Analyzing and Improving Fault Tolerance of Autonomous Navigation Systems

- From the Perspective of Hardware Faults

Zishen Wan

PhD Student, Department of ECE, Georgia Institute of Technology

zishenwan@gatech.edu

Acknowledge: Arijit Raychowdhury, Aqeel Anwar (Georgia Tech), Tianyu Jia (CMU), Yu-Shun Hsiao, Gu-Yeon Wei, David Brooks, Vijay Janapa Reddi (Harvard)

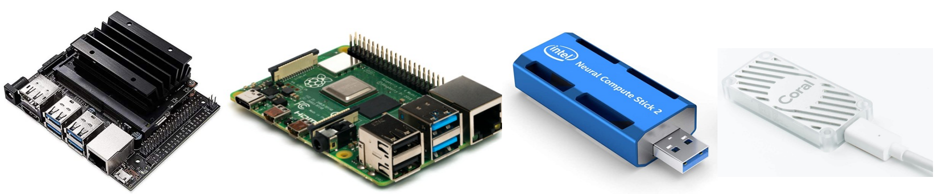
SRC® JUMP

C-BRIC

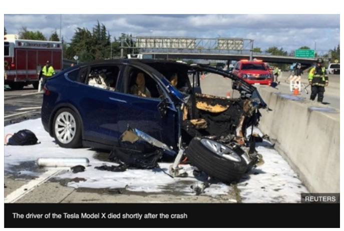
Safety of Autonomous Navigation

- Autonomous navigation systems are widely used.
- Specialized hardware accelerator is rising.
- Hardware Fault is increasing.
  - Transient fault
  - Permanent fault
- Traditional protection method incurs large overhead.
  - Hardware module redundancy

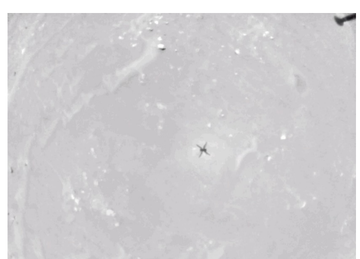
Safety of Autonomous Navigation



- Autonomous navigation systems are widely used.
- Specialized hardware accelerator is rising.
- Hardware Fault is increasing.
  - Transient fault
  - Permanent fault
- Traditional protection method incurs large overhead.
  - Hardware module redundancy



Tesla Autopilot



NASA Mars helicopter

C-BRIC

Safety of Autonomous Navigation

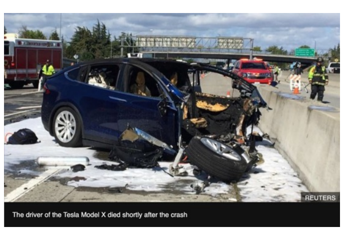
➢ Autonomous navigation systems are widely used.

How is the resilience of autonomous navigation system to hardware faults?

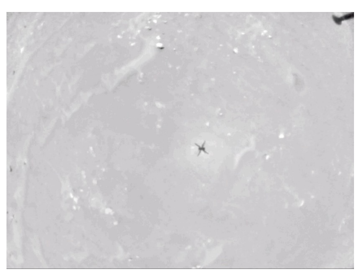
How do we detect and mitigate hardware faults?

➢ Traditional protection method incurs large overhead.

- Hardware module redundancy



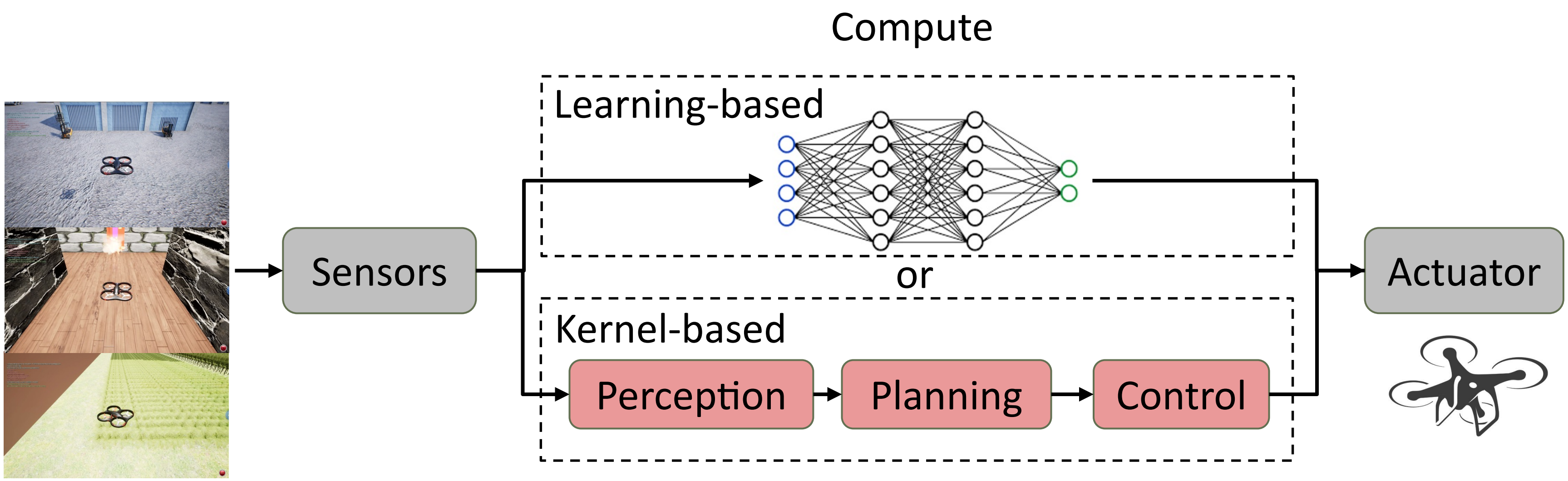
Tesla Autopilot



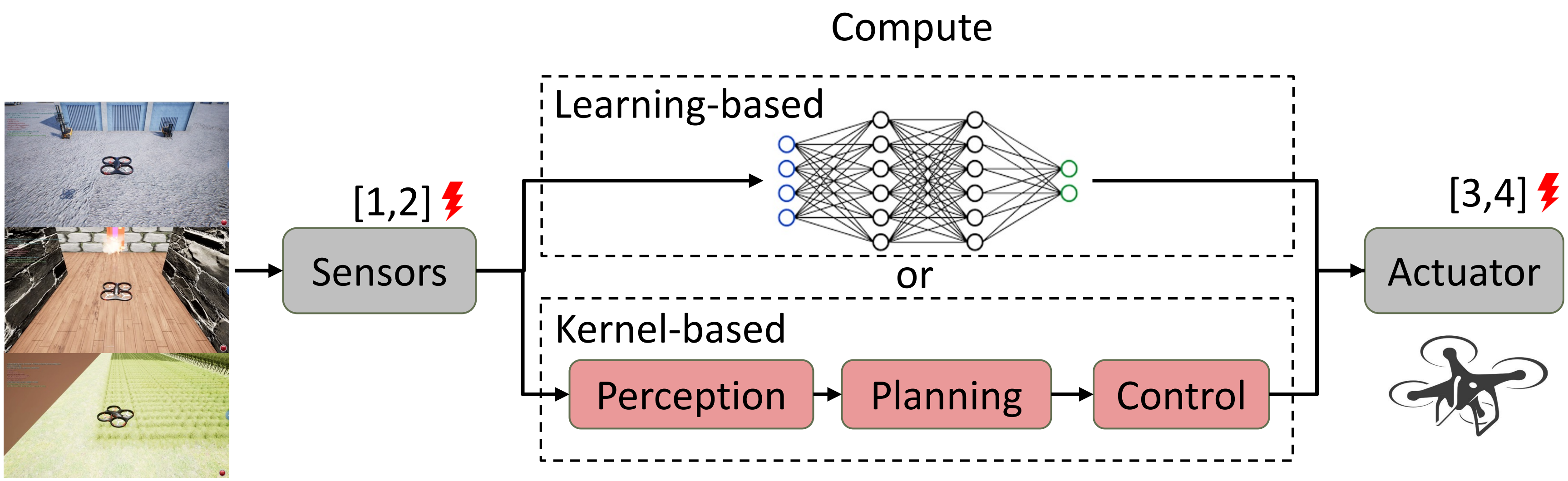
NASA Mars helicopter

C-BRIC

Autonomous Navigation System Paradigm

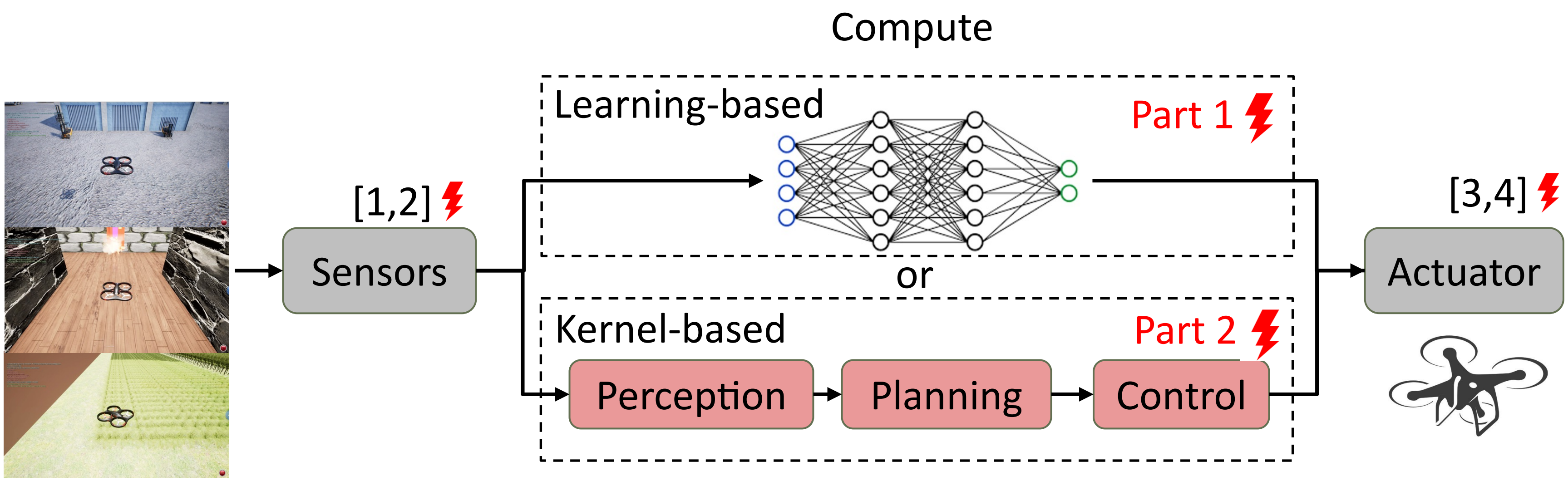


Autonomous Navigation System Paradigm



[1] A. Toschi, et al, “Characterizing perception module performance and robustness in production-scale autonomous driving system,” NPC, 2019.
[2] Y.Cao, et al, “Adversarial objects against lidar-based autonomous driving systems,” arXiv, 2019.
[3] J. A. Guzmán-Rabasa, et al, “Actuator fault detection and isolation on a quadrotor unmanned aerial vehicle modeled as a linear parameter-varying system,” Measurement and Control, 2019.
[4] X.Qi, et al, “Self-healing control framework against actuator fault of single-rotor unmanned helicopters,” Recent Progress in Aircraft Technologies, 2016.

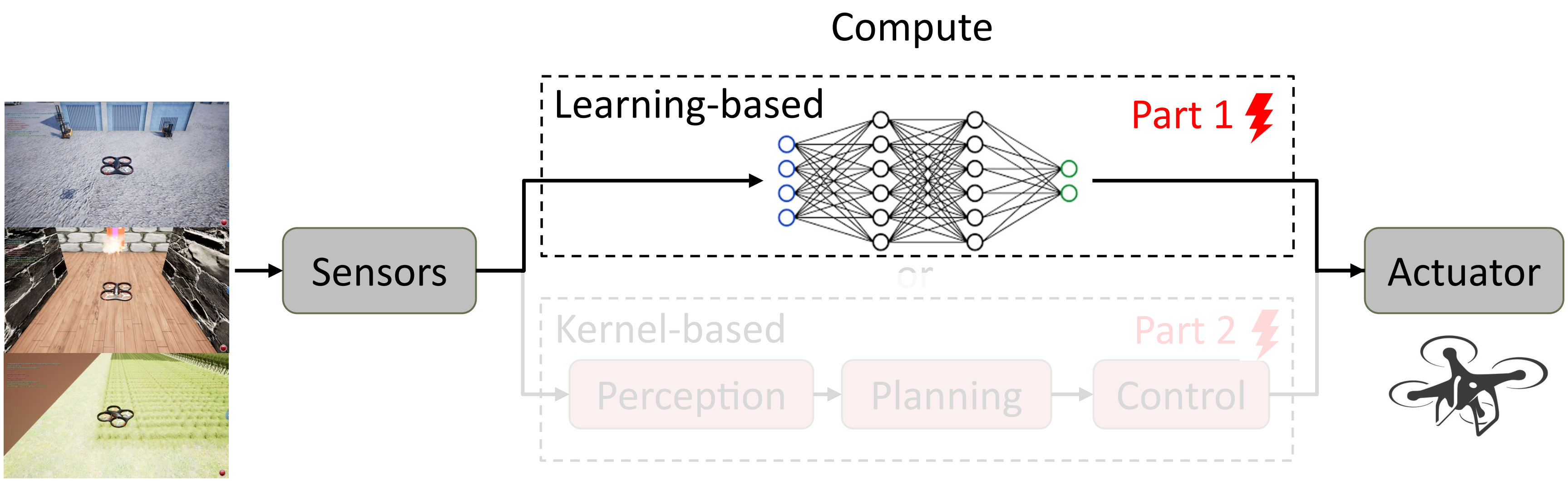
Autonomous Navigation System Paradigm



- Part1: Reliability of learning-based navigation pipeline
- Part2: Reliability of kernel-based navigation pipeline

[1] A. Toschi, et al, "Characterizing perception module performance and robustness in production-scale autonomous driving system," NPC, 2019.
[2] Y.Cao, et al, "Adversarial objects against lidar-based autonomous driving systems," arXiv, 2019.
[3] J. A. Guzmán-Rabasa, et al, "Actuator fault detection and isolation on a quadrotor unmanned aerial vehicle modeled as a linear parameter-varying system," Measurement and Control, 2019.
[4] X.Qi, et al, "Self-healing control framework against actuator fault of single-rotor unmanned helicopters," Recent Progress in Aircraft Technologies, 2016.

Autonomous Navigation System Paradigm



- Part1: Reliability of learning-based navigation system
- Part2: Reliability of kernel-based navigation system

[1] A. Toschi, et al, "Characterizing perception module performance and robustness in production-scale autonomous driving system," NPC, 2019.
[2] Y.Cao, et al, "Adversarial objects against lidar-based autonomous driving systems," arXiv, 2019.
[3] J. A. Guzmán-Rabasa, et al, "Actuator fault detection and isolation on a quadrotor unmanned aerial vehicle modeled as a linear parameter-varying system," Measurement and Control, 2019.
[4] X.Qi, et al, "Self-healing control framework against actuator fault of single-rotor unmanned helicopters," Recent Progress in Aircraft Technologies, 2016.

Related Work

- Fault characterization
  - Neural network in supervised learning: PytorchFI [1], Ares [2], SC'17 [3]
  - End-to-end reinforcement learning-based (Our)
- Fault mitigation
  - Hardware redundancy-based method: DMR, TMR
  - Application-aware method (Our)

[1] Mahmoud, A. et al. *Pytorchfi: A Runtime Perturbation Tool for DNNs*. In DSN, 2020.
[2] Reagen, B. et al. *Ares: A framework for quantifying the resilience of deep neural networks.* In DAC, 2018.
[3] Li, G. et al. *Understanding error propagation in deep learning neural network (DNN) accelerators and applications.* In SC, 2017.

This work

Analyzing and Improving fault tolerance of learning-based navigation systems, that is:

- A fault injection tool-chain for learning-based systems
- Hardware fault study in learning-based systems
- Fault mitigation techniques for learning-based systems

This work

Analyzing and Improving fault tolerance of learning-based navigation systems, that is:

- A fault injection tool-chain for learning-based systems
- Hardware fault study in learning-based systems
- Fault mitigation techniques for learning-based systems

Fault Model and Fault Injection

- Fault Type
  - Transient fault
    - Random bitflip
  - Permanent fault
    - Stuck-at-0
    - Stuck-at-1

Fault Model and Fault Injection

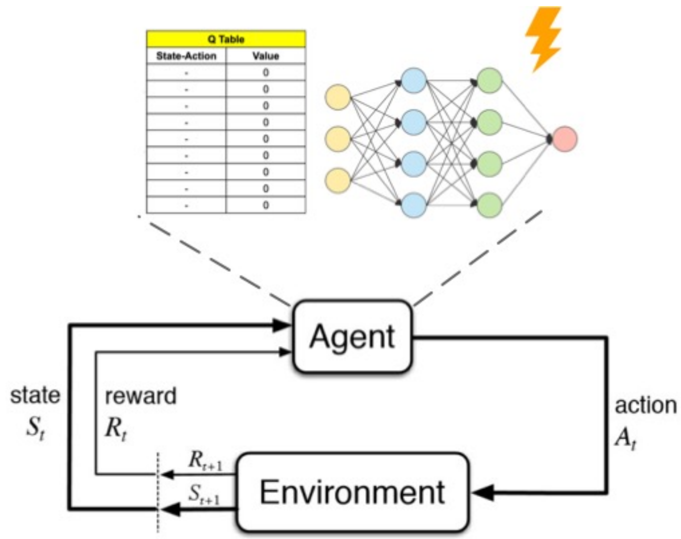
- Fault Type
  - Transient fault
    - Random bitflip
  - Permanent fault
    - Stuck-at-0
    - Stuck-at-1
- Fault Location
  - Memory [1,2]

[1] Reagen, B. et al. *Ares: A framework for quantifying the resilience of deep neural networks.* In DAC, 2018.
[2] Li, G. et al. *Understanding error propagation in deep learning neural network (DNN) accelerators and applications.* In SC, 2017.

Fault Model and Fault Injection

- Fault Type
  - Transient fault
    - Random bitflip
  - Permanent fault
    - Stuck-at-0
    - Stuck-at-1
- Fault Location
  - Memory [1,2]

- Fault Injection
  - Methodology
    - Static injection
    - Dynamic injection
  - Phases
    - Training
    - Inference



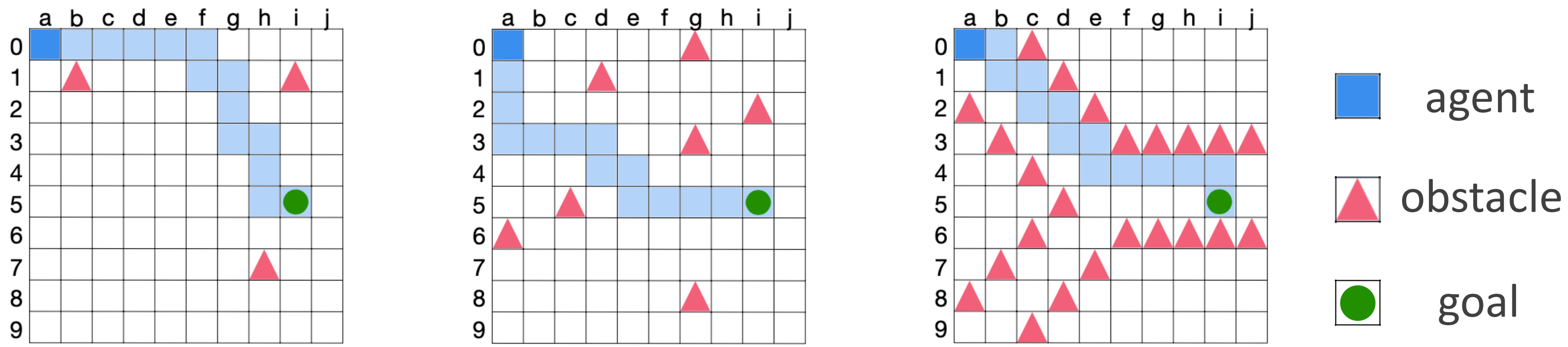
[1] Reagen, B. et al. *Ares: A framework for quantifying the resilience of deep neural networks.* In DAC, 2018.
[2] Li, G. et al. *Understanding error propagation in deep learning neural network (DNN) accelerators and applications.* In SC, 2017.

This work

Analyzing and Improving fault tolerance of learning-based navigation systems, that is:

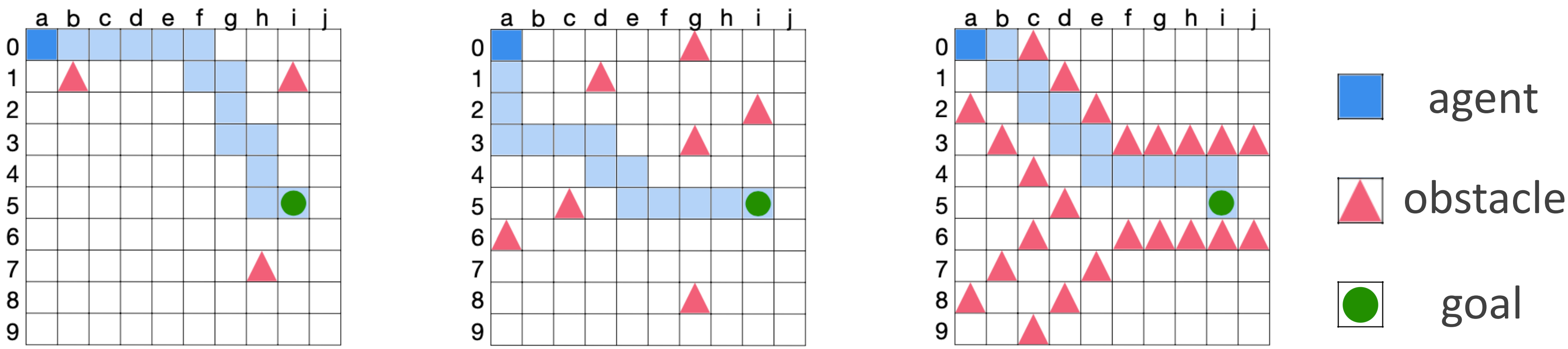
- A fault injection tool-chain for learning-based systems
- Hardware fault study in learning-based systems
- Fault mitigation techniques for learning-based systems

Grid-Based Navigation Problem



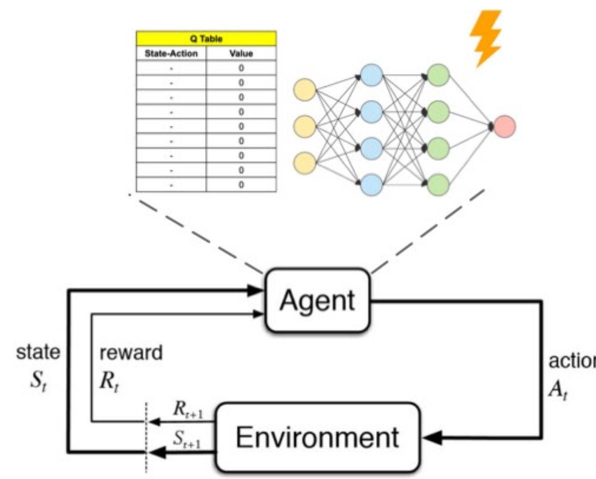
Low obstacle density   Middle obstacle density   High obstacle density

Grid-Based Navigation Problem



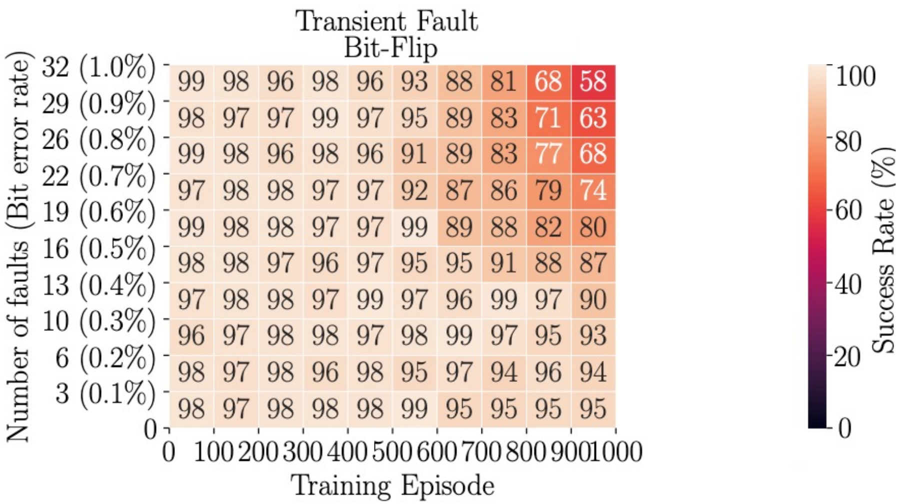
Low obstacle density | Middle obstacle density | High obstacle density

- Algorithm paradigm:
  - NN-based method
  - Tabular-based method
- Evaluation metric: agent’s success rate



Faults in Grid World (Training)

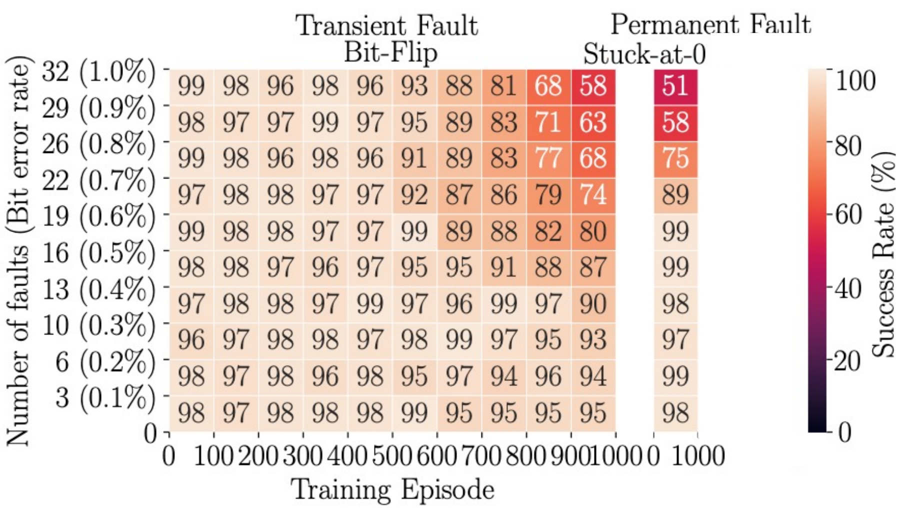
NN-based method:



➢ Transient fault occurred in later episodes with high BER has higher impact.

Faults in Grid World (Training)

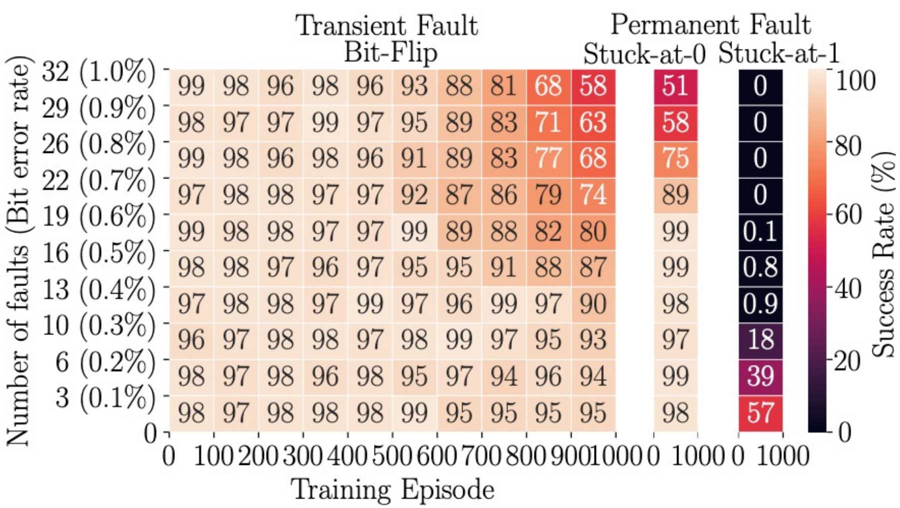
NN-based method:



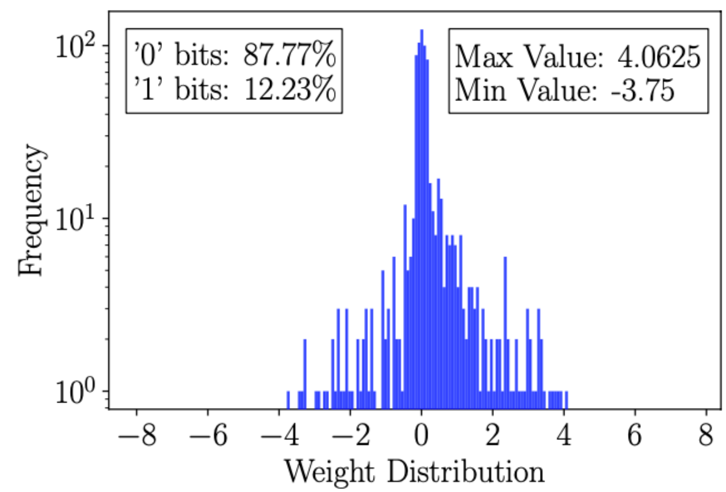
➤ Permanent fault stuck-at-0 has comparable impact as transient fault.

Faults in Grid World (Training)

NN-based method:



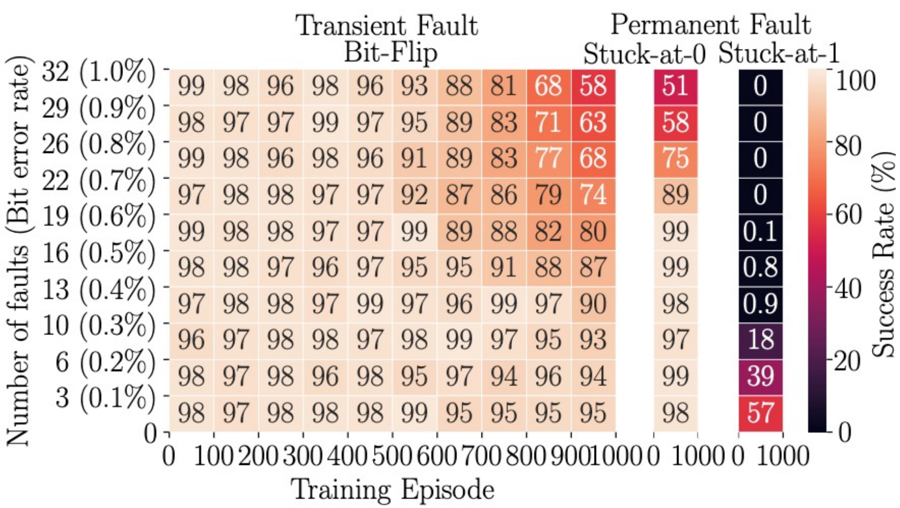
NN-based policy weight distribution:



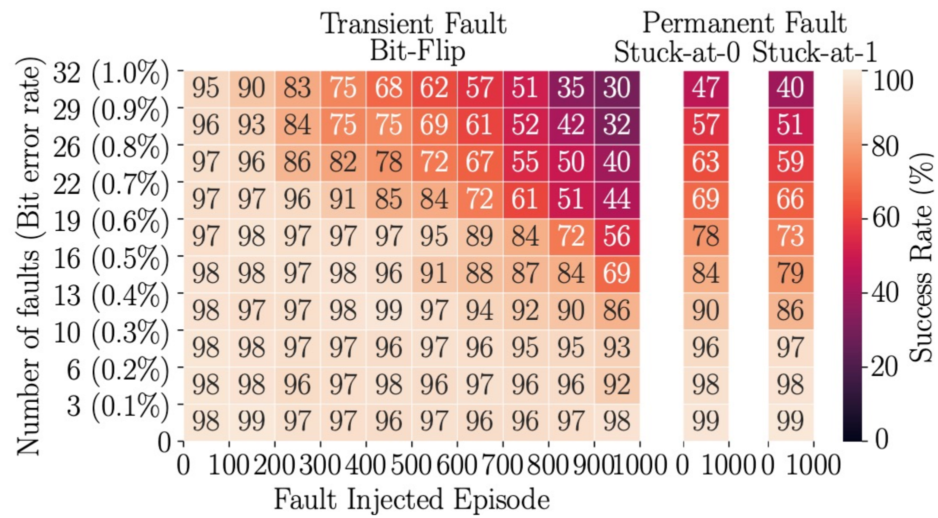
- ➢ Permanent fault stuck-at-1 has much severer impact than stuck-at-0.

Faults in Grid World (Training)

NN-based method:

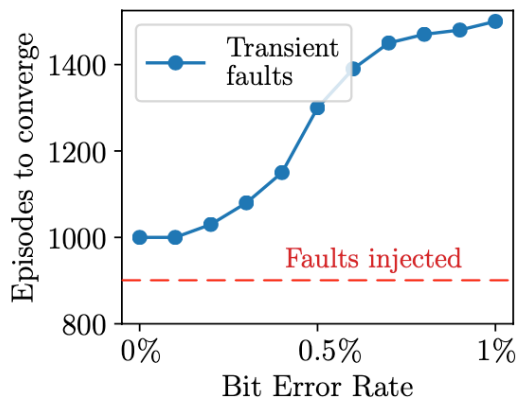


Tabular-based method:



- NN-based policy exhibit higher resilience than Tabular-based policy (except stuck-at-1).

Faults in Grid World (Convergence)

	NN-based method
Transient fault	

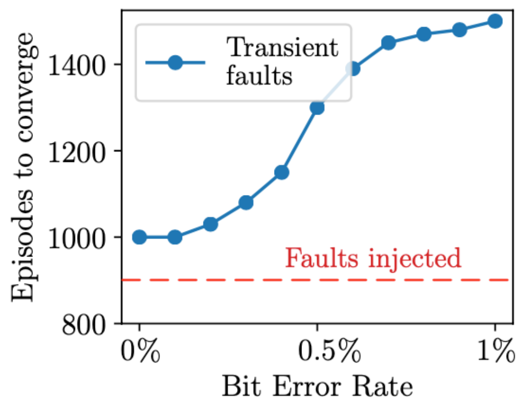
- System can finally achieve convergence (>95% success rate) after transient faults injected.

C-BRIC

Faults in Grid World (Convergence)

NN-based method

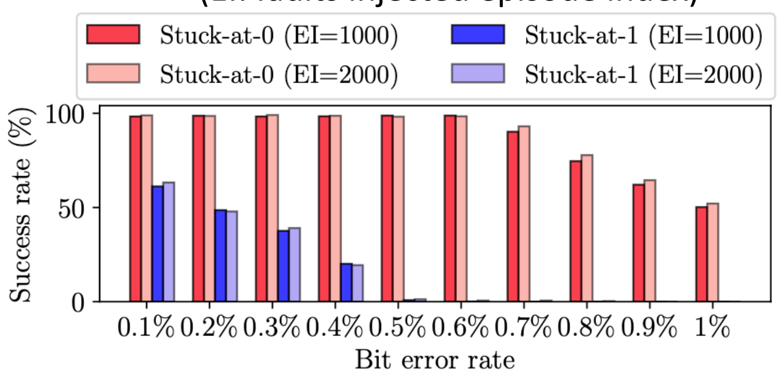
Transient fault



- System can finally achieve convergence (>95% success rate) after transient faults injected.

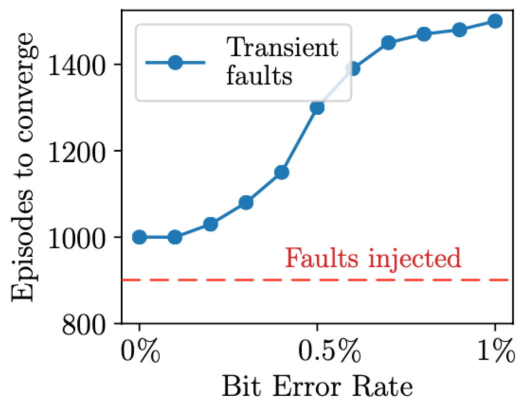
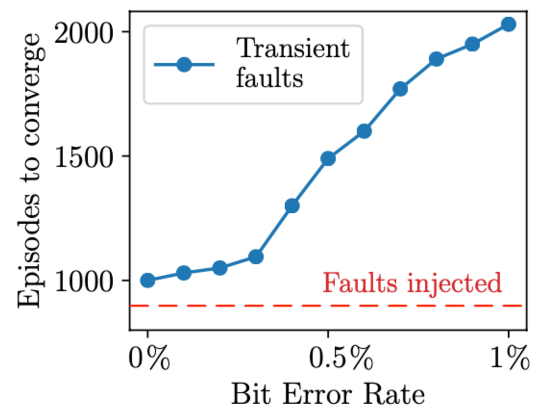
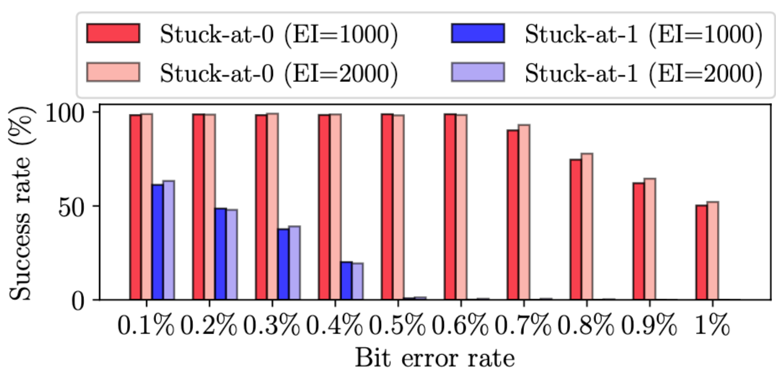
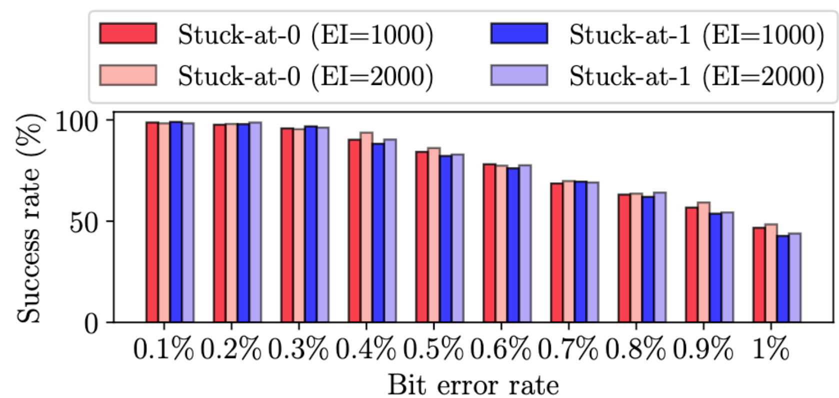
Permanent fault

(EI: faults injected episode index)



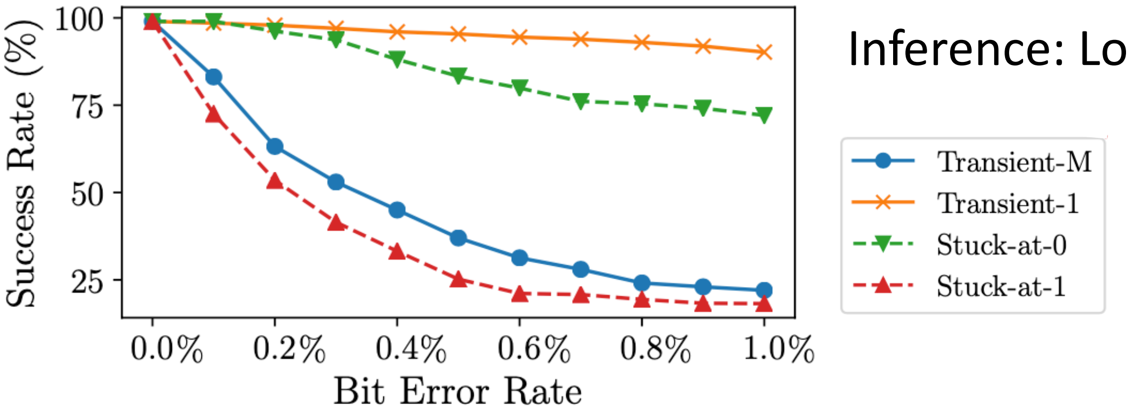
- Extra training time doesn't bring obvious improvements under permanent faults.

Faults in Grid World (Convergence)

	NN-based method	Tabular-based method
Transient fault		
Permanent fault		

Faults in Grid World (Inference)

NN-based method:



Inference: Long-term decision-making process

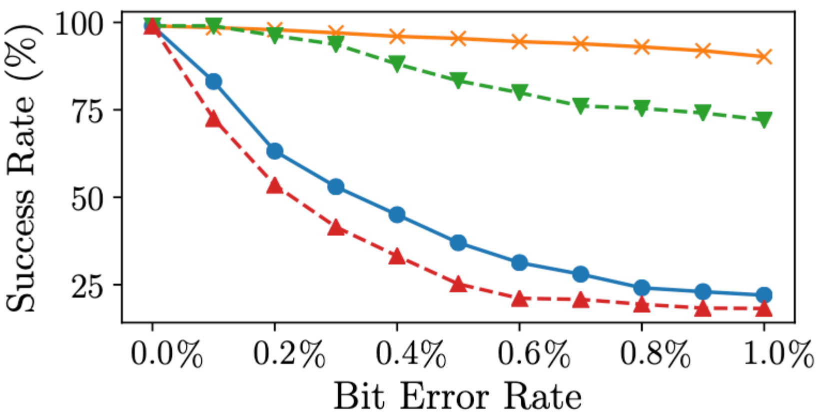
Transient-M: impact all steps

Transient-1: impact single step

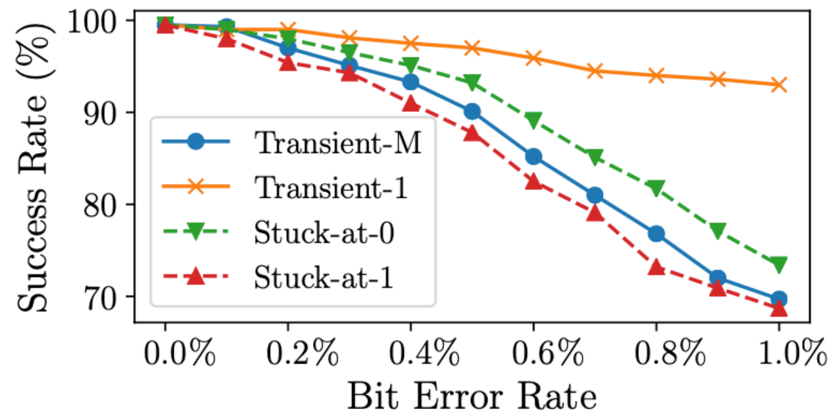
- Transient fault: Transient-1 has a negligible effect compared to Transient-M.
- Permanent fault: Stuck-at-1 has a much severe impact on policy than Stuck- at-0

Faults in Grid World (Inference)

NN-based method:



Tabular-based method:



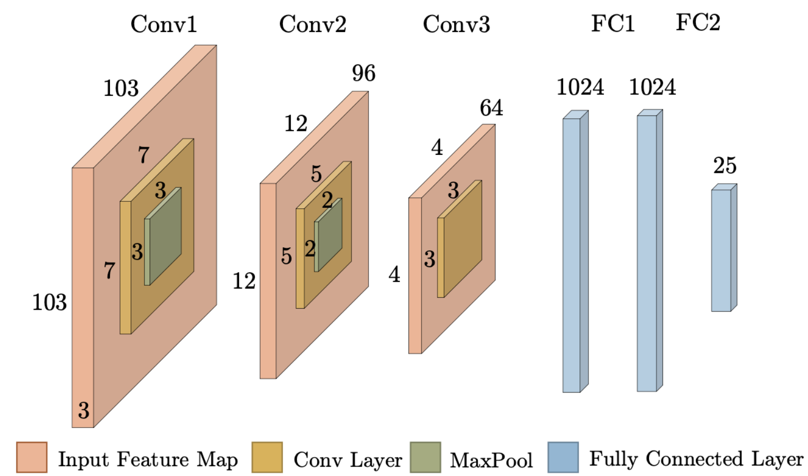
- Transient fault: Transient-1 has a negligible effect compared to Transient-M.
- Permanent fault: Stuck-at-1 has a much severe impact on policy than Stuck- at-0

Drone Autonomous Navigation Problem

Environments and demos:

(PEDRA: https://github.com/aqeelanwar/PEDRA)

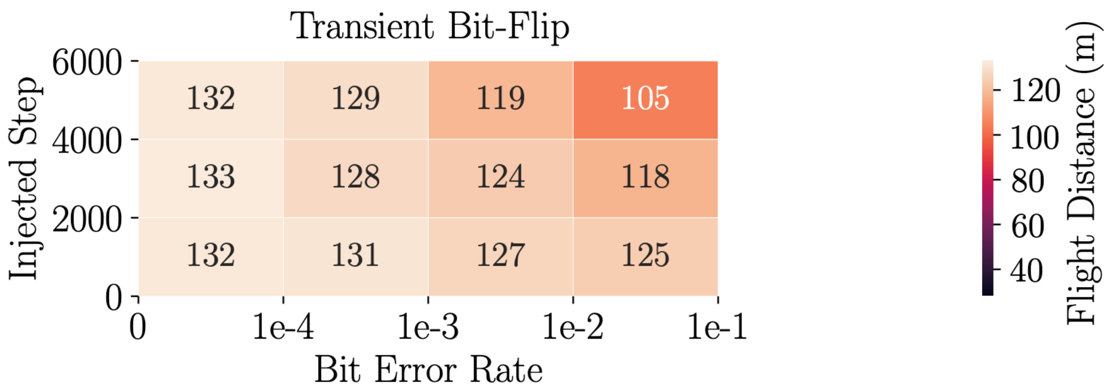
Policy architecture:



- ➢ Evaluation metric: drone safe flight distance (the longer, the better).

Part1: Reliability of learning-based autonomous navigation system

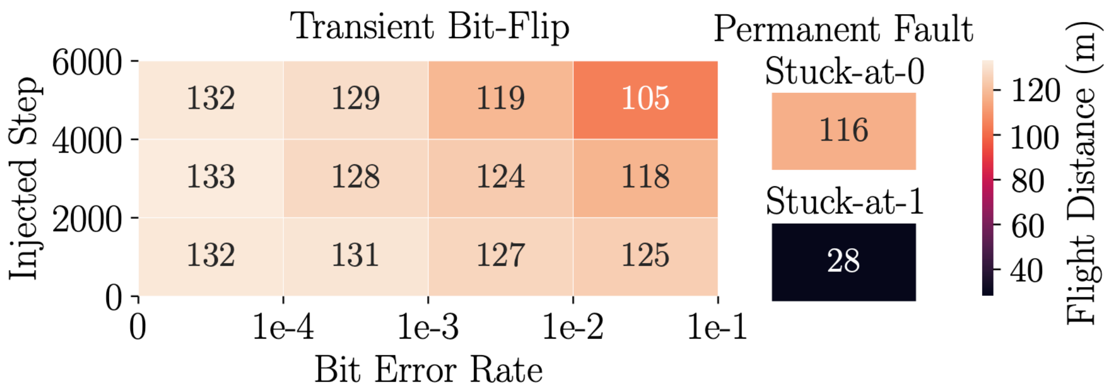
Faults in Drone Navigation (Training)



Higher (lighter) is better

- Training method: offline training -> online fine-tunning using transfer learning
- Transient fault: occurred at latter episodes with higher BER impact flight quality more.

Faults in Drone Navigation (Training)



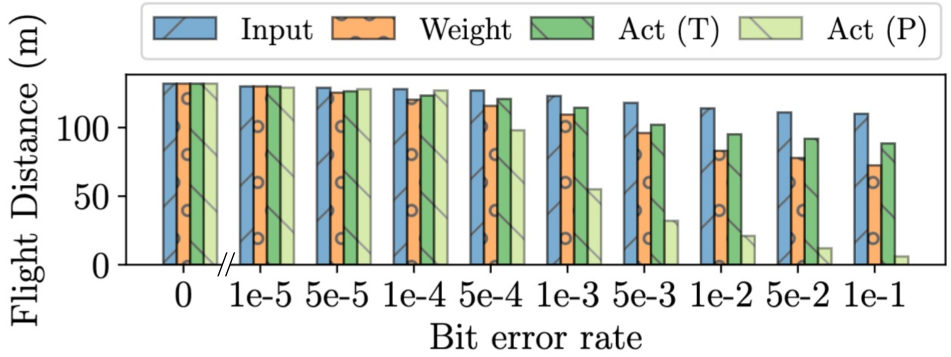
Higher (lighter) is better

- Training method: offline training -> online fine-tunning using transfer learning
- Transient fault: occurred at latter episodes with higher BER impact flight quality more.
- Permanent fault: stuck-at-1 has much severe impact than stuck-at-0

Faults in Drone Navigation (Inference)

Different data locations:

(the higher, the better)

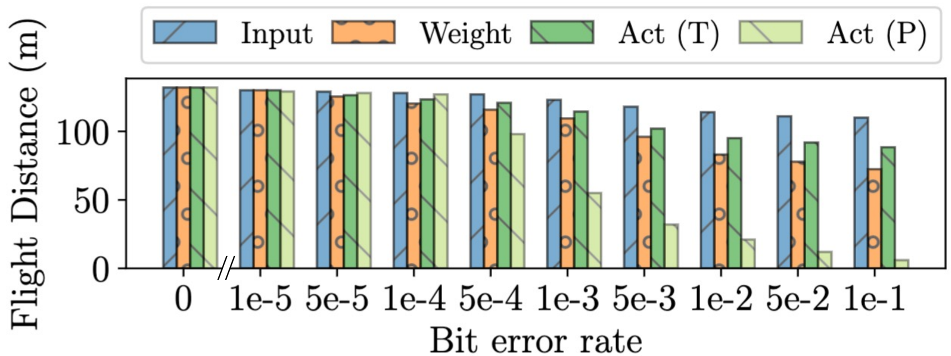


- Weights are sensitive to transient faults while input buffer is resilient.

Faults in Drone Navigation (Inference)

Different data locations:

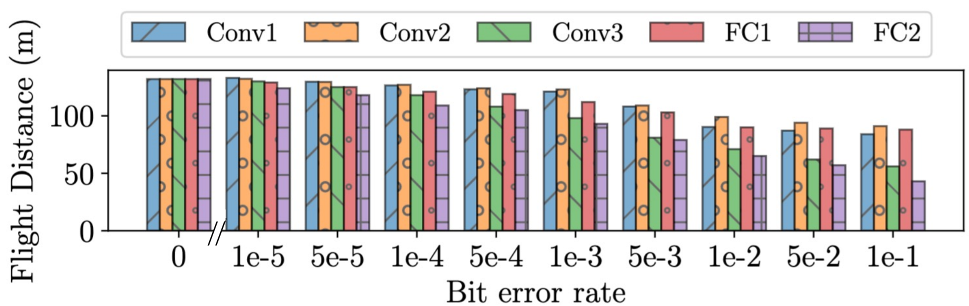
(the higher, the better)



- Weights are sensitive to transient faults while input buffer is resilient.

Different NN layers:

(the higher, the better)



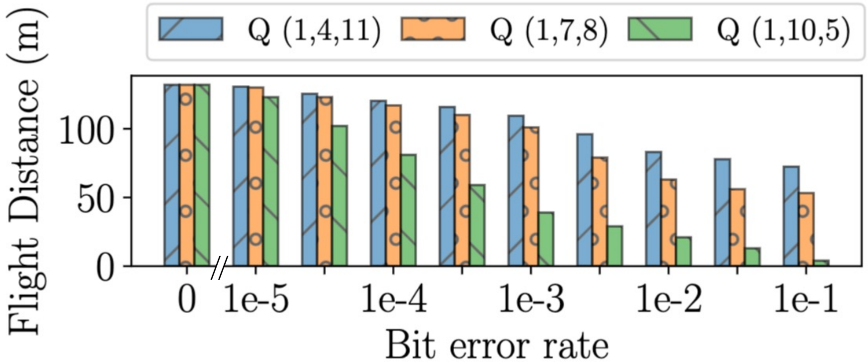
- Conv3: no followed pooling layer
- FC2: directly dictates the drone actions

C-BRIC

Faults in Drone Navigation (Inference)

Q (sign, integer, fraction)

Different data types:
(the higher, the better)



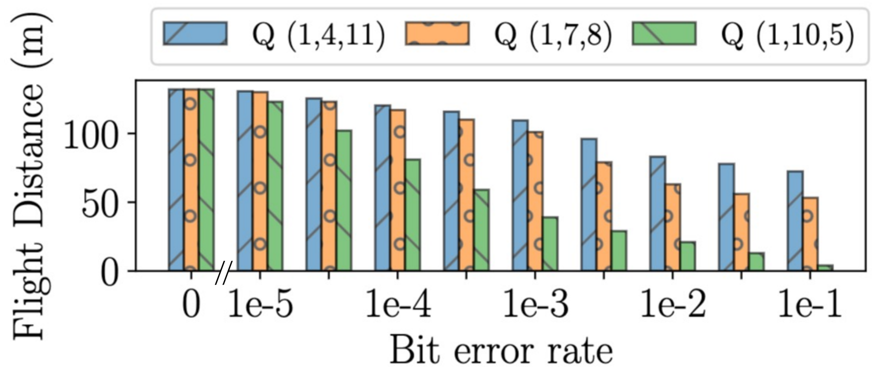
- Data types should optimally capture the value range rather than pursuing an unnecessarily large range

Different bit locations in Q (1,4,11):

Faults in Drone Navigation (Inference)

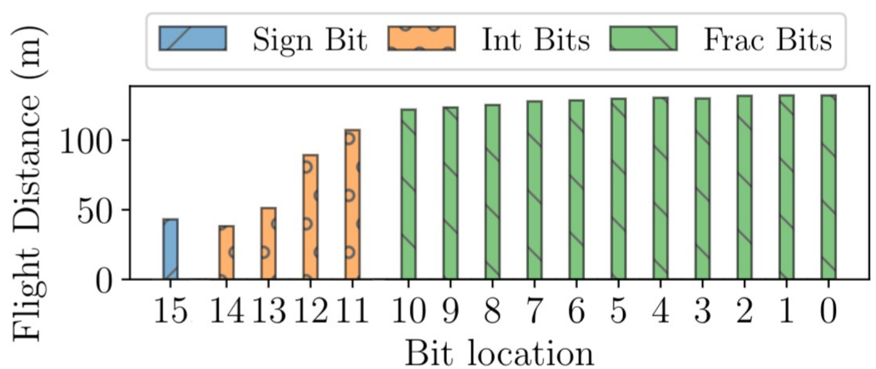
Different data types:
(the higher, the better)

Q (sign, integer, fraction)



- Data types should optimally capture the value range rather than pursuing an unnecessarily large range

Different bit locations in Q (1,4,11):
(the higher, the better)



- Only sign and high-order integer bits are vulnerable

This work

Analyzing and Improving fault tolerance of learning-based navigation systems, that is:

A fault injection tool-chain for learning-based systems

Hardware fault study in learning-based systems

Fault mitigation techniques for learning-based systems

Training: Adaptive Exploration Rate Adjustment

- Detection: change in cumulative reward

	Detection	Recovery
Transient fault	Reward drop exceeds *x*% within *y* continuous episodes	
Permanent fault	Reward is still low after going to steady-exploitation states	

Training: Adaptive Exploration Rate Adjustment

- Detection: change in cumulative reward
- Recovery: dynamically adjust exploration-to-exploitation ratio and speed

	Detection		Recovery
Transient fault	Reward drop exceeds *x*% within *y* continuous episodes	→	Increase exploration rate (ER)
Permanent fault	Reward is still low after going to steady-exploitation states	→	Revert the exploration rate to initial and slow down its decreasing speed by $2^n \times$

f(r): reward drop
f(t): fault occurrence time

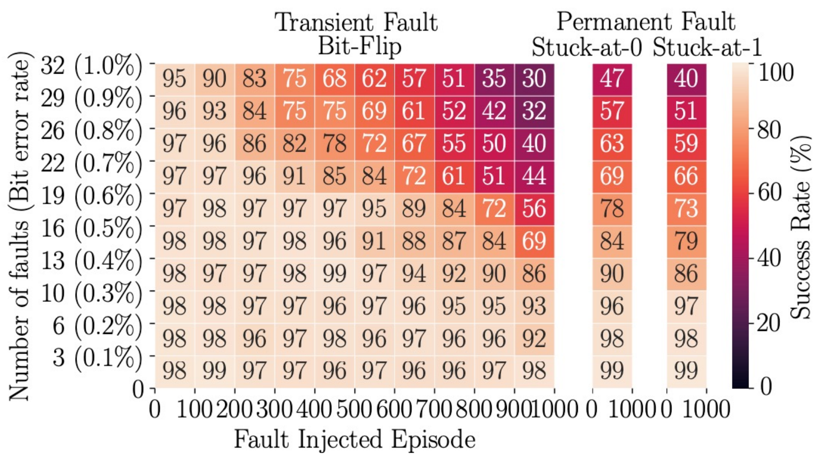
$$ER_{new} = ER_{old} + \delta(ER) = ER_{old} + \alpha \times min(f(r), f(r)f(t))$$

C-BRIC

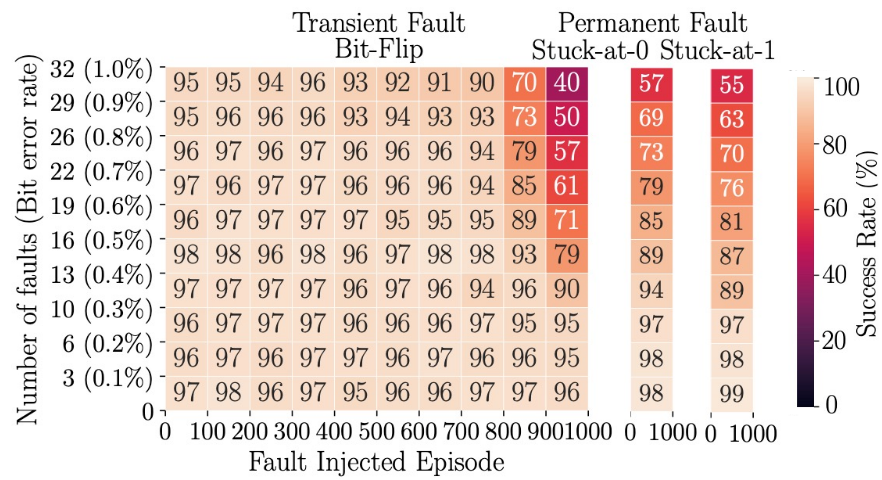
Training: Adaptive Exploration Rate Adjustment

- Evaluation:

Before fault mitigation:



After fault mitigation:



➢ The impact of both transient fault and permanent fault during training can be relieved.

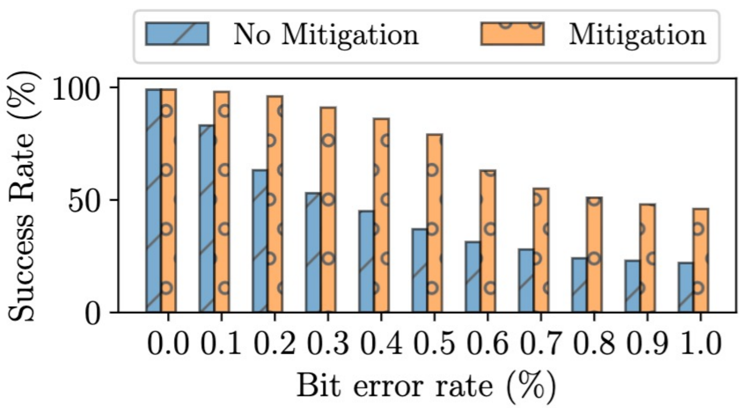
Inference: Value Range-Based Anomaly Detection

- Detection: statistically anomaly detection, (a_i, b_i) -> (1.1a_i, 1.1b_i)
- Recovery: skip faulty operations

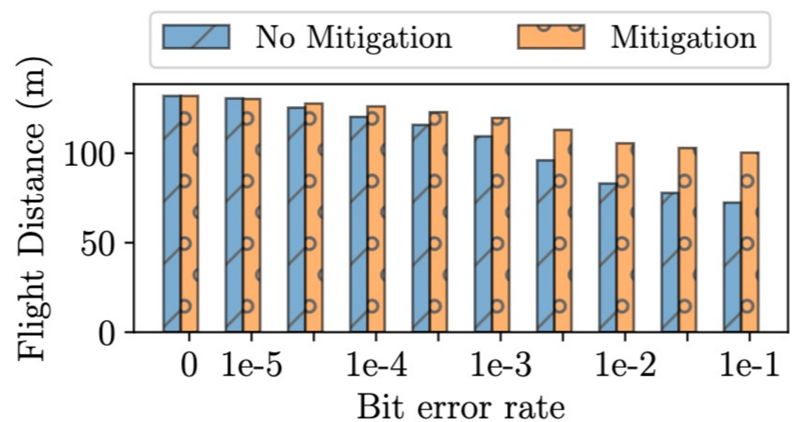
Inference: Value Range-Based Anomaly Detection

- Detection: statistically anomaly detection, (a_i, b_i) -> $(1.1a_i, 1.1b_i)$
- Recovery: skip faulty operations
- Evaluation:

Grid World navigation



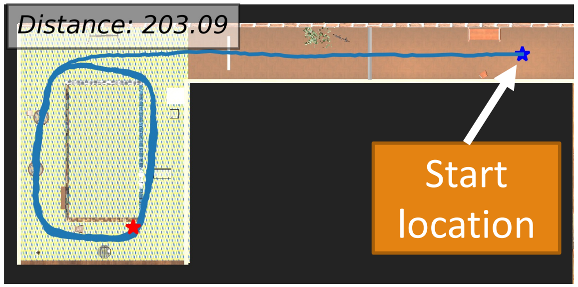
Drone autonomous navigation



- Grid World: agent's success rate increase by 2x
- Drone autonomous navigation: safe flight distance increases by 39%

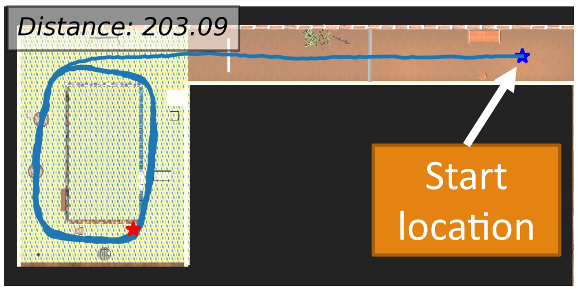
Drone Flight Trajectory Demo

No fault:

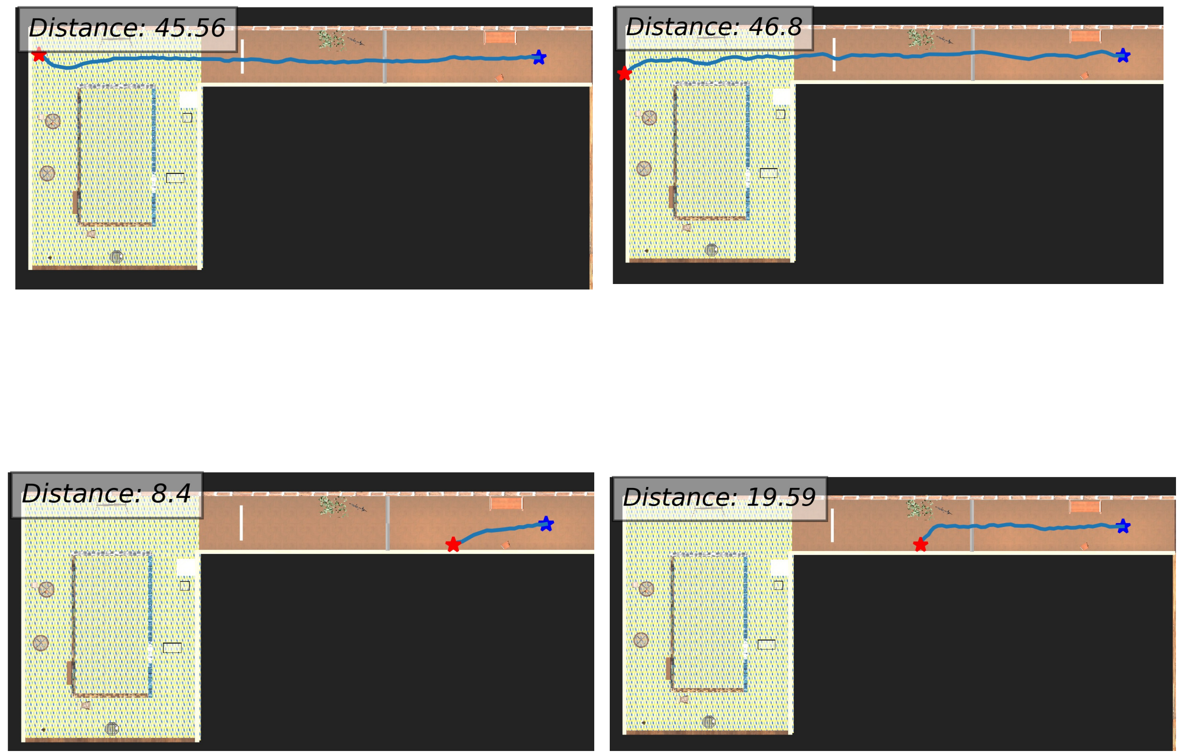


Drone Flight Trajectory Demo

No fault:

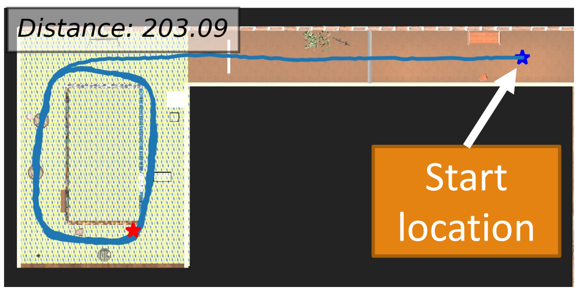


Fault injected:

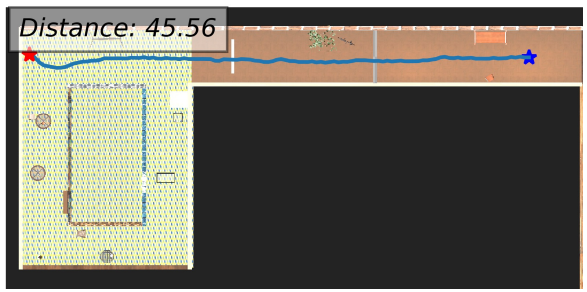


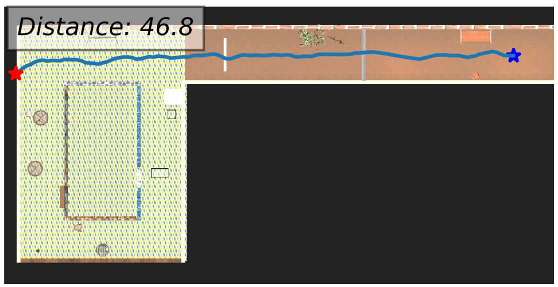
Drone Flight Trajectory Demo

No fault:

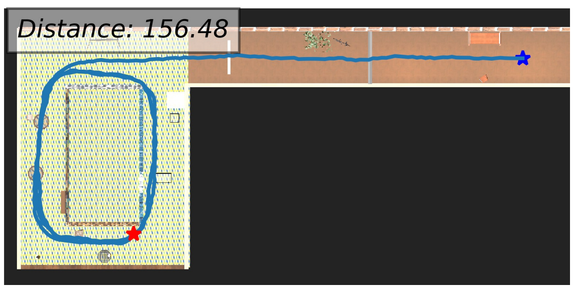


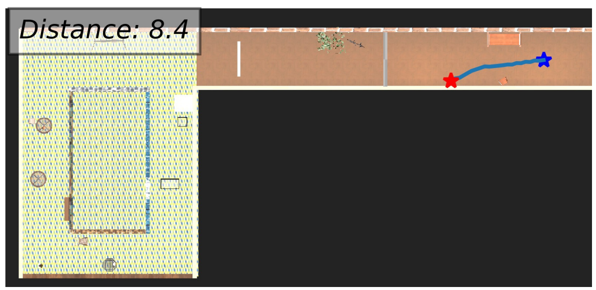
Fault injected:

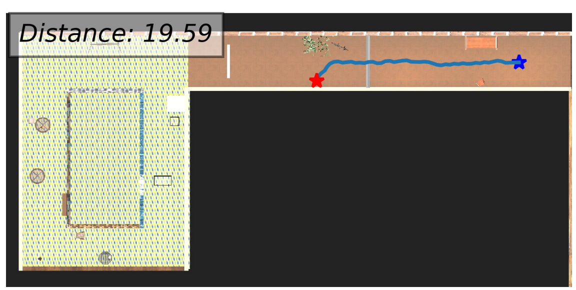




Fault mitigated:







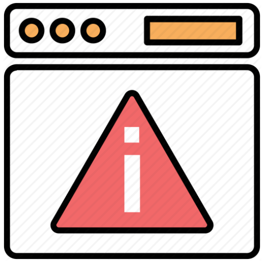
Part 1 Summary

Analyzing and Improving Fault Tolerance of Learning-Based Navigation System:

The **safety and reliability** of end-to-end **learning-based navigation systems** is important, but not well understood

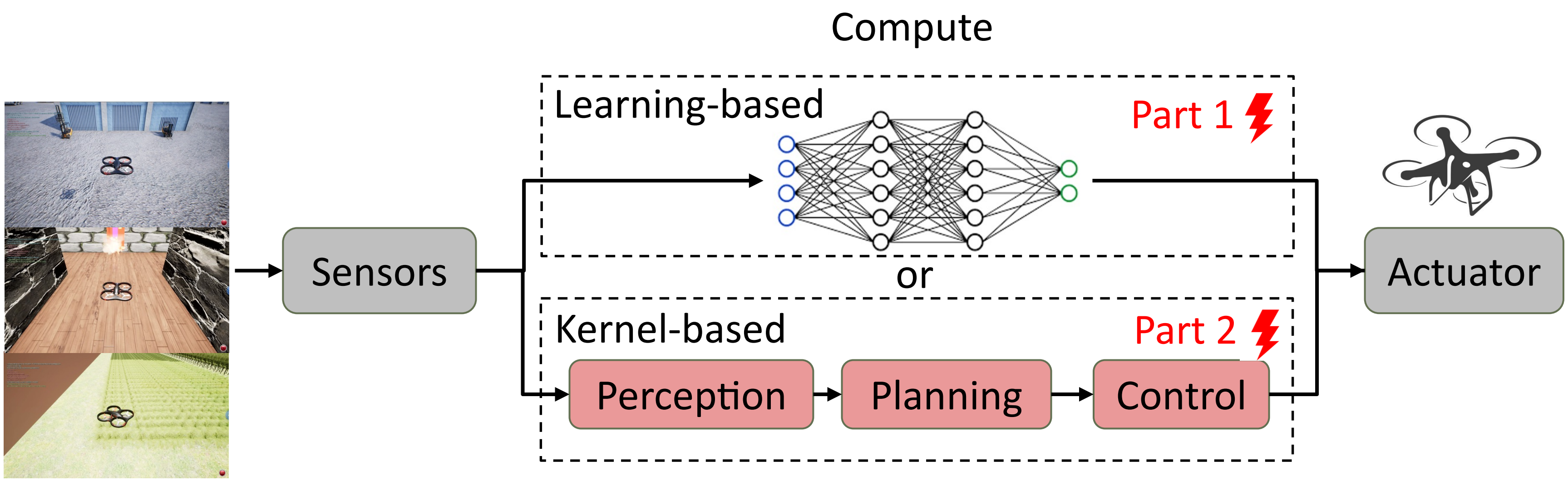
A **fault injection tool-chain** that emulates hardware faults and enables rapid fault analysis of learning-based navigation systems

Large-scale **fault injection study** in both training and inference stages of learning-based systems against permanent and transient faults



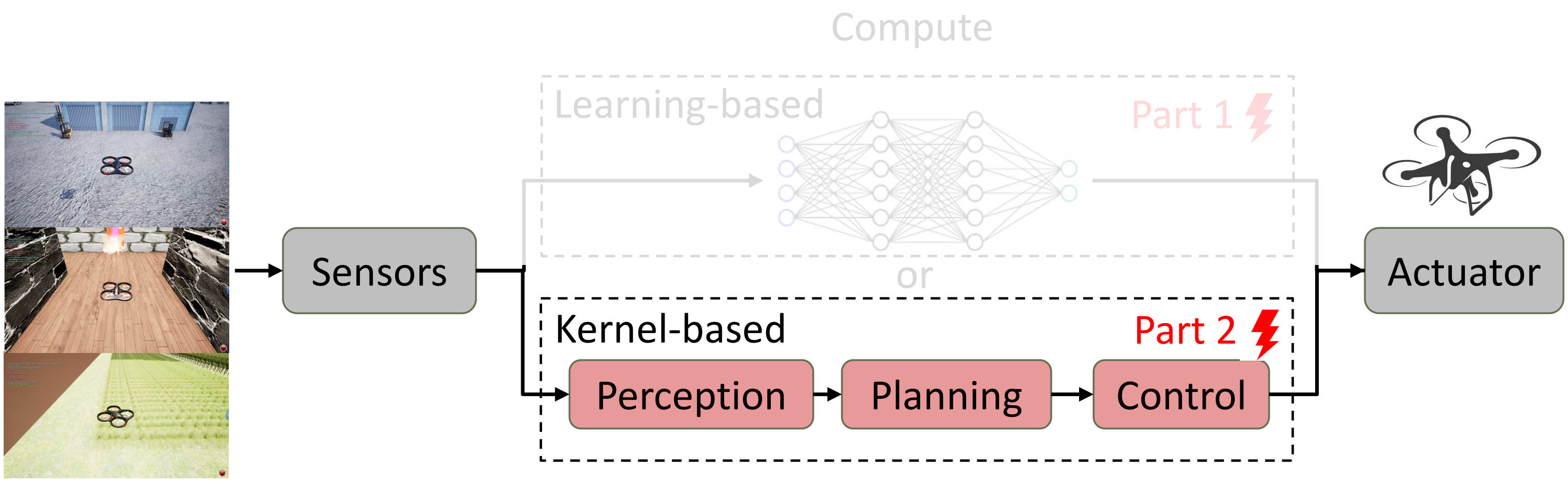
Low-overhead **fault detection and recovery techniques** for both training and inference

Autonomous Navigation System Paradigm



- Part1: Reliability of learning-based navigation pipeline
- Part2: Reliability of kernel-based navigation pipeline

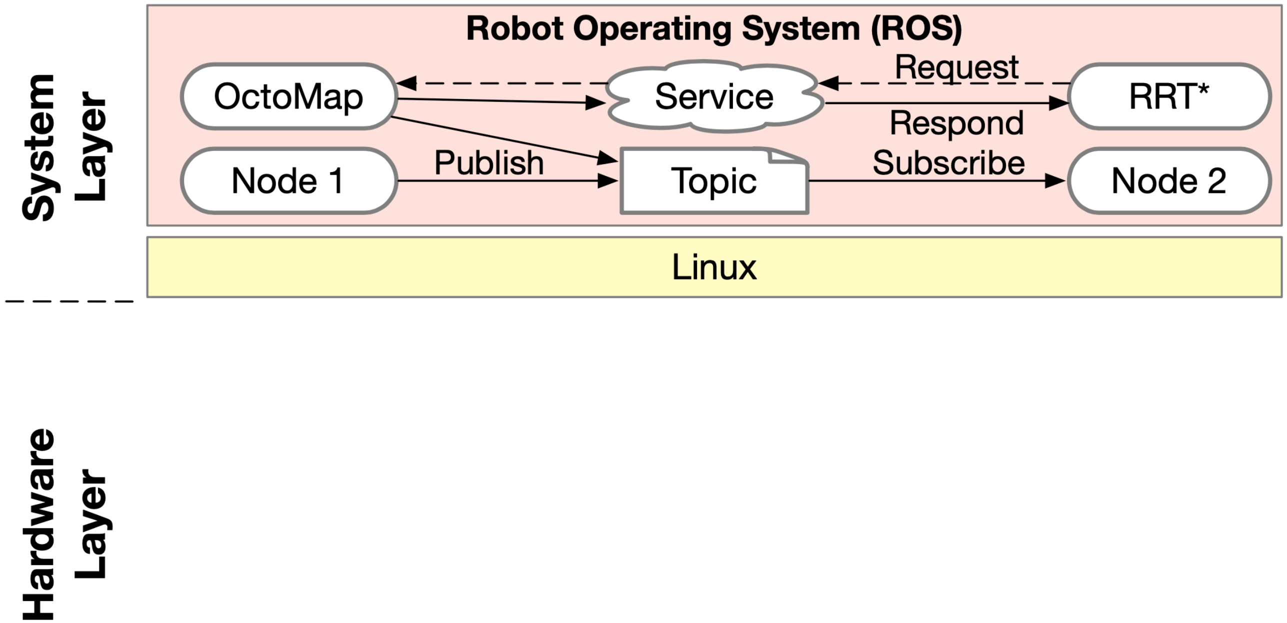
Autonomous Navigation System Paradigm



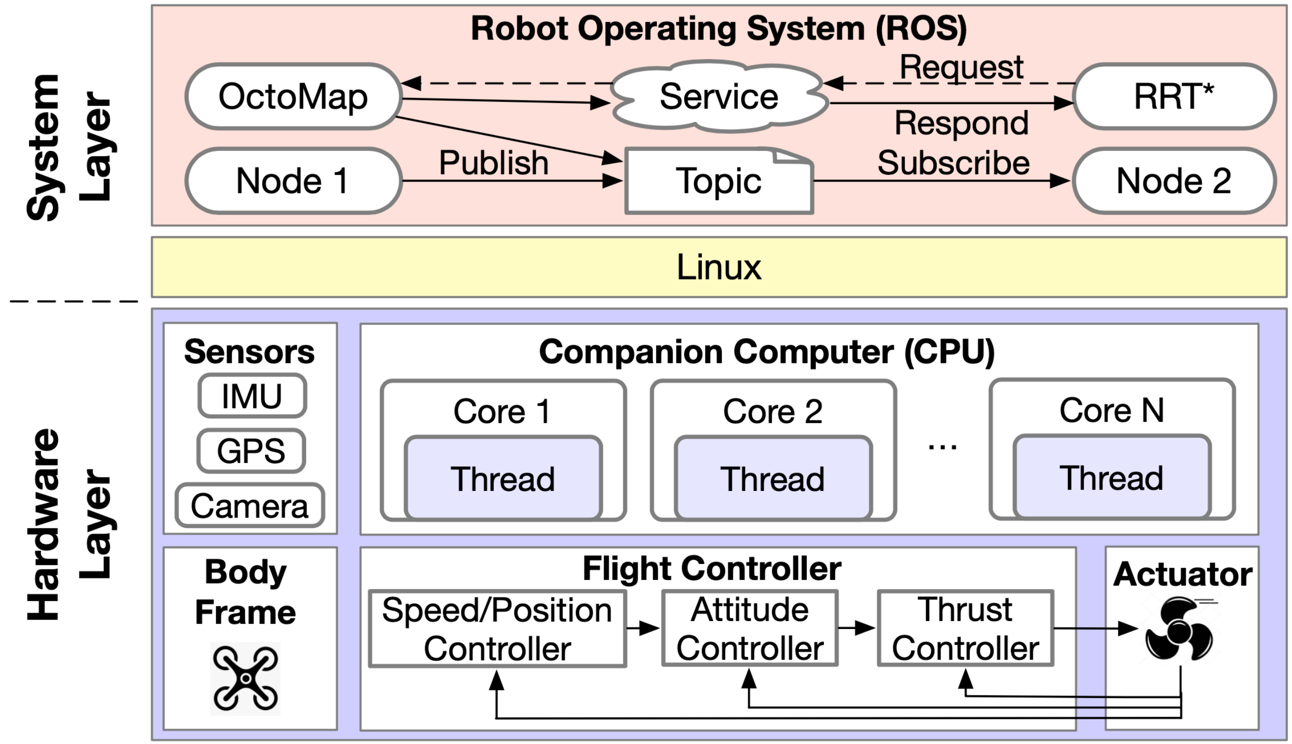
- Part1: Reliability of learning-based navigation system
- Part2: Reliability of kernel-based navigation system

C-BRIC

Drone Computing Stack



Drone Computing Stack



Part2: Reliability of kernel-based autonomous navigation system

This work

MAVFI: An End-to-End Fault Analysis Framework with Anomaly Detection and Recovery for Micro Aerial Vehicles

- A fault injection tool-chain for kernel-based systems
- Fault mitigation techniques for kernel-based systems
- Hardware fault study in kernel-based systems

This work

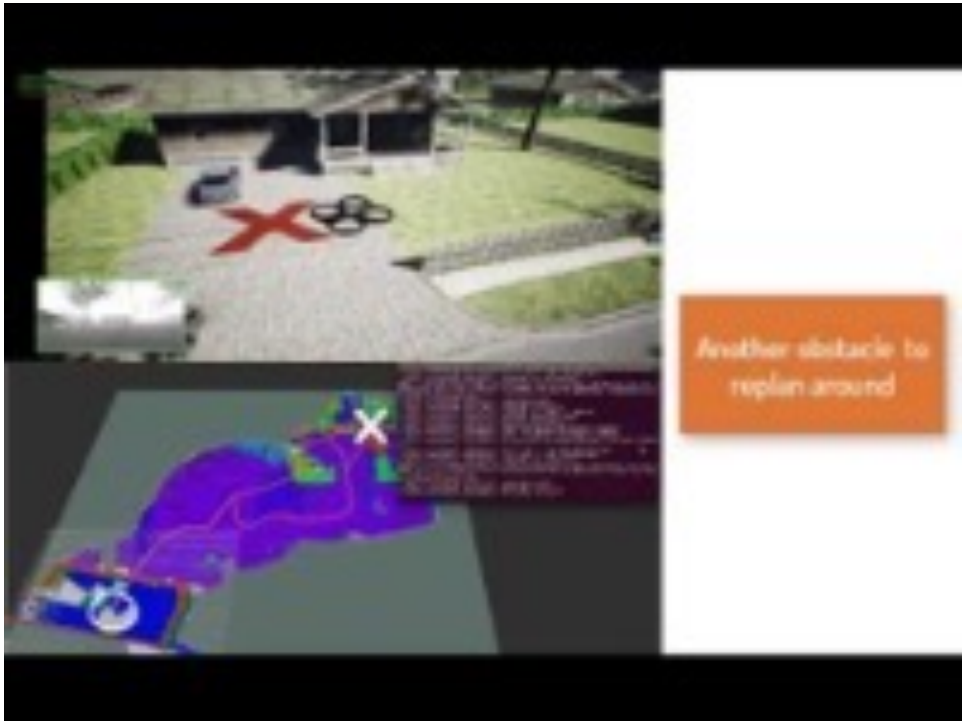
MAVFI: An End-to-End Fault Analysis Framework with Anomaly Detection and Recovery for Micro Aerial Vehicles

A fault injection tool-chain for kernel-based systems

Fault mitigation techniques for kernel-based systems

Hardware fault study in kernel-based systems

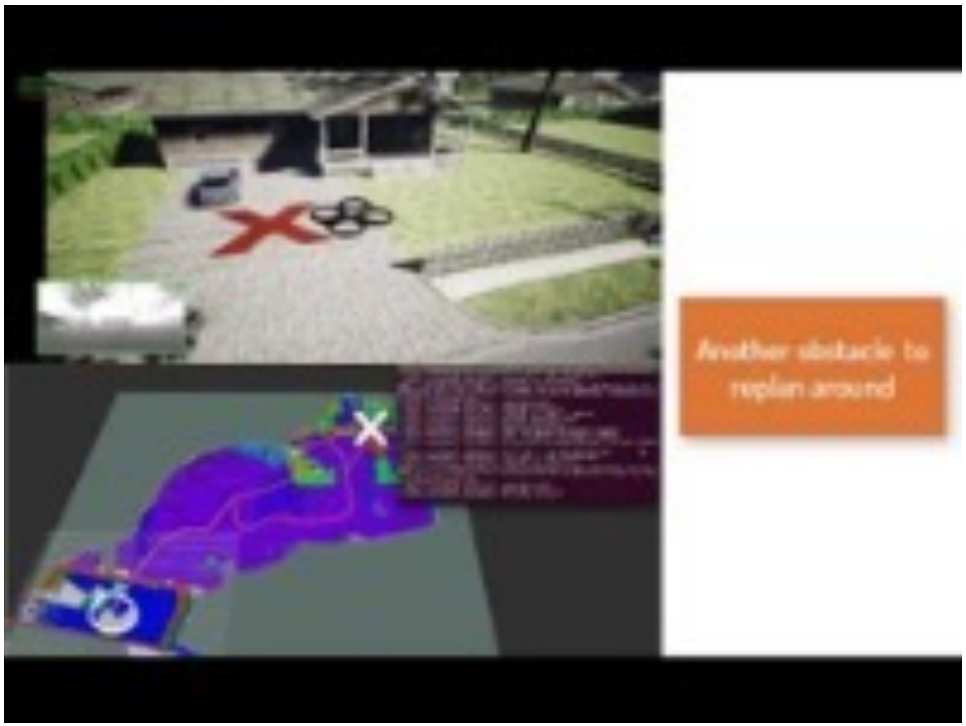
MAVFI Basis: Drone Simulator



MAVBench drone simulator

https://github.com/harvard-edge/MAVBench

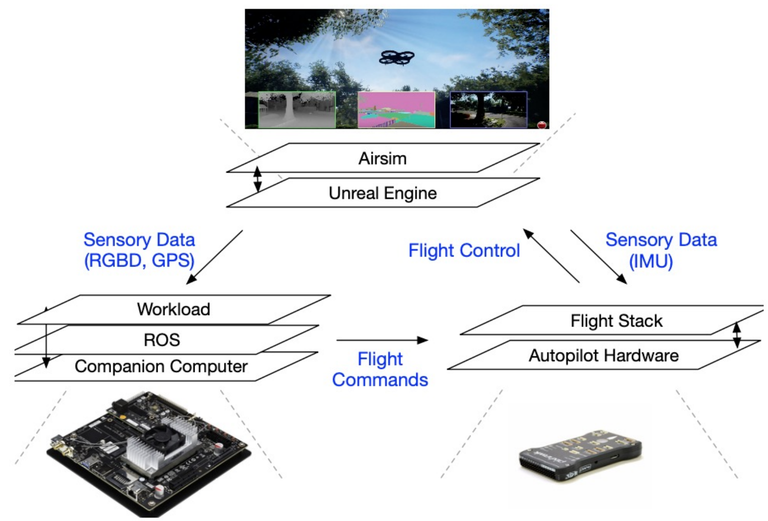
MAVFI Basis: Drone Simulator



MAVBench drone simulator

https://github.com/harvard-edge/MAVBench

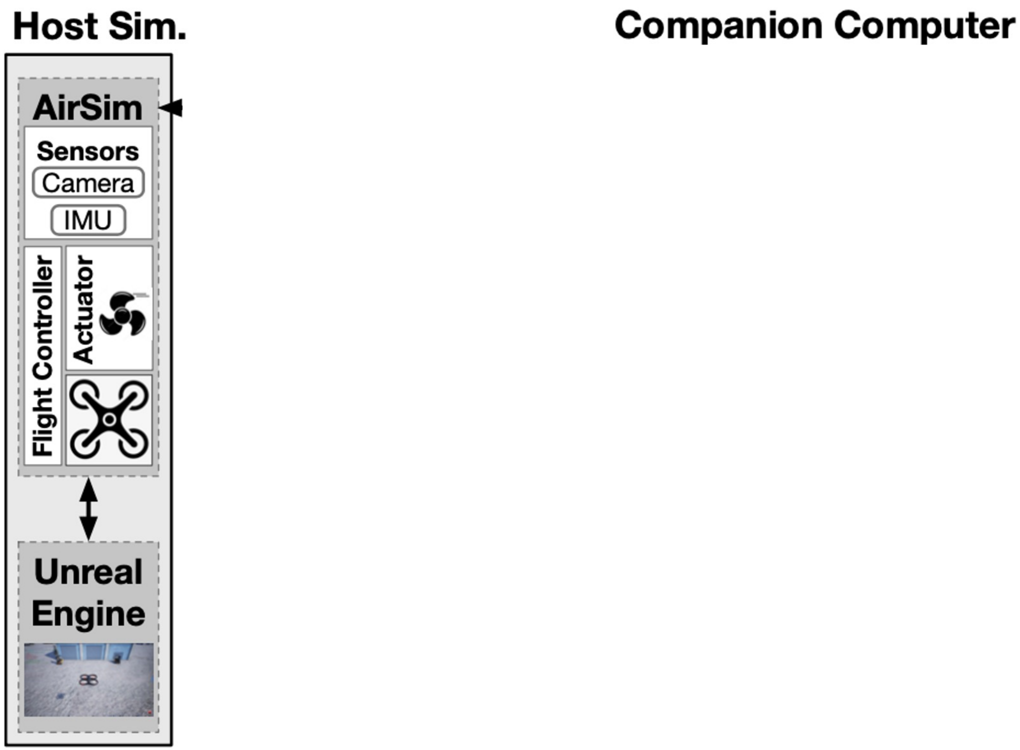
Environment
(Host simulator)



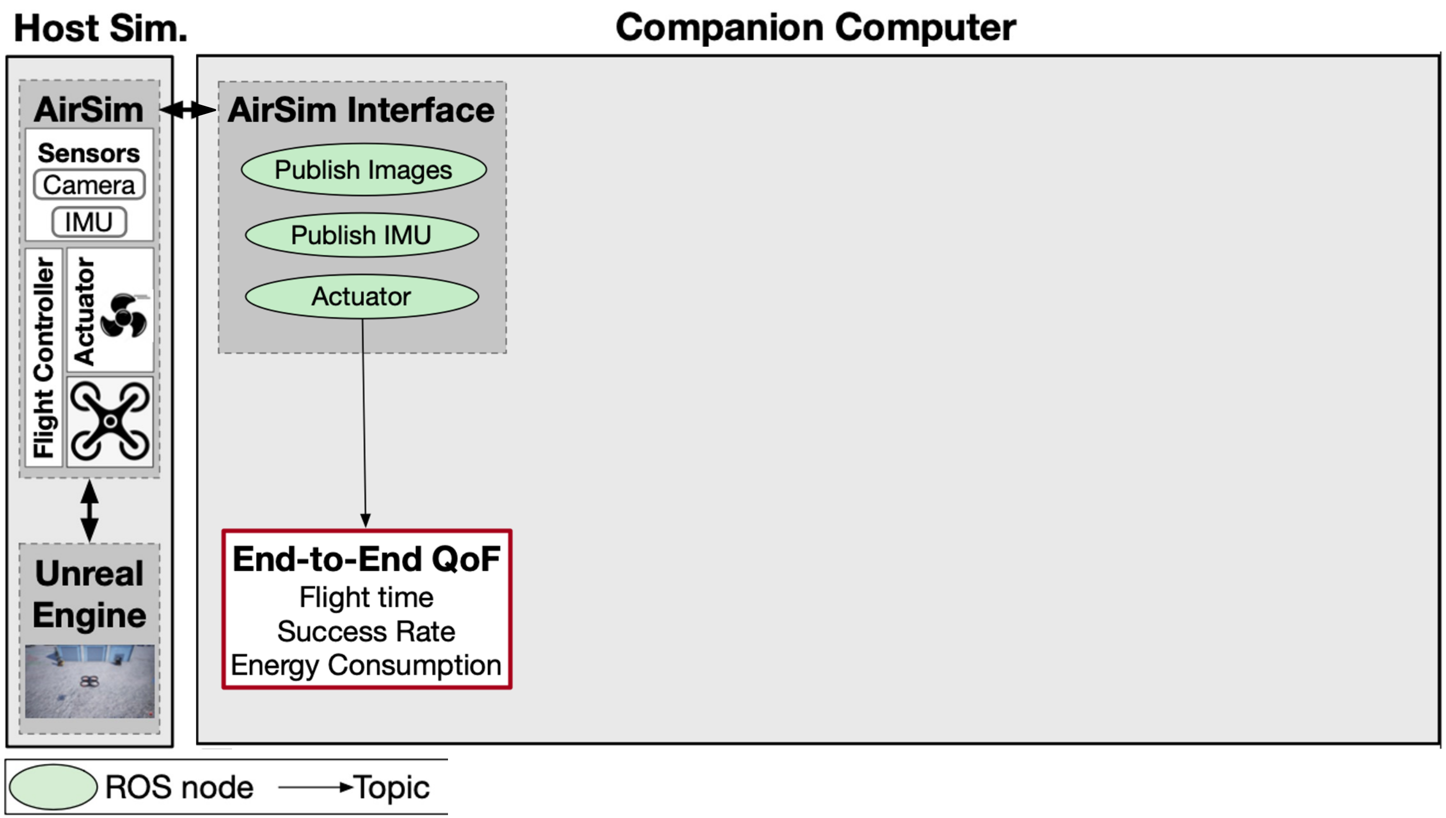
Compute System
(Companion Computer)

Flight controller

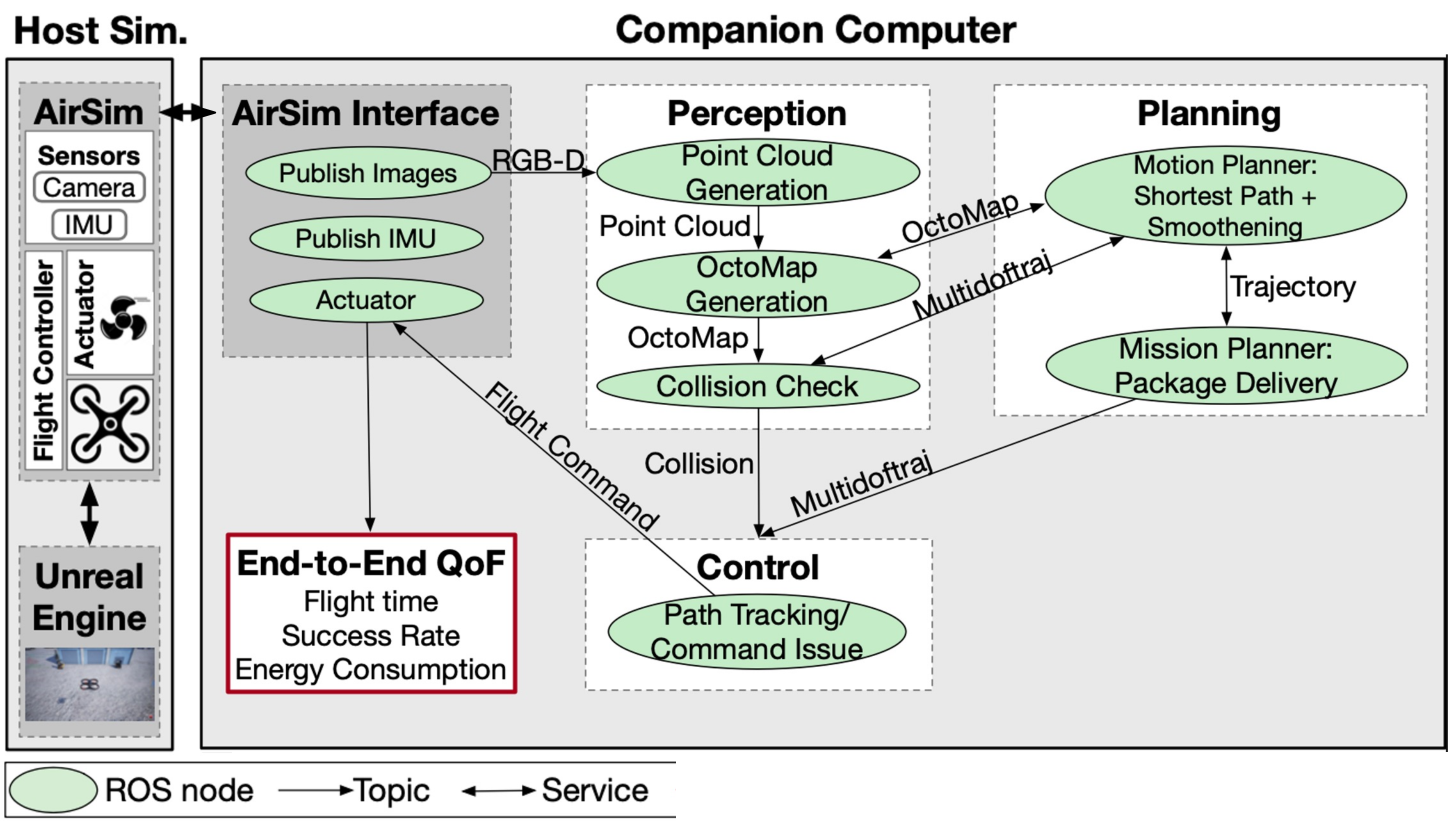
MAVFI Fault Injection Framework



MAVFI Fault Injection Framework

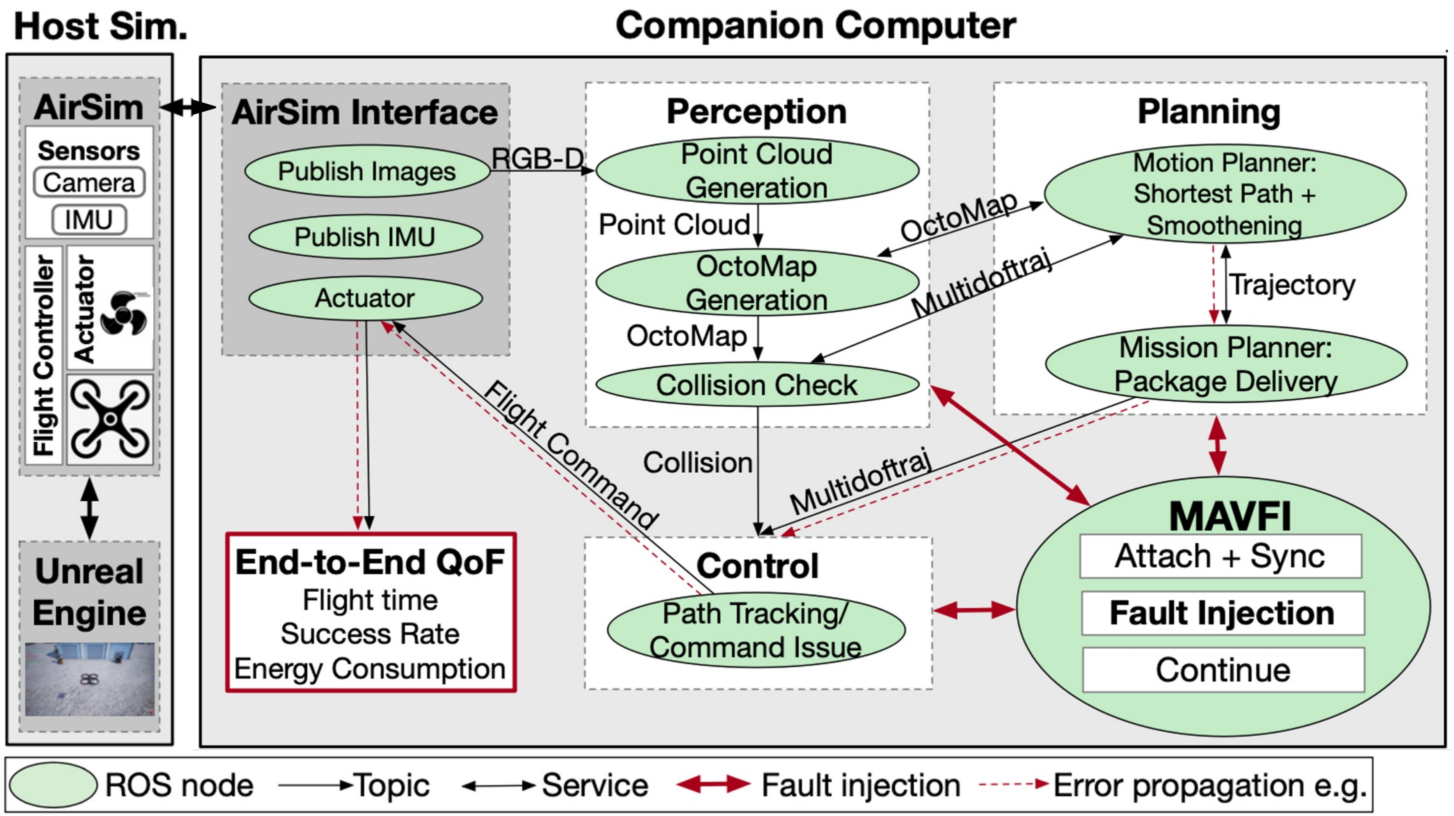


MAVFI Fault Injection Framework



Part2: Reliability of kernel-based autonomous navigation system

MAVFI Fault Injection Framework



Source code: https://github.com/harvard-edge/MAVBench/tree/mavfi

Fault Injection Methodology Details

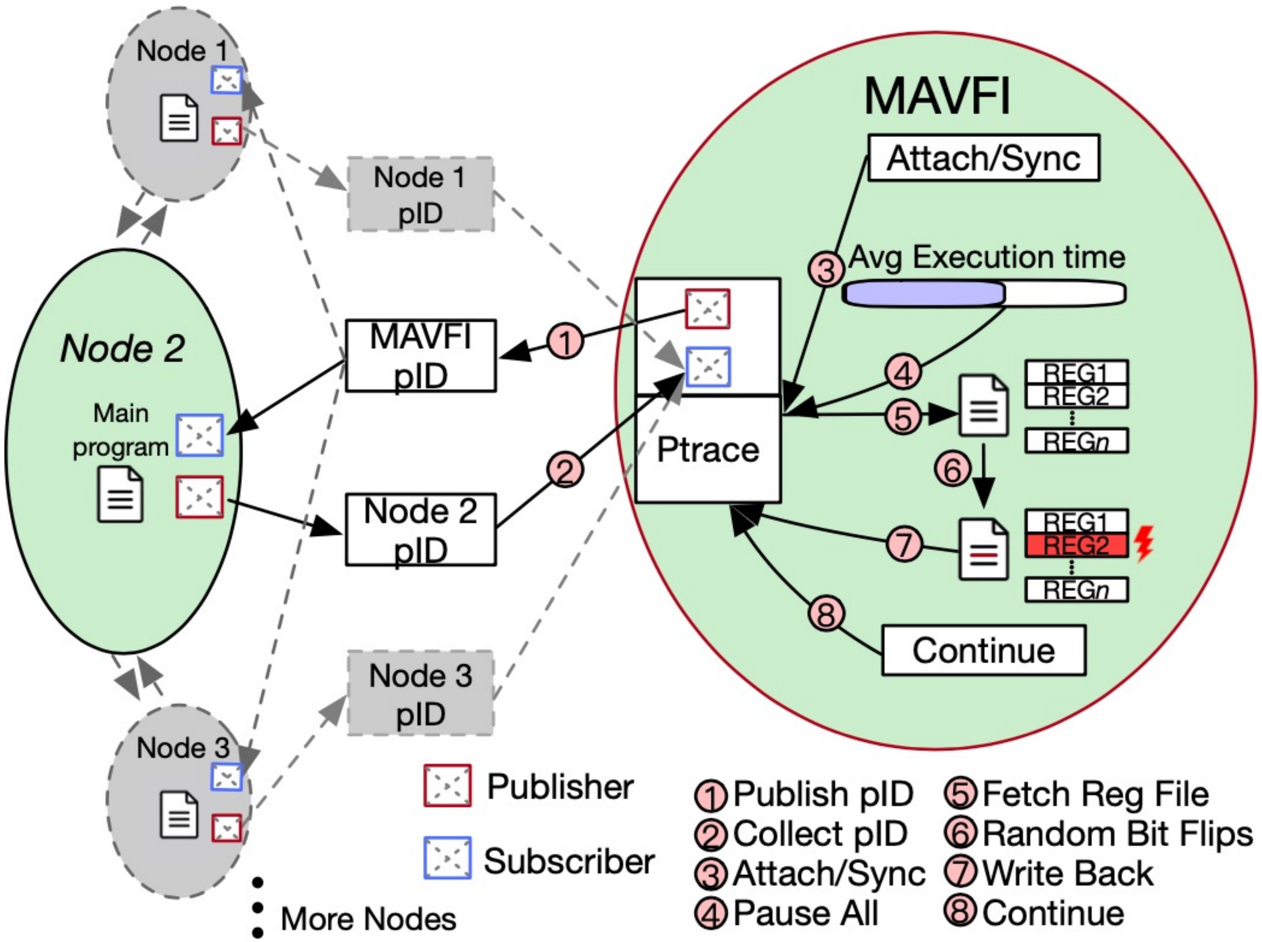


Figure 5: The design details for the interactions in MAVFI.

Fault Injection Methodology Discussion

Table 2: Comparison of fault injection techniques at various layers of abstraction.

Abstraction Layer	Platform	Perf. (cycles/sec)	E2E Exec. Time (1 run)	E2E Exec. Time (1000 runs)
RTL	IVM Alpha-like processor RTL simulation [58]	6×10^2	4.2×10^5 hours	1.74×10^7 days
Micro-architecture	gem5 simulator [10]	3×10^6	83.3 hours	3472 days
FPGA Emulation	OpenSPARC T1 FPGA emulation [67]	1×10^7	25 hours	1040 days
Architecture	TSIM SPARC simulator [19]	6×10^7	4.17 hours	173.6 days
Software (Ours)	x86 processor [84]	3×10^9	5 mins	3.48 days

- Software-level fault injection is necessary for end-to-end fault analysis

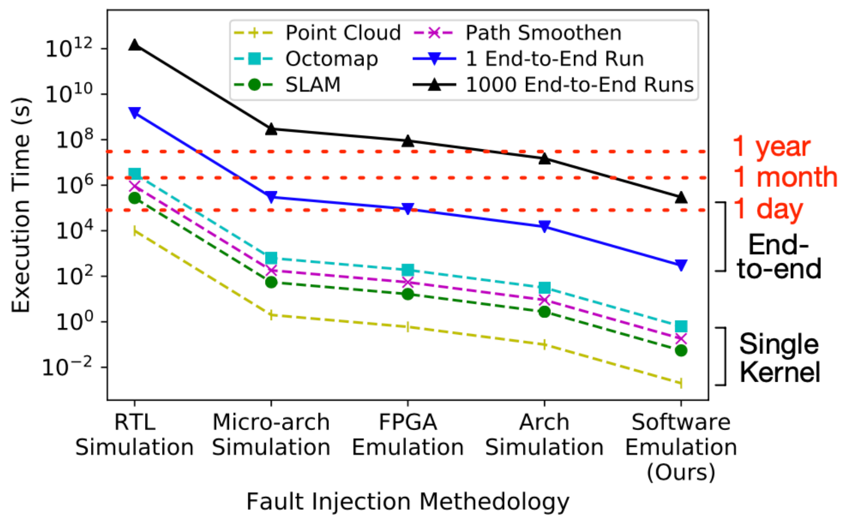


Figure 4: Comparison of fault injection techniques at various layers of abstraction.

This work

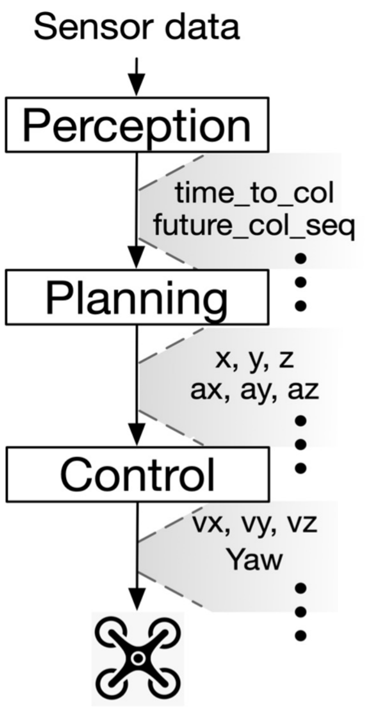
MAVFI: An End-to-End Fault Analysis Framework with Anomaly Detection and Recovery for Micro Aerial Vehicles

A fault injection tool-chain for kernel-based systems

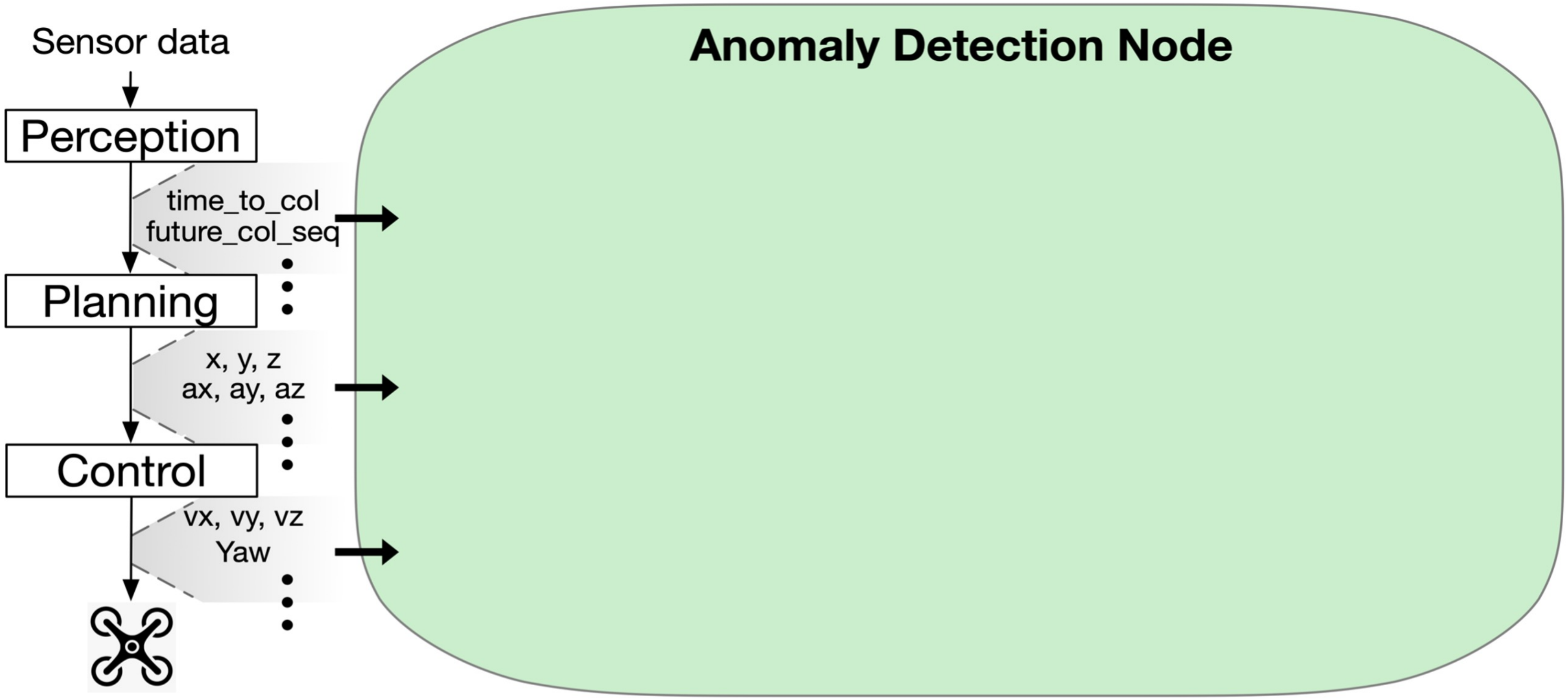
Fault mitigation techniques for kernel-based systems

Hardware fault study in kernel-based systems

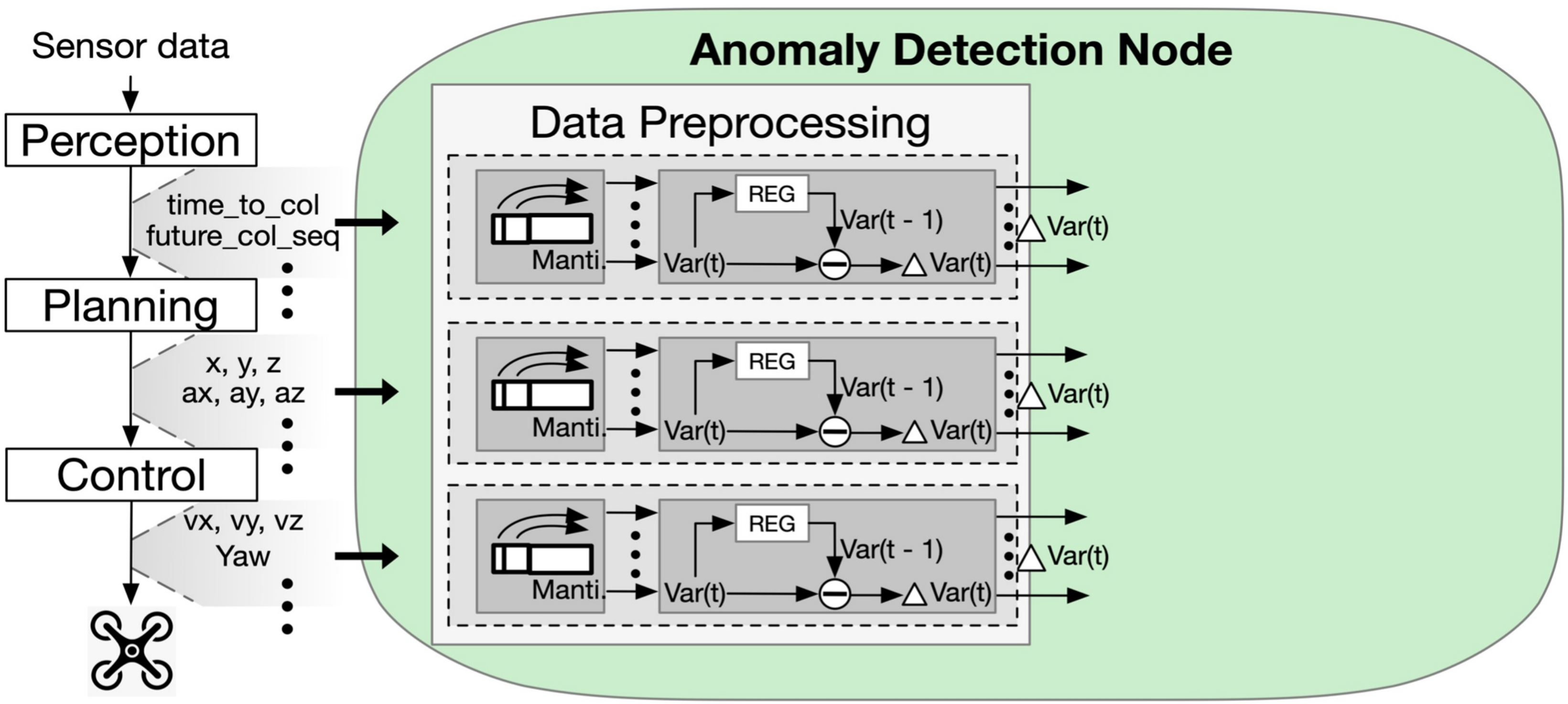
Anomaly Detection and Recovery



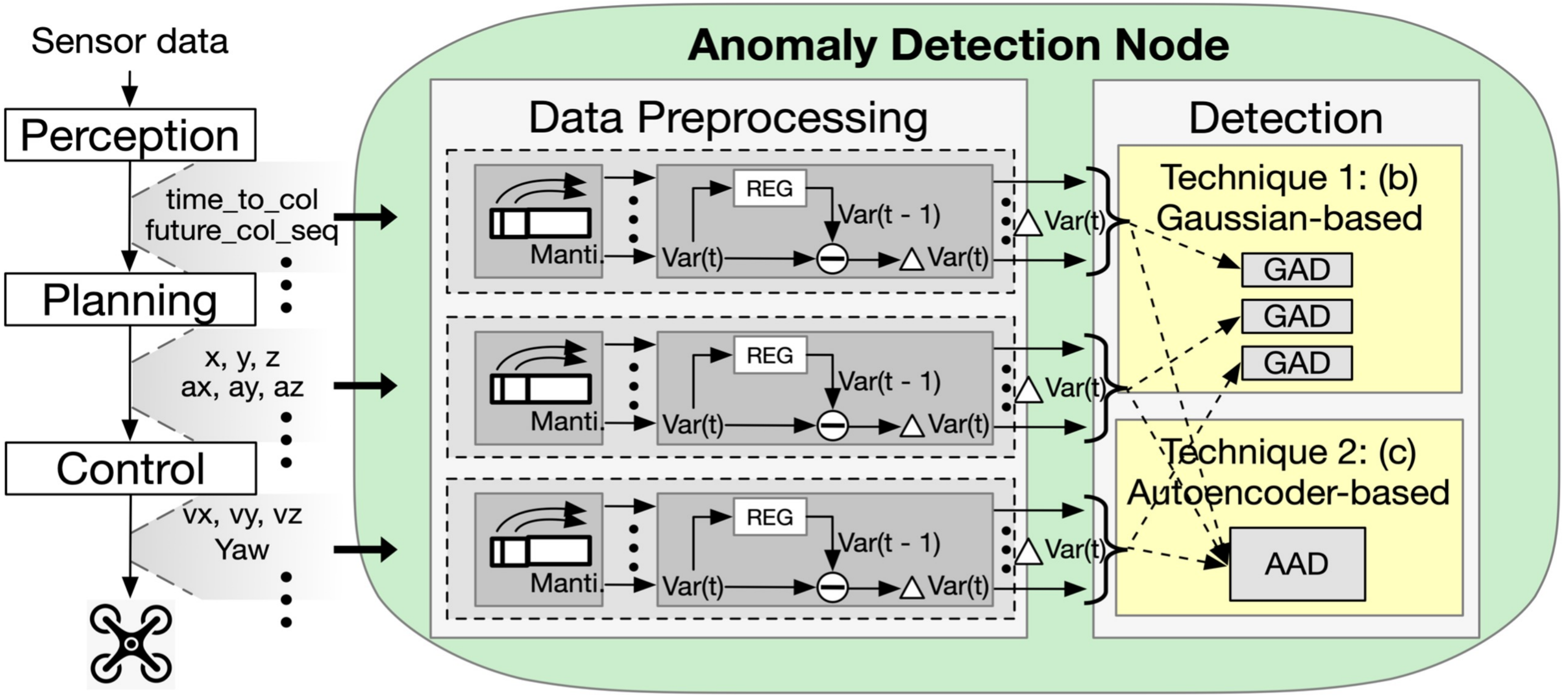
Anomaly Detection and Recovery



Anomaly Detection and Recovery

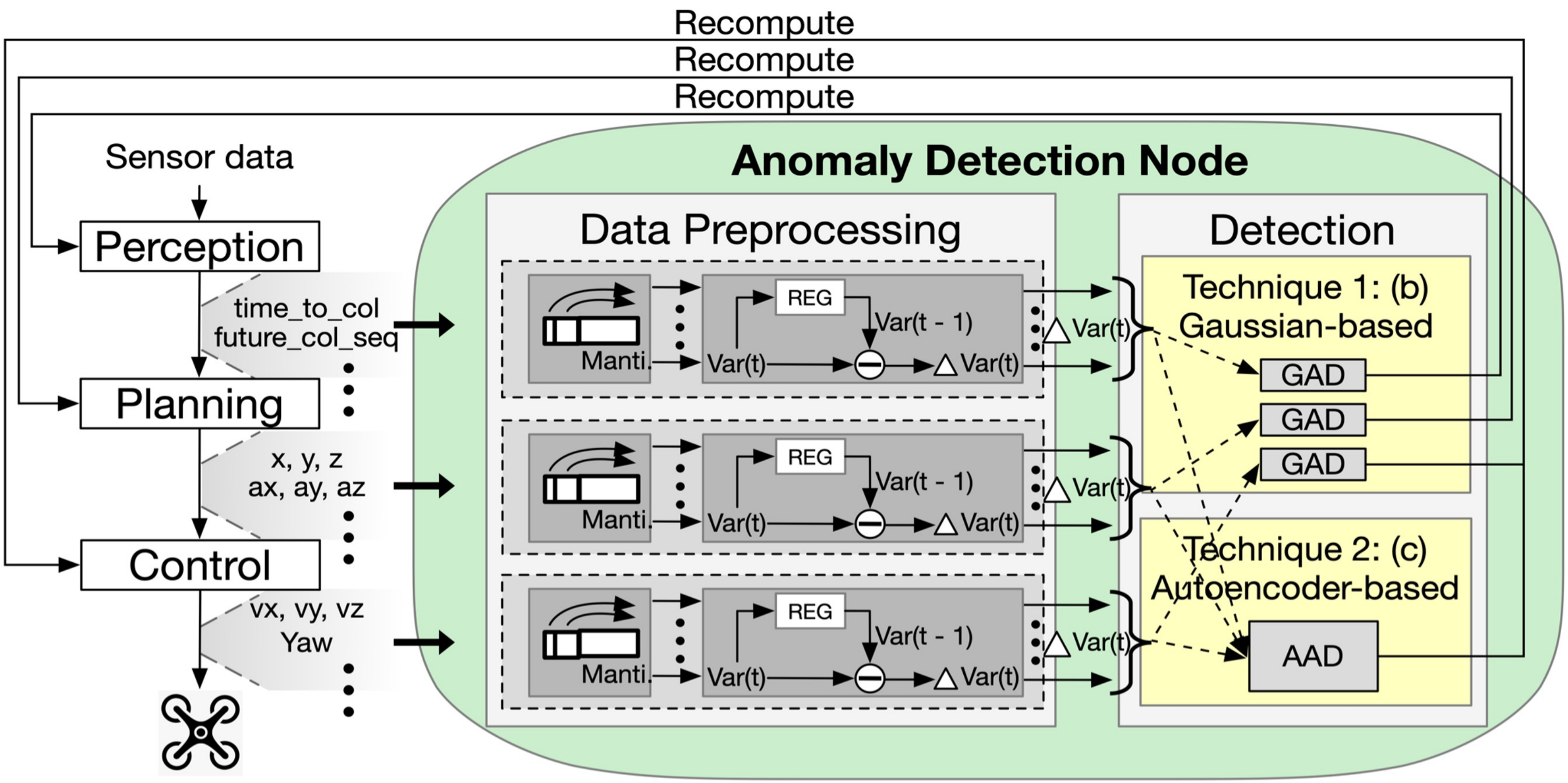


Anomaly Detection and Recovery



C-BRIC

Anomaly Detection and Recovery



This work

MAVFI: An End-to-End Fault Analysis Framework with Anomaly Detection and Recovery for Micro Aerial Vehicles

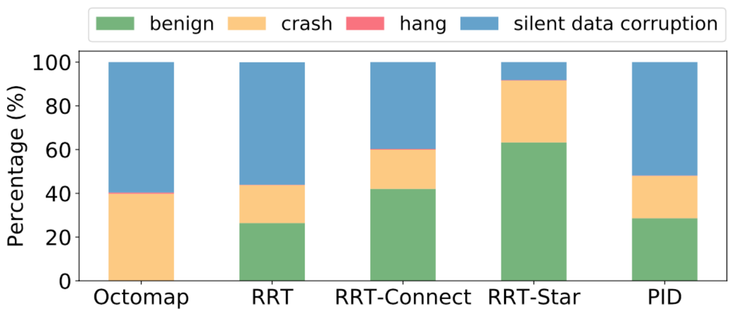
A fault injection tool-chain for kernel-based systems

Fault mitigation techniques for kernel-based systems

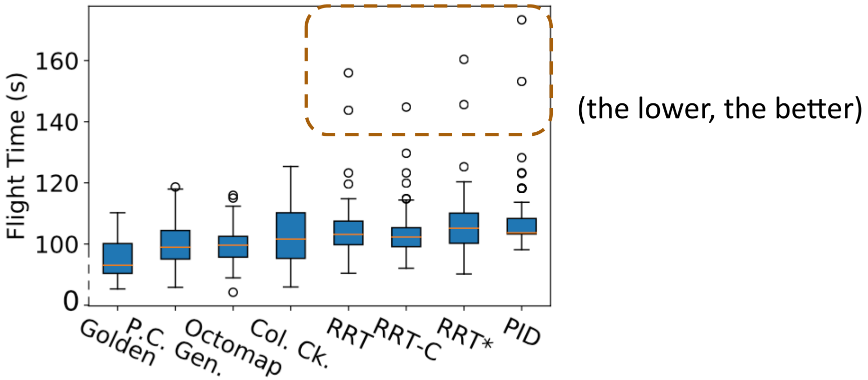
Hardware fault study in kernel-based systems

Key Takeaways from Results

➢ End-to-end fault analysis is essential to understand kernel vulnerability and fault's impact compared to conventional isolated analysis approach.



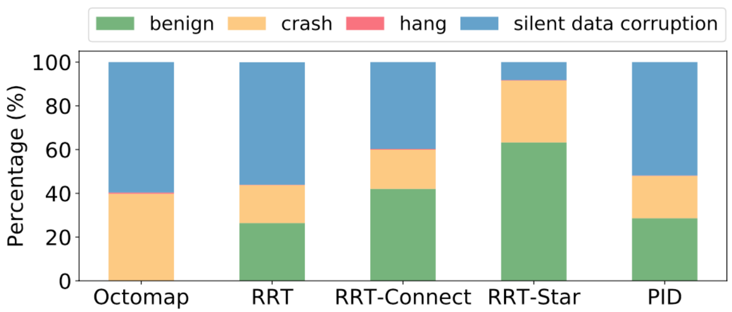
(isolated-kernel analysis)

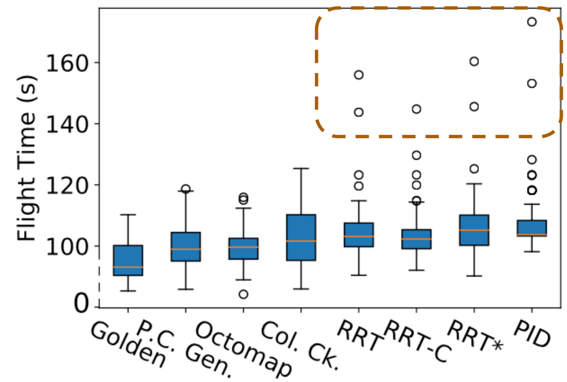


(end-to-end analysis)

Key Takeaways from Results

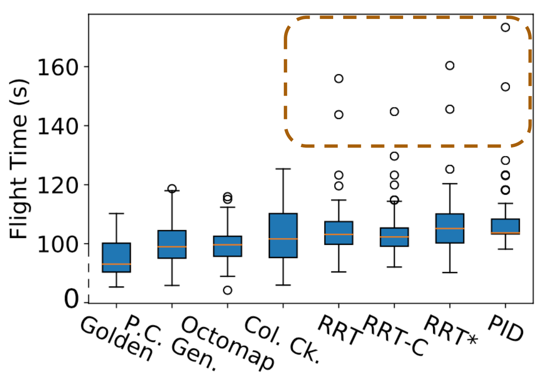
- End-to-end fault analysis is essential to understand kernel vulnerability and fault's impact compared to conventional isolated analysis approach.



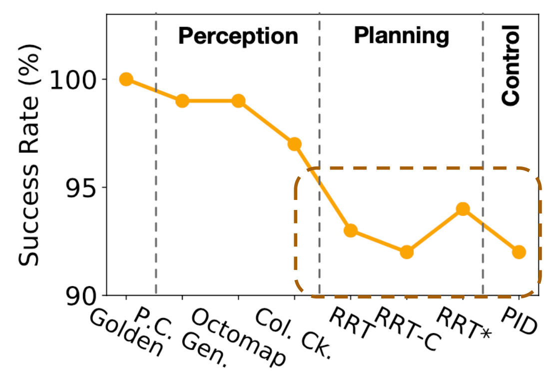


(the lower, the better)

- Planning and control stages are more vulnerable to faults



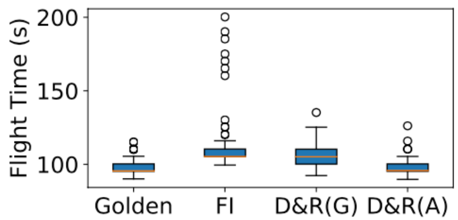
(the lower, the better)

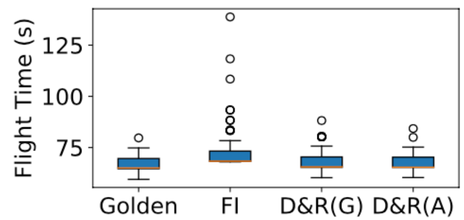


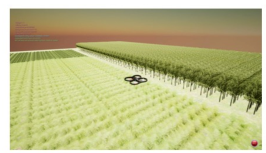
(the higher, the better)

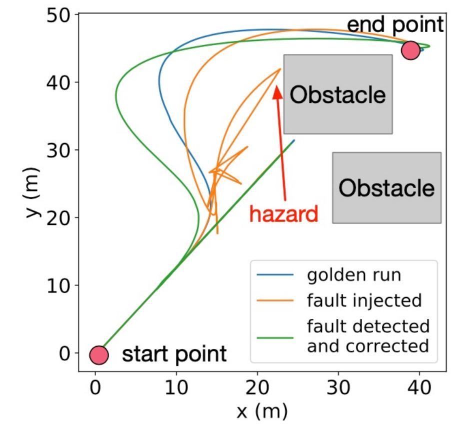
Key Takeaways from Results

- Anomaly detection and recovery enables autonomy reliability



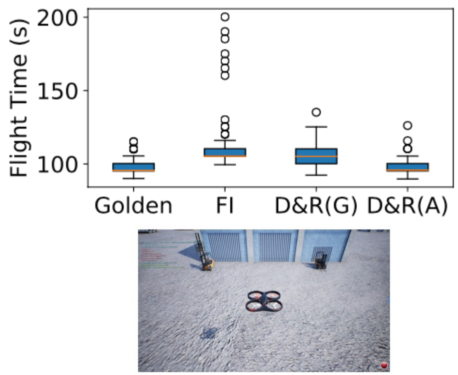


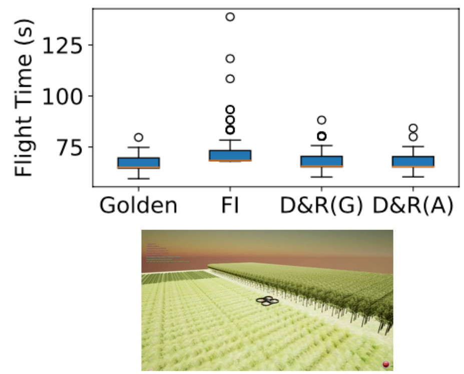


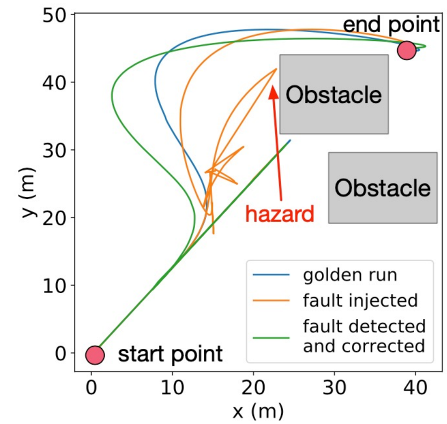


Key Takeaways from Results

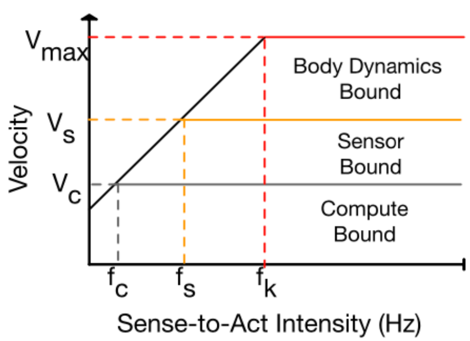
- Anomaly detection and recovery enables autonomy reliability



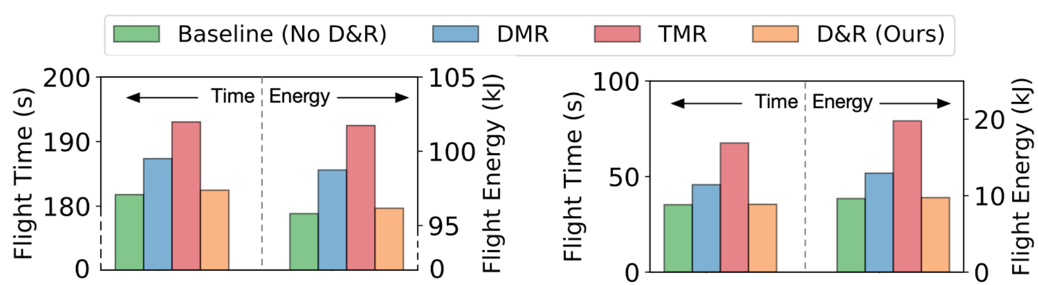




- The compute overhead of anomaly detection and recovery is negligible compared to redundancy-based scheme



[S. Krishnan, **Z. Wan**, K. Bhardwaj, et al., CAL'20]



(a) AirSim Drone

(b) DJI Spark.

(the lower, the better)

C-BRIC

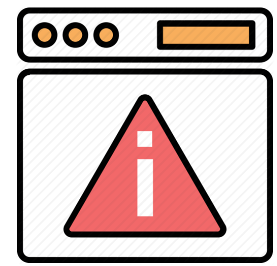
Part 2 Summary

MAVFI: An End-to-End Fault Analysis Framework with Anomaly Detection and Recovery for Micro Aerial Vehicles

The **safety and reliability** of end-to-end **kernel-based navigation systems** is important, but not well understood

A **fault injection tool-chain** that emulates hardware faults and enables rapid fault analysis of kernel-based navigation systems

Large-scale **fault injection study** in different kernels of kernel-based systems against hardware faults



Low-overhead **fault detection and recovery techniques** to enable autonomy robustness

References

- Part 1
  - **Zishen Wan**, Aqeel Anwar, Yu-Shun Hsiao, Tianyu Jia, Vijay Janapa Reddi, Arijit Raychowhury, “Analyzing and Improving Fault Tolerance of Learning-Based Navigation Systems”, to appear in *58th IEEE/ACM Design Automation Conference (DAC)*, 2021.

- Part 2
  - Yu-Shun Hsiao*, **Zishen Wan***, Tianyu Jia, Radhika Ghosal, Arijit Raychowhury, David Brooks, Gu-Yeon Wei, Vijay Janapa Reddi, “MAVFI: An End-to-End Fault Analysis Framework with Anomaly Detection and Recovery for Micro Aerial Vehicles”, *arXiv preprint arXiv: 2105.12882*, 2021. *(*equal contribution)*

C-BRIC

Thank You
Any Questions?

Email: zishenwan@gatech.edu
More info at website: https://zishenwan.github.io

This work was supported in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA