Fault Tolerant Control for Autonomous Surface Vehicles via Model Reference Reinforcement Learning

Zhang, Qingrui; Zhang, Xinyu; Zhu, Bo ; Reppa, V.

DOI
10.1109/CDC45484.2021.9683461

Publication date
2021

Document Version
Final published version

Published in
Proceedings of the 60th IEEE Conference on Decision and Control (CDC 2021)

2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Fault Tolerant Control for Autonomous Surface Vehicles via Model Reference Reinforcement Learning

Qingrui Zhang[1], Xinyu Zhang[1], Bo Zhu[1], and Vasso Reppa[2]

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683461

***Abstract*— A novel fault tolerant control algorithm is proposed in this paper based on model reference reinforcement learning for autonomous surface vehicles subject to sensor faults and model uncertainties. The proposed control scheme is a combination of a model-based control approach and a data-driven method, so it can leverage the advantages of both sides. The proposed design contains a baseline controller that ensures stable tracking performance at healthy conditions, a fault observer that estimates sensor faults, and a reinforcement learning module that learns to accommodate sensor faults using fault estimation and compensate for model uncertainties. The impact of sensor faults and model uncertainties can be effectively mitigated by this composite design. Stable tracking performance can also be ensured even at both the offline training and online implementation stages for the learning-based fault tolerant control. A numerical simulation with gyro sensor faults is presented to demonstrate the efficiency of the proposed algorithm.**

I. INTRODUCTION

With the impressive advancement in guidance, navigation, and control technologies, autonomous surface vehicles (ASVs) are becoming a possible alternative solution to human-operated vessels in diverse applications [1], [2]. In particular, the sudden burst of the COVID-19 pandemic makes it more urgent to develop ASVs for global shipping. In most applications, ASVs are expected to be running safely with little human intervention for a long period of time. It requires ASVs to have enough safety and reliability attributes for both avoiding catastrophic consequences and securing the deliverance of correct service [3]. However, ASVs are susceptible to malfunctions, degradation in system components, and sensor faults, *etc.*, thereby experiencing performance deterioration, instability, and even disastrous loss. Those issues motivate the extensive study of fault tolerant control (FTC), an efficient technique that can recover system performance or keep systems operational after encountering faults, and thus enhance the system safety significantly [4].

FTC algorithms are generally divided into two categories, namely passive and active FTC [5]. In passive FTC, a reliable controller is developed, which has sufficient robustness against all expected faults of low magnitude either by using a robust control approach or an adaptive control method [6]. Passive FTC algorithms demand no controller reconfiguration, so they need to accommodate both the healthy and faulty conditions [7]. However,faults would not occur most time in real-life applications. Hence, passive FTC is conservative and has limited fault tolerant capabilities [5], [8]. Different from passive FTC, active FTC algorithms can react actively to system faults by monitoring system health using a fault diagnosis and identification (FDI) mechanism [9]. Once a fault is detected, an active fault tolerant controller can reconfigure itself efficiently to recover the system performance. Most FTC algorithms belong to model-based approaches. Passive FTC needs to know the "worst-case" system faults for the design of robust control [10]. Active FTC needs the degraded system model under faults for control reconfiguration [11].

To reduce the dependence on system modelling, reinforcement learning (RL) has been discussed as an option for the design of FTC [12]–[14]. The major advantage of RL is the learning of an optimal control law from data samples without using models. Such an advantage is very suitable for ASVs subject to significant model uncertainties and environmental disturbances [15]. However, it is demanding for model-free RL to ensure closed-loop stability, if no extra assumption is made on the initial choices of control laws. Many existing RL algorithms for FTC learn a robust optimal FTC that ensures system stability under the “worst-case” faults [16], although model information is not necessary.

In this paper, we present a novel FTC algorithm for autonomous surface vehicles subject to sensor faults and model uncertainties based on reinforcement learning. Via the integration of RL with a model-based control approach, the proposed control scheme, termed as model reference reinforcement learning [17], [18], can ensure the closed-loop stability. It can also avoid learning the “worst-case” controller by incorporating a fault diagnosis and estimation mechanism to signal the occurrence and magnitude of sensor faults. Hence, our FTC algorithm can actively react to sensor faults, mitigate the impact of both sensor faults and model uncertainties on the trajectory tracking performance of ASVs, and ensure stable and safe tracking performance at both the offline training and the online implementation stages. In summary, two major contributions of this work are identified.

1) A new formulation framework following the model-reference structure is presented for the design of reinforcement learning-based control. With this new formulation, it is possible to combine model-based approaches with data-driven methods, and leverage the advantages from both sides.
2) A novel RL-based active FTC algorithm is presented for ASVs subject to sensor faults and model uncertainties. Numerical simulations show that the proposed FTC

[1]School of Aeronautics and Astronautics, Sun Yat-sen University, Guangzhou, Guangdong, China (zhangqr9@mail.sysu.edu.cn; zhangxy385@mail2.sysu.edu.cn; zhubo5@mail.sysu.edu.cn)

[2]Department of Maritime and Transport Technology, Delft University of Technology, Delft, the Netherlands (V.Reppa@tudelft.nl)

algorithm can efficiently mitigate the influence of sensor faults and model uncertainties.

The rest of the paper is organized as follows. Problem formulation is provided in Section II. Section III presents the reinforcement learning-based FTC scheme. In Section IV, we describe details on the real-life implementation of the proposed algorithm. Numerical simulations are provided in Section V. Concluding remarks are given in Section VI.

II. Problem Formulation

According to [19], [20], the ASV dynamics are

$$\begin{cases} \dot{\boldsymbol{\eta}} = \mathscr{R}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \mathscr{M}\dot{\boldsymbol{\nu}} + (\mathscr{C}(\boldsymbol{\nu}) + \mathscr{D}(\boldsymbol{\nu}))\boldsymbol{\nu} + \mathscr{G}(\boldsymbol{\nu}) = \mathscr{B}\boldsymbol{u} \end{cases} \tag{1}$$

where $\boldsymbol{\eta} = [x_p, y_p, \psi_p]^T \in \mathbb{R}^3$ is a generalized coordinate vector with x_p and y_p denoting the horizontal position coordinates of an ASV in the inertial frame and ψ_p the heading angle, $\boldsymbol{\nu} = [u_p, v_p, r_p]^T \in \mathbb{R}^3$ is the generalized speed vector with u_p and v_p being the linear velocities in surge (x-axis) and sway (y-axis), respectively, and r_p the heading angular rate, $\boldsymbol{u} = [\tau_u, \tau_r]^T \in \mathbb{R}^2$ is the control forces and moments, $\mathscr{G}(\boldsymbol{\nu}) = [\mathrm{g}_1(\boldsymbol{\nu}), \mathrm{g}_2(\boldsymbol{\nu}), \mathrm{g}_3(\boldsymbol{\nu})]^T \in \mathbb{R}^3$ is unmodeled dynamics due to gravitational and buoyancy forces and moments [19], $\mathscr{B} = \begin{bmatrix} 1, & 0, & 0 \\ 0, & 0, & 1 \end{bmatrix}^T$ is the input matrix. $\mathscr{M} = \mathscr{M}^T \in \mathbb{R}^{3\times3}$ is the inertia matrix,

$$\mathscr{M} = [\mathscr{M}_{ij}] = \begin{bmatrix} \mathscr{M}_{11} & 0 & 0 \\ 0 & \mathscr{M}_{22} & \mathscr{M}_{23} \\ 0 & \mathscr{M}_{32} & \mathscr{M}_{33} \end{bmatrix} \tag{2}$$

where $\mathscr{M}_{11} = m - \mathscr{X}_{\dot{u}}$, $\mathscr{M}_{22} = m - \mathscr{Y}_{\dot{v}}$, $\mathscr{M}_{33} = I_z - \mathscr{N}_{\dot{r}}$, and $\mathscr{M}_{32} = \mathscr{M}_{23} = mx_g - \mathscr{Y}_{\dot{r}}$. The matrix $\mathscr{C}(\boldsymbol{\nu}) = -\mathscr{C}^T(\boldsymbol{\nu})$ contains the Coriolis and centripetal terms, so

$$\mathscr{C} = [\mathscr{C}_{ij}] = \begin{bmatrix} 0 & 0 & \mathscr{C}_{13}(\boldsymbol{\nu}) \\ 0 & 0 & \mathscr{C}_{23}(\boldsymbol{\nu}) \\ -\mathscr{C}_{13}(\boldsymbol{\nu}) & -\mathscr{C}_{23}(\boldsymbol{\nu}) & 0 \end{bmatrix} \tag{3}$$

where $\mathscr{C}_{13}(\boldsymbol{\nu}) = -\mathscr{M}_{22}v - \mathscr{M}_{23}r$, $C_{23}(\boldsymbol{\nu}) = \mathscr{M}_{11}u$. The damping matrix $\mathscr{D}(\boldsymbol{\nu})$ is

$$\mathscr{D}(\boldsymbol{\nu}) = [\mathscr{D}_{ij}] = \begin{bmatrix} \mathscr{D}_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & \mathscr{D}_{22}(\boldsymbol{\nu}) & \mathscr{D}_{23}(\boldsymbol{\nu}) \\ 0 & \mathscr{D}_{32}(\boldsymbol{\nu}) & \mathscr{D}_{33}(\boldsymbol{\nu}) \end{bmatrix} \tag{4}$$

where $\mathscr{D}_{11}(\boldsymbol{\nu}) = -\mathscr{X}_u - \mathscr{X}_{|u|u}|u| - \mathscr{X}_{uuu}u^2$, $\mathscr{D}_{22}(\boldsymbol{\nu}) = -\mathscr{Y}_v - \mathscr{Y}_{|v|v}|v| - \mathscr{Y}_{|r|v}|r|$, $\mathscr{D}_{23}(\boldsymbol{\nu}) = -\mathscr{Y}_r - \mathscr{Y}_{|v|r}|v| - \mathscr{Y}_{|r|r}|r|$, $\mathscr{D}_{32}(\boldsymbol{\nu}) = -\mathscr{N}_v - \mathscr{N}_{|v|v}|v| - \mathscr{N}_{|r|v}|r|$, $\mathscr{D}_{33}(\boldsymbol{\nu}) = -\mathscr{N}_r - \mathscr{N}_{|v|r}|v| - \mathscr{N}_{|r|r}|r|$, and $\mathscr{X}_{(\cdot)}$, $\mathscr{Y}_{(\cdot)}$, and $\mathscr{N}_{(\cdot)}$ are hydrodynamic coefficients [19]. The rotation matrix $\mathscr{R}$ is

$$\mathscr{R}(\boldsymbol{\eta}) = \begin{bmatrix} \cos\psi_p & -\sin\psi_p & 0 \\ \sin\psi_p & \cos\psi_p & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

By defining $\boldsymbol{x} = \left[\boldsymbol{\eta}^T, \boldsymbol{\nu}^T\right]^T$, it yields

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & \mathscr{R}(\boldsymbol{\eta}) \\ 0 & \mathscr{H}(\boldsymbol{\nu}) \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ \mathscr{N} \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} 0 \\ -\mathscr{M}^{-1}\mathscr{G}(\boldsymbol{\nu}) \end{bmatrix} \tag{5}$$

where $\mathscr{H}(\boldsymbol{\nu}) = -\mathscr{M}^{-1}(\mathscr{C}(\boldsymbol{\nu}) + \mathscr{D}(\boldsymbol{\nu}))$ and $\mathscr{N} = -\mathscr{M}^{-1}\mathscr{B}$. The state measurement of the ASV system (1) is corrupted by noises and sensor faults, so it is expressed as

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n} + \boldsymbol{f}(t)$$

where $\boldsymbol{n} \in \mathbb{R}^6$ is the measurement noise and $\boldsymbol{f}(t) \in \mathbb{R}^6$ denotes the possible sensor fault. In this paper, we only consider sensor faults on the measurement of the heading angular rate r_p, so $\boldsymbol{f}(t) = [0, 0, 0, 0, 0, f_r(t)]^T$. The sensor fault $f_r(t)$ is given by

$$f_r(t) = \beta(t - T_f)\,\phi(t - T_f)$$

where $\phi(t - T_f)$ is the unknown function of the sensor fault that occurs at the time instant T_f, and $\beta(t - T_f)$ is the time profile with $\beta(t - T_f) = 0$ for $t \le T_f$ and $\beta(t - T_f) = 1 - e^{-k(t - T_f)}$ for $t > T_f$ where k is the evolution rate of the fault [21], [22]. Note that $k \to \infty$, if the occurrence of a sensor fault is abrupt, e.g., bias fault.

III. Reinforcement learning-based fault tolerant control scheme

This section starts with the presentation of a model-reference control structure, then provides preliminaries on RL, and eventually, presents the new FTC scheme.

A. Model-reference control structure

For most ASV systems, accurate nonlinear dynamic model is rarely available. Major uncertainties come from $\mathscr{M}$, $\mathscr{C}(\boldsymbol{\nu})$, and $\mathscr{D}(\boldsymbol{\nu})$ due to hydrodynamics, and $\mathscr{G}(\boldsymbol{\nu})$ due to gravitational and buoyancy forces and moments. Although the ASV dynamics are subject to uncertainties, a nominal model is still available based on the known information on the ASV dynamics (5). The nominal model of (5) is

$$\dot{\boldsymbol{x}}_m = \begin{bmatrix} 0 & \mathscr{R}(\boldsymbol{\eta}_m) \\ 0 & \mathscr{H}_m \end{bmatrix} \boldsymbol{x}_m + \begin{bmatrix} 0 \\ \mathscr{N}_m \end{bmatrix} \boldsymbol{u}_m \tag{6}$$

where $\mathscr{N}_m$ and $\mathscr{H}_m$ contain all the known constant parameters of the ASV dynamics (5), and $\boldsymbol{\eta}_m = [x_m, y_m, \psi_m]^T \in \mathbb{R}^3$ is a generalized coordinate vector of the nominal model. In this paper, $\mathscr{M}_m$ is given by $\mathscr{M}_m = diag\{\mathscr{M}_{11}, \mathscr{M}_{22}, \mathscr{M}_{33}\}$, $\mathscr{H}_m = \mathscr{M}_m^{-1}\mathscr{D}_m$ with $\mathscr{D}_m = diag\{-\mathscr{X}_u, -\mathscr{Y}_v, -\mathscr{N}_r\}$, and $\mathscr{N}_m = \mathscr{M}_m^{-1}\mathscr{B}$. Hence, in the nominal model, all the nonlinear terms in the inne-loop dynamics are ignored, so we end up with a linear and decoupled model for the dynamic equations of the generalized velocity vector $\boldsymbol{\nu}$. As the dynamics of the nominal model (6) are known, it is possible to design a control law $\boldsymbol{u}_m$ allowing the states of the nominal system (6) to converge to a reference signal $\boldsymbol{x}_r$, i.e., $\|\boldsymbol{x}_m - \boldsymbol{x}_r\|_2 \to 0$ as $t \to \infty$. The control law $\boldsymbol{u}_m$ can also be used by the full ASV dynamics (5) as a baseline control.

In the model-reference control structure, the objective is to design a control law allowing the states of (5) to track those of the nominal model (6), so the overall control law is

$$\boldsymbol{u} = \boldsymbol{u}_b + \boldsymbol{u}_l \tag{7}$$

where $\boldsymbol{u}_b$ is a baseline controller, and $\boldsymbol{u}_l$ is a control policy from the deep RL module. The baseline control $\boldsymbol{u}_b$ is employed to ensure some basic performance, i.e., local stability, while $\boldsymbol{u}_l$ is employed to compensate for system uncertainties and sensor faults. The baseline control $\boldsymbol{u}_b$ is designed based on the nominal model (6), so it has the same expression as $\boldsymbol{u}_m$ in (6). Note that $\boldsymbol{u}_b$ or $\boldsymbol{u}_m$ can be designed by any existing model-based method. Hence, we focus on the development of $\boldsymbol{u}_l$ using RL.

B. Reinforcement learning

The formulation of RL is based on a Markov decision process (MDP) denoted by a tuple $\mathcal{MDP} := \langle \mathcal{S}, \mathcal{U}, \mathcal{P}, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{U}$ specifies the action/input space, $\mathcal{P} : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to \mathbb{R}$ defines a transition probability, $R : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1)$ is a discount factor. In MDP, $\boldsymbol{s} \in \mathcal{S}$ contains all available signals affecting the RL control $\boldsymbol{u}_l \in \mathcal{U}$. In this paper, the transition probability is characterized by (1) and a reference signal, $\boldsymbol{x}_r$.

Let $\boldsymbol{s}_t$ be the state signal $\boldsymbol{s}$ at the time step t, and accordingly, $\boldsymbol{u}_{l,t}$ be the input from the RL-based control. The RL algorithm aims to maximize an action-value function, a.k.a., Q-function, given as

$$Q_\pi(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) = R_t + \gamma \mathbb{E}_{\boldsymbol{s}_{t+1}} [V_\pi(\boldsymbol{s}_{t+1})] \tag{8}$$

where R_t is the reward function with $R_t = R(\boldsymbol{s}_t, \boldsymbol{u}_{l,t})$, $\mathbb{E}_{\boldsymbol{s}_{t+1}} [V_\pi(\boldsymbol{s}_{t+1})] = \sum_{\boldsymbol{s}_{t+1}} \mathcal{P}_{t+1|t} [V_\pi(\boldsymbol{s}_{t+1})]$, and $V_\pi(\boldsymbol{s}_{t+1})$ is called state value function for $\boldsymbol{s}_{t+1}$, where

$$\begin{aligned} V_\pi(\boldsymbol{s}_t) = & \sum_{\boldsymbol{u}_{l,t}} \pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t) \mathbb{E}_{\boldsymbol{s}_{t+1}} \big[R_t + \gamma V_\pi(\boldsymbol{s}_{t+1}) + \\ & + \alpha \mathcal{H}(\pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t)) \big] \\ = & \mathbb{E}_\pi \big[\mathbb{E}_{\boldsymbol{s}_{t+1}} [R_t + \gamma V_\pi(\boldsymbol{s}_{t+1})] - \alpha \log \pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t) \big] \end{aligned} \tag{9}$$

where $\mathcal{H}(\pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t)) = -\mathbb{E}_\pi [\log(\pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t))]$ is the entropy of the policy, α is a temperature parameter, and $\pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t)$ is the control policy that is the probability of choosing an action $\boldsymbol{u}_{l,t} \in \mathcal{U}$ at a state $\boldsymbol{s}_t \in \mathcal{S}$ [18].

The objective in RL is to solve the optimization problem below.

$$\pi^* = \arg\max Q_\pi(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) \tag{10}$$

C. Fault diagnosis and estimation

For the clarity of the presentation, we only consider sensor faults in the measurement of inner-loop states that are $\boldsymbol{\nu}$. Hence, only the inner-loop dynamics of ASVs are considered in the development of the fault diagnosis and estimation. However, the proposed algorithm can be extended to more generic situations, for instance, sensor faults in the measurement of positions.

The overall control design is based on the model-reference control structure given in Section III-A, so the uncertain inner-loop dynamics of the ASV model (5) is rewritten as

$$\dot{\boldsymbol{\nu}} = \mathscr{H}_m \boldsymbol{\nu} + \mathscr{N}_m (\boldsymbol{u}_b + \boldsymbol{u}_l) + \boldsymbol{\beta}(\boldsymbol{\nu}) \tag{11}$$

where $\boldsymbol{\beta}(\boldsymbol{\nu})$ is the aggregation of all uncertainties in the inner-loop dynamics. Assume that $\boldsymbol{\beta}(\boldsymbol{\nu})$ is bounded. Let $\boldsymbol{e}_{\boldsymbol{\nu}} = \boldsymbol{\nu} - \boldsymbol{\nu}_m$. According to (6) and (12), one has

$$\dot{\boldsymbol{e}}_{\boldsymbol{\nu}} = \mathscr{H}_m \boldsymbol{e}_{\boldsymbol{\nu}} + \mathscr{N}_m (\boldsymbol{u}_b - \boldsymbol{u}_m) + \mathscr{N}_m \boldsymbol{u}_l + \boldsymbol{\beta}(\boldsymbol{\nu}) \tag{12}$$

Under healthy conditions, the model uncertainty term $\boldsymbol{\beta}(\boldsymbol{\nu})$ can be fully compensated using a learning-based control $\boldsymbol{u}_l$ according to [17], [18]. It implies that $\|\boldsymbol{e}_{\boldsymbol{\nu}}(t)\|_2 \leq \epsilon$ as $t \to \infty$, where ϵ is a certain positive small constant. If sensor faults happened, the error signal $\boldsymbol{e}_{\boldsymbol{\nu}}$ will be large than ϵ. A naive idea for the learning-based FTC is to treat the sensor faults as part of external disturbances. However, treating sensor faults as disturbances will result in a conservative learning-based control like the robust control. Hence, we introduce a fault diagnosis and estimation mechanism that allows the learning-based control to adapt to different scenarios: healthy and unhealthy conditions.

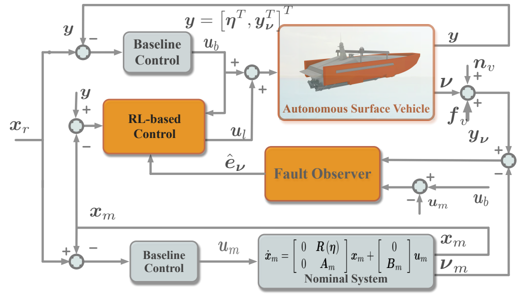

Fig. 1: RL-based FTC scheme

Let $\boldsymbol{y}_{\boldsymbol{\nu}} = \boldsymbol{\nu} + \boldsymbol{n}_{\boldsymbol{\nu}} + \boldsymbol{f}_{\boldsymbol{\nu}}$, where $\boldsymbol{n}_{\boldsymbol{\nu}}$ denotes the measurement noise and $\boldsymbol{f}_{\boldsymbol{\nu}}$ is the sensor fault. Furthermore, we define $\boldsymbol{e}_{y_\nu} = \boldsymbol{y}_{\boldsymbol{\nu}} - \boldsymbol{\nu}_m = \boldsymbol{e}_{\boldsymbol{\nu}} + \boldsymbol{n}_{\boldsymbol{\nu}} + \boldsymbol{f}_{\boldsymbol{\nu}}$ as the faulty residual vector. In real applications, $\boldsymbol{e}_{y_\nu}$ is measurable instead of $\boldsymbol{e}_{\boldsymbol{\nu}}$. The fault diagnosis and estimation mechanism is, therefore,

$$\dot{\hat{\boldsymbol{e}}}_{\boldsymbol{\nu}} = \mathscr{H}_m \hat{\boldsymbol{e}}_{\boldsymbol{\nu}} + \mathscr{N}_m (\boldsymbol{u}_b - \boldsymbol{u}_m) + \boldsymbol{L}(\boldsymbol{e}_{y_\nu} - \hat{\boldsymbol{e}}_{\boldsymbol{\nu}}) \tag{13}$$

where $\boldsymbol{L}$ is chosen such that $\mathscr{H}_m - \boldsymbol{L}$ is Hurwitz. The signal $\hat{\boldsymbol{e}}_{\boldsymbol{\nu}}$ performs as the indicator of the occurrence and strength of the sensor faults. Let $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}} = \boldsymbol{e}_{\boldsymbol{\nu}} - \hat{\boldsymbol{e}}_{\boldsymbol{\nu}}$, and we have

$$\dot{\boldsymbol{\varepsilon}}_{\boldsymbol{\nu}} = (\mathscr{H}_m - \boldsymbol{L}) \boldsymbol{\varepsilon}_{\boldsymbol{\nu}} + \mathscr{N}_m \boldsymbol{u}_l + \boldsymbol{\beta}(\boldsymbol{\nu}) + \boldsymbol{L}(\boldsymbol{n}_{\boldsymbol{\nu}} + \boldsymbol{f}_{\boldsymbol{\nu}}) \tag{14}$$

The following lemma exists for (14)

Lemma 1: Suppose that the model uncertainty term $\boldsymbol{\beta}(\boldsymbol{\nu})$ can be fully compensated using a RL-based control $\boldsymbol{u}_l$.The error dynamic model (14) is input-to-state stable with respect to the sensor fault $\boldsymbol{f}_{\boldsymbol{\nu}}$.

Proof: Let $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}} = \boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^1 + \boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^2$ with

$$\begin{aligned} \dot{\boldsymbol{\varepsilon}}_{\boldsymbol{\nu}}^1 &= (\mathscr{H}_m - \boldsymbol{L}) \boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^1 + \mathscr{N}_m \boldsymbol{u}_l + \boldsymbol{\beta}(\boldsymbol{\nu}) \\ \dot{\boldsymbol{\varepsilon}}_{\boldsymbol{\nu}}^2 &= (\mathscr{H}_m - \boldsymbol{L}) \boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^2 + \boldsymbol{L}(\boldsymbol{n}_{\boldsymbol{\nu}} + \boldsymbol{f}_{\boldsymbol{\nu}}) \end{aligned}$$

If $\boldsymbol{\beta}(\boldsymbol{\nu})$ is fully compensated by $\boldsymbol{u}_l$, it implies that $\mathscr{N}_m \boldsymbol{u}_l + \boldsymbol{\beta}(\boldsymbol{\nu})$ together can be treated as a bounded negligible external signal. Since $\mathscr{H}_m - \boldsymbol{L}$ is Hurwitz, $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^1$ will be input-to-state stable with respect to $\mathscr{N}_m \boldsymbol{u}_l + \boldsymbol{\beta}(\boldsymbol{\nu})$. It implies that $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^1$ will be negligible as well. Hence, the magnitude of $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}}$ mainly results from $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^2$. Since $\mathscr{H}_m - \boldsymbol{L}$ is Hurwitz, $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}}^2$ is input-to-state stable with respect to $\boldsymbol{f}_{\boldsymbol{\nu}}$. Hence, we can conclude that $\boldsymbol{\varepsilon}_{\boldsymbol{\nu}}$ is input-to-state stable with respect to $\boldsymbol{f}_{\boldsymbol{\nu}}$. ■

D. RL-based fault tolerant control

The RL-based fault tolerant control is developed using the output from the fault diagnosis and estimation mechanism (13) as shown in Figure 1. RL learns the control policies using data samples, including input and state data, at discrete time steps. Assume that the sample time step is fixed and denoted by δt. Without loss of generality, let $\boldsymbol{y}_t$, $\boldsymbol{u}_{b,t}$, $\boldsymbol{u}_{l,t}$, and $\hat{\boldsymbol{e}}_{\boldsymbol{\nu},t}$ be the ASV state, the baseline control action, the control action

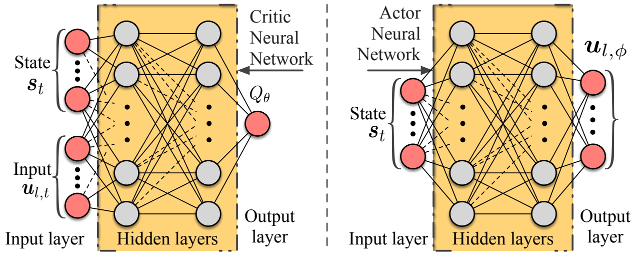

Fig. 2: MLP networks for Q_θ and π_ϕ

from RL, and the output of the fault diagnosis and estimation mechanism at the time step t, respectively. The state signal $\boldsymbol{s}$ at the time step t is thus $\boldsymbol{s}_t = \left[\boldsymbol{x}_{m,t}^T - \boldsymbol{y}_t^T, \boldsymbol{u}_{b,t}^T, \hat{\boldsymbol{e}}_{\boldsymbol{\nu},t}^T\right]^T$.

The learning-based fault tolerant robust control $\boldsymbol{u}_l$ will be learned instead of designed based on RL according to Section III-B. The training/learning process of RL will repeatedly execute policy evaluation and policy improvement. In the policy evaluation, the Q-value in (8) is computed by applying a Bellman operation $Q_\pi(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) = \mathcal{T}^\pi Q_\pi(\boldsymbol{s}_t, \boldsymbol{u}_{l,t})$ where

$$\mathcal{T}^\pi Q_\pi(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) = R_t + \gamma \mathbb{E}_{\boldsymbol{s}_{t+1}}\{\mathbb{E}_\pi[Q_\pi(\boldsymbol{s}_{t+1}, \boldsymbol{u}_{l,t+1}) \\ -\alpha \ln(\pi(\boldsymbol{u}_{l,t+1}|\boldsymbol{s}_{t+1}))]\} \quad (15)$$

In the policy improvement, the policy is updated by

$$\pi_{new} = \arg\min_{\pi' \in \Pi} \mathscr{D}_{KL}\left(\pi'(\cdot|\boldsymbol{s}_t) \middle\| Z^{\pi_{old}} e^{\frac{1}{\alpha} Q^{\pi_{old}}(\boldsymbol{s}_t, \cdot)}\right) \quad (16)$$

where Π denotes a policy set, π_{old} denotes the policy from the last update, $Q^{\pi_{old}}$ is the Q-value of π_{old}, $\mathscr{D}_{KL}$ denotes the Kullback-Leibler (KL) divergence, and $Z^{\pi_{old}}$ is a normalization factor. The objective can be transformed into

$$\pi^* = \arg\min_{\pi \in \Pi} \mathbb{E}_\pi\left[\alpha \ln(\pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t)) - Q(\boldsymbol{s}_t, \boldsymbol{u}_{l,t})\right] \quad (17)$$

More details on how (17) is obtained can be found in [23].

IV. Algorithm Implementation Using Deep Neural Networks

In this paper, both $Q_\pi(\boldsymbol{s}_t, \boldsymbol{u}_{l,t})$ and $\pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t)$ are approximated by fully connected multiple layer perceptrons (MLP) with rectified linear unit (ReLU) nonlinearities as the activation functions. The ReLU function is

$$\underline{relu}(z) = \max\{z, 0\}$$

The ReLU activation function outperforms other activation functions like sigmoid functions [24]. For a vector $z = [z_1, \ldots, z_n]^T$, $\underline{relu}(z) = [\underline{relu}(z_1), \ldots, \underline{relu}(z_n)]^T$. More details on the MLP can be found in [18].

The whole training process will be offline. At each time step $t+1$, we collect data samples, such as an input from the last time step $\boldsymbol{u}_{l,t}$, a state from the last time step $\boldsymbol{s}_t$, a reward R_t, and a current state $\boldsymbol{s}_{t+1}$. Those historical data will be stored as a tuple $(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}, R_t, \boldsymbol{s}_{t+1})$ at a replay memory $\mathcal{D}$ [25]. At each policy evaluation or improvement step, we randomly sample a batch of historical data, $\mathcal{B}$, from the replay memory $\mathcal{D}$ for the training of the parameters θ and ϕ. Starting the training, we apply the baseline control policy $\boldsymbol{u}_b$ to an ASV system to collect the initial data $\mathcal{D}_0$ as shown in Algorithm 1. The initial data set $\mathcal{D}_0$ is used for the initial fitting of Q-value functions. When the initialization is over, we execute both $\boldsymbol{u}_b$ and the latest updated reinforcement learning policy $\pi_\phi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t)$ to run the ASV system.

The parameters θ are trained to minimize

$$J_Q(\theta) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) - Y_{target}\right)^2\right] \quad (18)$$

where $(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) \sim \mathcal{D}$ implies that we randomly pick data samples $(\boldsymbol{s}_t, \boldsymbol{u}_{l,t})$ from a replay memory $\mathcal{D}$, and

$$Y_{target} = R_t + \gamma \mathbb{E}_{\boldsymbol{s}_{t+1}}\left[\mathbb{E}_\pi\left[Q_{\bar{\theta}}(\boldsymbol{s}_{t+1}, \boldsymbol{u}_{l,t+1}) - \alpha \log(\pi_\phi)\right]\right]$$

where $\bar{\theta}$ is the target parameter which will be updated slowly. The DNN parameters θ are obtained by applying the stochastic gradient descent. to (18) on a data batch $\mathcal{B}$ with a fixed size denoted by $|\mathcal{B}|$. In the final implementation, we use two critics which are parameterized by θ_1 and θ_2, respectively. The two critics are introduced to reduce the over-estimation issue in the training of critic neural networks [26], so

$$Y_{target} = R_t + \gamma \min\Big\{Q_{\bar{\theta}_1}(\boldsymbol{s}_{t+1}, \boldsymbol{u}_{l,t+1}), \\ Q_{\bar{\theta}_2}(\boldsymbol{s}_{t+1}, \boldsymbol{u}_{l,t+1})\Big\} - \gamma \alpha \log(\pi_\phi) \quad (19)$$

The policy improvement is to minimize the following objective function using data samples from the replay memory.

$$J_\pi(\phi) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}) \sim \mathcal{D}}\Big(\alpha \log(\pi_\phi) - Q_\theta(\boldsymbol{s}_t, \boldsymbol{u}_{l,t})\Big) \quad (20)$$

Parameter ϕ is trained using a stochastic gradient descent technique. At the training stage, the actor neural network is

$$\boldsymbol{u}_{l,\phi} = \bar{\boldsymbol{u}}_{l,\phi} + \boldsymbol{\sigma}_\phi^{\frac{1}{2}} \odot \boldsymbol{\xi} \quad (21)$$

where $\bar{\boldsymbol{u}}_{l,\phi}$ is the parameterized control law to be learned, $\boldsymbol{\sigma}_\phi^{\frac{1}{2}}$ is the standard deviation of the exploration noise, $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is the Gaussian noise, and "$\odot$" is the Hadamard product. The exploration noise $\boldsymbol{\xi}$ is only applied to the training stage. Once the training is done, we only need $\bar{\boldsymbol{u}}_{l,\phi}$ in the implementation. Hence, at the training stage, $\boldsymbol{u}_l$ in Figure 1 is equal to $\boldsymbol{u}_{l,\phi}$. Once the training is over, we have $\boldsymbol{u}_l = \bar{\boldsymbol{u}}_{l,\phi}$.

The temperature parameter α is also updated by minimizing

$$J_\alpha = \mathbb{E}_\pi\left[-\alpha \log \pi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t) - \alpha \bar{\mathcal{H}}\right] \quad (22)$$

where $\bar{\mathcal{H}}$ is a target entropy. The entire algorithm is summarized in Algorithm 1, where ι_Q, ι_π, and ι_α are positive learning rates (scalars), and $\kappa > 0$ is a constant scalar.

V. Numerical Simulations

In this section, the proposed learning-based control algorithm is implemented to the trajectory tracking control of a supply ship model presented in [20]. Model parameters can be found in [18]. The unmodeled dynamics in the simulations are given by $g_1 = 0.279uv^2 + 0.342v^2r$, $g_2 = 0.912u^2v$, and $g_3 = 0.156ur^2 + 0.278urv^3$, respectively. The baseline control $\boldsymbol{u}_b$ is designed based on a nominal model in (6) in

Algorithm 1 Reinforcement learning for fault tolerant control

Initialize parameters θ_1, θ_2 for Q_{θ_1} and Q_{θ_2}, respectively, and ϕ for the actor network (21).
Assign values to the the target parameters $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$, $\mathcal{D} \leftarrow \emptyset$, $\mathcal{D}_0 \leftarrow \emptyset$,
Get data set $\mathcal{D}_0$ by running $\boldsymbol{u}_b$ on (5) with $\boldsymbol{u}_l = \mathbf{0}$
Turn off the exploration and train initial critic parameters θ_1^0, θ_2^0 using $\mathcal{D}_0$ according to (18).
Initialize the replay memory $\mathcal{D} \leftarrow \mathcal{D}_0$
Assign initial values to critic parameters $\theta_1 \leftarrow \theta_1^0$, $\theta_2 \leftarrow \theta_2^0$ and their targets $\bar{\theta}_1 \leftarrow \theta_1^0$, $\bar{\theta}_2 \leftarrow \theta_2^0$
repeat
  for each data collection step **do**
    Choose an action $\boldsymbol{u}_{l,t}$ according to $\pi_\phi(\boldsymbol{u}_{l,t}|\boldsymbol{s}_t)$
    Run both (5), (6), and (13) & collect $\boldsymbol{s}_{t+1} = \{\boldsymbol{x}_{t+1}, \boldsymbol{x}_{m,t+1}, \boldsymbol{u}_{b,t+1}\}$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{\boldsymbol{s}_t, \boldsymbol{u}_{l,t}, R(\boldsymbol{s}_t, \boldsymbol{u}_{l,t}), \boldsymbol{s}_{t+1}\}$
  end for
  for each gradient update step **do**
    Sample a batch of data $\mathcal{B}$ from $\mathcal{D}$
    $\theta_j \leftarrow \theta_j - \iota_Q \nabla_\theta J_Q(\theta_j)$, and $j = 1, 2$
    $\phi \leftarrow \phi - \iota_\pi \nabla_\phi J_\pi(\phi)$,
    $\alpha \leftarrow \alpha - \iota_\alpha \nabla_\alpha J_\alpha(\alpha)$
    $\bar{\theta}_j \leftarrow \kappa\theta_j + (1-\kappa)\bar{\theta}_j$, and $j = 1, 2$
  end for
until convergence (i.e. $J_Q(\theta) <$ a small threshold)

TABLE I: RL configurations

Parameters	Values
Learning rate ι_Q	0.001
Learning rate ι_π	0.0001
Learning rate ι_α	0.0001
κ	0.01
Actor neural network	fully connected with three hidden layers ($256 \times 128 \times 64$ neurons)
critic neural networks	fully connected with two hidden layers ($256 \times 256 \times 32$ neurons)
Replay memory capacity	1.5×10^6
Sample batch size	512
γ	0.998
Training episodes	1500
Steps per episode	1000
time step size δt	0.1

terms of the PID control method. The reference signal is assumed to be produced by the following motion planner,

$$\dot{\boldsymbol{\eta}}_r = \boldsymbol{R}(\boldsymbol{\eta}_r)\boldsymbol{\nu}_r \quad \dot{\boldsymbol{\nu}}_r = \boldsymbol{a}_r \tag{23}$$

where $\boldsymbol{\eta}_r = [x_r, y_r, \psi_r]^T$, $\boldsymbol{\nu}_r = [u_r, 0, r_r]^T$, and $\boldsymbol{a}_r = [\dot{u}_r, 0, \dot{r}_r]^T$. The initial position vector is $\boldsymbol{\eta}_r(0) = \left[0, 0, \frac{\pi}{4}\right]^T$. We set $u_r(0) = 0.4\ m/s$, $r_r(0) = 0\ rad/s$, and $\dot{u}_r = 0$. The derivative of the reference angular rate $\dot{r}_r$ is

$$\dot{r}_r = \begin{cases} \frac{\pi}{300}\ rad/s^2 & \text{if } 25\ s \le t < 35\ s \\ -\frac{\pi}{300}\ rad/s^2 & \text{if } 35\ s \le t < 45\ s \\ -\frac{\pi}{300}\ rad/s^2 & \text{if } 65\ s \le t < 75\ s \\ \frac{\pi}{300}\ rad/s^2 & \text{if } 75\ s \le t < 85\ s \\ 0\ rad/s^2 & \text{otherwise} \end{cases} \tag{24}$$

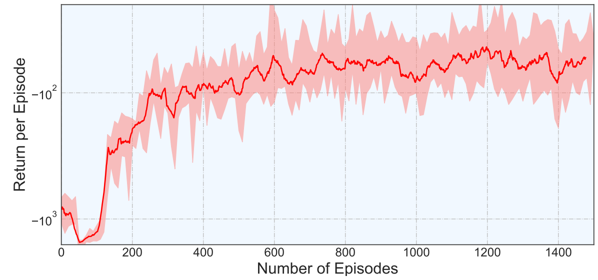

Fig. 3: Learning curves of the RL-based FTC

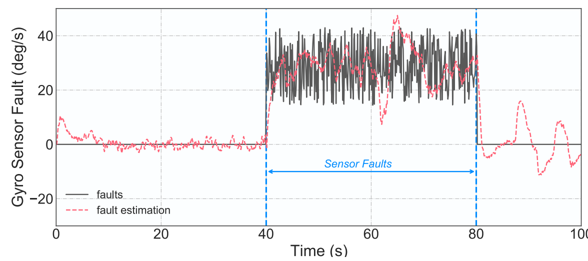

Fig. 4: Sensor fault and its estimation

The bias sensor faults of interest are defined as

$$f_r(t) = \beta(t - T_f)(\phi(t - T_f) + n_\phi) \tag{25}$$

where n_ϕ is a random noise with a uniformly random distribution. In the simulation, Gaussian measurement noises $\boldsymbol{n} \sim \mathcal{N}(0, \boldsymbol{\sigma}_n)$ are added to the measurement, where $\boldsymbol{\sigma}_n = diag\{0.025, 0.025, 0.01, 0.01, 0.005, 0.005\}$.

At the training stage, we run the ASV system for $100\ s$, and the training processes are repeated for 1500 times (i.e., 1500 episodes). Figure 3 shows the learning curves of the proposed algorithm. At each episode, we uniformly randomly sample $x(0)$ and $y(0)$ from $(-1.5, 1.5)$, $\psi(0)$ from $(-0.25\pi, 0.25\pi)$ and $u(0)$ from $(0.1, 0.6)$, and we choose $v(0) = 0$ and $r(0) = 0$. The proposed control algorithm is compared with

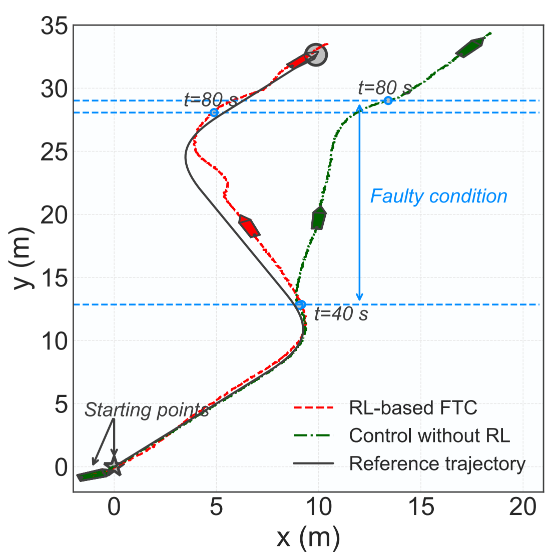

Fig. 5: Trajectory tracking performance

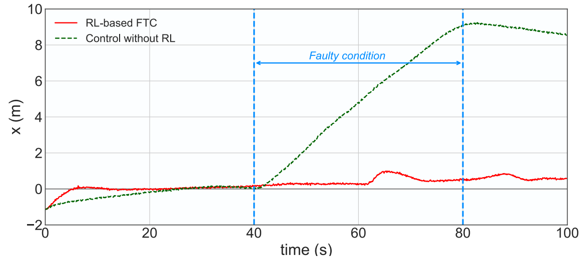

Fig. 6: Position tracking error in the x coordinate

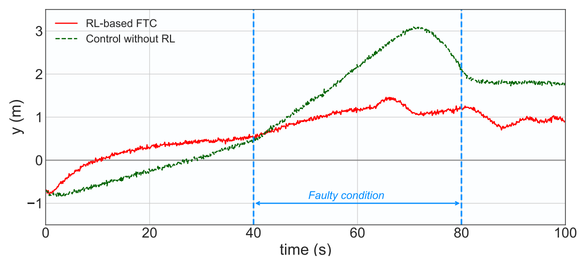

Fig. 7: Position tracking error in the y coordinate

a benchmark design in which only the baseline control $\boldsymbol{u}_b$ is considered. Configurations for the training and neural networks are found in Table I. The matrix $\boldsymbol{G}$ and $\boldsymbol{H}$ are chosen to be $\boldsymbol{G} = diag\{0.025, 0.025, 0.0016, 0.005, 0.001, 0\}$ and $\boldsymbol{H} = diag\left\{1.25e^{-4}, 1.25e^{-4}, 8.3e^{-5}\right\}$, respectively.

At the evaluation stage, the sensor fault profile shown in Fig. 4 is implemented to the ASV. The simulation results are summarized in Figs. 5–7. With our new design, the ASV learns to adapt to a bias sensor fault that happens to the measurement of the angular rate r_p. The trajectory tracking under faulty scenario is significantly improved.

VI. Conclusions

In this paper, a novel reinforcement learning-based fault tolerant control algorithm is presented for ASV systems subject to model uncertainties and sensor faults. The new algorithm is obtained by combining a model-reference reinforcement learning with a fault diagnosis and estimation mechanism. With the new RL-based fault tolerant control, we ensured the ASV can learn to adapt to bias faults in the gyro and recover the trajectory tracking performance under faulty conditions.

References

[1] P. Švec, A. Thakur, E. Raboin, B. C. Shah, and S. K. Gupta, "Target following with motion prediction for unmanned surface vehicle operating in cluttered environments," *International Journal of Robotics Research*, no. 36, pp. 383 – 405, Apr. 2014.

[2] D. D. Bloisi, F. Previtali, A. Pennisi, D. Nardi, and M. Fiorini, "Enhancing automatic maritime surveillance systems with visual information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 824 – 833, Apr. 2017.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, Mar. 2004.

[4] X. Jin, "Fault tolerant finite-time leader-follower formation control for autonomous surface vessels with los range and angle constraints," *Automatica*, vol. 68, pp. 228–236, Jun. 2016.

[5] R. J. Patton, "Fault-tolerant control," in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds., Jul. 2015, pp. 201–213.

[6] H. Yang, B. Jiang, H. H. T. Liu, H. Yang, and Q. Zhang, "Attitude synchronization for multiple 3-DoF helicopters with actuator faults," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 2, pp. 597 – 608, Apr. 2019.

[7] K. Zhou and Z. Ren, "A new controller architecture for high performance, robust, and fault-tolerant control," *IEEE Transactions on Automatic Control*, vol. 46, no. 1, pp. 1613–1618, Oct. 2001.

[8] M. Blanke, M. Kinnaert, M. Staroswiecki, and J. Lunze, *Diagnosis and Fault-Tolerant Control*, 2nd ed. Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2006.

[9] S. Yin, H. Gao, J. Qiu, and O. Kaynak, "Descriptor reduced-order sliding mode observers design for switched systems with sensor and actuator faults," *Automatica*, vol. 76, pp. 282–292, Feb. 2017.

[10] H. Hamadi, B. Lussier, I. Fantoni, C. Francis, and H. Shraim, "Comparative study of self tuning, adaptive and multiplexing ftc strategies for successive failures in an octorotor uav," *Robotics and Autonomous Systems*, vol. 133, Nov. 2020.

[11] J. Jiang and Y. Zhang, "Accepting performance degradation in fault-tolerant control system designg," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 2, pp. 156–172, Mar. 2006.

[12] Z. Wang, L. Liu, Y. Wu, and H. Zhang, "Optimal fault-tolerant control for discrete-time nonlinear strict-feedback systems based on adaptive critic design," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2179–2191, Jun. 2018.

[13] I. Ahmed, H. Khorasgani, and G. Biswas, "Comparison of model predictive and reinforcement learning methods for fault tolerant control," *IFAC-Papers OnLine*, vol. 51, no. 24, pp. 233–240, Dec. 2018.

[14] H. Zhang, K. Zhang, Y. Cai, and J. Han, "Adaptive fuzzy fault-tolerant tracking control for partially unknown systems with actuator faults via integral reinforcement learning method," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 7, pp. 1986 – 1998, 2019.

[15] W. Shi, S. Song, C. Wu, and C. L. P. Chen, "Multi pseudo q-learning-based deterministic policy gradient for tracking control of autonomous underwater vehicles," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 12, pp. 3534 – 3546, Dec. 2019.

[16] D. Mguni, "Cutting your losses: Learning fault-tolerant control and optimal stopping under adverse risk," *arXiv:1902.05045*, 2019.

[17] Q. Zhang, W. Pan, and V. Reppa, "Model-reference reinforcement learning control of autonomous surface vehicles," in *Proc. of 59th IEEE Conference on Decision and Control*, Jeju, Korea (South), Dec. 2020.

[18] ——, "Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2021, (Early access, doi:10.1109/TITS.2021.3086033).

[19] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Inc., 2011.

[20] R. Skjetne, T. I. Fossen, and P. V. Kokotović, "Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory," *Mathematics of Operations Research*, vol. 41, pp. 289 – 298, 2005.

[21] V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, "Decentralized isolation of multiple sensor faults in large-scale interconnected nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1582 – 1596, Mar. 2015.

[22] ——, *Sensor Fault Diagnosis*. Now Foundations and Trends, 2016.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. of the 35th International Conference on Machine Learning*, vol. 80, Stockholmsmässan, Stockholm Sweden, Jul. 2018, pp. 1861–1870.

[24] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proc. of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, May 2013.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, and et. al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[26] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of the 35th International Conference on Machine Learning Conference*, vol. 80, Stockholmsmässan, Stockholm Sweden, Jul. 2018, pp. 1587–1596.