From Control Model to Program: Investigating Robotic Aerial Vehicle Accidents with MAYDAY

Taegyu Kim[1], Chung Hwan Kim[2], Altay Ozen[1], Fan Fei[1], Zhan Tu[1], Xiangyu Zhang[1], Xinyan Deng[1], Dave (Jing) Tian[1], Dongyan Xu[1]

[1]Purdue University
[2]UT Dallas

Drone (Robotic Aerial Vehicle) Accidents

Drone crashes into Virginia bull run crowd

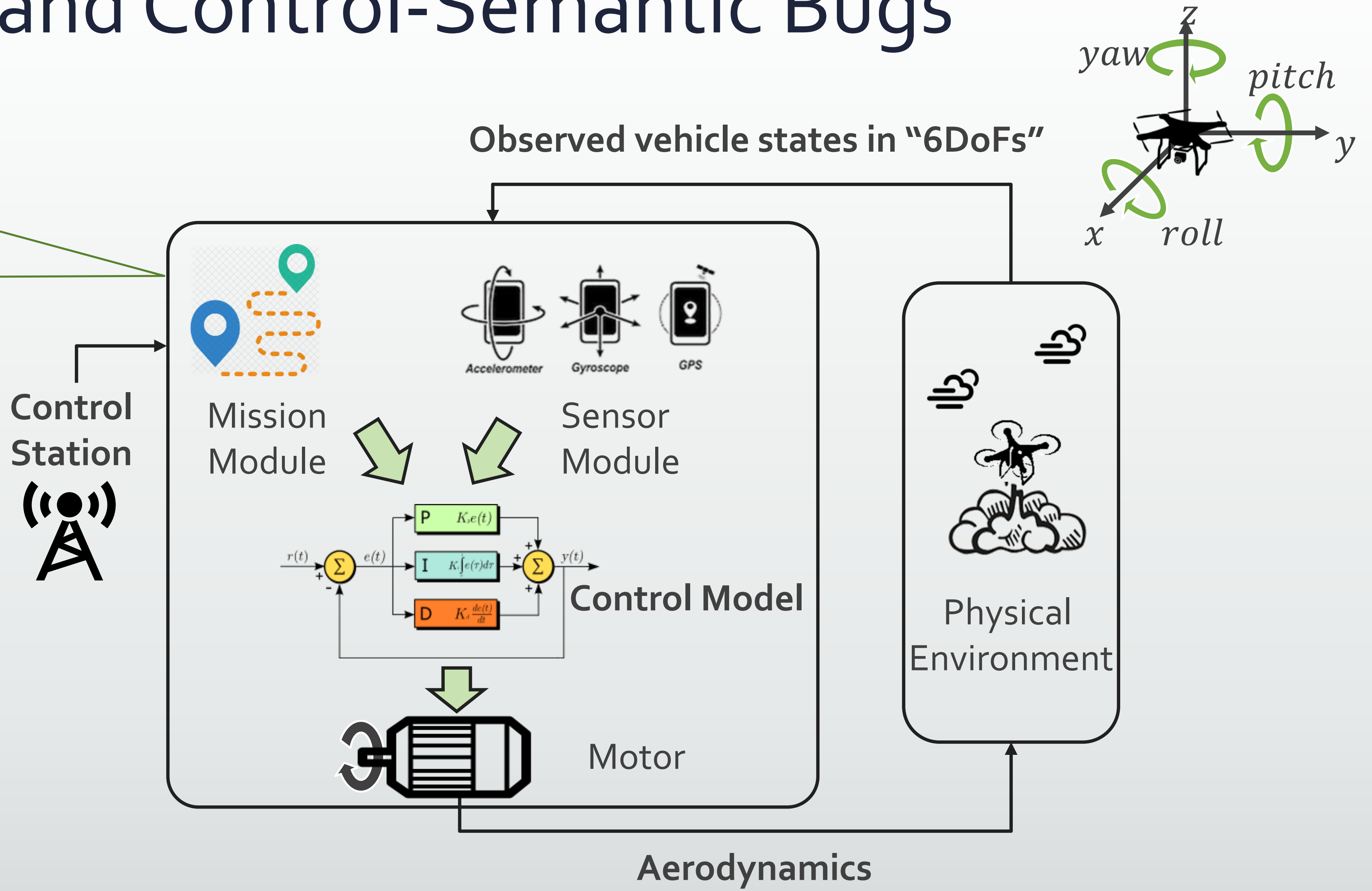
A drone crashed into the grandstand at Virginia Motorsports Park during Saturday's Great Bull Run.

Drone That Crashed at White House Was Quadcopter

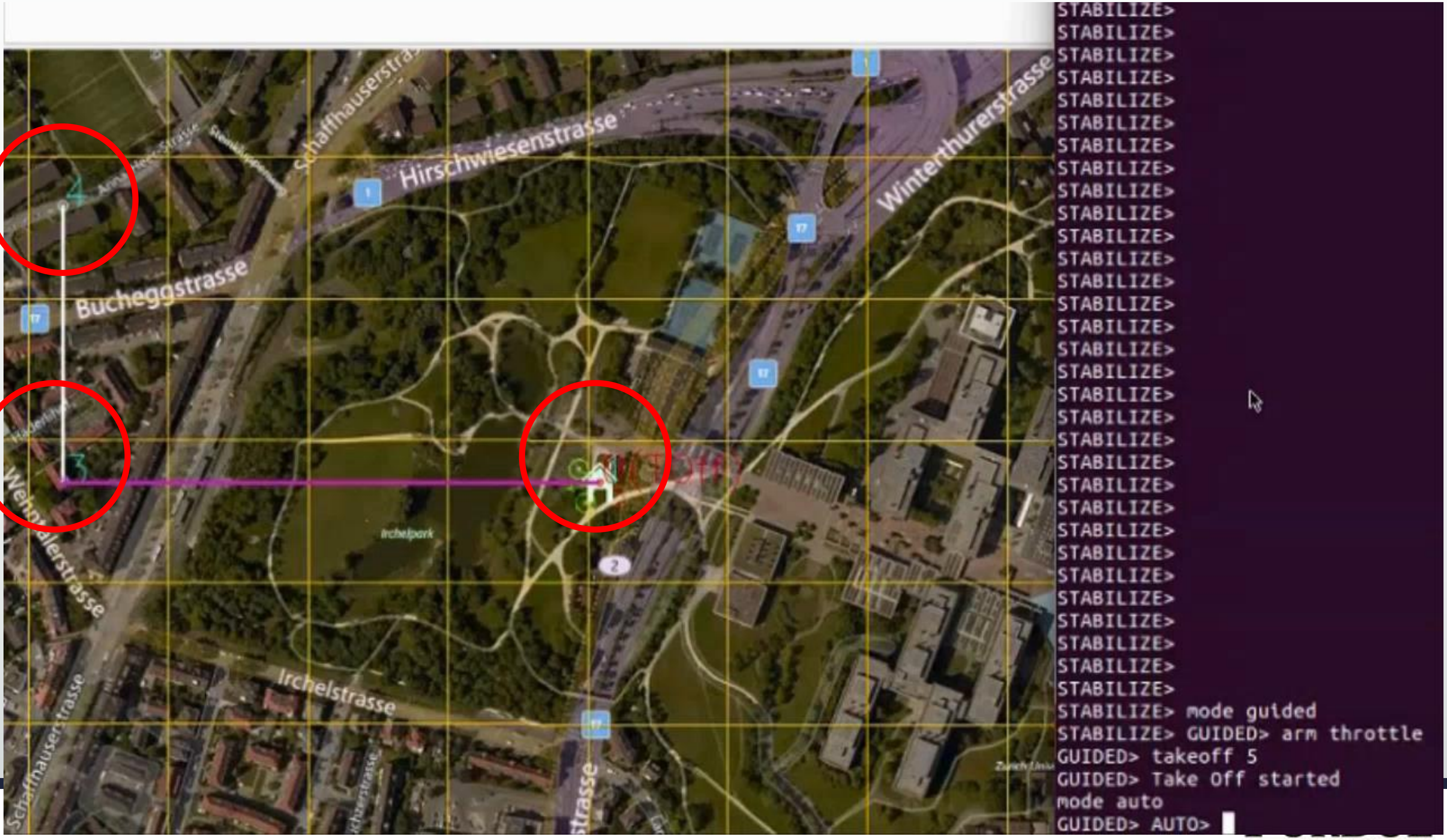
RAV Control and Control-Semantic Bugs

Control-Semantic Bug

- Accident root cause inside control program
- Incorrect or incomplete implementation of control model



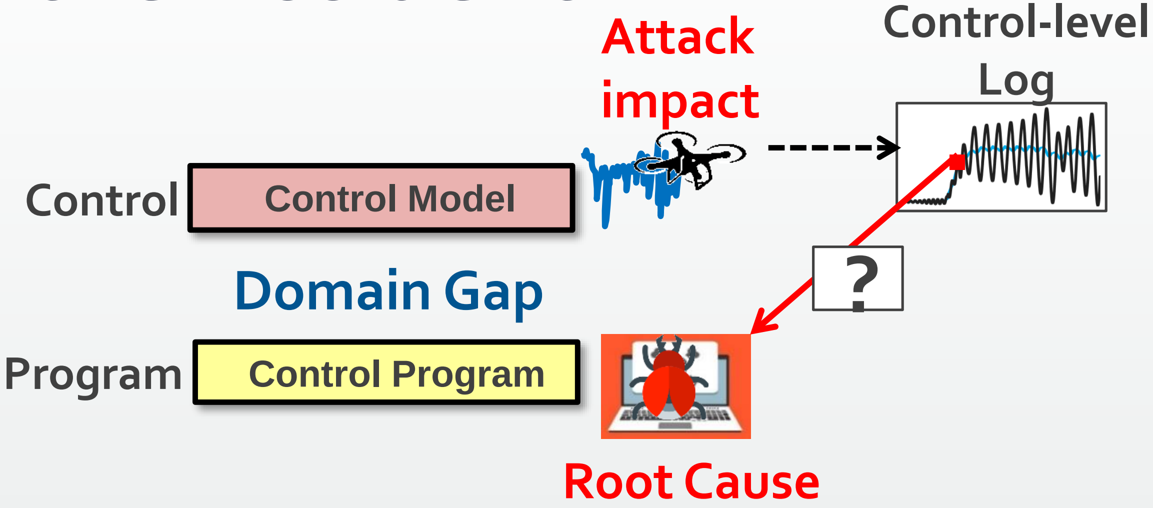
A Motivating Accident

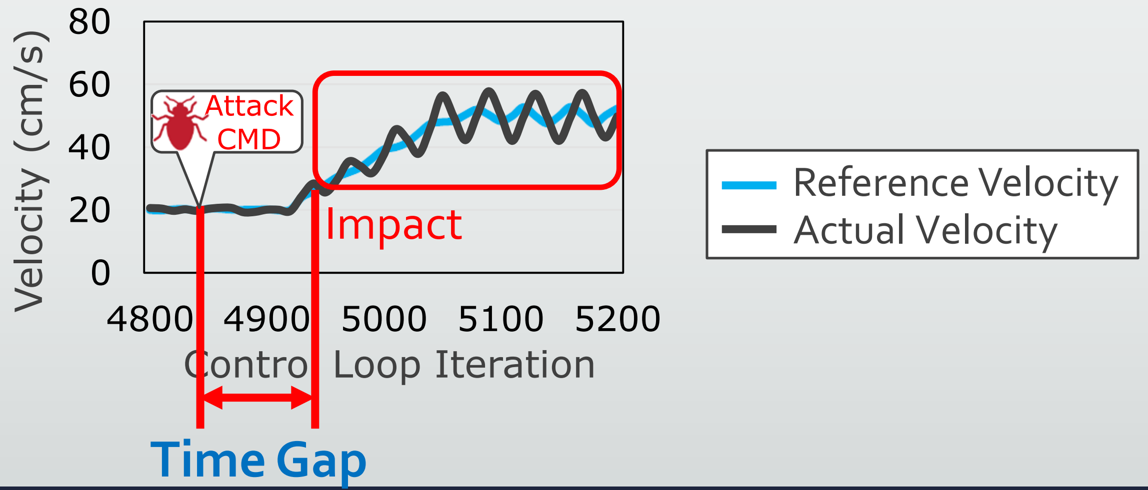


Challenges in Investigating the Accident

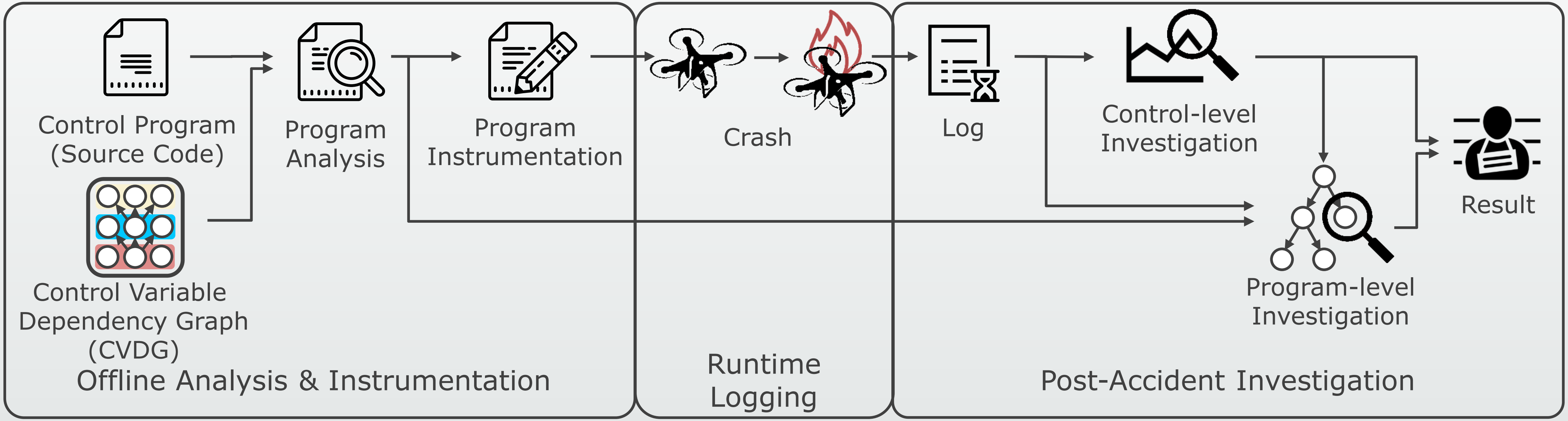
- "Two Gaps"
  - Domain gap
    - **Control** domain → **Program** domain
  - Time gap
    - **Attack** time → **Impact** time

- Our solution: MAYDAY
  - Bridge the gaps
  - Enable cross-domain investigation

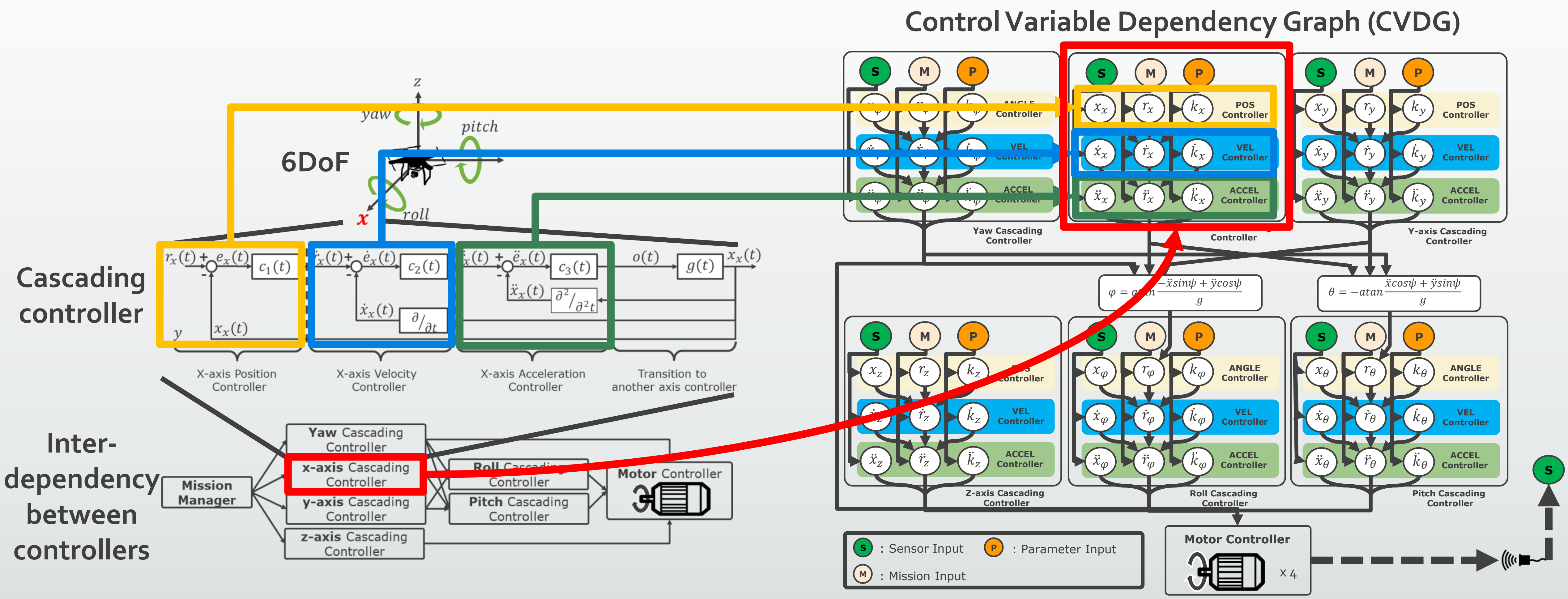




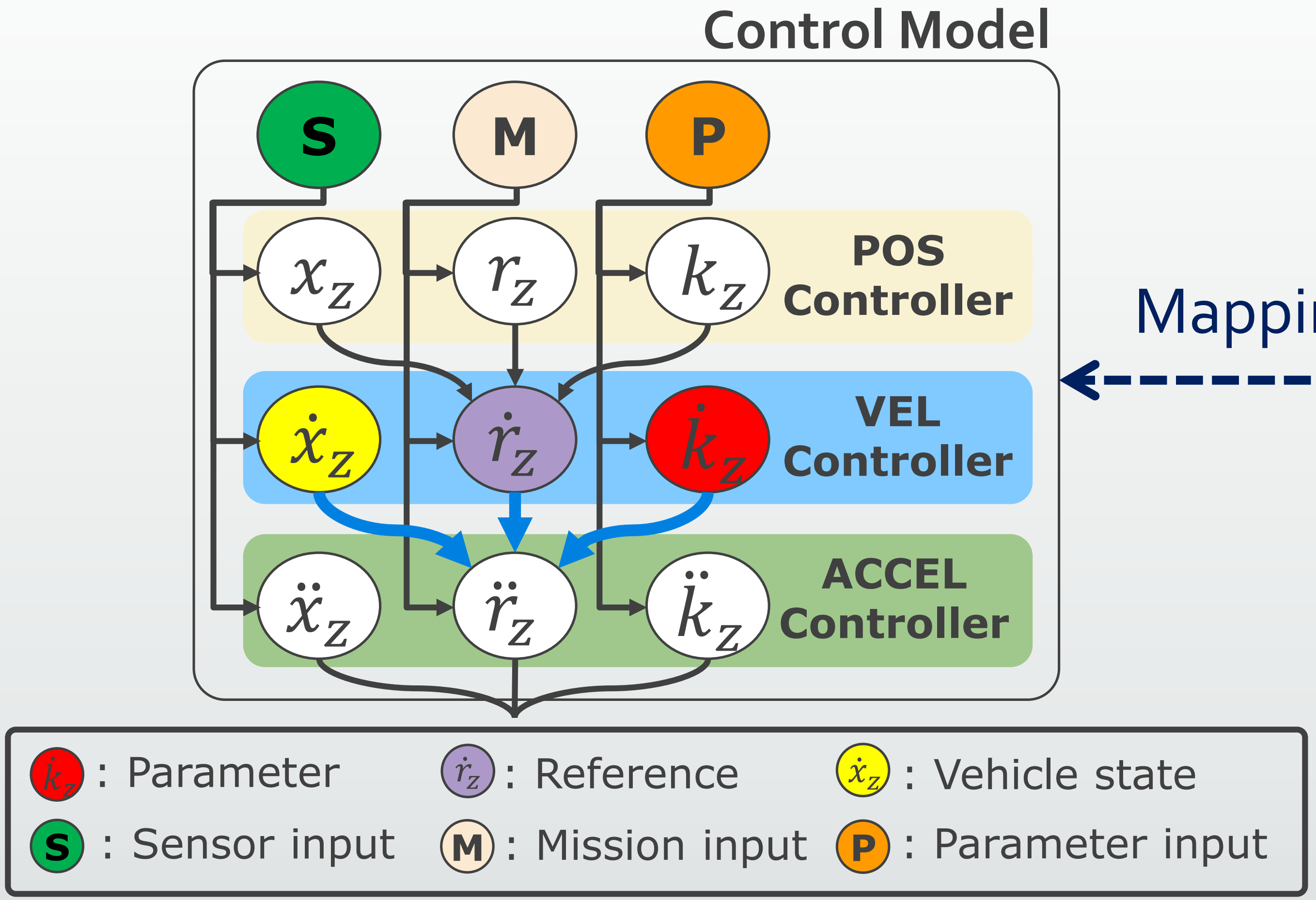
MAYDAY Workflow



RAV Control Model



Mapping Control Model to Control Program



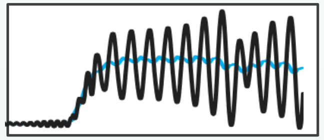
Mapping

Control Program

```
void AC_PosControl::rate_to_accel_z(
   ...
   vel_err.z = vel_target.z - cur_vel.z
   p = _p_velz._kP() * vel_err.z;
   accel_target.z = accel_ff.z + p ;
   ...
```

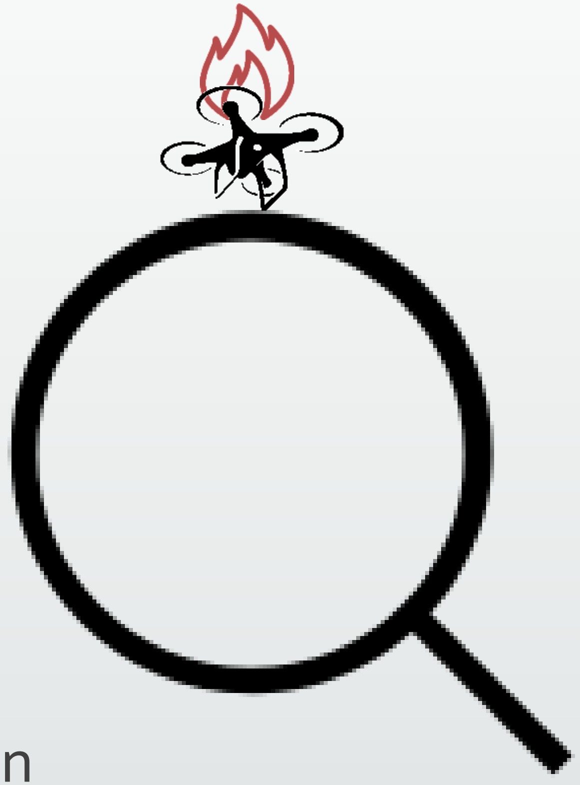
- Control model variable → *Control program variable*
- Control model data flow → *Control program execution paths*

Logging Enhancement

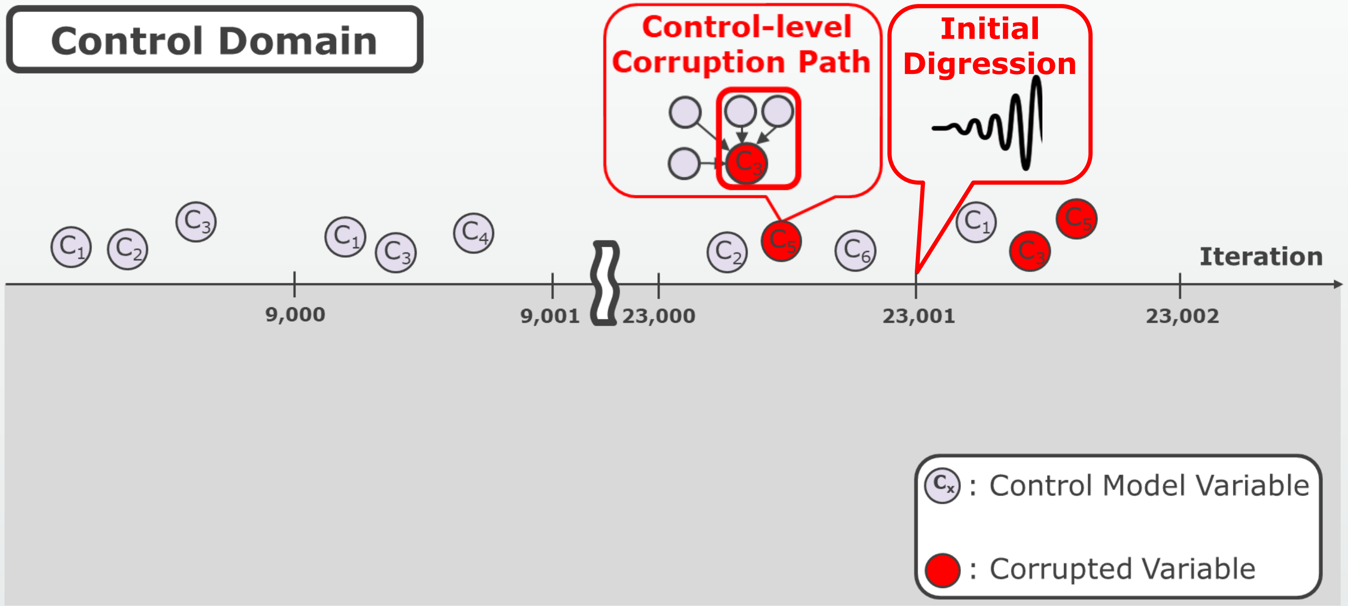
- Control/vehicle operation log 
  - Recorded by default
    - Supported by major drone control programs
    - Recorded by *control-level logging functions*
- Program execution log

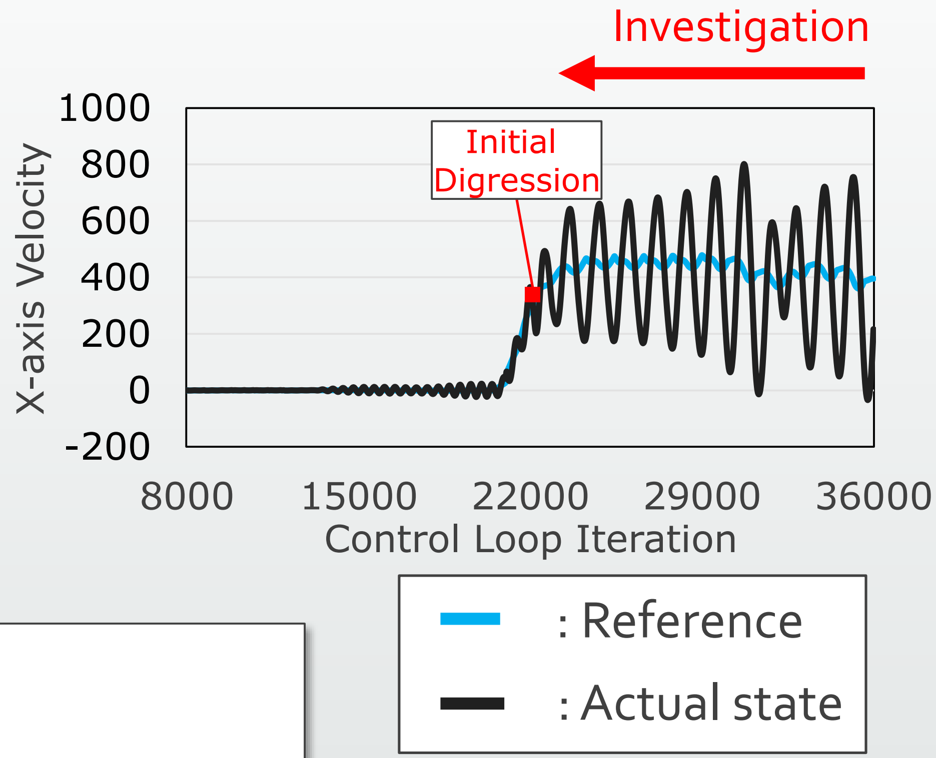
```
If      err.z -= cur.z;
else  err.z = 0.0;
 p = kP* err.z;
```

  - Enabled by MAYDAY
    - Logging functions inserted via LLVM-level instrumentation
    - Guided by mapping between control model and program



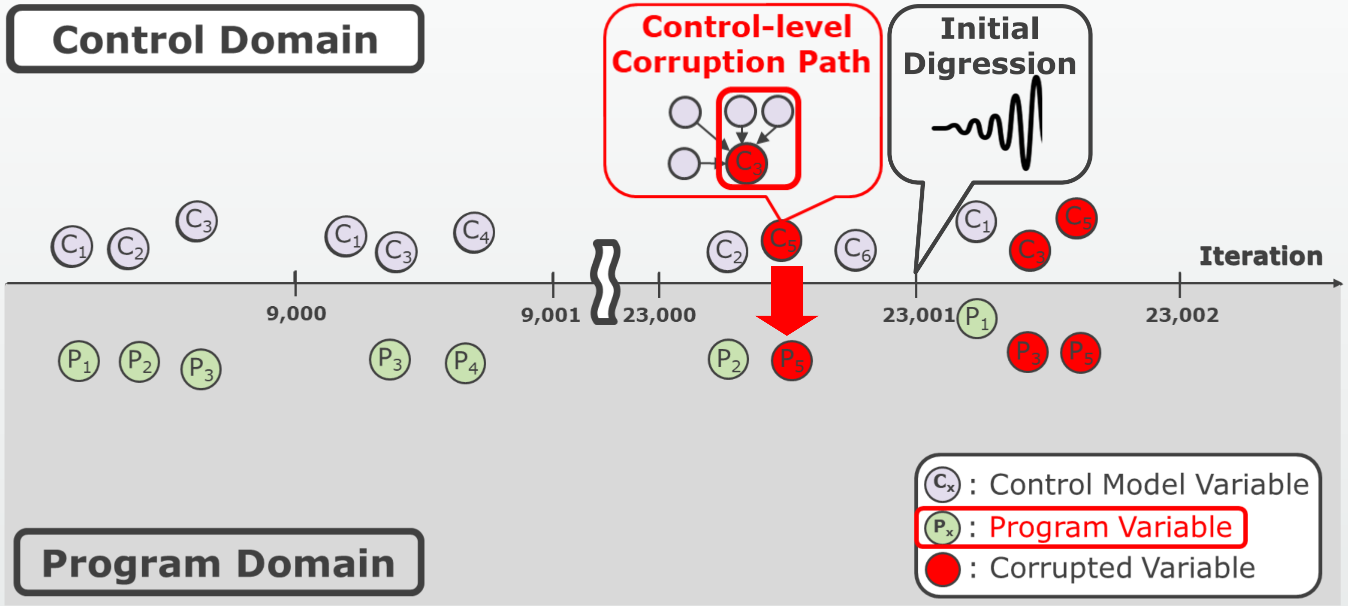
Control-Level Investigation

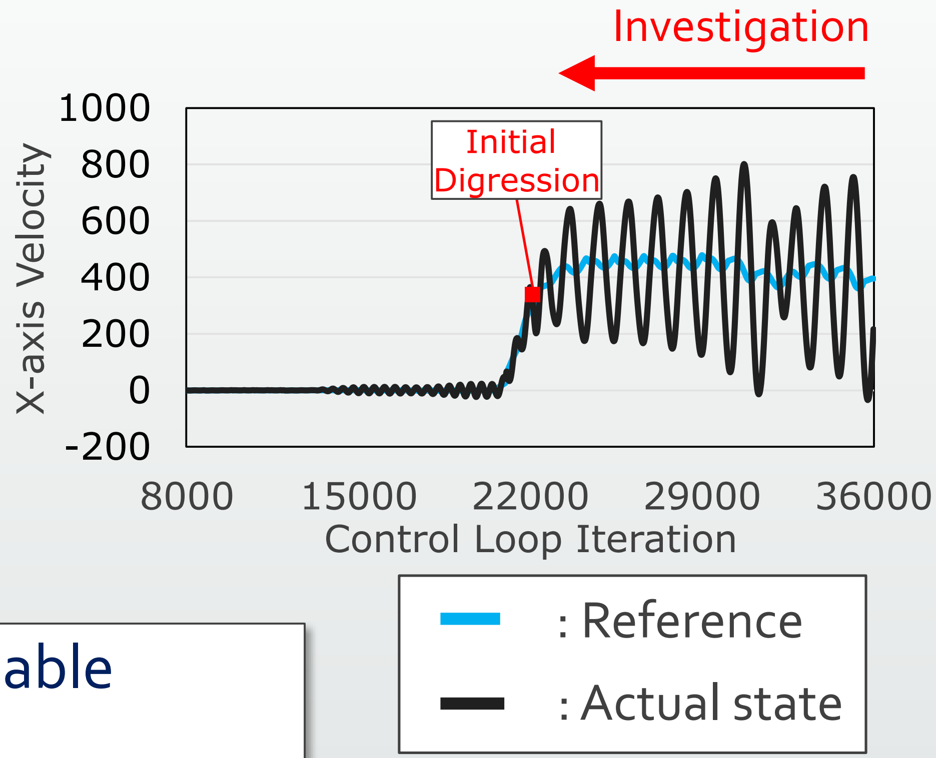




- Identify initial digressing controller
  - [Controller, corrupted variable, initial digression time]
- Infer control-level corruption path based on CVDG

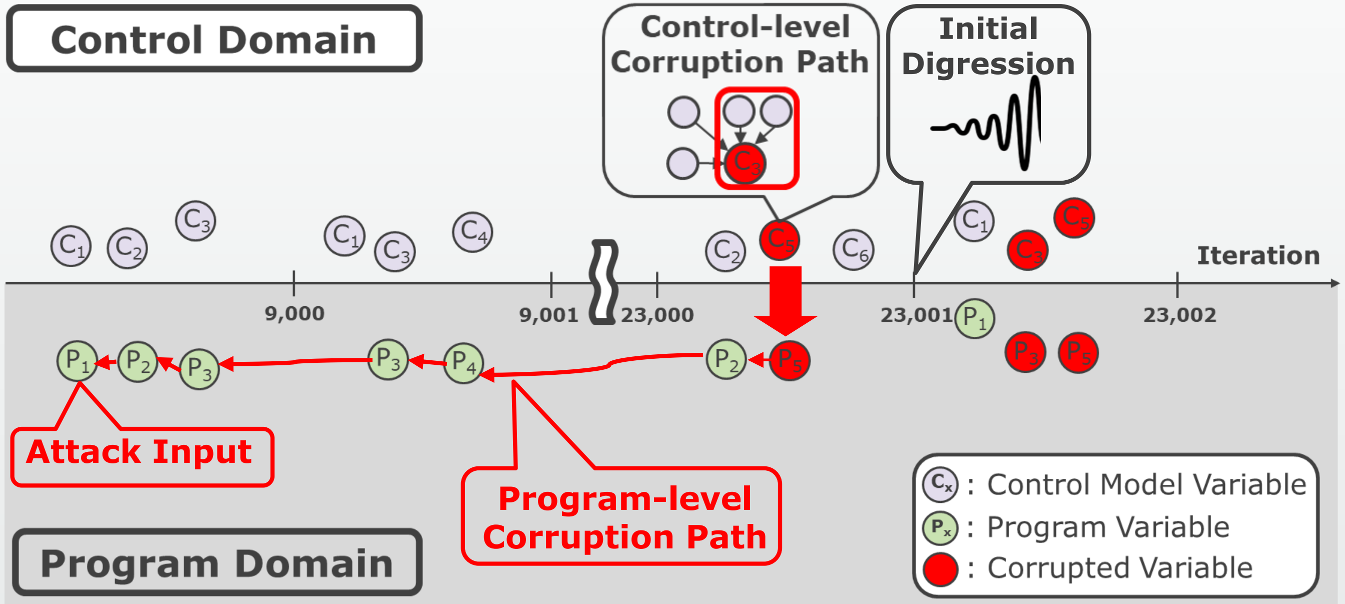
Moving from Control Domain to Program Domain

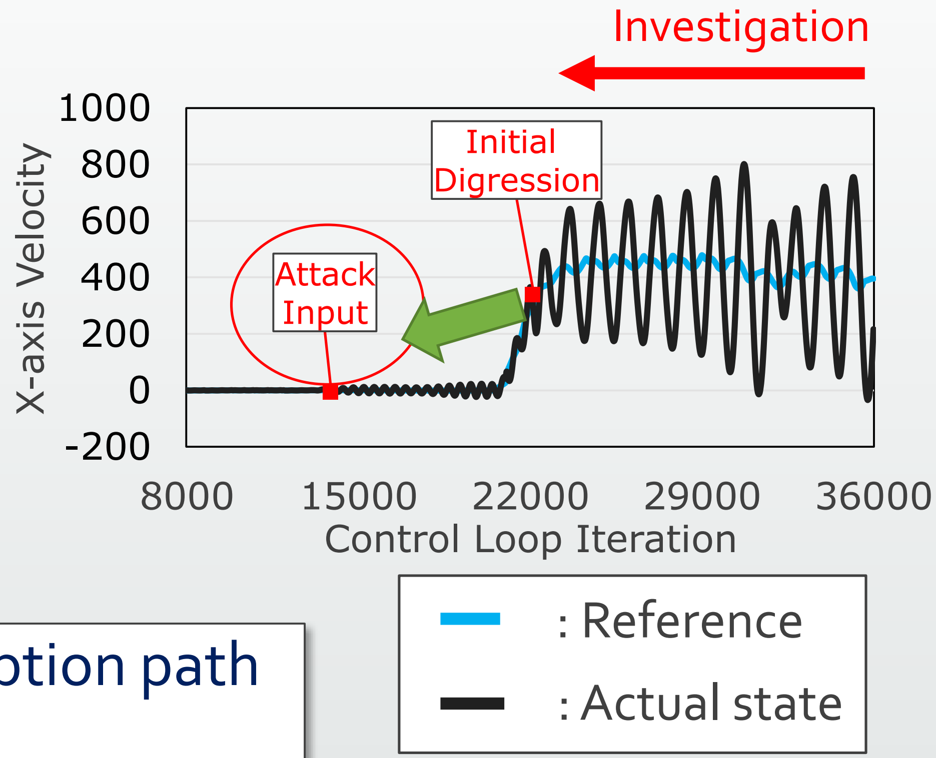




- Corrupted control variable → Corrupted program variable

Program-Level Investigation





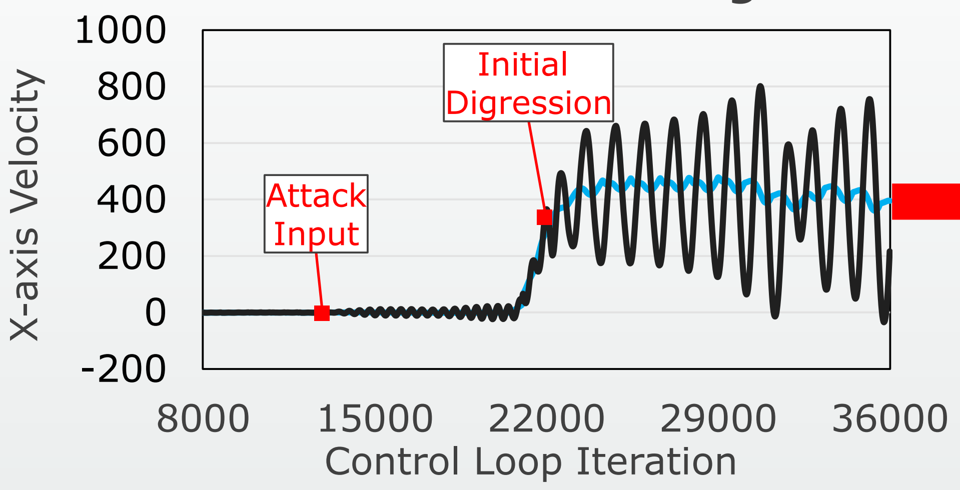
- Control-level corruption path → Program-level corruption path
  - From initial digression to attack input
  - Bug localized in basic blocks that implement the corruption path

Evaluation: Effectiveness of MAYDAY

Case ID	Control-Level Investigation			Program-Level Investigation		
	Initial Digressing Controller	CVDG-Level Corruption Path	# of Iterations from Initial Corruption to Initial Digression	# of Basic Blocks	SLoC	Bug Found?
1	x, y-axis Velocity	$P \rightarrow \dot{k}_{xy} \rightarrow \ddot{r}_{xy}$	x ($x \geq 4$)	34	89	✓
2	z-axis Velocity	$P \rightarrow \dot{k}_{z} \rightarrow \ddot{r}_{z}$	x ($x \geq 4$)	32	85	✓
3	Roll Angle	$P \rightarrow k_{roll} \rightarrow \dot{r}_{roll}$	x ($x \geq 4$)	50	121	✓
4	Pitch Angle	$P \rightarrow k_{pitch} \rightarrow \dot{r}_{pitch}$	x ($x \geq 4$)	50	121	✓
5	x, y-axis Velocity	$M \rightarrow \dot{r}_{xy}$	x ($x \geq 4$)	12	44	✓
6	x, y-axis Position	$M \rightarrow r_{xy}$	x ($x \geq 4$)	48	137	✓
7	z-axis Position	$M \rightarrow r_{z}$	x ($x \geq 4$)	48	135	✓
8	z-axis Position	$P \rightarrow k_{z} \rightarrow \dot{r}_{z}$	4	9	30	✓
9	x, y-axis Position	$P \rightarrow k_{xy} \rightarrow \dot{r}_{xy}$	4	41	94	✓
10	Roll, Pitch, Yaw Angle	$S \rightarrow x_{rpy} \rightarrow \dot{r}_{rpy}$	1	7	22	✓

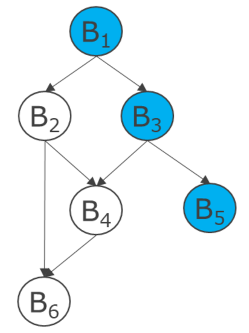
Evaluation: Solving the Earlier Case

Control-Level Log



- **Initial digressing controller:** X, Y-axis velocity controller
- **Corrupted control variable:** X, Y-axis acceleration reference
- **Control-level corruption path:** $P \rightarrow \dot{k}_{xy} \rightarrow \ddot{r}_{xy}$

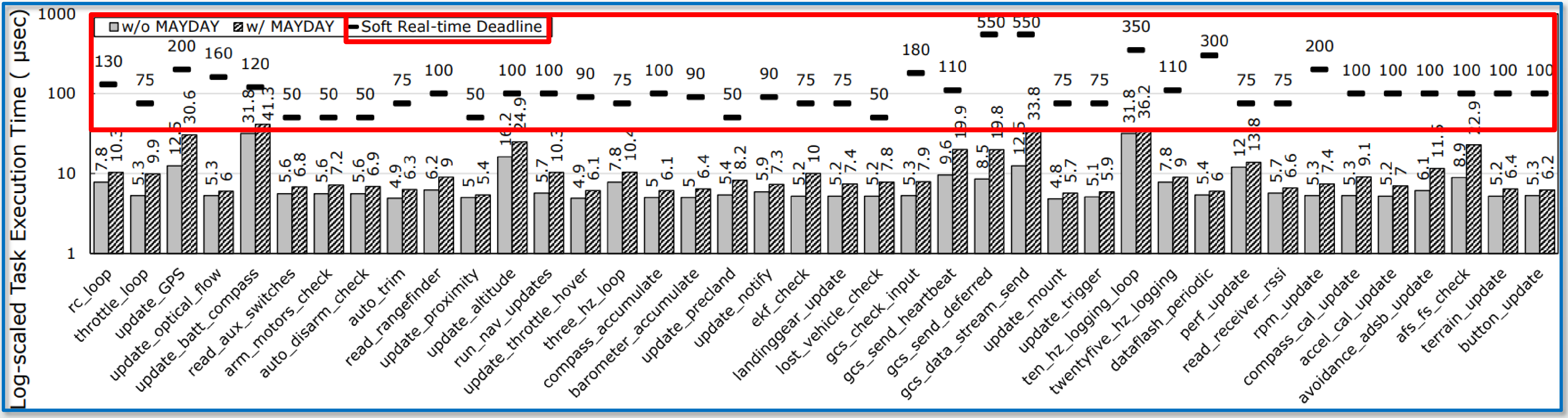
Program-Level Log



- **Attack input:**
  Control gain k_P
- **Number of BBs on corruption path:** 34
- **Source LoC:** 89

```
void GCS_MAVLINK::handle_param_set(..//Parameter update
   ...
   //No range check
   vp->set_float(packet.param_value, var_type);
Vector2f AC_PI_2D::get_p() const{
   ...
   return (_input * _kp)           //No range check
void AC_PosControl::rate_to_accel_xy(... //controller
   ...
   //Access parameter _kp
   vel_xy_p = _pi_vel_xy.get_p(); //No range check
```

Evaluation: Runtime Overhead of MAYDAY



Conclusion

- Drone accident may be caused by control semantic bugs
- Control-level logs alone are not sufficient for bug-tracing
- **MAYDAY**: a cross-domain accident investigation tool
  - Bridging the domain gap and the time gap
  - Mapping control model to control program
  - Integrating control-level and program-level logging
  - Connecting control-level and program-level investigation

Thank you!

This work was supported in part by ONR Grant #N00014-17-1-2045.

tgkim@purdue.edu