Detecting multi sensor fusion errors in advanced driver assistance systems (ISSTA 2022)

Ziyuan Zhong
ziyuan.zhong@columbia.edu
Columbia University
United States

Zhisheng Hu
zhishenghu@baidu.com
Baidu Security
United States

Shengjian Guo
sjguo@baidu.com
Baidu Security
United States

Xinyang Zhang
xinyangzhang@baidu.com
Baidu Security
United States

Zhenyu Zhong
edwardzhong@baidu.com
Baidu Security
United States

Baishakhi Ray
rayb@cs.columbia.edu
Columbia University
United States

Motivation

- Multi-sensor Fusion (MSF) is widely used in ADS/ADAS to smooth out the uncertainties brought by sensor outputs
- Sometimes MSF results are unreliable since it does not know which sensor output to trust, leading to serious consequences.
- Popular MSF methods in an industry-grade ADAS can mislead the car control and result in serious safety hazards

Contributions

- Provide a definition for *fusion error*
- **FusED**: A novel grey-box fuzzing technique for detecting *fusion error*
- Analyze the causes of the fusion errors using causal analysis
- Source Code: https://github.com/AIasd/FusED

Background

Fusion in Driving Automation

- Fused sensors include camera, radar, lidar
- High Level Fusion: Fuse the high-level object attributes produced by different sensors (most widely used)
- Low Level Fusion: Fuse raw sensor data

Fusion in OpenPilot

- Fuse data from Camera and Radar to detect the leading vehicle, based on which longtitudinal control is decided
- Latitudinal control is decided solely by camera inputs, no fusion is used

- Fusion error can be introduced by faulty logics in the implementation of the fusion algorithm

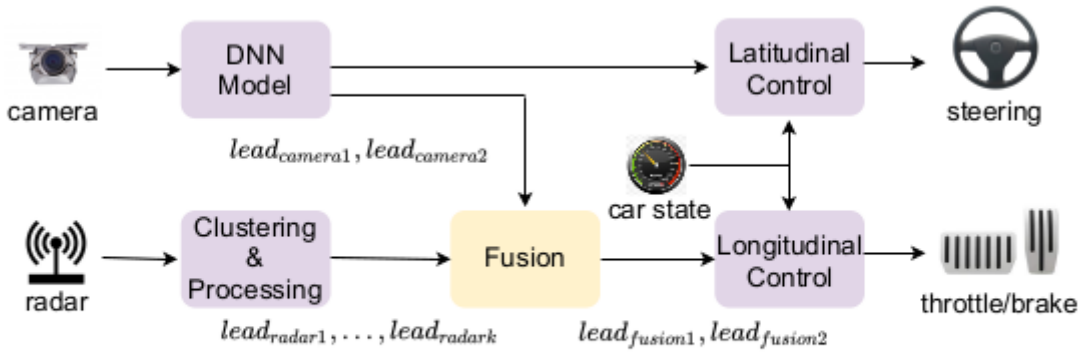

Figure 2: The role of fusion in OPENPILOT.

Challenges

1. Hard to trigger failure (i.e. crash) in commercial ADAS
2. When a failure occurs, it is hard to conclude its root cause is an incorrect fusion logic

Approach

Fusion Fault:
$$\min_{j \in {1,...,m}}\bold{dist}(S_{tj},GT_t)+th_{err} <\bold{dist}(F_t, GT_t)$$
$\bold{dist}(x, y)😒 l1 Distance
$S_tj:$ j-th sensor output at timestamp t
$GT_t:$ Ground-truth at timestamp t
$F_t:$ Fusion result at timestamp t

Fusion Error: Must satisfy two properties:

1. Failure-inducing
2. Fusion-induced

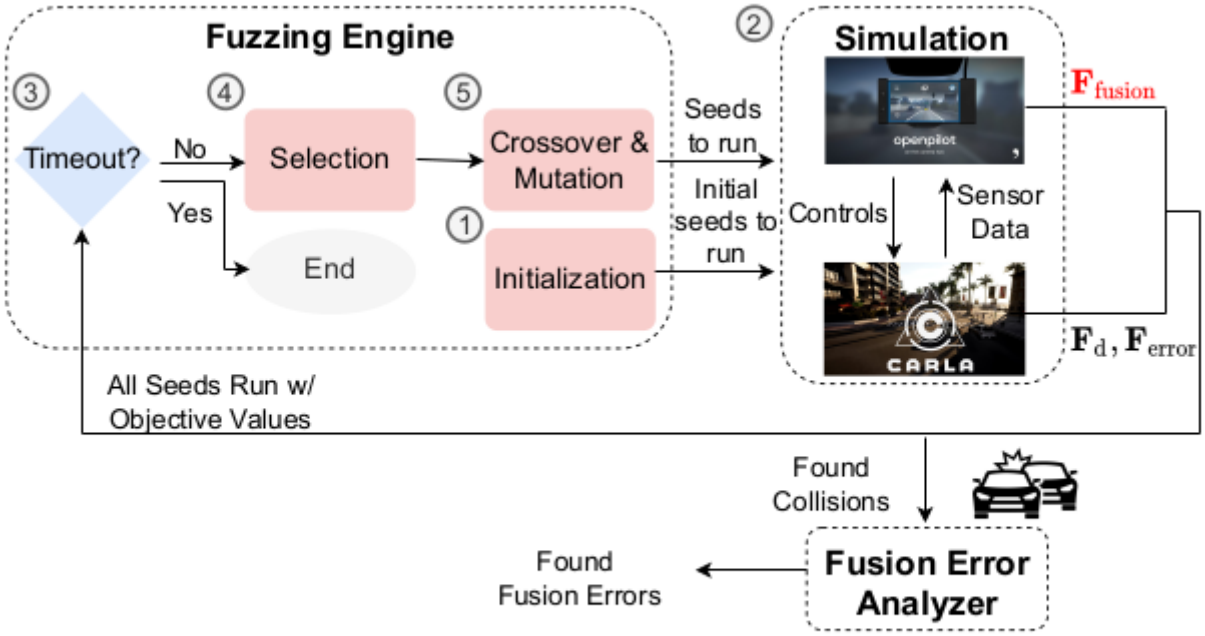

Figure 7: The overall workflow of *FusED*.

Fuzzing to Find Collisions

FusED adopts an evolutionary-based search algorithm with a domain-specific fitness function promoting fusion errors finding.
Fitness Function: Three objectives

- Minimizing the distance between the ego car and the leading vehicle (subtracted by the ego car's minimum stopping distance)
- Maximizing the occurence of collisions
- Maximizing the number of fusion-fault frames

Selection: Binary tournament selection, which has shown effectiveness in previous ADAS testing
Crossover and Mutation: Simulated binary crossover & polynomial mutation

Root Cause Analysis

To identify whether the root cause is sensor error, the authors develop an alternative *best fusion method*: an ideal fusion that always returns the minimal distance of the leading vehicles. If replacing the original fusion method with the *best fusion method* eliminates the collisions, then the root cause should be fusion error.
One major problem with the above method is that some uncontrollable factors other than fusion methods (i.e. communication delays between different SUT components) affect the test outcome. To cope with this issue, the authors set the communication delay to a fixed value (feasible in Carla).

Experiment

Environment. Carla 0.9.11 + OpenPilot 0.8.5
Baselines and Metrics. We use random search (random) and genetic algorithm without Ffusion in the fitness function (ga) as two baselines. We set the number of scenarios causing fusion errors and distinct fusion errorsas two evaluation metrics.
RQ1:Evaluating Performance. How effectively can FusED find fusion errors in comparison to baselines?
Result 1 Under each of the four settings, at the 0.05 significance level, FusED finds more distinct fusion errors (as well as fusion errors) than the best baseline method. The difference has a medium effect size at 90% confidence interval.
RQ2:Case Study of Fusion Errors. What are the representative causes of the fusion errors found?
Result 2. The representative fusion errors found by FusED for the two fusion methods are due to the dominance of camera over radar, their mismatch, or the faulty prediction selection method.
RO3:Evaluating Repair Impact. How to improve MSF in Openpilot based on our observations on found fusion errors?
Result 3: Based on the observations of the found fusion errors, we adjust the fusion method we study and enable it to avoid more than 50% of the initial fusion errors.

Conclusion

In this work, we formally define, expose, and analyze the root causes of fusion errors on two widely used MSF methods in a commercial ADAS. To the best of our knowledge, our work is the first study on finding and analyzing failures causally induced by MSF in an end- to-end system. We propose a grey-box fuzzing framework, FusED, that effectively detects fusion errors. Lastly, based on the analysis of the found fusion errors, we provide several learned suggestions on how to improve the studied fusion methods.

Questions

1. The target of this work is an ADAS OpenPilot whose function is simple lane-keeping and fuses only radar and camera. Is their approach applicable to more complex ADSs that have much richer functionalities and fuse more sensors?
2. In root cause analysis the authors only consider the communication delay as the sole factor of influencing the test outcome other than the fusion method, but is there any other factor that the authors do not take into consideration?