Learn-to-Recover: Retrofitting UAVs with Reinforcement Learning-Assisted Flight Control Under Cyber-Physical Attacks

Fan Fei[+], Zhan Tu[+], Dongyan Xu[++] and Xinyan Deng[+]

***Abstract*— In this paper, we present a generic fault-tolerant control (FTC) strategy via reinforcement learning (RL). We demonstrate the effectiveness of this method on quadcopter unmanned aerial vehicles (UAVs). The fault-tolerant control policy is trained to handle actuator and sensor fault/attack. Unlike traditional FTC, this policy does not require fault detection and diagnosis (FDD) nor tailoring the controller for specific attack scenarios. Instead, the policy is running simultaneously alongside the stabilizing controller without the need for on-detection activation. The effectiveness of the policy is compared with traditional active and passive FTC strategies against actuator and sensor faults. We compare their performance in position control tasks via simulation and experiments on quadcopters. The result shows that the strategy can effectively tolerate different types of attacks/faults and maintain the vehicle's position, outperforming the other two methods.**

I. Introduction

Autonomous aerial and ground vehicles have been increasingly applied to many tasks due to advancement in their sensors and actuators, together with sophisticated algorithms and computing power. With their expanding mission capabilities, their attack surface exposed to both cyber and physical attacks grows accordingly. This poses a significant threat particularly in its control system which is responsible for vehicle stabilization and performance.

Current research has been focused on securing the cyber aspect of cyber-physical systems such as memory protection [1] or firmware hardening [2]. However, if attacks are launched against the physical components of mobile robots/vehicles such as GPS spoofing [3]–[5] or using sound wave to resonate the IMU sensor [6], cyber (software/firmware) techniques can no longer protect the system and ensure safe operation of the vehicle. It has been found that signal spoofing is among the most likely and most effective attacks against UAV systems [7]–[9]. As a result, undesirable performance or even loss of control would occur. Given that attacks/faults cannot be fully prevented, flight recovery strategies and fault-tolerant capability are highly desirable for such autonomous mobile systems.

In this work, we introduce a novel reinforcement learning assisted flight recovery strategy against cyber-physical attacks. Unlike traditional Fault Tolerant Control (FTC) strategies where specific estimator and controller need to be designed for detection and recovery under particular fault, the original stabilizing controller can be kept as is. The uniqueness of this strategy is that the model-free nonlinear FTC policy is designed to be an add-on to the closed-loop system, which can be trained and implemented on an existing legacy system through minimally intrusive software retrofitted like the BlueBox [10]. For training, the vehicle dynamic is generally well known, the original control software can be either extract through binary reverse engineering or setup as software/hardware in the loop. The vehicle dynamics under various fault conditions can be difficult to model explicitly, however, it can be viewed as a partially observable Markov decision process, where the attacks or faults are considered as unobservable states. The policy can be optimized to handle both actuator and sensor attacks, and it would recover the vehicle flight to normal operation. The policy is constantly running alongside the vehicle's flight controller, and it does not require fault detection and activation. When the vehicle is operating normally, the policy generates no or minimum control command adjustment and does not interfere with the operation. When the fault condition arises or the vehicle is under attack, the policy takes the state inputs and generate appropriate actuator command adjustment with little to no delay to compensate for the fault/attack condition and maintains vehicle position/trajectory tracking.

To test the proposed strategy, we use a custom build quadcopter UAV as test platform running ArduPilot [11] as flight controller. To evaluate its performance, we compare its position control results against the BlueBox [10] and a robust controller (passive FTC). The results show that the RL based policy can maintain vehicle stability under sensor and actuator attack, and outperforms the other two methods.

II. Related Work

A. Fault-Tolerant Control

Traditionally, there are two types of fault-tolerant control: passive fault-tolerant control (PFTC) and active fault-tolerant control (AFTC) [12]. AFTC has a fault detection and diagnostics component (FDD) to identify the source of the fault, and the controller is reconfigured to compensate such fault. The FDD component is usually an observer [13]–[15], and can generate residual signals to indicate the fault [14], [16]. Both sensor and actuator attacks or failure can be detected with system model [10], [17]. If actuator fault is detected, the control torque can be redistributed accordingly [18] base on estimation or simple gain scheduling to improve performance when the motor fault is only loss of effectiveness [16], [19]. Mueller et al. showed that quadcopter can be controlled when loosing up to three rotors [20].

Meanwhile, PFTC does not have an FDD mechanism, but aiming at improving the controller's robustness to tolerate

[+]School of Mechanical Engineering, Purdue University
[++] Department of Computer Science, Purdue University
This work was supported in part by ONR under Grant N00014-17-1-2045.
(Email: xdeng@purdue.edu).

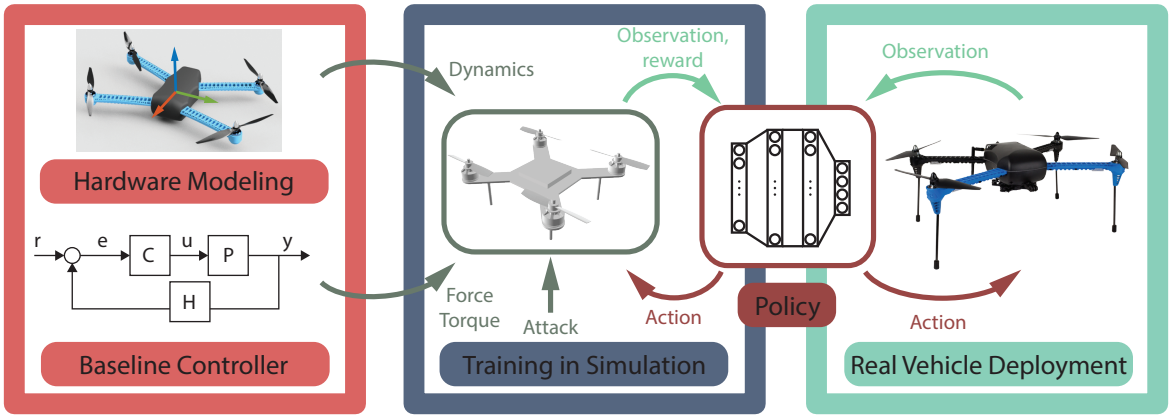

Fig. 1. The overview of the RL-assisted fault-tolerant control. The vehicle model is identified and the control algorithm is obtained through source code or reverse engineering. The information is used in a dynamic simulation where randomized attacks are injected. Using reinforcement learning, a fault-tolerant policy is optimized to maintain vehicle control under various attacks.

fault condition or attack [19]. These two types have their pros and cons. AFTC can pinpoint the fault and act accordingly, but if the FDD is not designed with care, the implementation could lead to delay in detection or false positives and greatly affects the performance. PFTC cannot isolate faults but could potentially achieve robust performance.

B. Reinforcement Learning

Machine learning and reinforcement learning has been explored in developing FTC strategies. Farivar et al. proposed methods using learning based observer and RL based controller to treat bounded additive sensor and actuator fault [21]. Liu and Wang et al. introduced a controller design incorporating an online RL policy assuming all states are measurable [22], [23]. Ahmadzadeh et al. use a model based approach to recover an underwater vehicle from actuator fault [24]. However, these methods were only evaluated in simulation, and their real-world performance is unknown.

Deploy reinforcement learning policy onto real systems, or commonly known as sim-to-real transfer, is a very difficult task and has gained a lot of attention recently. Controlling an unstable system such as quadcopter is especially challenging. Hwangbo et al. [25] achieved quadcopter position tracking using an RL policy with a weak attitude controller, while in [26], attitude control is tested with different RL algorithms. In [27], using a model-based reinforcement learning policy to control a small quadcopter is explored. More recently, [28] showed a generalized policy that can be transferred to multiple quadcopters. In this work, we demonstrate the RL policy proposed here trained in simulation can be transferred to real vehicle to recover from sensor and actuator fault.

III. Framework Procedure

Here, we propose a novel model-free fault and attack-resilient control strategy using reinforcement learning. The control method relies on a stabilizing controller for nominal flight control, and a model-free fault-tolerant policy optimized to generate additional torque necessary to maintain the vehicle's normal operation. The mathematical justification is similar to Structure Control Network [29], while only the nonlinear fault-tolerant policy is learned here. The learned policy can be viewed as an optimized FDD and FTC control approximated by a neural network. The overview of the procedure is shown in Fig. 1 and each component is described as follows.

A. Vehicle Model

For quadcopter UAV, the vehicle dynamics can be described by Newton-Euler equation:

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v}, & m\dot{\mathbf{v}} &= \mathbf{R}\mathbf{f}^b + mg\mathbf{e}_3, \\ \dot{\mathbf{R}} &= \mathbf{R}\hat{\boldsymbol{\omega}}^b, & \mathbf{J}\dot{\boldsymbol{\omega}}^b &+ \boldsymbol{\omega}^b \times \mathbf{J}\boldsymbol{\omega}^b = \boldsymbol{\tau}^b. \end{aligned} \tag{1}$$

where $\mathbf{p} = [x, y, z]^T$ is the vehicle's position in the inertial frame, $\mathbf{R}$ is the rotation matrix, $\mathbf{f}^b = [0, 0, F_z]^T$ is the body force, $\mathbf{e}_3$ is the unit vector $[0, 0, 1]^T$, $\boldsymbol{\omega}^b = [p, q, r]^T$ is the body angular velocity, $\hat{\cdot}$ denotes the skew-symmetric matrix mapping, $\mathbf{J}$ is the inertia matrix and $\boldsymbol{\tau}^b$ is the body torque.

The body force can be approximated by a simple fitting

$$\begin{aligned} F_z &= K_{F_z} u_z^2 + F_{z0} \\ \boldsymbol{\tau} &= \begin{bmatrix} K_{roll} u_{roll} + \tau_{x0} \\ K_{pitch} u_{pitch} + \tau_{y0} \\ K_{yaw} u_{yaw} + \tau_{z0} \end{bmatrix} \end{aligned} \tag{2}$$

and the voltage command of each motor is generated as

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} u_z \\ u_{roll} \\ u_{pitch} \\ u_{yaw} \end{bmatrix} \tag{3}$$

where K's are the lumped input gains; u_z, u_{roll}, u_{pitch} and u_{yaw} are the input voltage command for thrust, roll, pitch and yaw; F_{z0}, τ_{x0}, τ_{y0} and τ_{z0} are the thrust and torque offset introduced either by mechanical imperfection or fault; u_{1-4} are the voltage command for each motor.

Our test vehicle is a generic quadcopter assembled in-house with a commodity microcontroller. System identification is performed to obtain these dynamic model parameters. Each physical component is carefully measured, weighted and modeled in CAD software, and mass/inertia parameters are calculated. The thrust and drag constants of the motor and propeller are measured. For sensor fusion and feedback

control, a Vicon multi-camera position tracking system is used to provide position feedback.

B. Baseline Controller

The baseline controller is ArduPilot 3.3 multicopter flight controller [11]. It is a cascading PID controller. The altitude controller generates thrust by controlling the z axis acceleration with an inner loop PID controller. The outer loop PID controller uses altitude error to generate velocity then subsequently acceleration target. The x-y position is controlled by a proportional controller that generates velocity targets, where the velocity is controlled by a PID controller to generate roll pitch angle target. The attitude is also a cascading controller where the outer loop is a proportional controller generating angular velocity and the inner loop is a PID controller generating torque command.

C. Simulation Setup

A high-fidelity simulation tool was developed for the training and evaluation of the control strategies similar to [30]. Using the physical test vehicle as a blueprint, we recreated a virtual quadcopter in the simulation environment. Virtual sensors are implemented with noise characteristics similar to that of the real vehicle. Identical sensor fusion and control algorithms were implemented in the simulation so the closed-loop dynamics of the vehicle in the simulation can approximate the real vehicle.

The simulation is validated statically through open-loop and closed-loop tests. We sample the measured trajectories and compare the state transition against the simulation. The overall state transition error within 0.2s window ($\gg$ vehicle time constant) is $\leq 10\%$ across all state variables, which proves the simulation can capture the main dynamic effects accurately. Closed-loop flight tests were conducted and similar tracking error is observed on both experimental and simulated trajectories.

D. Model-free Fault Tolerant Policy

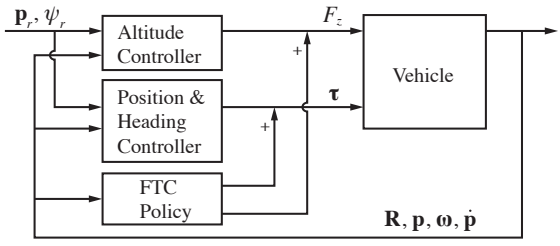

Fig. 2. The reinforcement learning based FTC control structure. The trained FTC policy is running alongside the normal baseline controller. When fault/attack occurs, it will adjust the output to maintain vehicle position.

1) Problem Setup: The fault resilient policy optimization is formulated with a standard reinforcement learning problem. The agent is the quadcopter, and the environment is the simulation world. Their interaction is a finite-horizon MDP with state space $\mathcal{S} \subset \mathbb{R}^{22}$ and action space $\mathcal{A} \subset \mathbb{R}^4$. At time t, the state $s_t \in \mathcal{S}$ and the action $a_t \in \mathcal{A}$ are

$$s_t = [R_{11}, ..., R_{33}, e_x, e_y, e_z, p, q, r, \dot{x}, \dot{y}, \dot{z}, u_1, ..., u_4]^T \quad (4)$$

$$a_t = [\Delta u_1, \Delta u_2, \Delta u_3, \Delta u_4]^T \quad (5)$$

The simulation starts at a random state $s_0 \sim p(s_0)$. Given an observation of the environment o_t, the agent takes an action a_t according to a deterministic policy $\pi_\theta(a_t|o_t)$. Given the state transition dynamics $p(s_{t+1}|s_t, a_t)$, which is the closed loop dynamics of the vehicle with sensor or actuator fault, the environment will transition to the next state s_{t+1} and generate a reward $r(s_t, a_t)$. During each rollout, the total return is the sum of the discounted reward $\mathcal{R} = \Sigma_{i=0}^{T} \gamma^{i-1} r(s_i, a_i)$ at the end of the episode at time T. The goal is to optimize the policy π_θ such that the expected return $J = \mathbb{E}_{a_i \sim \pi_\theta}[\mathcal{R}]$ is maximized.

To train the fault resilient control policy, we set the observation $o_t = s_t$. For simplicity, we let the vehicle to hover at $\mathbf{p}_0 = [0, 0, 0]^T$. The reward is

$$r_t = \left((f_{s_t} + f_{c_t})^2 + \varepsilon_f\right)^{-1} \quad (6)$$

where ε_f is a small number, f_{s_t} is the stability cost which equals zero when the vehicle is hovering stable, and f_{c_t} is the control cost for regularization. They are given by

$$f_{s_t} = \lambda_{\mathbf{p}}||\mathbf{e_p}|| + \lambda_{\mathbf{v}}||\dot{\mathbf{p}}|| + \lambda_{\mathbf{R}}||\mathbf{e_R}|| + \lambda_{\boldsymbol{\omega}}||\boldsymbol{\omega}|| \quad (7)$$

$$f_{c_t} = \lambda_a||a_t|| + \lambda_a||\dot{a}_t|| \quad (8)$$

By maximizing the reward, the policy will learn to minimize the vehicle's position tracking error even when faulty states are observed by the controller and policy.

2) Attack Injection: Attack/fault on attitude, gyro, GPS and motor signals are considered in this work. We manipulate these signals to simulate their effects. Note that the attack only affects the linear controller and the policy's observation, the reward is calculated with the true state value during training. The attacks are implemented as worse case scenario, i.e. replacing the actuator signal or the sensor value with a random number. We will show that this training method can effectively force the agent vehicle to experience the faulty condition and learn to recover from weaker attacks. At the beginning of each episode, a random choice of attack with an equal chance is selected, and a random attack value sampled from a uniform distribution is generated. The selected target sensor reading or the motor signal is replaced by this constant value during the entire length of the episode. The range of the attack values of each state during training is listed in table I. The policy will be optimized to cope with the different faulty condition regardless of the type and severity of the attack, and maintain the position of the vehicle. We include 25% rollouts with no attacks so the policy will learn to not disrupt normal operation.

TABLE I
ATTACKS AND THEIR RANGES IN TRAINING

	Motor (% throttle)	Position (m)	Attitude (deg)	Gyro (deg/s)
Minimum	70	0	30	0
Maximum	80	1.0	40	180

3) Training: Since the system dynamic is largely deterministic, we use actor-critic algorithm deep deterministic policy gradient (DDPG) for training [31]. Fully connected MLPs (multilayer perceptron) were used as function approximators. The actor network has 96×96 hidden layers

with tanh hidden and output activation. The critic network is 128×128 with tanh hidden activation and linear output activation. Both observation and output are normalized. The simulation is solving the dynamics at 2kHz. To be consistent with experimental implementation, control is down sampled to 500Hz. Each rollout has a maximum length of 2,500 samples which corresponds to 5 seconds, and each epoch has the maximum length of 10,000 samples. The algorithm implementation is based on [32], with hyperparameters from [31]. Similar to randomization techniques in [33], we inject randomness into the physical parameters such as mass/inertia , motor parameters and thrust/drag constants, etc. as well as adding noise to the sensor and actuation signals to improve the robustness of the training and transferability to the real vehicle. The learning curve is shown in Fig. 3.

IV. Evaluation and Results

We compare the effectiveness of the proposed method against two other fault-tolerant strategies. The first strategy is our previous work BlueBox, which is Active FTC [10]. The other strategy is a robust control adapted from [34], which is Passive FTC. The robust controller has built-in model compensation for actuator attack and a sliding surface variable constructed such that certain sensor corruption can be tolerated.

To evaluate the effectiveness of the three different fault/attack resilient control methods, we use the following four types of signal spoofing for each sensor and actuator.

- Total failure. The sensor or motor signal is zero.
- Biased signal. The signal supplied to the motor or the state value is being added with an additional value.
- Multiplicative. The signal is being multiplied with a number. This could be viewed as actuator loss of effectiveness or sensor resonating.
- Replacement. The signal is being replaced with a value.

For motor attack, since the vehicle we used has a relatively low lift-to-weight ratio (71% thrust needed for hovering), unlike [20], it cannot tolerate large thrust loss. Therefore motor failure (no thrust) is not tested.

We also include time-varying signals for motor attacks to test whether each method can tolerate/adapt to the varying fault condition. The signals are generated using Ornstein-Uhlenbeck process follows the stochastic equation

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t \tag{9}$$

where θ and σ are the parameters and dW_t denotes the Wiener process. For each type of attack, slow and fast changing attack signals are tested, with the attack signal varying within the same range as the corresponding fixed value attacks. The parameters used to generate the attack signals is shown in table II and a sample attack signal is shown in Fig. 4.

To simplify the comparison, we only evaluate attacks on roll angle, x axis gyro measurement and y position, since these all cause lateral deviation in y direction. The effectiveness of the fault-tolerant is measured by y position control error. The attack values of each sensor and motor attack cases and their position control errors are listed in the evaluation summary in table III. The results are averaged over 5 sample flight of 10s for each case.

A. Motor Attack

The motor signal replacement attack result is shown in Fig. 5. This is the most severe attack as the controller no longer has control over the actuator. The traditional fault-tolerant control strategy performs the worst. The FDD delay, estimator error and a low lift-to-weight ratio of the vehicle contributed to the suboptimal performance. Robust control and reinforcement learning strategy performed adequately well. The Robust controller essentially treats the overridden motor thrust as normal output plus disturbances, and tries to compensate for it. However, as the attack value deviates from its nominal value (0.71), it cannot adapt and compensate for such large disturbances fast enough, and finally leads to crashing. The reinforcement learning policy has the best performance among the three methods with minimum $x - y$ position error. Observing the flight pattern, the RL policy learned to fly at the equilibrium condition described in [20], where it spins about a near constant axis.

Bias and multiplication (loss of effectiveness) attacks/failures are the most common types of failure studied in

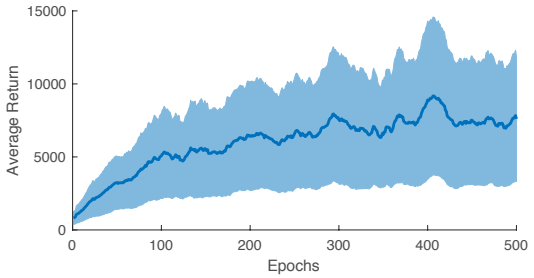

Fig. 3. Training performance of the fault-tolerant policy averaged over five runs with different random seeds. The learning curve is noisy since the vehicle is experiencing vastly different attack scenarios given the large attack space.

TABLE II

Motor time varying attacks

	Replacement		Bias		Multiplication	
	Slow	Fast	Slow	Fast	Slow	Fast
μ	0.75		0.05		1.15	
θ	5e-4	1e-2	5e-4	1e-2	5e-4	1e-2
σ	5e-4	2e-3	3.5e3	1.2e-2	1e-2	2e-2

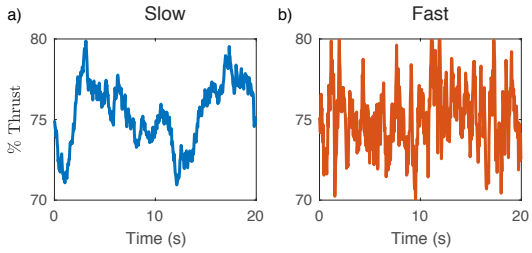

Fig. 4. Samples of motor replacement attacks signals. a) show the slow varying attack signal and b) shows the fast changing attack signal. Both signals roughly varying between 70% to 80%.

TABLE III

POSITION TRACKING ERROR UNDER ATTACK IN 10 SECONDS

Y position error (mm)			BlueBox (no redundancy)			Robust Control			Reinforcement Learning		
Attack Target	Attack Type	Attack Value	Mean	RMS	Max	Mean	RMS	Max	Mean	RMS	Max
Motor	Replacement	0.72	28.6	106.8	656.4	**2.8**	**14.8**	**45.8**	5.7	25.9	71.3
		0.75	91.5	1034.2	3032.6	17.5	82.6	405.2	**-0.1**	**20.0**	**63.5**
		0.77	785.5	1618.8	3854.5	8.0	85.6	318.7	**3.4**	**18.9**	**60.8**
		0.80	3028.6	4159.6	8474.4	454.3	1048.5	3407.6	**-4.8**	**13**	**60.2**
	Bias	-0.1	-4.5	11.4	25.8	**4.5**	**7.4**	**11.5**	29.3	68	271.7
		0.1	6.6	14.1	32.7	**5.4**	**13.9**	**20.1**	9.1	26.1	64.4
		0.2	17.2	30.4	69.5	**9.4**	**14.7**	**29.7**	17.5	28.2	71.2
	Multiplication	0.9	6.2	9.4	21.8	**2.6**	**6.1**	**15.3**	8.6	71.3	221.9
		1.2	11.7	16.6	38.8	**8.2**	**10.9**	**17.6**	10.3	23.1	67.4
		1.4	15.8	30.3	71.0	**11.2**	**13.8**	**21.3**	14.6	25.6	66.4
	Replacement	slow	1119.3	1648.7	3186.5	7.0	88.1	242.1	**5.5**	**22.5**	**72.8**
	Time varying	fast	53.1	235.9	797	14.2	83.1	268.9	**1.5**	**20.9**	**50.6**
	Bias	slow	**3.6**	**15.2**	**34.3**	9.4	64.8	106.6	-11.1	34.1	91.5
	Time varying	fast	**2.7**	**12.3**	**26.7**	22.2	35.6	61.3	-1.3	25.2	69.3
	Multiplication	slow	**18.8**	**31.2**	**111.4**	40.4	80.9	153.6	19.2	39.5	148.7
	Time varying	fast	**5.7**	**21.2**	**58.2**	24.0	34.8	54.2	14.5	26.1	72.6
Attitude	Zero	0°	278.2	1133.2	2748.1	**2.0**	**3.5**	**8.4**	-138.3	151.6	233.7
	Replacement	5°	430.0	959.5	3087.6	**3.6**	**6.2**	**11.7**	38.3	39.6	45.2
	Bias	32°	8382.9	10646	22212	31884	12148	38139	**-45.3**	**48.2**	**54.9**
	Multiplication	30	-4.1	8570.4	26065	1161.1	6459.1	12104	**-8.0**	**8.8**	**11.1**
Gyro	Zero	$0^\circ/s$	123.3	596.3	1704.5	236.2	595.0	1890.5	**-55.2**	**62.1**	**86.1**
	Replacement	$90^\circ/s$	20.2	161.3	424.7	667.1	1466.7	4064.8	**-71.5**	**92.9**	**150.7**
	Bias	$180^\circ/s$	159.5	162.9	192.3	322.4	341.6	497.7	**-121.5**	**124.7**	**139.3**
	Multiplication	10	-1.2	490.2	1515.3	**-1.5**	**15.7**	**39.7**	-179.6	205.9	398.7
Position	Zero	$0(mm)$	20.2	31.0	82.5	**20.8**	**22.9**	**33.3**	64.4	126.4	272.9
	Replacement	$500(mm)$	3.1	4248.5	7404	-4.1	622.5	1130.2	**-64.7**	**67.1**	**72.7**
	Bias	$500(mm)$	-0.8	476.6	502.8	-5.5	1363.4	1878.3	**-67.4**	**68.7**	**73.9**
	Multiplication	5	2.7	80.7	143.5	12.6	16.6	32.2	**4.1**	**12.7**	**19.9**

literature. All three methods performed well with small error. Since these attacks affect the motor thrust in a consistent way, robust control outperforms the other two methods as it can accurately adapt to the constant change in thrust.

Under time varying attacks, the RL policy has the best performance under the strongest replacement attack. For bias and multiplication attacks, robust control loses its advantage as it has to constantly update the disturbance estimation.

B. Sensor Attack

Different types of sensor attacks are evaluated and the results are shown in table III. Corresponding sample flights with RL-assisted FTC are shown in Fig. 6.

1) Attitude Attack: Without redundancy, the traditional estimation method can only detect sensor failure. Without special treatment, it cannot recover the real state value and therefore cannot defend against such attacks. This is explored in a parallel study [35]. The robust controller does not control the attitude directly. However, it still relies on attitude information to position error projections. Therefore it can tolerate subtle attacks that do not affect error projection significantly, but not sever attacks such as bias and large multiplication. The RL policy is only trained with replacement attack, however, it can tolerate all four types of attacks. The vehicle is able to hover near the target with a very small offset.

2) Gyro Attack: Since both the baseline controller and the robust controller rely heavily on the angular velocity measurement, both performances degraded severely when gyroscope measurement is tampered with. The RL policy can maintain the position mostly while only shows oscillation under multiplication attack, which is mostly due to the large output from the baseline controller.

3) Position Attack: With no position feedback, no controller can accurately maintain the vehicle's position. Even using techniques such as dead reckoning, longterm position drift is inevitable. However with single-axis measurement being attacked, y position in our case, other state variables could be used to estimate the y position. The RL policy can leverage the attitude and acceleration measurement to minimize the drift, while traditional controller cannot.

C. Real Vehicle Verification

The reinforcement learning FTC policy is implemented on the real vehicle to demonstrate successful sim-to-real transfer. To avoid damaging the vehicle in our testing environment with limited space, only mild attack cases were tested, as stronger attacks could result in drastic movements. Motor replacement attack with fixed 73% thrust is shown in Fig. 7a) and gyro replacement attack with $p = 90^\circ/s$ is shown in Fig. 7b). The experimental evaluation exhibits similar behavior as the simulation result, further validating the fidelity of the simulation and the effectiveness of the training.

D. Discussion

Motor FDD and remapping cannot deal with strong motor attacks where the motor signal is being replaced, especially when the vehicle is carrying payloads and has a low lift-to-weight ratio. The performance degrades severely when the attack value deviates from its normal operating point. With no redundant sensor, special treatment is needed to isolate sensor fault and subsequently recover. Generally,

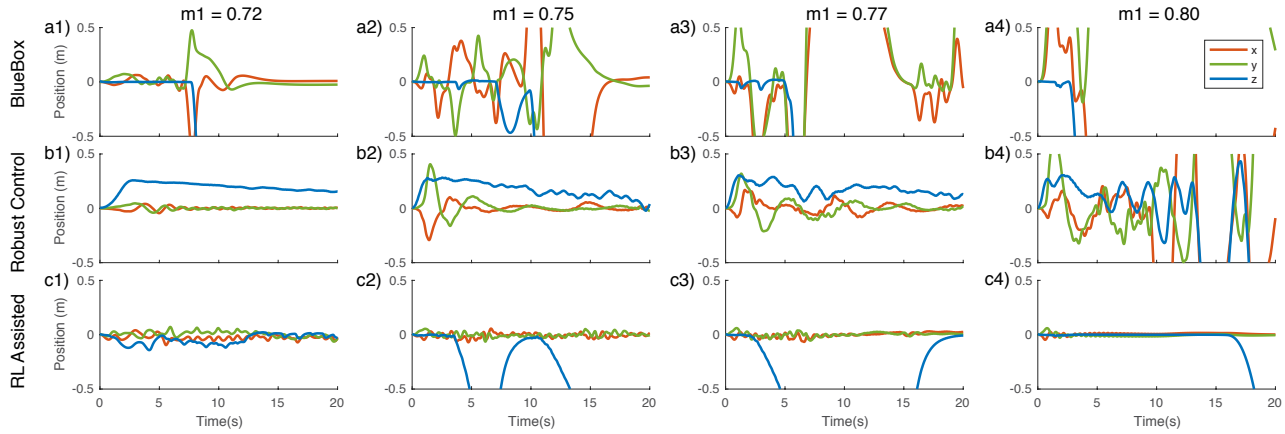

Fig. 5. The position tracking control of the quadcopter under actuator attack is shown. All samples above show the position of the vehicle while motor 4 is under signal replacement attack. Figure a) are the control result with BlueBox, figure b) are the results using robust control, and figure c) are the results using the RL-assisted FTC. Figure 1), 2), 3) and 4) are showing attacks with different intensities.

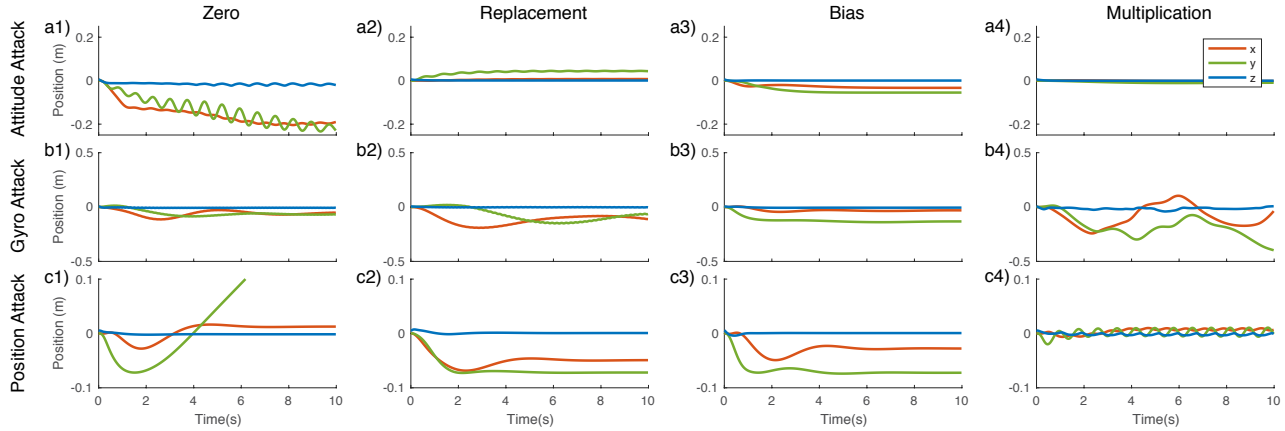

Fig. 6. The position tracking control of the quadcopter under different types of sensor attacks with RL-assisted FTC. Figure a), b) and c) corresponds to attitude, gyro and position attacks and 1), 2), 3) and 4) are different types of attacks.

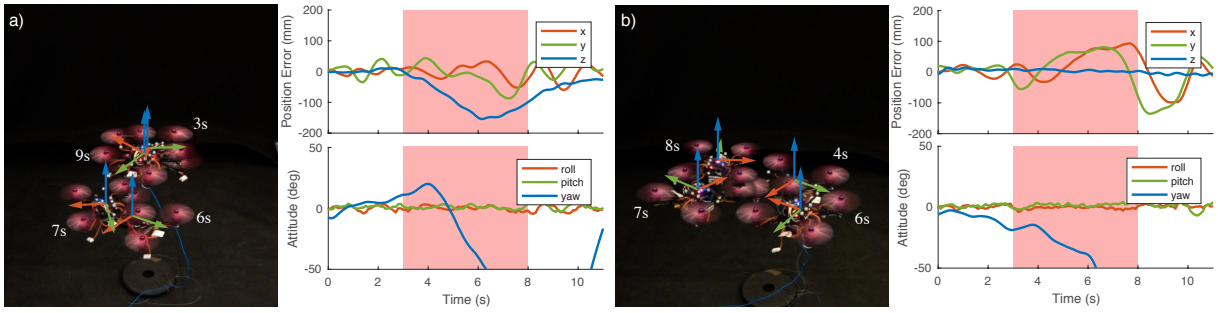

Fig. 7. a) Motor attack on the test vehicle. The attack initiates at 3s and ends at 8s. The altitude will drop and the yaw angle will lose control under such attack, however the lateral position control is stable. b) Gyro attack on the test vehicle. Similar to that observed in simulation, the position control is drifting, when the attack is in effect. Attitude stability is maintained.

robust control can tolerate most motor attacks. By mixing all state variables in the sliding surface variable, the controller can tolerate some sensor attacks without crashing. The reinforcement learning policy assisted FTC can tolerate both motor and sensor fault.

V. Conclusions

In this work, we presented a novel model-free reinforcement learning-based hybrid fault-tolerant strategies for UAVs to recover flight under cyber-physical attacks. This is realized by training an MLP policy that provides extra control efforts on top of the existing stabilizing controller while injecting attacks. By selecting the position tracking error as cost, the policy learned to maintain position control regardless of whether an attack is present or not, or the type of attack it encounters. We evaluate the effectiveness of this method against two tradition strategies: fault detection and robust control. The result shows that, without explicit modeling, the trained policy is able to decide when and how to provide appropriate action to counter the attack/fault. Among all the attacks cases, the RL strategy survives the most cases and has the least position error.

REFERENCES

[1] C. H. Kim, T. Kim, H. Choi, Z. Gu, B. Lee, X. Zhang, and D. Xu, "Securing real-time microcontroller systems through customized memory view switching," in *Network and Distributed Systems Security Symp.(NDSS)*, 2018.

[2] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004, pp. 298–307.

[3] J. S. Warner and R. G. Johnston, "A simple demonstration that the global positioning system (gps) is vulnerable to spoofing," *Journal of Security Administration*, vol. 25, no. 2, pp. 19–27, 2002.

[4] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, and P. M. Kintner, "Assessing the spoofing threat: Development of a portable gps civilian spoofer," in *Radionavigation Laboratory Conference Proceedings*, 2008.

[5] D. P. Shepard, J. A. Bhatti, T. E. Humphreys, and A. A. Fansler, "Evaluation of smart grid and civilian uav vulnerability to gps spoofing attacks," in *Proceedings of the ION GNSS Meeting*, vol. 3, 2012, pp. 3591–3605.

[6] Y. M. Son, H. C. Shin, D. K. Kim, Y. S. Park, J. H. Noh, K. B. Choi, J. W. Choi, and Y. D. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *24th USENIX Security symposium*. USENIX Association, 2015.

[7] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in *Homeland Security (HST), 2012 IEEE Conference on Technologies for*. IEEE, 2012, pp. 585–590.

[8] N. M. Rodday, R. d. O. Schmidt, and A. Pras, "Exploring security vulnerabilities of unmanned aerial vehicles," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 993–994.

[9] A. Kim, B. Wampler, J. Goppert, I. Hwang, and H. Aldridge, "Cyber attack vulnerabilities analysis for unmanned aerial vehicles." in *Infotech@ Aerospace*, 2012, pp. 1–30.

[10] F. Fei, Z. Tu, R. Yu, T. Kim, X. Zhang, D. Xu, and X. Deng, "Cross-layer retrofitting of uavs against cyber-physical attacks," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*. IEEE, 2018, pp. 550–557.

[11] "Arduplane, arducopter, ardurover source," Aug 2019. [Online]. Available: https://github.com/ArduPilot/ardupilot/tree/Copter-3.3.2

[12] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual reviews in control*, vol. 32, no. 2, pp. 229–252, 2008.

[13] H. Aguilar-Sierra, G. Flores, S. Salazar, and R. Lozano, "Fault estimation for a quad-rotor mav using a polynomial observer," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 455–468, 2014.

[14] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja, "Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5266–5271.

[15] S. Mallavalli and A. Fekih, "A fault tolerant control approach for a quadrotor uav subject to time varying disturbances and actuator faults," in *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE, 2017, pp. 596–601.

[16] I. Sadeghzadeh, A. Mehta, A. Chamseddine, and Y. Zhang, "Active fault tolerant control of a quadrotor uav based on gainscheduled pid control," in *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*. IEEE, 2012, pp. 1–4.

[17] G. Heredia, A. Ollero, M. Bejar, and R. Mahtani, "Sensor and actuator fault detection in small autonomous helicopters," *Mechatronics*, vol. 18, no. 2, pp. 90–99, 2008.

[18] F. Sharifi, M. Mirzaei, B. W. Gordon, and Y. Zhang, "Fault tolerant control of a quadrotor uav using sliding mode control," in *Control and Fault-Tolerant Systems (SysTol), 2010 Conference on*. IEEE, 2010, pp. 239–244.

[19] A. Milhim, Y. Zhang, and C.-A. Rabbath, "Gain scheduling based pid controller for fault tolerant control of quad-rotor uav," in *AIAA infotech@ aerospace 2010*, 2010, p. 3530.

[20] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 45–52.

[21] F. Farivar and M. N. Ahmadabadi, "Continuous reinforcement learning to robust fault tolerant control for a class of unknown nonlinear systems," *Applied Soft Computing*, vol. 37, pp. 702–714, 2015.

[22] Z. Wang, L. Liu, H. Zhang, and G. Xiao, "Fault-tolerant controller design for a class of nonlinear mimo discrete-time systems via online reinforcement learning algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 5, pp. 611–622, 2015.

[23] L. Liu, Z. Wang, and H. Zhang, "Adaptive fault-tolerant tracking control for mimo discrete-time systems via reinforcement learning algorithm with less learning parameters," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 299–313, 2016.

[24] S. R. Ahmadzadeh, P. Kormushev, and D. G. Caldwell, "Multi-objective reinforcement learning for auv thruster failure recovery," in *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2014, pp. 1–8.

[25] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[26] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, p. 22, 2019.

[27] N. O. Lambert, D. S. Drew, J. Yaconelli, R. Calandra, S. Levine, and K. S. Pister, "Low level control of a quadrotor with deep model-based reinforcement learning," *arXiv preprint arXiv:1901.03737*, 2019.

[28] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," *arXiv preprint arXiv:1903.04628*, 2019.

[29] M. Srouji, J. Zhang, and R. Salakhutdinov, "Structured control nets for deep reinforcement learning," in *International Conference on Machine Learning*, 2018, pp. 4749–4758.

[30] F. Fei, Z. Tu, Y. Yang, J. Zhang, and X. Deng, "Flappy hummingbird: An open source dynamic simulation of flapping wing robots and animals," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9223–9229.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[32] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[33] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[34] F. Fei, Z. Tu, J. Zhang, and X. Deng, "Learning extreme hummingbird maneuvers on flapping wing robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 109–115.

[35] Z. Tu, F. Fei, M. Eagon, D. Xu, and X. Deng, "Flight recovery of mavs with compromised imu," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.