Blind Spot in Sensor Fusion

Exploiting an Architectural Vulnerability to Hijack and Precisely Control UAVs

Jongsoo Han, Seulbae Kim
Pohang University of Science and Technology (POSTECH)

CompSec Lab

Unmanned Aerial Vehicle (UAV)s Are No Longer Toys

Military

Operation Spider's Web
(Ukraine, 2025)

Agriculture

Crop sprayers
(DJI Agras, ArduPilot)

Logistics

Urban delivery UAVs
(Google, Amazon)

UAV failures have real-world consequences.
UAV security is now mission-critical.

Img src: By Ssu.gov.ua, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=166885127

Sensor Attacks on UAVs

Core sensors of UAVs

GPS

Measures position & velocity

Magnetometer

Measures heading (magnetic North)

IMU (Gyro + Accel)

Measures angular rate & acceleration

Attack!

Crash / Destabilize UAVs **+** **Targeted steering** higher-value

Our Goal!

Two Requirements for Targeted Steering

Two Requirements for Targeted Steering

For the gyroscope sensor…

Two Requirements for Targeted Steering

For the gyroscope sensor…

For targeted steering,
a spoofing strategy is necessary.

Studied
(Acoustic, Laser, etc.)

Not yet studied

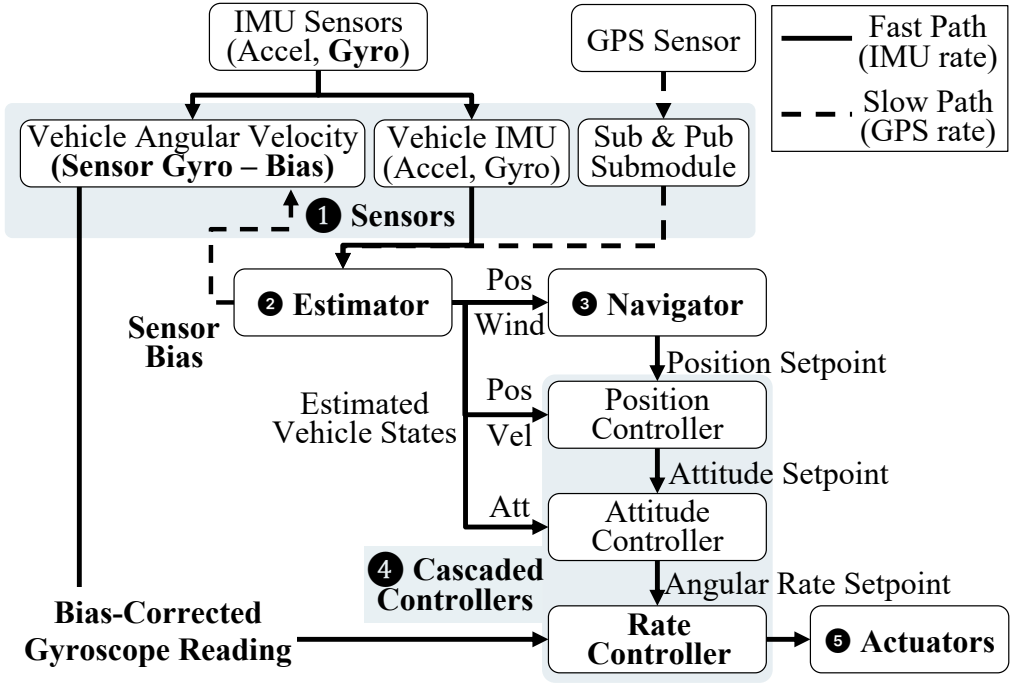
The Control Pipeline of PX4 and ArduPilot

PX4 and ArduPilot

- Two major open-source UAV autopilots.
- Share the same architectural pattern.

Modules and Control Flow

1. Sensors
2. Estimator (Sensor Fusion)
3. Navigator
4. Cascaded Controllers

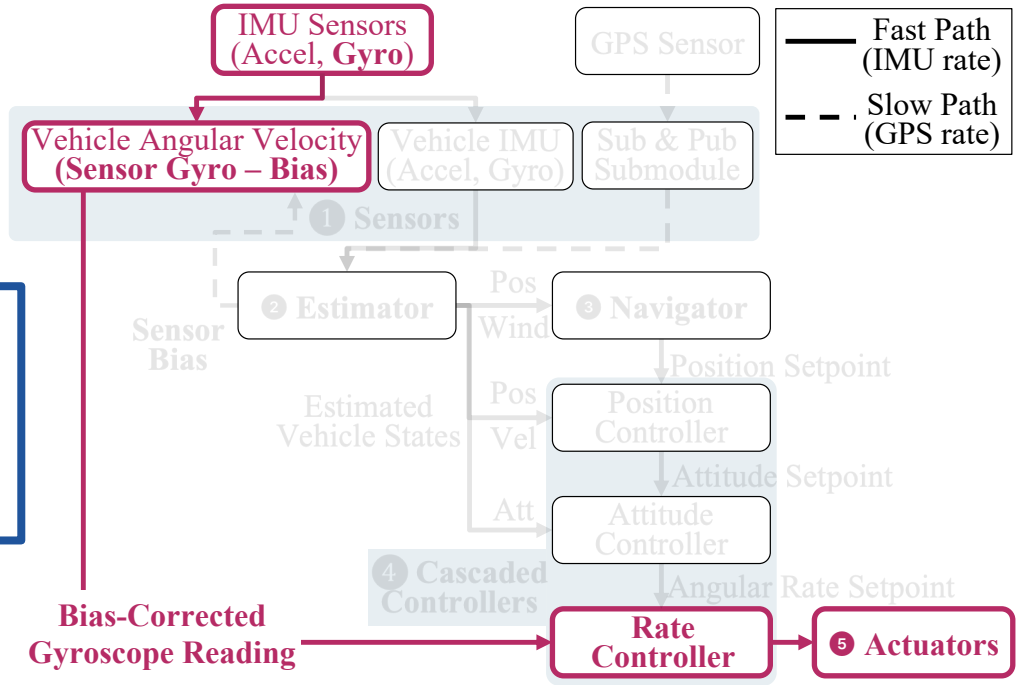


Control pipeline shared by PX4 and ArduPilot.

The Control Pipeline of PX4 and ArduPilot

Findings & Hypothesis V1:

Gyroscope readings are directly used to calculate motor commands



Control pipeline shared by PX4 and ArduPilot.

Correction of Gyroscope in Flight

Sensor fusion

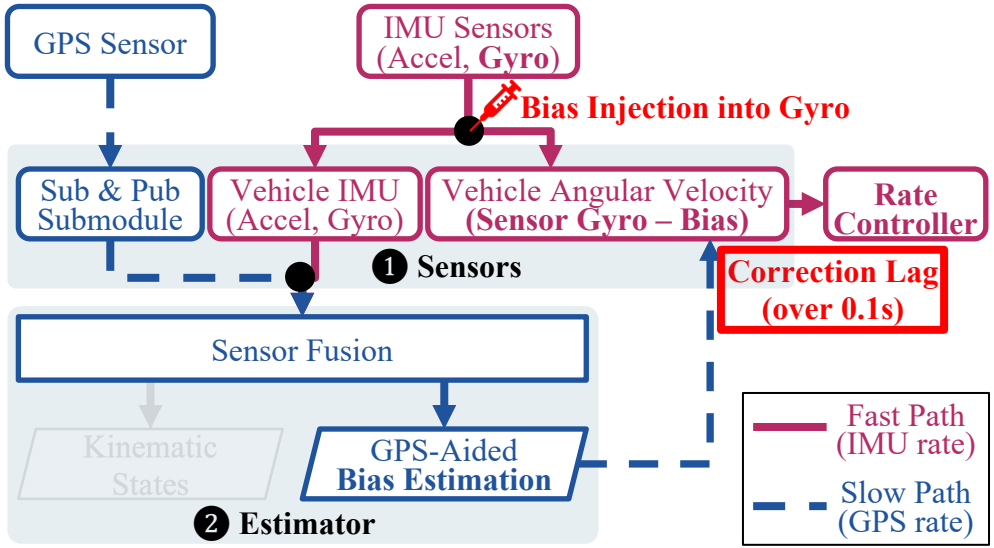
- Correct sensor errors
- **Depends on GPS (5-10 Hz)**

Gyroscope (200 Hz)

- **Unique angular velocity sensor**

Gyroscope correction

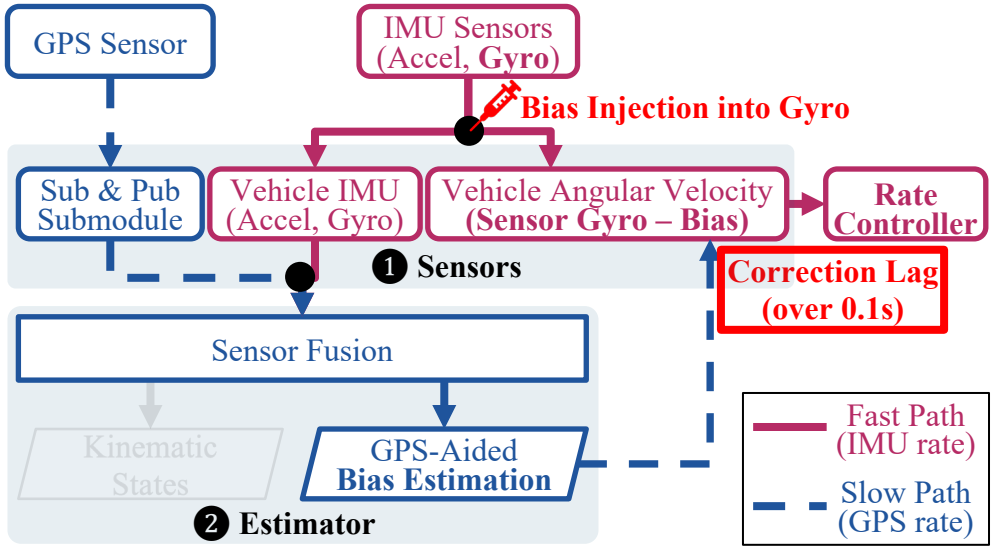
- Error propagation
  - Ang_Vel → Att → Vel → Pos
- Pos ↔ GPS **indirect comparison**
- 10 Hz vs. 200 Hz → **0.1 s lag**



Gyroscope correction lag.

Correction of Gyroscope in Flight

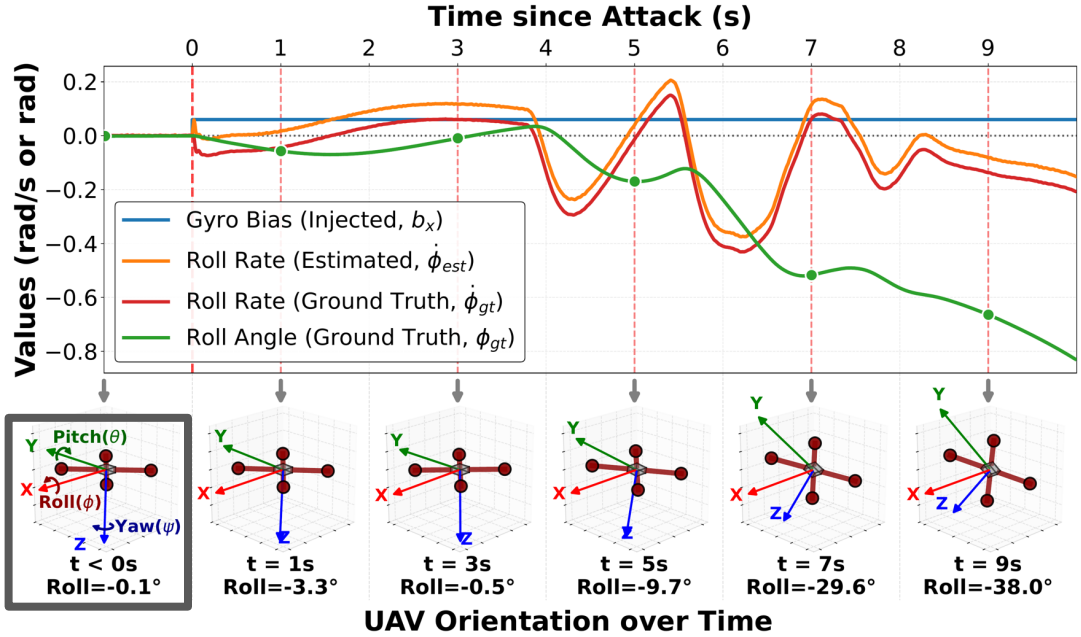
Findings & Hypothesis V2, V3:
Gyroscope correction is delayed (V2) and indirect (V3)



Gyroscope correction lag.

V1 - Immediate Physical Reaction

Initial/Target state: HOLD
Maintain a stable hover.



Rate controller's immediate reaction to gyroscope bias.

V1 - Immediate Physical Reaction

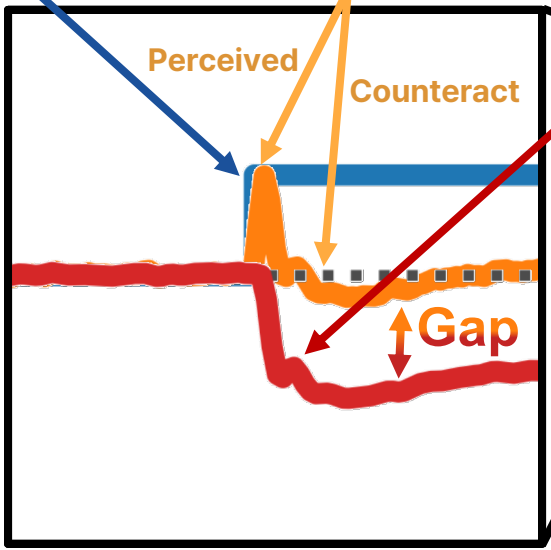
Believing now stabilized.
In fact, ➡

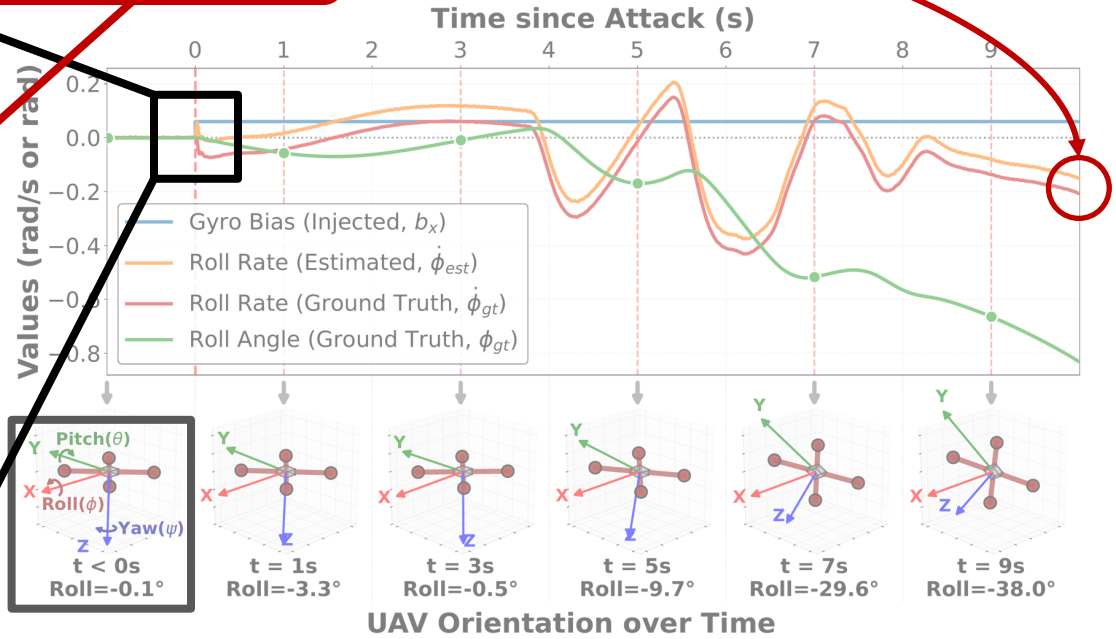
Spoof on x-axis by 0.06 rad/s

Rate controller counteraction

Ground truth: Rotating in -0.06 rad/s

Correction delay!





Initial/Target state: HOLD
Maintain a stable hover.

Rate controller's immediate reaction to gyroscope bias.

V2·V3 - Delayed & Indirect Correction

Accelerometer (200 Hz)

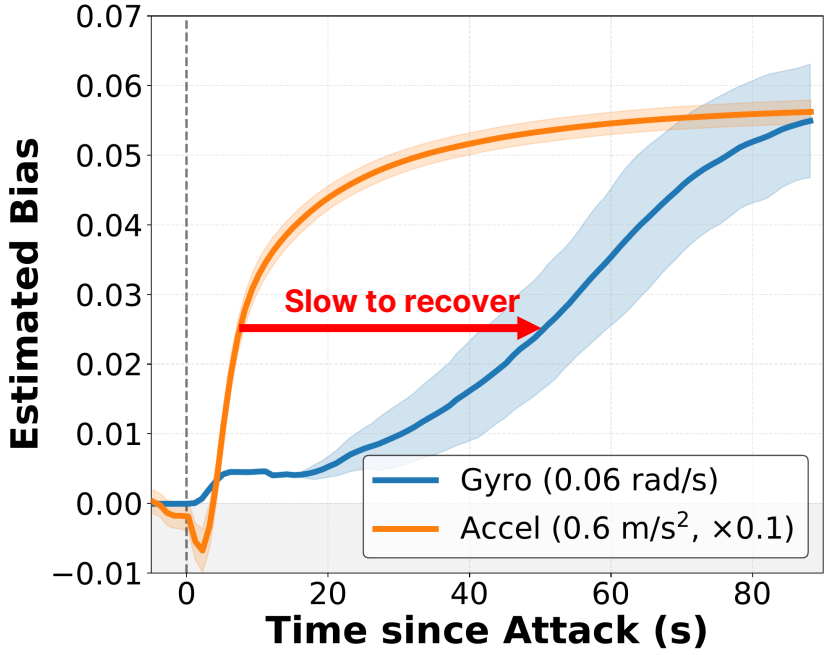
- Measures acceleration
- Direct comparison to GPS
- $a_y \cong 0.6\ m/s^2$ ($\omega_x = 0.06$ rad/s applied for 1s)

Gyroscope vs. Accelerometer bias

Detection
- **17.9×** slower
- Gyro: 2.46 s
- Accel: 0.14 s

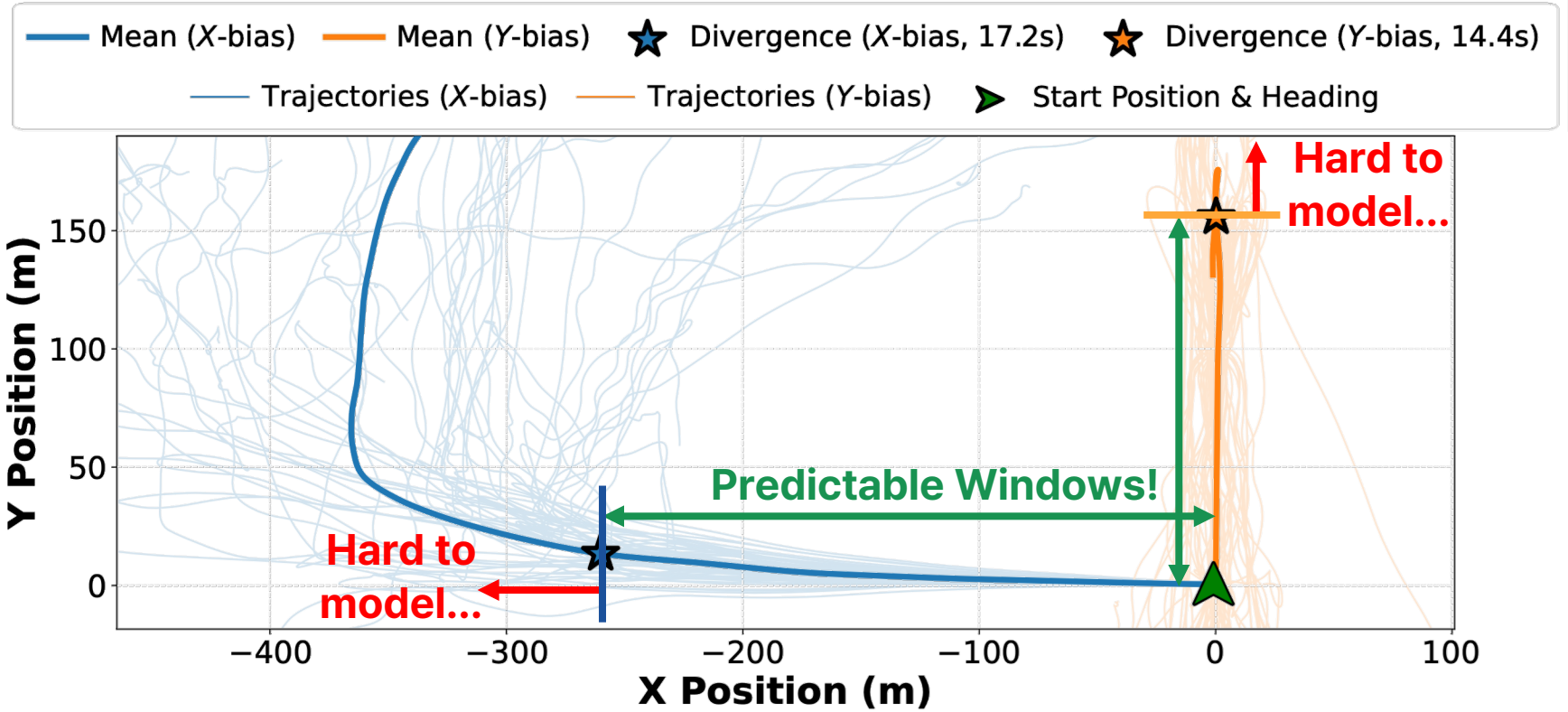
Correction
- **16.3×** longer
- Gyro: 49 s
- Accel: 3 s

Recover stabilized hover



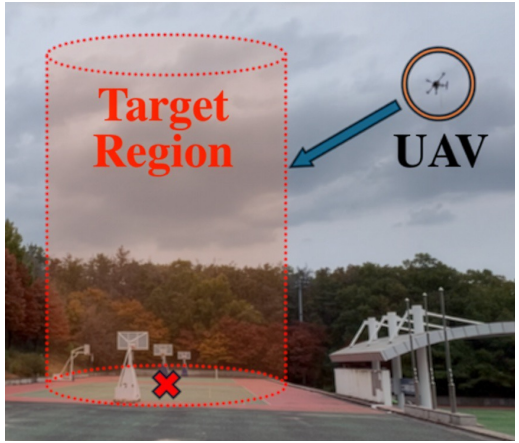
Bias estimation process.

A Predictable Attack Window



UAV trajectories observed across 50 bias injection trials.

Threat Model



Black-Box Adversary

- ✓ **Inject controlled gyro bias (X/Y)** at 1 Hz
- ✓ Observe **external state** (Pos, Att, etc.)
- ✗ Can't read internal states (fusion, controllers)

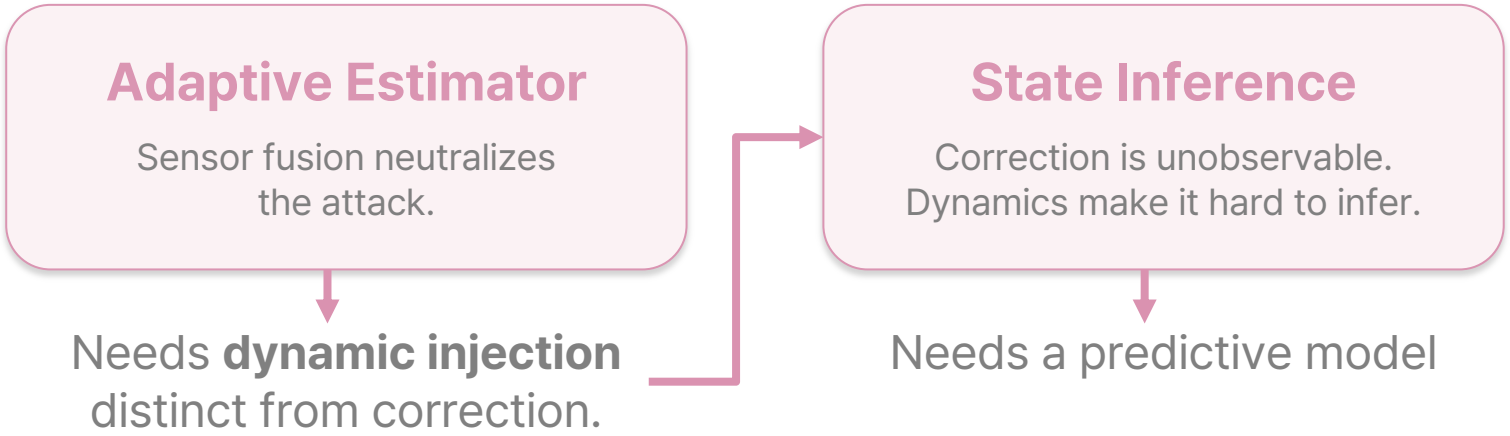
Target

- UAV with **core sensors** (GPS, IMU, MAG)

Goal

- Steer UAV into **attacker-chosen region**
- Reach the goal **within a 10 m radius**

Two Challenges for the Strategy

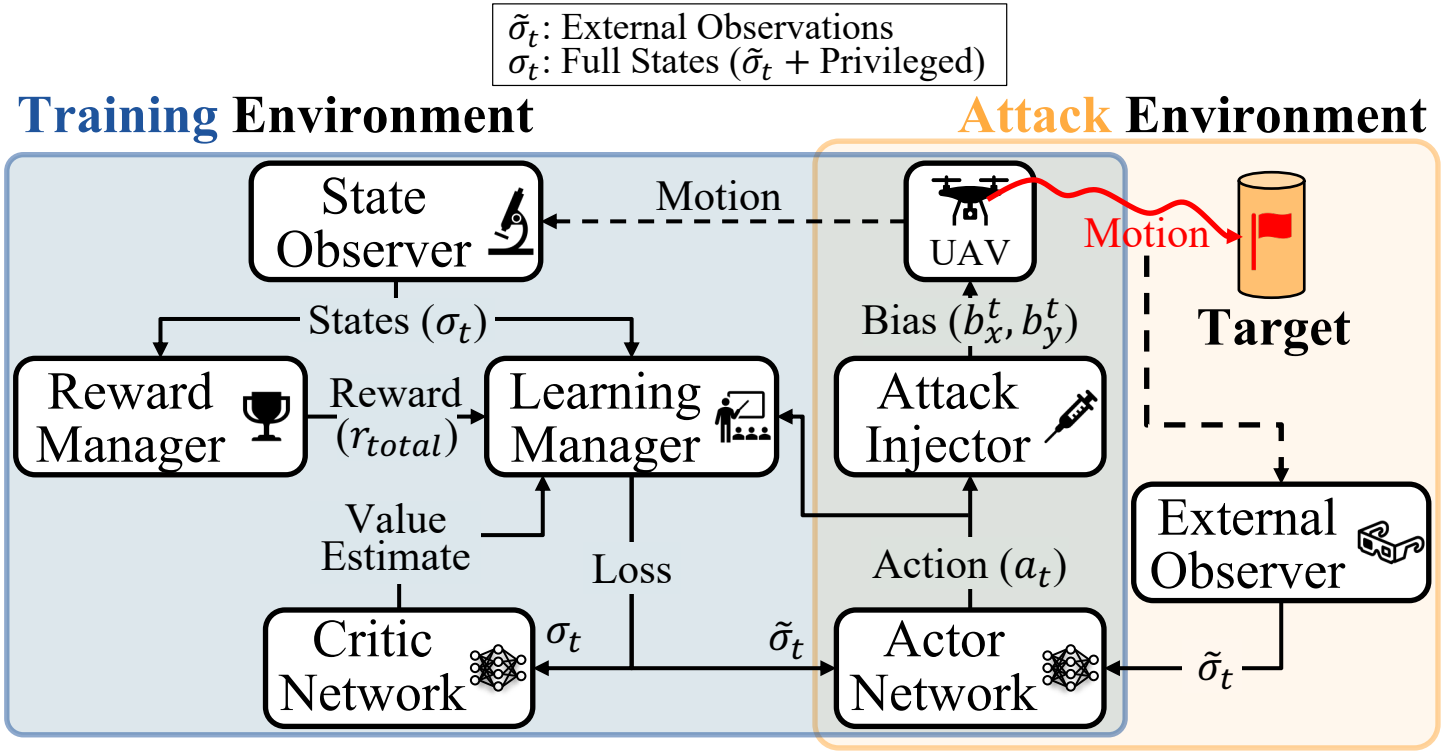


Solution: **Deep Reinforcement Learning**

Learns closed-loop bias control automatically

GAP: Gyroscope Attack Policy

Design of Gyroscope Attack Policy (GAP)



Design and workflow of GAP.

Training in Simulation

Initial setup

- Hover at altitude of 60 m
- Random spawn position along a circle

Success condition

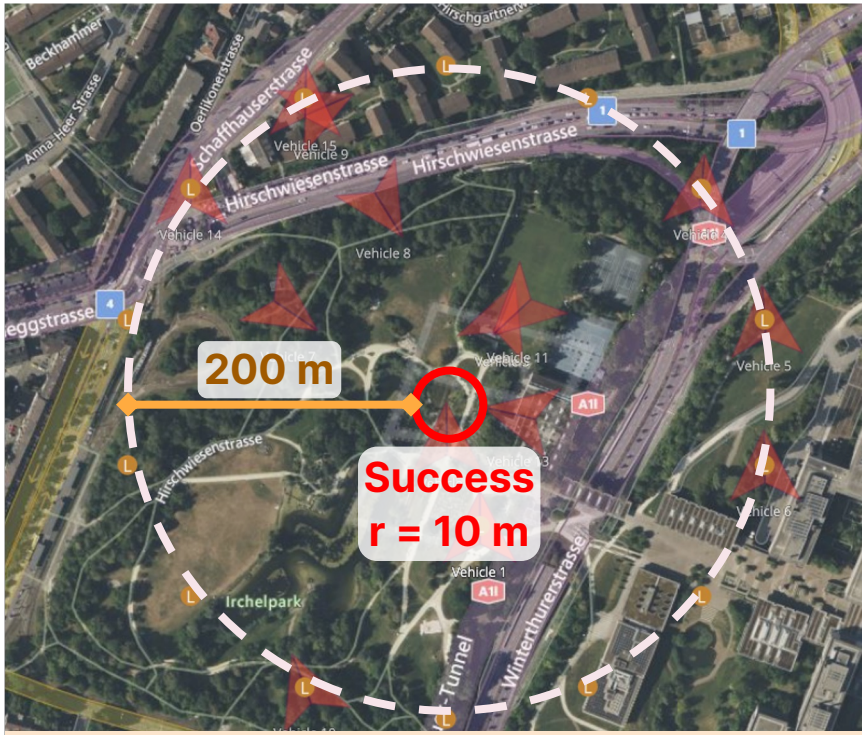
- Reach 10 m radius cylinder target region

Actions

- $bias \in [-0.06, 0.06]\ rad/s$
- updated every 1 s

Reward

- $r_{\text{dist}} + r_{\text{vel}} + r_{\text{time}} + r_{\text{termination}}$



Goal: Attack and make the UAV reach the success circle

Evaluation Overview

PERFORMANCE

- **RQ1:** Can GAP steer better than baselines?

ROBUSTNESS & TRANSFER

- **RQ2:** Is it robust to observation and injection noise?
- **RQ3:** Does it transfer across platforms?

DEFENSES & REAL-WORLD

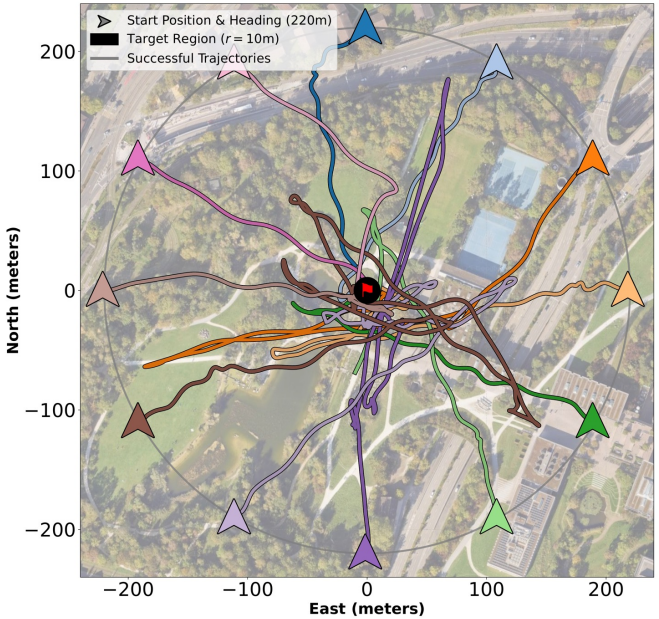
- **RQ4 & RQ5:** How does GAP cope with existing defenses?
- **RQ6:** Does it work on real UAVs?

RQ1 - GAP vs. Baselines

Cyl: Cylinder target, Sph: Sphere target

Attack	Cyl 20	Sph 20	Cyl 10	Sph 10
Random	0.0	0.0	0.0	0.0
Directional	4.2	4.2	0.0	0.0
Adaptive	41.7	29.2	20.8	8.3
GAP	**93.2**	**85.0**	**85.1**	**59.1**

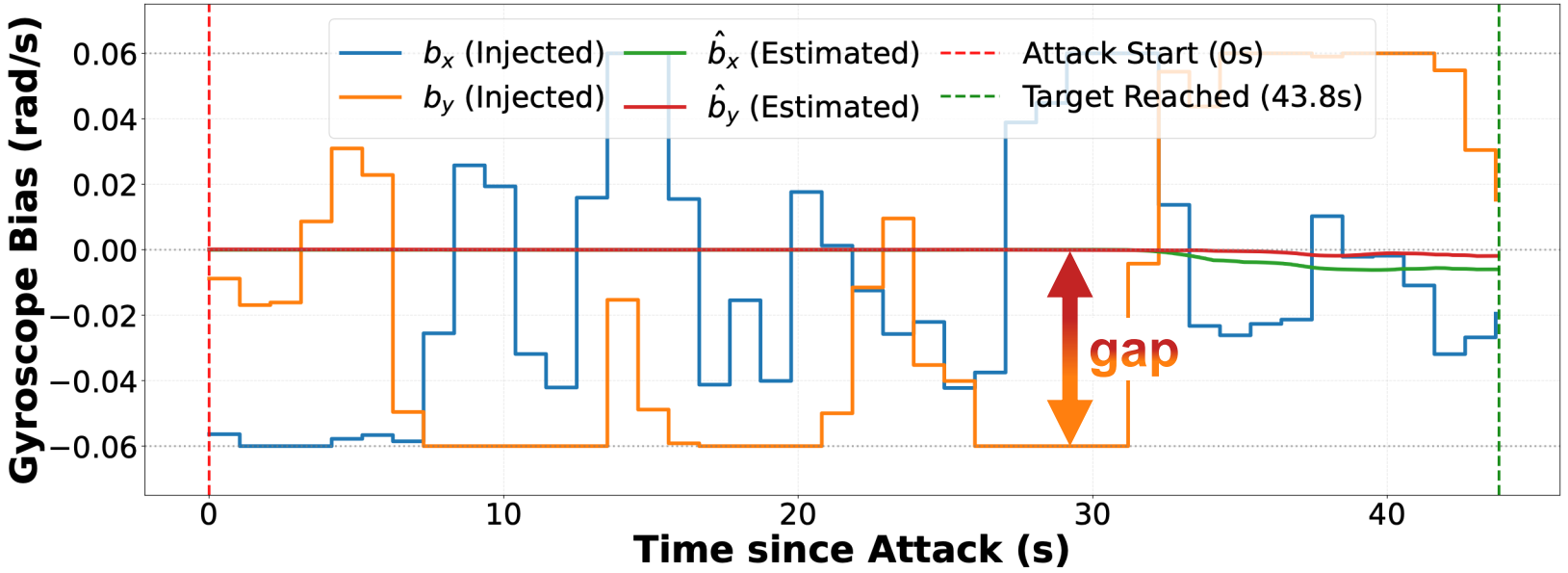
Success rate (%) by attack and target geometry.



Successful trajectories

GAP outperformed all the baselines, reaching the target at least 64% faster

Anatomy of GAP Steering



Bias injection sequence analysis

GAP dynamically modulates bias to sustain the gap, evading neutralization

RQ2 - Noise Robustness

Condition	Cyl 10 m Success
None	85.1 %
Tracking Noise	**89.2 %**
Delay & Loss	**87.3 %**
Both	**87.2 %**

1200 episodes evaluated for each condition

Two noise sources:

- **External observation noise**
  - $\sigma_p = 0.295m$, $\sigma_\theta = 1.033°$
- **Injection delay/loss**
  - 12.5% command loss, period jitter

GAP successfully steered even under noises

RQ3 - Transferability

Platform	Airframe	Cyl 10 m (%)
PX4	x500 (Gazebo)	83.3
ArduPilot	octaquad	**87.5**
ArduPilot	singlecopter	83.3
ArduPilot	hexa	79.2
ArduPilot	quad	79.2
ArduPilot	tri	79.2
ArduPilot	y6	79.2
ArduPilot	octa	75.0
ArduPilot	dodeca-hexa	70.8
ArduPilot	coaxcopter	54.2

24 episodes evaluated for each airframe and platform

A single policy generalizes

- Across autopilots (PX4 → ArduPilot)
- Multiple airframes

GAP exploits a shared architectural pattern

RQ4 & 5 - Detection by Existing Defenses

Failsafe detector

Emergency responses to abnormal flight state.

0% Gyro fail, 20.3% other detection

Control Invariant detector

Monitors physics-based invariants.

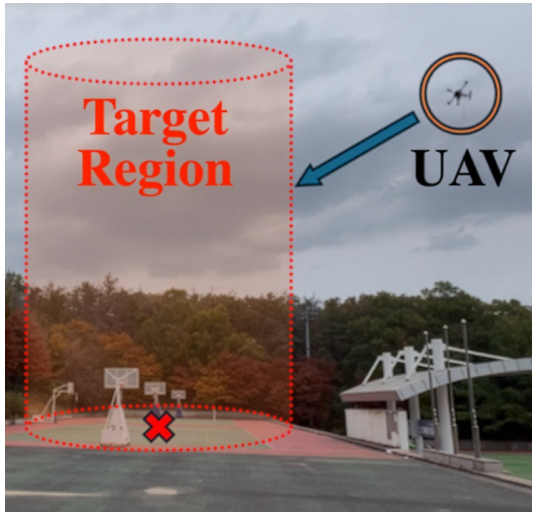
24% detection

Neither method reliably detected the attack in our experiments.

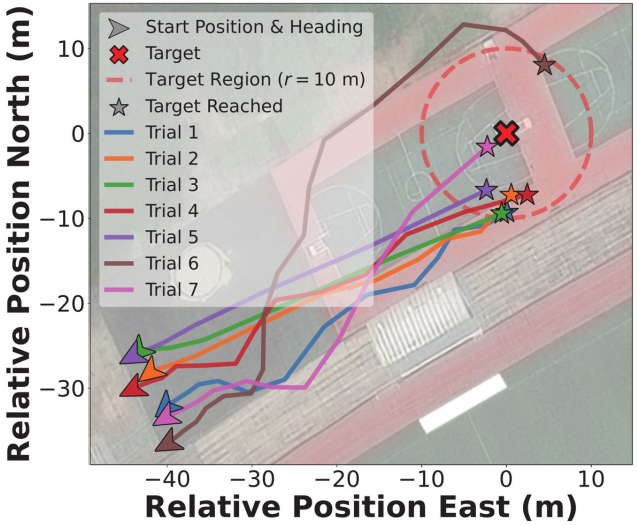
RQ5 - Real-World

Experimental setup

- **Airframes**
  - Holybro X500, S500
- **Target**
  - 50 m away
  - 10 m radius cylinder
- **Emulated**
  - Spoofing, observation
- **Wind**
  - 1.7-4.4 m/s



Field experiment setup.

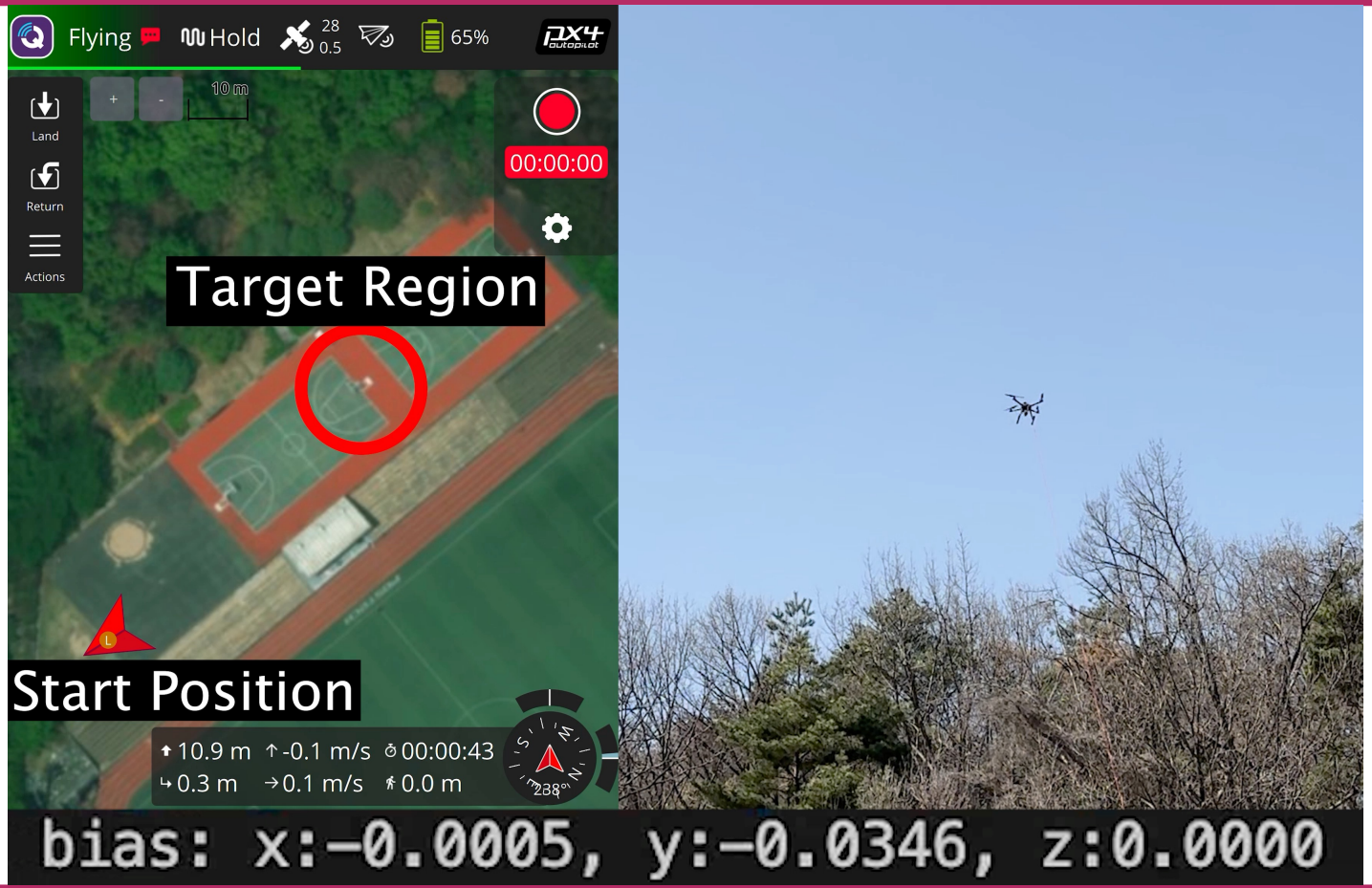


Real-world trajectories

Results

Trials successful:	Network jitter:	Failsafe resilience:
7 / 7	12.5% command loss	never triggered

Evaluation - Real-World Demo



Summary

1. Architectural vulnerability analysis

- Immediate reaction, sensor fusion lag

2. RL-based black-box Gyroscope Attack Policy (GAP)

- Asymmetric actor-critic, 85% success, 10 m radius precision, defense evasion

3. Generalization + Real-world validation

- Transfers across platforms/airframes, 7 real-world flights succeeded

Q & A

Contact: Jongsoo Han
- hanjs@postech.ac.kr
- https://jsoone24.github.io