complex operations). In this work, the length of the window was determined by the method described in [23], [30]. The covariance is used to derive the variance of the error between the prior and posterior check state. Thus for the check state C_k, one has $M(C_k^{prior}, C_k^{post}) = [\sigma^2(C_k^{prior}), \sigma_{cov}^2; \sigma_{cov}^2, \sigma^2(C_k^{post})]$ as the covariance matrix. From this representation, the following relationship is used to determine the variance: $\sigma^2(C_k^{prior} - C_k^{post}) = \sigma^2(C_k^{prior}) + \sigma^2(C_k^{post}) - 2cov(C_k^{prior}, C_k^{post})$.

After the mean and variance are computed, the statistical bound of the error is determined by:

$$\mu \pm \rho\sigma \tag{4}$$

where, μ and σ are mean and standard deviation of the error $e_x(k)$ respectively for the window described earlier, and ρ controls the error bound beyond which system anomalies are flagged. Note that, the quantities μ and σ are time varying in nature, whereas the quantity ρ is fixed and is determined by the method explained in VI. This results in a *time-varying threshold for the check employed*. In addition to the check proposed in this paper, a simple state check can also be adapted from [15] with time-varying coding vectors and anomaly

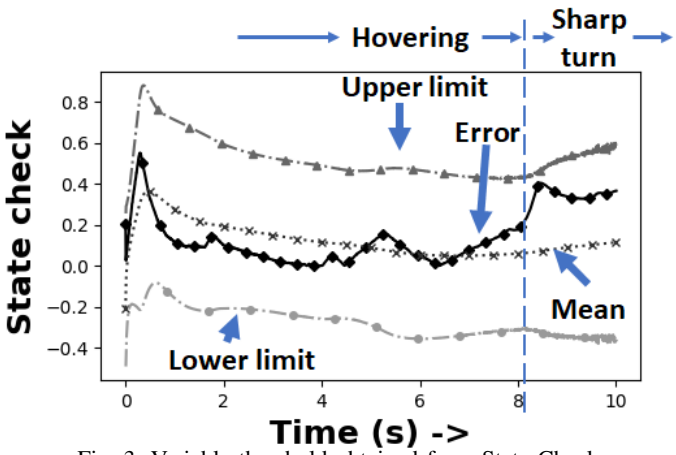

Fig. 3: Variable threshold obtained from State Check

detection thresholds, to address detection of slow-changing parametric failures.

An example use of a variable check threshold is shown in Figure 3 where a quadcopter hovers starting at $t = 0$ sec when we observe a gradual decrease in the check threshold. However, the system goes through a sharp turn starting at $t = 8.1$ sec, and the threshold starts to increase. In this manner, the threshold adapts dynamically to the operating environment of the quadcopter, maximizing anomaly coverage while minimizing false alarms.

B. Variance Check

The 'Variance check' is constructed using the covariance matrix which is given in Equation 3. From the properties of random variables, the variance of the check state C_k can be expressed as: $Var(C_k) = Var(\Sigma_{i=1}^{n}\alpha_{ik}s_{ik}) = \Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\alpha_{ik}\alpha_{jk}cov(s_{ik}, s_{jk})$. As can be observed, both sides of the above relation will agree during nominal operation of the system (limited by noise and modelling inaccuracy). However, they will differ significantly when the system experiences anomalies as changes in the state trajectories impact state covariance values. Hence, we propose the following as the definition of a variance check $e_v(k)$:

$$e_v(k) = Var(C_k) - \Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\alpha_{ik}\alpha_{jk}cov(s_{ik}, s_{jk})$$

Similar to the state check, we also determine the statistical bound for the variance check and this is compared against these bounds at every time instant k to flag errors and failures.

V. Test Cases: Overview and EKF Checks

A. Quadcopter

A quadcopter is modeled as a nonlinear state variable system with 12 state variables: $[x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ (see Figure 4), where, x, y and z represent the position of the quadcopter in 3 dimensional inertial reference frame and ϕ, θ and ψ represent the roll, pitch and yaw angle in the body frame. It has an inertial measurement unit (IMU) as the prime sensor and comprises of an accelerometer and a gyroscope. Four brushless DC motors are used as actuators. The dynamics of a quadcopter system are given below:

$$\begin{aligned}[\ddot{x}, \ddot{y}, \ddot{z}]^T &= [0, 0, -mg]^T + RT_B + F_D \\ [\ddot{\phi}, \ddot{\theta}, \ddot{\psi}]^T &= I^{-1}(\tau - \omega \times (I\omega))\end{aligned} \quad (5)$$

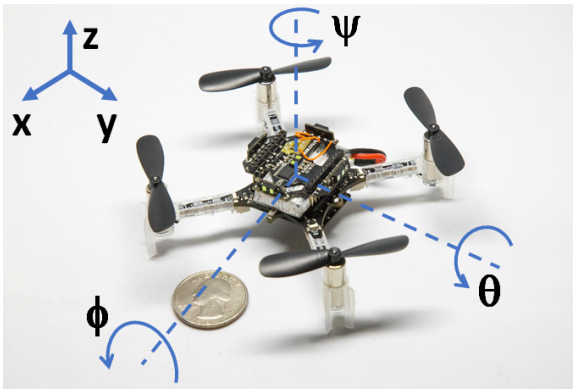

Fig. 4: An example quadcopter system (adopted from Crazyflie [31])

In the above, m = mass of the quadcopter, g = acceleration due to gravity, R = rotational matrix, T_B = thrust vector, F_D = drag force, I = inertia matrix, τ = external torque vector, ω = angular velocity vector. The detailed expressions for these quantities can be found in [32]. The quadcopter system modulates motor speeds to perform maneuvers in the inertial frame according to Equation 5.

To implement the EKF on quadcopter system, the Jacobian F_k is computed. The full size of the Jacobian F_k is 12×12 with the following non-zero components:

$F_{k(0:2,3:5)} = F_{k(6:8,9:11)} = \mathbb{I}(3), F_{k(3,6)} = T(-c_\psi s_\theta s_\phi + s_\psi c_\phi)/m,$

$F_{k(3,7)} = T(c_\psi c_\theta c_\phi)/m,\ F_{k(3,8)} = T(-s_\psi s_\theta c_\phi + c_\psi s_\phi)/m,$

$F_{k(4,6)} = T(-s_\psi s_\theta s_\phi - c_\psi c_\phi)/m,\ F_{k(4,7)} = T(s_\psi c_\theta c_\phi)/m,$

$F_{k(4,8)} = T(c_\psi s_\theta c_\phi + s_\psi s_\phi)/m,\ F_{k(5,6)} = T(-c_\theta s_\phi)/m,$

$F_{k(5,7)} = T(-s_\theta c_\phi)/m,\ F_{k(9,10)} = -(I_z - I_y)\dot{\psi}/I_x,$

$F_{k(9,11)} = -(I_z - I_y)\dot{\theta}/I_x,\ F_{k(10,9)} = -(I_x - I_z)\dot{\psi}/I_y,$

$F_{k(10,11)} = -(I_x - I_z)\dot{\phi}/I_y,\ F_{k(11,9)} = -(I_y - I_x)\dot{\theta}/I_z,$

$F_{k(11,10)} = -(I_y - I_x)\dot{\phi}/I_z,\ F_{k(3:5,3:5)} = -k_D\mathbb{I}(3)/m$

where, $\mathbb{I}(.)$ = identity matrix, T = Thrust = $T_B[2]$, $[I_x, I_y, I_z] = diag(I)$ (T_B and I are defined earlier), and c_* and s_* represents $\cos(*)$ and $\sin(*)$, respectively. For an appropriately selected coding vector $CV_k \in \mathbb{R}^{12}$, the expression for the State Check $e_x(k)$ becomes $e_x(k) = \Sigma_{i=1}^{12}(CV_{k-1}x_k^{prior} - CV_k x_k^{post})$, and that for the variance check $e_v(k)$ becomes $e_v(k) = Var(C_k) - CV_k F_k CV_k^T$. The coding vector CV_k is updated as suggested by Lemma 1, and the statistical bounds for these two checks were determined.

B. Steer by Wire System

An automotive subsystem, Steer-by-Wire (SbW) is considered as the second test case. As shown in Figure 5, a SbW system can be decomposed into two parts, namely hand wheel and front wheel subsystems. The hand wheel subsystem is comprised of the hand wheel, the hand wheel angle sensor, and the feedback motor. The front wheel subsystem is comprised of the steering motor, the pinion angle sensor, the rack and pinion gearbox, and the steered front wheels. These two subsystems are connected through the 'Electronic Control Unit' (ECU). The hand wheel motor generates the feedback torque as a representation of 'road feel', and the front wheel

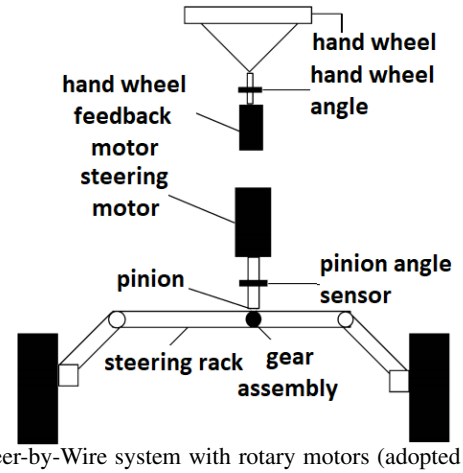

Fig. 5: Steer-by-Wire system with rotary motors (adopted from [33])

steering motor generates the actual torque to steer the front wheels. Figure 5 shows the block diagram of a SbW system.

The dynamics of the hand wheel and front wheel subsystems are given in the following:

$$J_h\ddot{\theta}_h + B_h\dot{\theta}_h + c_h\theta_h + \tau_r = \tau_h$$

$$J_{eq}\ddot{\delta}_f + B_{eq}\dot{\delta}_f + F_s sign(\dot{\delta}_f) + \tau_e = \tau_{eq}$$

The vehicle's slip angle β and yaw rate γ dynamics are:

$$\begin{bmatrix}\dot{\beta}\\ \dot{\gamma}\end{bmatrix} = \begin{bmatrix} -\frac{C_f+C_r}{MV_{CG}} & -1+\frac{C_r l_r - C_f l_f}{MV_{CG}^2} \\ \frac{C_r l_r - C_f l_f}{I_Z} & -\frac{C_r l_r^2 + C_f l_f^2}{I_Z V_{CG}} \end{bmatrix}\begin{bmatrix}\beta\\ \gamma\end{bmatrix} + \begin{bmatrix}\frac{C_f}{MV_{CG}}\\ \frac{C_f l_f}{I_Z}\end{bmatrix}\delta_f$$

where, $J_h(J_{eq})$ = front (equivalent) wheel moment of inertia, $B_h(B_{eq})$ = front (equivalent) wheel viscous friction, c_h = torsional stiffness of the hand wheel shaft, θ_h = hand wheel rotational angle, $\tau_h(\tau_r)$ = hand wheel input (feedback) torque, N_θ = ratio between the hand wheel rotational angle and the front-wheel steering angle, F_s = Coulomb friction constant, $C_f(C_r)$ = front (rear) wheel cornering stiffness, $l_c(l_p)$ = mechanical (pneumatic) trail, $l_f(l_r)$ = front (rear) wheel center to center of gravity (CG) distance, M = vehicle mass, I_Z = vehicle inertia around CG, V_{CG} = vehicle velocity. Further details of J_{eq} and B_{eq} are omitted here for the sake of brevity and can be found in [33].

The hand wheel subsystem is controlled by a proportional and derivative (PD) controller, which produces the response θ_h. This is fed to ECU, which generates the reference angle $\theta_{hr} = \theta_h/N_\theta$ for the front-wheel subsystem. The input to this subsystem is τ_{eq} which is computed from reference front-wheel angle θ_{hr} by a sliding mode controller (SMC). To implement the EKF on the frontwheel subsystem of the SbW, the whole system was constructed with systems states $x_k = [\delta_f, \dot{\delta}_f, \beta, \gamma]^T$ at time step k. the Jacobian F_k was computed which has following non-zero elements:

$F_{k(0,1)} = 1$, $F_{k(1,0)} = -C_f(l_c + l_p)/J_{eq}$,

$F_{k(1,1)} = -(2F_s\Delta(0) + B_{eq})/J_{eq}$, $F_{k(1,2)} = c_f(lc + lp)/J_{eq}$,

$F_{k(1,3)} = c_f(l_c + l_p)l_f/(J_{eq}v_{CG})$, $F_{k(2,0)} = c_f/(Mv_{CG})$,

$F_{k(2,2)} = -(c_f + c_r)/(Mv_{CG})$,

$F_{k(2,3)} = -1 + (c_r l_r - c_f l_f)/(Mv_{CG}^2)$,

$F_{k(3,0)} = c_f l_f/I_z$, $F_{k(3,2)} = (c_r l_r - c_f l_f)/I_z$,

$F_{k(3,3)} = -(c_r l_r^2 + c_f l_f^2)/(I_z v_{CG})$

where, $\Delta(.)$ is a 'Kronecker delta' function which is 1 if the argument is zero and 0 otherwise. The coding vectors $CV_k \in \mathbb{R}^4$ were updated as suggested by Lemma 1 and the state check, variance check and statistical bounds for these two quantities were computed using previously described methods.

VI. Experimental Results

The quadcopter used in this work has following parameters (defined in [32]): $L = 0.3\,m$, $r = 0.1\,m$, $m = 1.2\,kg$, $b = 0.0245$, $d = 10\,in$, and $pitch = 4.5\,in$. The steer-by-wire system used in this work (adopted from [33]) has the following parameters: $J_{fw} = 2.6\,kg.m^2$, $B_{fw} = 12\,Nms/rad$, $J_{eq} = 9.498\,kg.m^2$, $B_{eq} = 24.312\,Nms/rad$, $c_h = 0.2\,Nm/rad$, $N_\theta = 12$, $F_s = 2.68\,Nm$, $C_f = C_r = 45000\,N/rad$, $l_f = 1.2\,m$, $l_c = 0.016\,m$, $l_p = 0.023\,m$, $l_r = 1.05\,m$, $M = 2000\,kg$, $I_Z = 1300\,Kg.m^2$, and $V_{CG} = 10\,m/sec$. The initial coding vector CV_0 (initial CV at $t = 0$) for quadcopter was chosen to be $[1/4, 1/4, 1/4, 1, 1, 1/8, 4, 4, 4, 2, 2, 2]$, and that for SbW system was chosen to be $[1, 1, 1, 1]$. Subsequent coding vectors and check detection thresholds are computed as per the discussion in Section IV. The proposed error detection framework scales across a variety of error and failure mechanisms. We discuss the statistical analysis based coding parameter and threshold selection strategy in the following. This is followed by results for transient errors in sensor values, Kalman filter computations and control program execution and offset errors in actuators. Comparison with other state of the art methodologies is discussed and hardware validation experiments are presented.

Statistical analysis based best bound selection:

Nonlinear systems having the form shown in Figure 1 are subject to process and measurement noise. Having a check whose threshold is low results in vulnerability to noise. For this reason, the proposed Kalman filter check uses both the *'State check'* and *'Variance check'*. In general, the quantity ρ is selected to minimize false positives while maximizing error coverage. Low threshold values result in higher '*false positive*' counts (detection of errors when there are none). On the other hand, high values of threshold (high values of ρ) result in loss of error coverage and hurts performance as well. As a consequence, a proper value of ρ needs to be determined in Equation 4 that balances this tradeoff. We use a statistical approach to systematically address this issue. We vary the value of ρ and determine the '*Precision*' and '*Recall*' of the model. '*Precision*' is defined as $\frac{TP}{TP+FP}$ and '*Recall*' is defined as $\frac{TP}{TP+FN}$, where TP, FP, and FN are defined as TP = *True Positive* = the number of errors that cause failures in performance that are detected, FP = *False Positive* = the number of events (error injections) that do not cause failures in performance but are flagged as errors and FN = *False Negative* = The number of events that cause failures in performance but are flagged as error-free. Performance failure for a quadcopter is defined to occur when the L_∞ norm of the difference in the x, y and z co-ordinates of the quadcopter deviates from its intended trajectory by 5 inches, where the diameter of the spherical volume of the quadcopter is 10

inches. Similarly, for the Steer-by-Wire system, performance failure occurs when the lateral acceleration of the vehicle exceeds its maximum permissible bound, (0.4 g at any speed, set by United States Federal Highway Administration [34]).

To determine the best value of ρ for the quadcopter check, we performed 2000 experiments where faults were injected in 1000 experiments (both the fault models and the fault value were selected in random) and the remaining experiments were fault-free. From these experiments, the value of '*Precision*' and '*Recall*' were determined. The '*F1 score,* F_1' was determined from these two quantities which is defined as $F_1 = \frac{2}{Recall^{-1}+Precision^{-1}}$. The maximum value of F_1 corresponds to the best balance between '*Precision*' and '*Recall*' and this represents the selected value of ρ. For the quadcopter, the values of *F1 score*, *Precision*, and *Recall* are plotted in Figure 6. As seen from this plots, the value ρ = 4.1 corresponds to the best threshold bound.

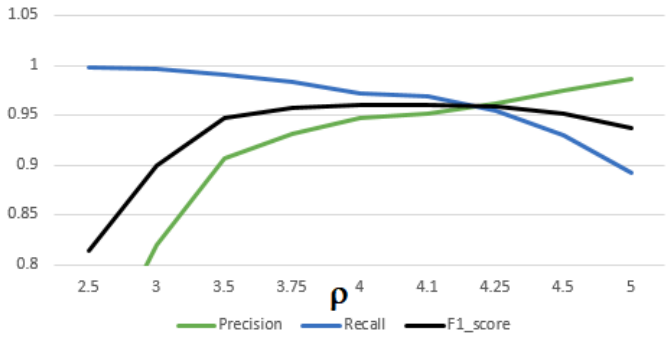

Fig. 6: F1 score profile for quadcopter system

Similar to the previous experiment, we performed F_1 *score* based analysis for the Steer-by-Wire system, and the values of F_1 *score*, *Precision*, and *Recall* are plotted in Figure 7. As can be seen from these plots, ρ = 3 corresponds to the best threshold bound.

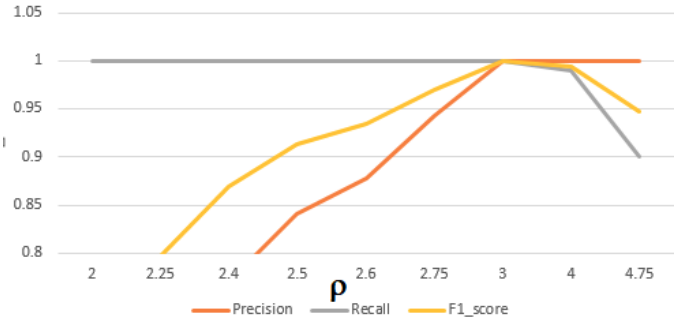

Fig. 7: F1 score profile for Steer-by-Wire system

In the following, we illustrate the detection of failures in sensors, actuators, covariance computation and control program execution with examples. This is followed by a discussion of the benefits of the proposed methodology vis-a-vis the state of the art.

A. Sensor Failure

Multiple bit-flips were introduced into the sensor readings (inertial measurement unit (IMU) reading) of the quadcopter from time t = 3 sec to t = 4.2 sec of a planned flight path. The state check and variance check obtained for this failure are shown in Figure 8, and it is observed from the plots that the injected error is almost instantaneously detected at t = 3 sec. Similarly, random bit-flips were injected into the sensor readings of the SbW system at t = 0.7 sec, and both the state and variance checks are shown in the Figure 9. As can be seen from these plots, the error is detected at t = 0.701 sec with only 1 ms detection latency. As discussed earlier, both the state check and the variance check are adaptively encoded with time-varying detection thresholds.

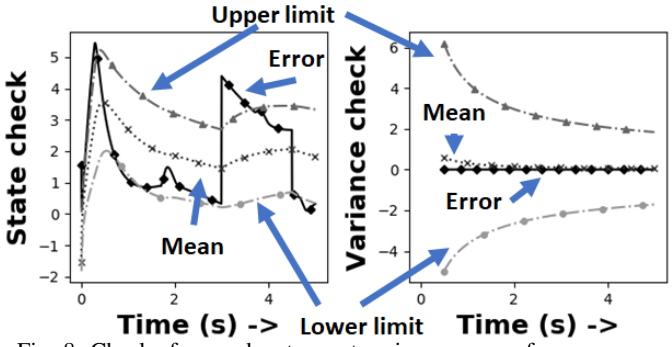

Fig. 8: Checks for quadcopter system in presence of sensor error

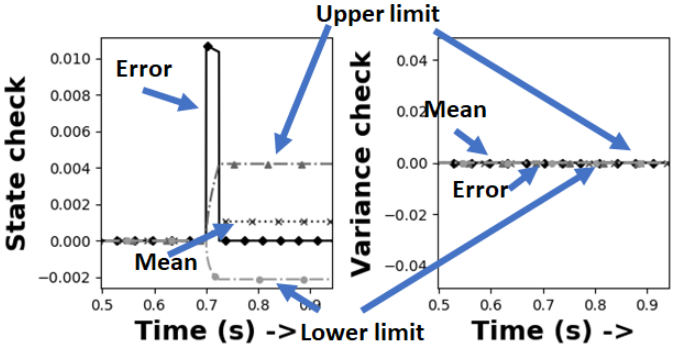

Fig. 9: Checks for SBW system in presence of sensor error

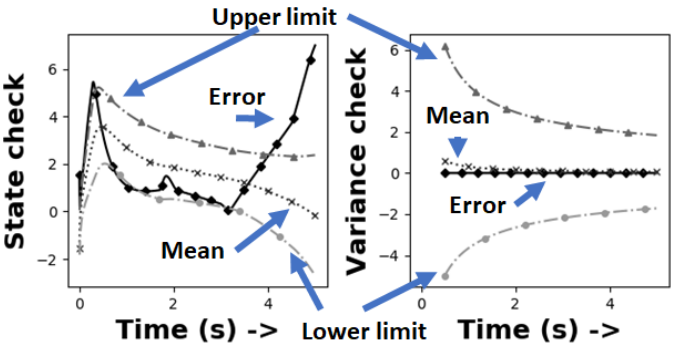

Fig. 10: Checks for quadcopter system in presence of actuator error

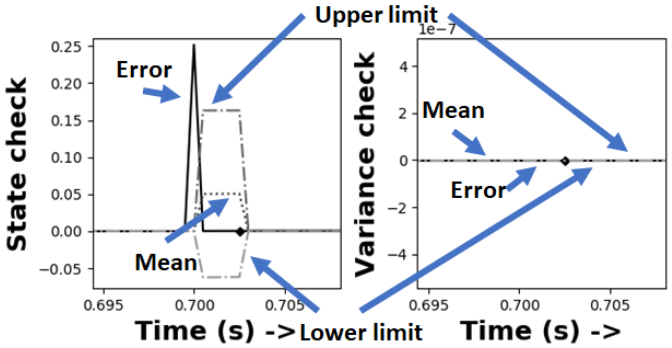

Fig. 11: Checks for SBW system in presence of actuator error

B. Actuator Error

Actuator failures for the quadcopter systems are modeled as offsets in actuator values (motor speed). Figure 10 shows plots for state and variance checks when the system experiences an offset in actuator output (failure) at t = 3.15 sec. The state check shows a clear breach of bounds at t = 4 sec which indicates the detection of failure within 0.85 sec. For the SbW system, an offset was introduced into the steering motor (brushless DC motor) output at t = 0.7 sec. The corresponding state and variance checks are shown in Figure 11 which shows that the error was detected with a latency of only 1 ms. Most other injected failures were detected with low latency (within 10 ms).

C. Covariance Computation Failure

We also considered computation errors in the covariance matrix block of the Kalman filter. Computation of this matrix is particularly important as the *coding vectors* are scaled according to the diagonal elements of this matrix and errors in computation directly impact the error checking process. Figures 12 and 13 show plots for the state and variance

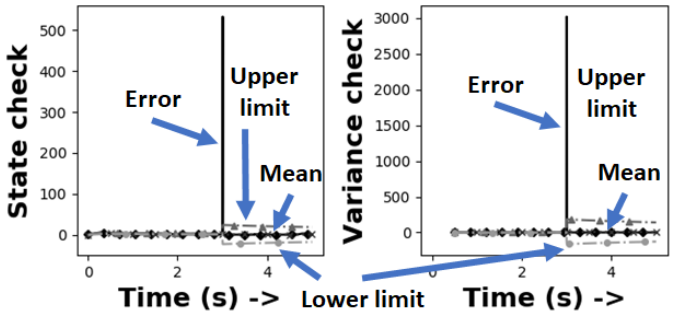

Fig. 12: State Check for quadcopter system in presence of covariance computation error

checks for the quadcopter and the SbW system respectively, when the covariance matrix stored in memory is corrupted due to soft errors. The errors for the quadcopter system and the SbW system were injected at t = 3 sec and t = 0.7 sec, respectively. As seen from these plots, both the state and variance checks were able to flag the computation error in real-time at t = 3.001 sec and t = 0.701 respectively, resulting in corresponding error detection latencies of 1 ms in both cases.

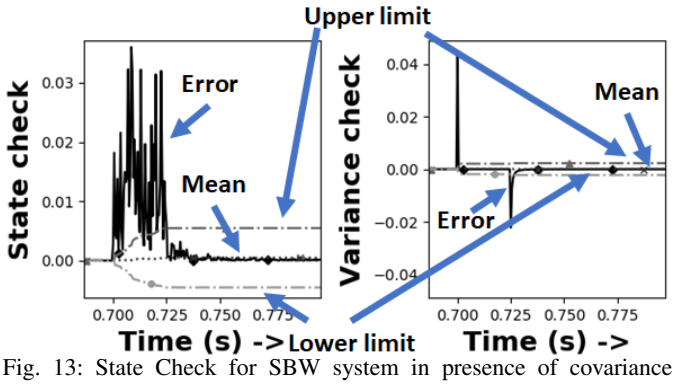

Fig. 13: State Check for SBW system in presence of covariance computation error

D. Control Software Error

Soft errors involve fault injection in the controller (digital processor core) block of Figure 1. These errors are modeled as spurious bit flips in the processor core on which the control algorithm is executed. In a simulation environment, the plant, controller and checking algorithms are coded together in a combined manner and executed on a common processor platform. However, in reality, only the controller and the fault detection methodologies are executed on an embedded processor since the rest of the simulation software mimics the actual physical behavior of the system (plant). For this reason, the location of the control action computations was analyzed carefully in assembly level code and faults were injected into the registers involved in control action computations. The range of least significant bits of the registers which are affected by error injection is referred to as the '*injection range*'. 1000 injection experiments for single and multiple bit error each were performed and for each set of experiments, the number of errors detected and number of experiments that resulted in silent data corruption (SDC) are shown in Tables I and II respectively. As seen in this case, the lower value of

TABLE I: Error detection experiments on quadcopter system

	Multiple bit-flip		Single bit-flip	
Injection range	Potential crashes with 100% detection	# of SDC	Potential crashes with 100% detection	# of SDC
32	846	154	542	458
16	576	424	68	932
8	522	478	0	1000

TABLE II: Error detection experiments on SbW system

	Multiple bit-flip		Single bit-flip	
Injection range	Errors detected	# of SDC	Errors detected	# of SDC
32	864	136	627	373
16	766	234	421	579
8	100	900	21	979

injection range for both multiple and single bit-flips resulted in low value of errors detected (52.2% and 0% respectively) and potential crashes, as these bit-flips did not have strong manifestation into system failure. Additionally, single bit-flips resulted in fewer errors in all experiments. For silent data corruption, the errors do not change the performance of the

system and hence the proposed check did not detect those events. However, as the injection range increases, the system failure also increases which are detected by the proposed approach. As seen from the Table I, the error detection accuracy increases to 84.6% (for multiple bit error) and 54.2% (for single bit error) when injection range is 32. For the SbW case, similar to the quadcopter case, fewer multiple and single bit-flip errors were detected at low injection range of 8 (10% and 2.1% respectively), which increases to 86.4% and 62.7%, respectively when injection range is 32 as reported in Table II.

E. Comparison with state of the art detection methods

In the following, we compare the proposed error detection methodology using Kalman filter checks with variable coding vectors and variable thresholds against two most relevant error detection techniques: (a) use of Kalman filter checks with constant coding vector and constant threshold [15] and (b) use of a machine learning based state encoding technique proposed in [23]. In both (a) and (b), linear sums of the system states were used for state encoding and the state encoding was constant over time. Also, in both (a) and (b), fixed time-independent error detection thresholds were used. The error threshold was set to worst case magnitude of the coding error determined for a fault free system under anticipated operating conditions. We considered faults in sensors (accelerometer and gyroscope, selected in random), actuators, and control action computation. We performed 1000 experiments for each of these failure modes and simulated each of the above error detection techniques under the same set of operating conditions. For each of these combinations, we determined detection accuracy (defined as $\frac{Number\ of\ true\ events\ detected}{Number\ of\ total\ events}$) and average detection latency. The results are given in Table III.

TABLE III: Comparative study for quadcopter system

Comparison metric	Proposed approach	Kalman check with constant CV and Threshold [15]	Machine learning based approach [23]
Comparison of detection accuracy			
Sensor fault	99.7%	97.2%	99.1%
Actuator fault	83.7%	12.5%	83.3%
Control program fault	95%	97.2%	88%
Comparison of average detection latency			
Sensor fault	< 1 ms	3.8 ms	1.9 sec
Actuator fault	0.89 sec	1.9 sec	1.4 sec
Control program fault	0.28 sec	0.28 sec	0.2 sec
Other comparisons			
Training time for machine learned model*	NA	NA	15 min
Computation Overhead	21.17 %	21.05 %	646.88 %
Memory Overhead	15.81 %	15.81 %	4842.79 %

*Training was done in Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, 16GB RAM, without GPU

As observed from Table III, our proposed approach had superior or comparable performance to the two other state of the art error detection methods in the majority of metrics considered. In each case, the proposed technique was overwhelmingly superior to the machine learning based checking approach of [23]. In only one case, the Steer-by-Wire actuator detection latency was 44.2 ms vs 27.2 ms for the Kalman check with constant coding vector. This is due to the extra computations needed by the proposed approach (can be fine tuned by code optimization) but allows significant improvements in failure coverage (98.2% vs. 7.5%).

TABLE IV: Comparative study for Steer-by-Wire system

Comparison metric	Proposed approach	Kalman check with constant CV and Threshold [15]	Machine learning based approach [23]
Comparison of detection accuracy			
Sensor fault	100%	76.2%	99.6%
Actuator fault	98.2 %	7.5%	9.2%
Control program fault	92%	2%	2.2%
Comparison of average detection latency			
Sensor fault	7.8 ms	40.9 ms	501 ms
Actuator fault	44.2 ms	27.2 ms	252 ms
Control program fault	65.7 ms	146.5 ms	605 ms
Other comparisons			
Training time for machine learned model*	NA	NA	9 min 10 sec
Computation Overhead	60.56 %	61.76 %	589.44%
Memory Overhead	43.75 %	43.75 %	4902.08%

*Same computer mentioned in Table IV was used here

F. Hardware validation

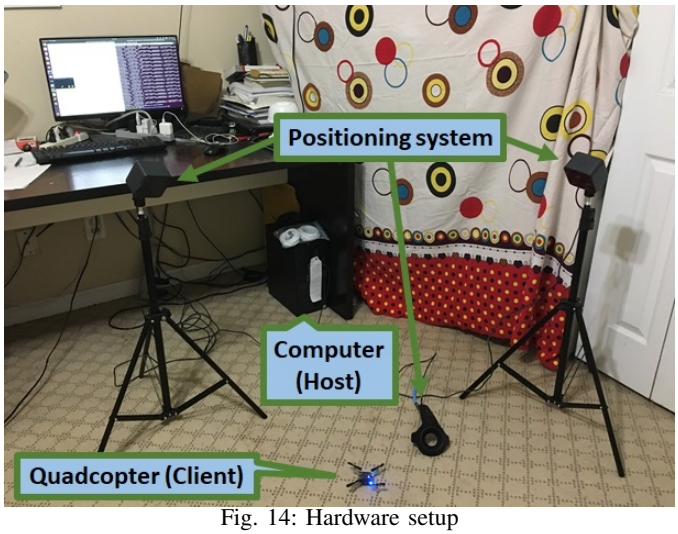

Fig. 14: Hardware setup

We implemented the *Kalman Filter Check* on a '*Crazyflie 2.1*' [31] quadcopter system (see Figure 14). The hardware configuration of the system is shown in Table V. The Crazyflie quadcopter communicates with a PC via Bluetooth interface and a propriety Crazy RealTime Communication Protocol (CRTP). The server and client platforms for the setup were

TABLE V: Hardware configuration of Crazyflie 2.1

Physical parameters/Components	Specification/Model
System mass	27 gm
Size (W × H × D)	92 × 92 × 29 mm
Radio Band	2.4 GHz
Coomunication type	Bluetooth
Coomunication protocol	Crazy RealTime Protocol (CRTP)
Main microcontroller Unit	STM32F405 Cortex-M4 Clock frequency: 168MHz RAM: 192KB Flash: 1MB
Radio and power management unit	nRF51822
3 axis accelerometer / gyroscope	BMI088
Pressure sensor	BMP388
Actuator	4 DC coreless motors

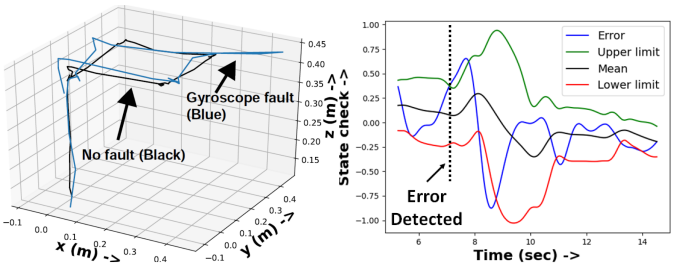

Fig. 15: (a) Trajectory of the quadcopter under gyroscope fault and (b) corresponding error plot

implemented using python and C respectively. The server was used only to send commands to the client and all the necessary computations related to control action generation, necessary system level consistency check for the client, Kalman filter based state estimation, our proposed checks etc. were implemented in the CrazyFlie Microcontroller unit (MCU). The client performed the necessary maneuver according to its objective and the check values were read back from the client in the form of a log variable. The minimum period for a log variable to be read from the client was 10 ms. Failures were injected into the accelerometer, gyroscope, actuator circuit, and control program of the quadcopter and the obtained results are discussed below:

1) Gyroscope fault

Offsets in gyroscope readings (the most common form of gyroscope faults) were used to emulate gyroscope failures. The planned trajectory of the quadcopter under nominal conditions and with gyroscope fault are shown in Figure 15a. The corresponding state check, $e_x(k)$ is shown in the Figure 15b. From the error plot, it is seen that the fault was injected at time t = 7.11 sec and the check produced a non-zero output above the threshold at time t = 7.12 sec and the error was detected.

2) Accelerometer fault

Accelerometer faults were emulated and an offset (the most common form of accelerometer faults) was introduced in the accelerometer reading at t = 5.4 sec. It was successfully detected at t = 5.71 sec as seen at Figure 16b with an error detection latency of 0.31 sec.

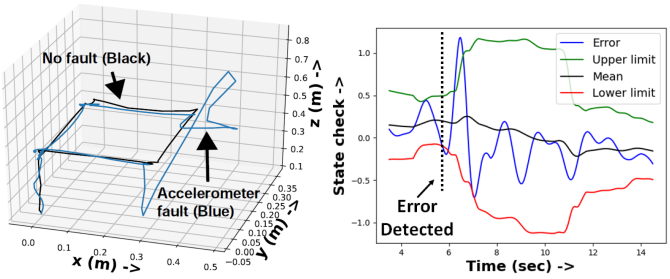

Fig. 16: (a) Trajectory of the quadcopter under Accelerometer fault and (b) corresponding error plot

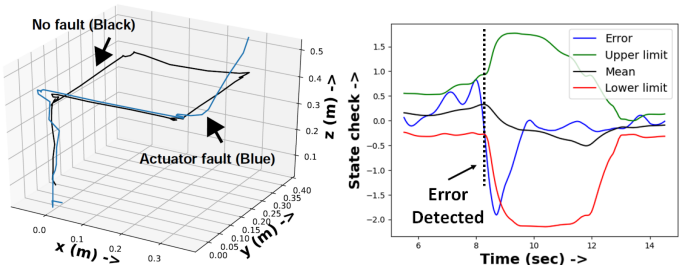

Fig. 17: (a) Trajectory of the quadcopter under Actuator fault and (b) corresponding error plot

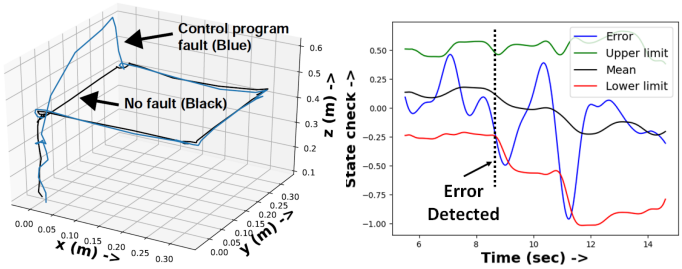

Fig. 18: (a) Trajectory of the quadcopter under Control program fault and (b) corresponding error plot

3) Actuator fault

The quadcopter system has DC motors which produces necessary thrust from its input voltage. The generated thrust from the actuator can vary due to numerous reasons which include change in terminal resistance of the motor or temporary variations at the propeller driven by the DC motors. In this experiment, we introduced temporary speed changes in one of the DC motors which results in change of thrust. The fault was injected at t = 8 sec and was detected by the state check at t = 8.25 sec as seen in Figure 17b.

4) Control program fault

We introduced failure in control program execution in the form of spurious bit-flips in the processor core. Introduction of spurious bit-flips results in incorrect quadcopter control action. In this experiment, a fault was introduced at time t = 8.23 sec and caused unwanted deviation of the quadcopter from its desired trajectory. As seen from the Figure 18b, the proposed check was able to capture this at t = 8.69 sec and the fault

was successfully detected.

VII. Conclusion and Future Work

An encoded Extended Kalman Filter based internal anomaly detection framework has been presented in this research. The proposed approach has been successfully implemented for two different test cases: (a) a quadcopter and (b) a steer-by-wire system. A detailed comparative study has been performed and it is found that the implementation overhead is very low compared to earlier machine learning based checking schemes for nonlinear systems. At the same time, higher coverage of errors and failures is achieved. Simulation data and hardware validation experiments obtained for different failure mechanisms in different subsystems corroborate the efficacy of the proposed technique. Future work will focus on extending this approach to other pervasive failure mechanisms and external security attacks.

Acknowledgment

This research was supported by the U.S. National Science Foundation under Grant S&AS:1723997 and by the Semiconductor Research Corporation under Auto Task 2892.001.

References

[1] K. Kaur and G. Rampersad, "Trust in driverless cars: Investigating key factors influencing the adoption of driverless cars," *Journal of Engineering and Technology Management*, vol. 48, pp. 87–96, 2018.

[2] A. Eskandarian, *Handbook of intelligent vehicles*. Springer, 2012, vol. 2.

[3] S. Banerjee, B. Samynathan, J. Abraham, and A. Chatterjee, "Real-time error detection in nonlinear control systems using machine learning assisted state-space encoding," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[5] B. T. Thumati and S. Jagannathan, "A model-based fault-detection and prediction scheme for nonlinear multivariable discrete-time systems with asymptotic stability guarantees," *IEEE Transactions on Neural Networks*, vol. 21, no. 3, pp. 404–423, 2010.

[6] E. Lampiri, "Sensor anomaly detection and recovery in a nonlinear autonomous ground vehicle model," in *2017 11th Asian Control Conference (ASCC)*. IEEE, 2017, pp. 430–435.

[7] K. Bouibed, A. Aitouche, and M. Bayart, "Sensor fault detection by sliding mode observer applied to an autonomous vehicle," in *2009 International Conference on Advances in Computational Tools for Engineering Applications*. IEEE, 2009, pp. 621–626.

[8] P. Goel, G. Dedeoglu, S. I. Roumeliotis, and G. S. Sukhatme, "Fault detection and identification in a mobile robot using multiple model estimation and neural network," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3. IEEE, 2000, pp. 2302–2309.

[9] L. Cork and R. Walker, "Sensor fault detection for uavs using a nonlinear dynamic model and the imm-ukf algorithm," in *2007 Information, Decision and Control*. IEEE, 2007, pp. 230–235.

[10] P. Guo, H. Kim, N. Virani, J. Xu, M. Zhu, and P. Liu, "Roboads: Anomaly detection against sensor and actuator misbehaviors in mobile robots," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 574–585.

[11] M. R. Napolitano, G. Silvestri, D. A. Windon, J. Casanova, and M. Innocenti, "Sensor validation using hardware-based on-line learning neural networks," *IEEE transactions on aerospace and electronic systems*, vol. 34, no. 2, pp. 456–468, 1998.

[12] M. Borairi and H. Wang, "Actuator and sensor fault diagnosis of nonlinear dynamic systems via genetic neural networks and adaptive parameter estimation technique," in *Proceedings of the 1998 IEEE International Conference on Control Applications (Cat. No. 98CH36104)*, vol. 1. IEEE, 1998, pp. 278–282.

[13] M. I. Momtaz, S. Banerjee, and A. Chatterjee, "On-line diagnosis and compensation for parametric failures in linear state variable circuits and systems using time-domain checksum observers," in *2017 IEEE 35th VLSI Test Symposium (VTS)*, April 2017, pp. 1–6.

[14] M. I. Momtaz and A. Chatterjee, "Hierarchical check based detection and diagnosis of sensor-actuator malfunction in autonomous systems: A quadcopter study," in *25th IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)*, July 2019, pp. 316–321.

[15] S. Pandey, S. Banerjee, and A. Chatterjee, "Concurrent error detection and tolerance in kalman filters using encoded state and statistical covariance checks," in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2016, pp. 161–166.

[16] C. N. Amarnath, M. I. Momtaz, and A. Chatterjee, "Encoded check driven concurrent error detection in particle filters for nonlinear state estimation," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2020, pp. 1–6.

[17] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computers*, vol. C-33, no. 6, pp. 518–528, June 1984.

[18] J.-Y. Jou and J. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures," *Proceedings of the IEEE*, vol. 74, no. 5, pp. 732–741, May 1986.

[19] V. Nair and J. Abraham, "Real-number codes for fault-tolerant matrix operations on processor arrays," *Computers, IEEE Transactions on*, vol. 39, no. 4, pp. 426–435, Apr 1990.

[20] A. Chatterjee and M. A. d'Abreu, "The design of fault-tolerant linear digital state variable systems: theory and techniques," *IEEE Transactions on Computers*, vol. 42, no. 7, pp. 794–808, Jul 1993.

[21] A. Chatterjee, "Concurrent error detection and fault-tolerance in linear analog circuits using continuous checksums," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 1, no. 2, pp. 138–150, June 1993.

[22] M. I. Momtaz, S. Banerjee, S. Pandey, J. Abraham, and A. Chatterjee, "Cross-layer control adaptation for autonomous system resilience," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, July 2018, pp. 261–264.

[23] S. Banerjee, A. Chatterjee, and J. A. Abraham, "Efficient cross-layer concurrent error detection in nonlinear control systems using mapped predictive check states," in *2016 IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–10.

[24] D.-L. Yu, T. K. Chang, and D.-W. Yu, "Fault tolerant control of multivariable processes using auto-tuning pid controller," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 1, pp. 32–43, Feb 2005.

[25] M. I. Momtaz, S. Banerjee, and A. Chatterjee, "Real-time dc motor error detection and control compensation using linear checksums," in *2016 IEEE 34th VLSI Test Symposium (VTS)*, April 2016, pp. 1–6.

[26] ——, "Probabilistic error detection and correction in switched capacitor circuits using checksum codes," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, July 2017, pp. 271–276.

[27] H.-G. Stratigopoulos and Y. Makris, "Concurrent detection of erroneous responses in linear analog circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 5, pp. 878–891, May 2006.

[28] R. Kalman, "Contributions to the theory of optimal control," 1960.

[29] E. L. Lehmann and J. P. Romano, *Testing statistical hypotheses*, 3rd ed., ser. Springer Texts in Statistics. New York: Springer, 2005.

[30] M. I. Momtaz and A. Chatterjee, "Hierarchical state space checks for errors in sensors, actuators and control of nonlinear systems: Diagnosis and compensation," in *2019 IEEE 28th Asian Test Symposium (ATS)*, 2019, pp. 141–146.

[31] "Crazyflie 2.1." [Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1/

[32] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," Master's thesis, KTH Royal Institute of Technology, 2015.

[33] H. Wang, H. Kong, Z. Man, D. M. Tuan, Z. Cao, and W. Shen, "Sliding mode control for steer-by-wire systems with ac motors in road vehicles," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 3, pp. 1596–1611, March 2014.

[34] W. H. Levison, J. L. Campbell, K. Kludt, A. C. Bittner, I. B. Potts, D. W. Harwood, J. M. Hutton, D. Gilmore, J. G. Howe, J. Christos *et al.*, "Development of a driver vehicle module (dvm) for the interactive highway safety design model (ihsdm)," United States. Federal Highway Administration. Office of Research and ..., Tech. Rep., 2007.