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

Taegyu Kim¹, Chung Hwan Kim², Altay Ozen¹, Fan Fei¹, Zhan Tu¹, Xiangyu Zhang¹, Xinyan Deng¹, Dave (Jing) Tian¹, Dongyan Xu¹

¹Purdue University
²UT 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 Program

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

Sensor Module
- Observed vehicle states in “6DoFs”

Mission Module
- Control Model
- Aerodynamics

Motor

Physical Environment

Control Station
A Motivating Accident
Challenges in Investigating the Accident

• “Two Gaps”
  • Domain gap
    • Control domain $\rightarrow$ Program domain
  • Time gap
    • Attack time $\rightarrow$ Impact time

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

Control Program (Source Code) → Program Analysis → Program Instrumentation → Crash → Log → Control-level Investigation → Result

Offline Analysis & Instrumentation

Control Variable Dependency Graph (CVDG)
RAV Control Model

Control Variable Dependency Graph (CVDG)

Inter-dependency between controllers

Cascading controller

RAW CONTROL MODEL

6DoF

Mission Manager

Motor Controller

x-axis Position Controller

X-axis Velocity Controller

X-axis Acceleration Controller

Transition to another axis controller

Pitch Cascading Controller

Roll Cascading Controller

Yaw Cascading Controller

x-axis Cascading Controller

y-axis Cascading Controller

z-axis Cascading Controller

\[ \varphi = \arctan \frac{\dot{y}\sin \theta + \dot{z}\cos \theta}{\dot{z}\sin \theta - \dot{y}\cos \theta} \]

\[ \theta = -\arctan \frac{\dot{x}\cos \varphi + \dot{y}\sin \varphi}{\dot{y}\cos \varphi - \dot{x}\sin \varphi} \]

\[ \psi = \arctan \frac{\dot{y}\sin \varphi + \dot{z}\cos \varphi}{\dot{z}\sin \varphi - \dot{y}\cos \varphi} \]
Mapping Control Model to Control Program

**Control Model**

- **S**: Sensor input
- **M**: Mission input
- **P**: Parameter input
- **param**: Parameter
- **ref**: Reference
- **state**: Vehicle state

**Control Program**

```c
void AC_PosControl::rate_to_accel_z(

  vel_err.z = vel_target.z - curr_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
  • Enabled by MAYDAY
    • Logging functions inserted via LLVM-level instrumentation
    • Guided by mapping between control model and program

If err.z = cur.z;
else err.z = 0.0;
p = kP*err.z;
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 $\rightarrow$ 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

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Initial Digressing Controller</th>
<th>CVDG-Level Corruption Path</th>
<th># of Iterations from Initial Corruption to Initial Digression</th>
<th>Program-Level Investigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x, y-axis Velocity</td>
<td>$P \rightarrow k_{xy} \rightarrow \dot{r}_{xy}$</td>
<td>$x \geq 4$</td>
<td>34, 89, ✓</td>
</tr>
<tr>
<td>2</td>
<td>z-axis Velocity</td>
<td>$P \rightarrow k_z \rightarrow \dot{r}_z$</td>
<td>$x \geq 4$</td>
<td>32, 85, ✓</td>
</tr>
<tr>
<td>3</td>
<td>Roll Angle</td>
<td>$P \rightarrow k_{roll} \rightarrow \dot{r}_{roll}$</td>
<td>$x \geq 4$</td>
<td>50, 121, ✓</td>
</tr>
<tr>
<td>4</td>
<td>Pitch Angle</td>
<td>$P \rightarrow k_{pitch} \rightarrow \dot{r}_{pitch}$</td>
<td>$x \geq 4$</td>
<td>50, 121, ✓</td>
</tr>
<tr>
<td>5</td>
<td>x, y-axis Velocity</td>
<td>$M \rightarrow \dot{r}_{xy}$</td>
<td>$x \geq 4$</td>
<td>12, 44, ✓</td>
</tr>
<tr>
<td>6</td>
<td>x, y-axis Position</td>
<td>$M \rightarrow r_{xy}$</td>
<td>$x \geq 4$</td>
<td>48, 137, ✓</td>
</tr>
<tr>
<td>7</td>
<td>z-axis Position</td>
<td>$M \rightarrow r_z$</td>
<td>$x \geq 4$</td>
<td>48, 135, ✓</td>
</tr>
<tr>
<td>8</td>
<td>z-axis Position</td>
<td>$P \rightarrow k_z \rightarrow \dot{r}_z$</td>
<td>4</td>
<td>9, 30, ✓</td>
</tr>
<tr>
<td>9</td>
<td>x, y-axis Position</td>
<td>$P \rightarrow k_{xy} \rightarrow \dot{r}_{xy}$</td>
<td>4</td>
<td>41, 94, ✓</td>
</tr>
<tr>
<td>10</td>
<td>Roll, Pitch, Yaw Angle</td>
<td>$S \rightarrow x_{rpy} \rightarrow \dot{r}_{rpy}$</td>
<td>1</td>
<td>7, 22, ✓</td>
</tr>
</tbody>
</table>
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**: $P \rightarrow \dot{k}_{xy} \rightarrow j_{xy}$

**Program-Level Log**

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