PID-Piper: Recovering Robotic Vehicles from Physical Attacks
NSE ML+Security Reading Group

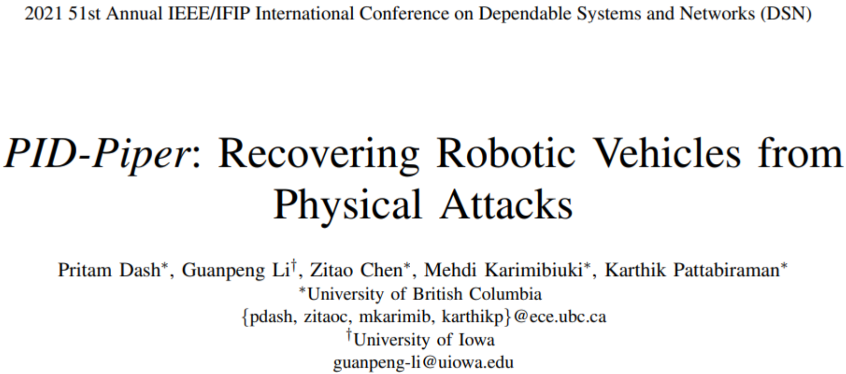
Kim Hammar

kimham@kth.se

Division of Network and Systems Engineering
KTH Royal Institute of Technology

April 19, 2024

Context of the Paper



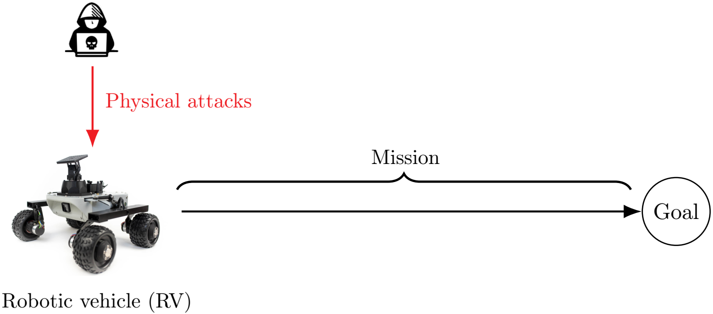
2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)

PID-Piper: Recovering Robotic Vehicles from Physical Attacks

Pritam Dash[*], Guanpeng Li[†], Zitao Chen[*], Mehdi Karimibiuki[*], Karthik Pattabiraman[*]
[*]University of British Columbia
{pdash, zitaoc, mkarimib, karthikp}@ece.ubc.ca
[†]University of Iowa
guanpeng-li@uiowa.edu

Published in DSN'21. **Best paper awardee** ($<$ 0.003% out of 350 submissions.)

Context of the Paper



Physical attacks

Mission

Goal

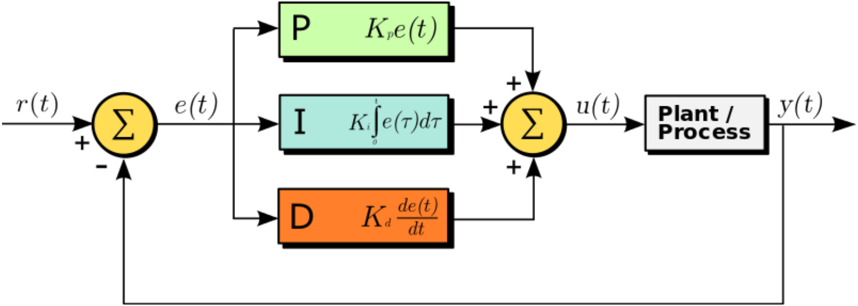
Robotic vehicle (RV)

- PID-PIPER is a framework to automatically **recover RVs from physical attacks.**
- Allows RV's to complete their missions despite attacks.

Demo

https://bit.ly/3oswuTc

Problem Setup

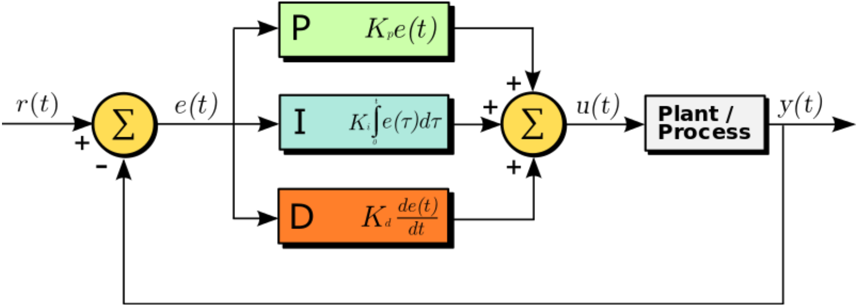


- The RV's are controlled using PID control:

$$u(t) = K\left(pe(t) + i\int_0^t e(t)\mathrm{d}t + d\frac{de(t)}{dt}\right)$$

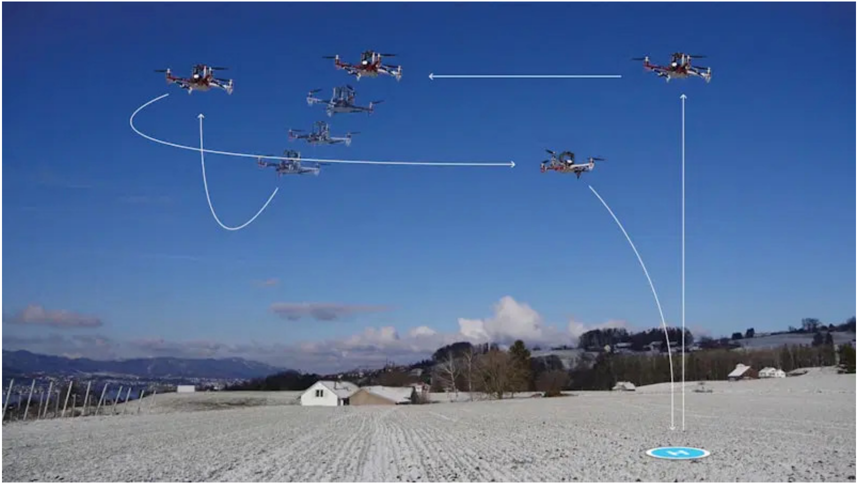
$u(t)$ is the control signal (e.g., motor commands), $y(t)$ is the system output, w is noise, $r(t)$ is the target state, $e(t) = r(t) - y(t)$ is the error.

Threat Model



- An adversary can manipulate **some** of the sensor measurements $y(t)$, e.g., spoof GPS or gyroscope.
- The PID controller is designed to handle i.i.d zero-mean noise
- By systematically manipulating the sensors, the adversary can **cause the controller to destabilize and fail its mission.**

Current Solutions



- Detect the attack and enter fail-safe mode (e.g., force landing).
    - **Limitation:** Fail-safe means that the mission fails.

Proposed Solution: PID-PIPER

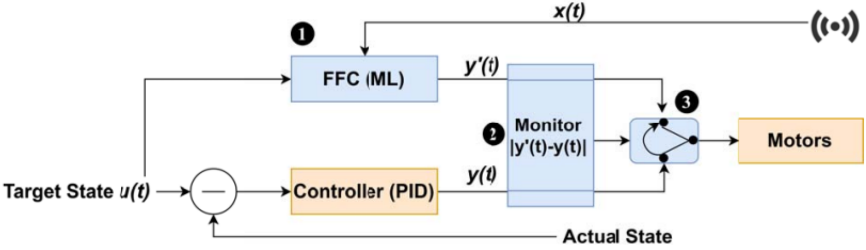
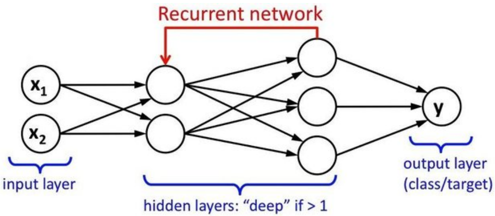


Fig. 4: *PID-Piper* architecture.

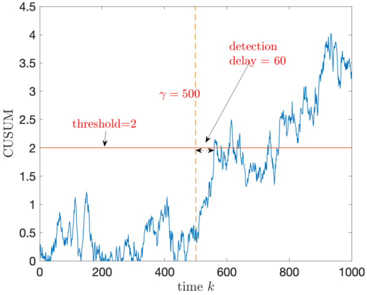
- PIP Piper uses **two controllers**: (1) a PID controller for normal operation; and (2) an ML controller for recovery
- When an attack is detected the control switches to the ML controller.
- **Remark:** the ML controller is feed-forward rather than feed-back.

Training the ML Controller



- The ML controller **takes as input**: the current state $x(t)$ (possibly manipulated measurements) and the target state $u(t)$
- Predicts what the PID controller would do if there was no attack $y'(t)$
- The controller is a **2-layer LSTM**
- Trained offline based on data from normal missions (no attacks).

Deciding When to Switch Controllers



- **Quickest change detection** problem: switch controllers when the deviation between the PID controller and the ML controller becomes large.
  - *Specific type of stopping problem where the change point is geometrically distributed, distributions before and after change are i.i.d, and reward is defined using Lorden's formulation (minimized detection delay subject to a false alarm constraint.)*
- Approach: the CUSUM algorithm, i.e., switch controllers when the cusum statistic exceeds a fixed threshold τ.

Evaluation

(a) Pixhawk Drone (b) Aion R1 Rover (c) Sky-viper Drone

Fig. 5: Real RV Systems used for Experiments.

- To evaluate PID-piper, they define missions for the RV's and **emulate physical attacks with software programs.**
- A mission is successful if the total deviation from the target destination is less than 10m.
- Baselines: CI, Savior, and SRR. Feeback-recovery approaches proposed in prior work. CI/Savior use fail-safe modes. SRR tries to recover.

Evaluation - Recovery from Overt Attacks

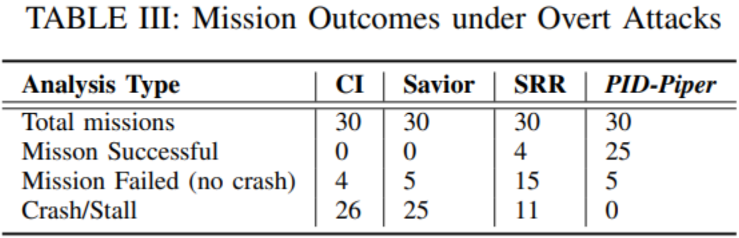


TABLE III: Mission Outcomes under Overt Attacks

Analysis Type	**CI**	**Savior**	**SRR**	***PID-Piper***
Total missions	30	30	30	30
Misson Successful	0	0	4	25
Mission Failed (no crash)	4	5	15	5
Crash/Stall	26	25	11	0

- PID-PIPER **achieves 83% successful mission completion.**
- Baselines achieve 0% and 13%
- Overt attacks are non-stealthy attacks that aim significantly manipulate the sensors to try to make the RV crash suddenly.

Evaluation - Recovery from False Positives

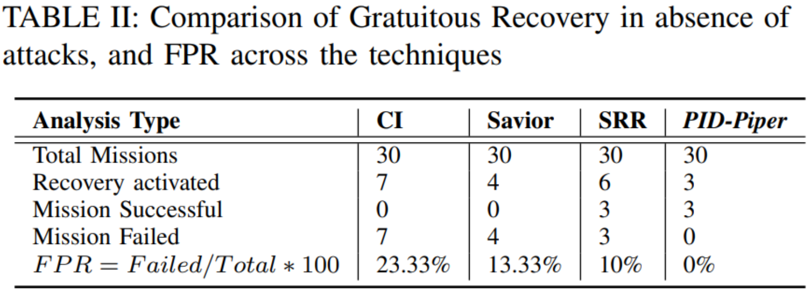
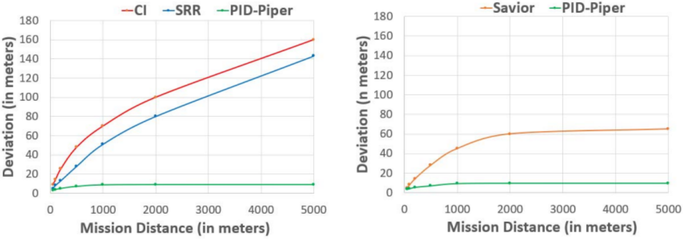


TABLE II: Comparison of Gratuitous Recovery in absence of attacks, and FPR across the techniques

Analysis Type	**CI**	**Savior**	**SRR**	***PID-Piper***
Total Missions	30	30	30	30
Recovery activated	7	4	6	3
Mission Successful	0	0	3	3
Mission Failed	7	4	3	0
$FPR = Failed/Total * 100$	23.33%	13.33%	10%	0%

- PID-PIPER **have 0% failures when no attacks occur.**
- Baselines have 23.33%, 13.33% and 10% failures.

Evaluation - Robustness against Stealthy Attacks



(a) (b)

Fig. 9: Deviation due to stealthy attacks. (a) Comparison between *PID-Piper*, SRR, and CI on ArduCopter. (b) Comparison between *PID-Piper* and Savior on PX4 Solo.

- PID-PIPER **limits deviation to less than 10m, even for missions up to 5km length.**
- Baselines deviate significantly.
- Stealthy attacks are attacks that make small perturbations to the sensor inputs and which does not trigger the recovery threshold.

Discussion - Why does PID-PIPER Beat the Baselines?

- PID-PIPER uses a smaller set of features (selected through **feature engineering**) which makes it less vulnerable to sensor perturbations. Shown to improve the performance significantly.
- By using a deep learning controller, they are able to predict control outputs accurately, which means that the recovery threshold can be low $\implies$ more robust against stealthy attacks.

Discussion - Feedback vs Feedforward

- An evaluation in the paper shows that feed-forward control is more robust against attacks than feed-back control.
  - Feed-forward control avoids the overcompensation issue.

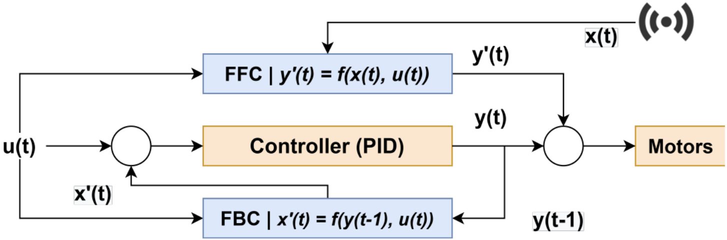


Fig. 3: FBC and FFC controller design

Discussion - Limitations

- Assumes that the RV is segmented such that even if an attacker controls some sensors, it cannot access the recovery module and the firmware.
- Only considers attacks where the **attacker can target one sensor at a time**, i.e., cannot control all sensors simultaneously.
- PID-PIPER is vulnerable to adversarial attacks: what if the attacker has access to the ML controller logic?
  - Attacker can perturb the sensor measurements to fool the ML-controller.
  - Making the controller robust to such attacks may involve adversarial training and game-theoretic analysis.

Conclusions

- This paper PID-PIPER: a framework for **automated recovery of robotic vehicles from physical attacks**
- Uses a separate ML-controller that is invoked whenever an attack is detected
- Achieves state-of-the-art results on 3 real RV's and 3 simulated RV's