Testing and Debugging Autonomous Driving: Experiences with Path Planner and Future Challenges

Fuyuki Ishikawa
National Institute of Informatics
f-ishikawa@nii.ac.jp

NII 大学共同利用機関法人 情報・システム研究機構 国立情報学研究所 National Institute of Informatics

ERATO MMSD

TOC

- Preliminary
- Testing and Debugging a Path Planner
- Future Perspectives

Autonomous Driving: Engineering Challenges

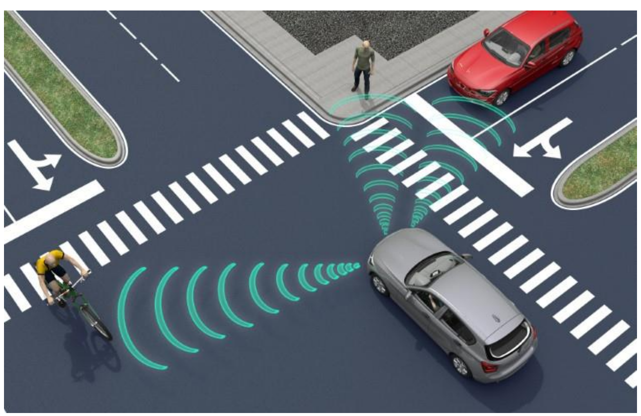
- Smart functionality demonstrated to be feasible

➡ Concerns on safety and reliability

- and the engineering process to make assurance

How do we tackle with our weapon?

(e.g., techniques from the ISSRE community)



[http://www.dailymail.co.uk/news/article-3677101/Tesla-told-regulators-fatal-Autopilot-crash-nine-days-happened.html]

Transferring Techniques for Software Systems

Existing: search, analyze, and repair program bugs

Discrete
Clear oracle

Complex programs

l.1	l.2	l.3	…	Result
✓	✓		…	PASS
✓		✓	…	FAIL
		✓	…	FAIL
…	…	…	…	…

"Intelligent testing" (e.g., search-based)

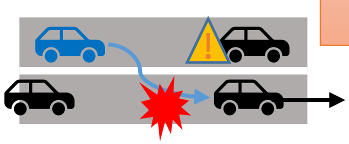
Line	Fault impact
l.3	0.8
l.5	0.72
l.9	0.6
…	…

Fault localization (e.g., spectrum-based)

Automated repair (e.g., search-based)

Transfer to (Autonomous) Automotive Systems

Continuous
Fuzzy/open world



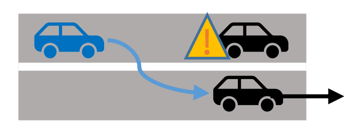
Driving systems

X	Y	Z	…	Danger
0.2	0.8	0.4	…	0.2
0.8	0.3	0.1	…	0.9
0.4	0.2	0.7	…	0.6
…	…	…	…	…

"Intelligent testing" for safety

Factor	Safety impact
Large X	0.8
Small Y	0.72
Small Z	0.6
…	…

Fault localization in continuous world

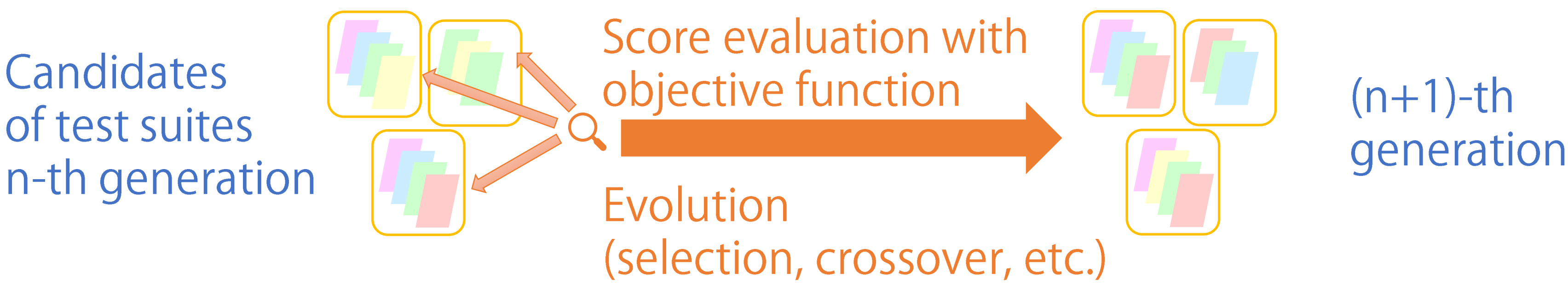


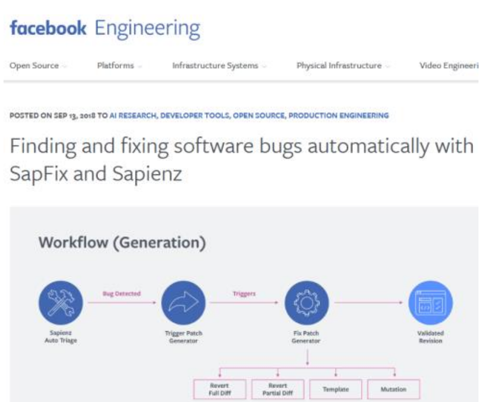
Automated repair of continuous behavior

Note: Search-based Software Engineering

- Reduce SE problems to optimization
  - Test input generation, program repair, configuration, …
  - Use of metaheuristics such as evolutionary computation

Case of test suite generation





Application in Facebook
(test input generation and repair)

[S. Ali et al., Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation, 2010]

[https://code.fb.com/developer-tools/finding-and-fixing-software-bugs-automatically-with-sapfix-and-sapienz/]

Our Project: ERATO-MMSD

led by Ichiro Hasuo (NII) (2016-2022)

https://group-mmm.org/eratommsd/

Group 0: Metamathematical Integration

T + e → $T(e)$ ← Category Theory

Generalize / *Instantiate*

Group 1: Heterogenous Formal Methods

Transfer from discrete to continuous

T_1 + e_1 → $T_1(e_1)$

T_2 + e_2 → $T_2(e_2)$

…

Computer Science

Control Theory

HERE!

Group 3: Formal Methods and Intelligence

Heuristics, Evolutionary, Search-based approaches

Evolutionary Computation

Machine Learning

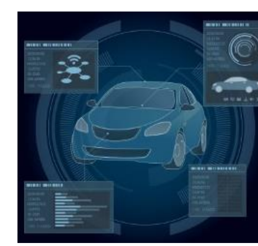
Software Engineering

Reliability Engineering

Group 2: Formal Methods in Industry

Advanced setting in autonomous driving

Automotive Industry



Practical setting to improve present practices

TOC

- Preliminary
- Testing and Debugging a Path Planner
- Future Perspectives

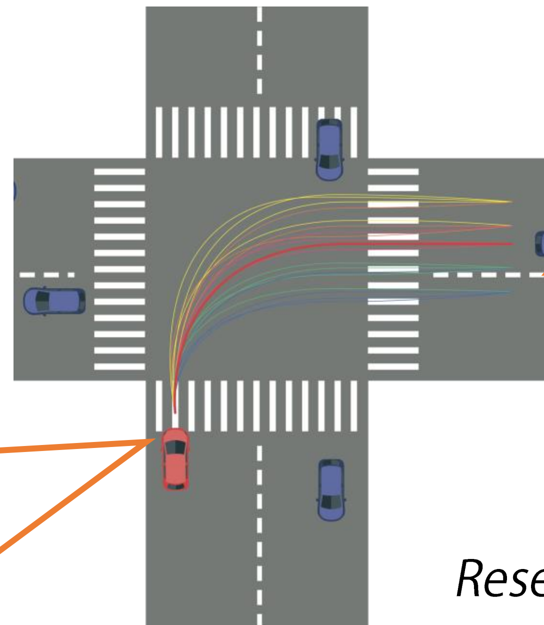
Target: Path Planning Software

- Path planning in autonomous driving
  - Short-term decision on steering and acceleration
  - Here, optimization-based

Dev. time

Weight design in cost function

Distance to objects	10000
Abrupt steering	200
…	



Runtime

Cost of each path candidate

1. 150
2. 30000
3. …

Research software provided by Mazda

Testing and debugging weight design?

Search-based Collision Detection?

We can search for and detect collision cases by using a "danger score" !

Search space

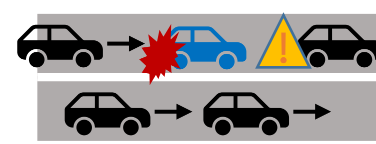
Simulator configuration
- Road shape
- Movement of pedestrians and other cars
- Initial location and velocity
- …

Objective function

$$danger(s^e_{t_i}, s^j_{t_i}) = \begin{cases} \vec{v}^{\,t_i}_{e|j} + K & \text{if } collision(s^e_{t_i}, s^j_{t_i}), \\ \dfrac{\vec{v}^{\,t_i}_{e|j}}{\|s^e_{t_i}.p, s^j_{t_i}.p\|^2} & \text{otherwise.} \end{cases}$$

Collision case: bad if the relative speed is high
Non-collision case: bad if the relative speed is high and the distance is small

Detected collisions are not due to the ego-car
- *Even "attacks" by other cars*
- *But "collision of our fault" is non-specifiable*



Detection of "Avoidable" Collision

Intuition

A collision is likely to our fault if it can be avoid by very small change of the weight design

Search space

Simulator configuration

- Road shape
- Movement of pedestrians and other cars
- Initial location and velocity
- …

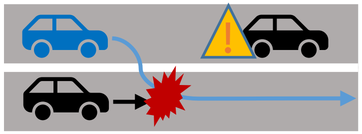
+ Weight repair

Objective function

1. The weight repair largely changes the "danger score" (especially, changing a collision case into a non-collision case)
2. The weight repair is small

Note: scenarios (e.g., overtaking) can be specifiable by an objective or the initial setting

For each scenario, we could generate collision cases that need to be fixed



[ICST'20]

Debugging (1) Automated Repair

- Want to discover a repair in the weight design that deals with all the detected collision cases
  - Previous "repair" was only for avoiding "too difficult" collision cases that are probably due to the environment

➡ Applying the search-based repair

Search space
Weight repair

Objective function

1. The weight repair largely changes the "danger score" values in the input collision cases
2. The weight repair is small

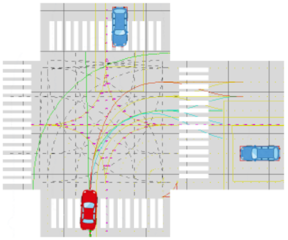
➡ Discovered a repair to avoid all the 7 collision can

- In most cases 80～90% (includes randomness)
- 7 cases detected in each scenario (e.g., overtaking)

[GECCO'20]

Debugging (2) Explanation of Factors

- Generating many collision cases in the same scenario and analyzing their factors
  - Extending spectrum-based fault localization (next slide)



(a) $\mathcal{S}_{RightTurn}$

Record of variations with different weight designs and simulator configurations

Analyze, extract, and explain factors

1. Greater weight values for too much lateral acceleration
   → Higher danger scores
2. Higher danger scores
   → Curvature and deceleration go beyond the thresholds

Explanation: collisions were caused by too strict restriction of large steering behavior for avoiding them

[ISSRE'20]

Foundation of Explanation

Transfer of spectrum-based fault localization

Spectrum for programs

l.1	l.2	l.3	…	Result
✓	✓		…	PASS
✓		✓	…	FAIL
		✓	…	FAIL
…	…	…	…	…

X	Y	Z	…	Danger
0.2	0.8	0.4	…	0.2
0.8	0.3	0.1	…	0.9
0.4	0.2	0.7	…	0.6
…	…	…	…	…

Line 3 was rarely used in PASS cases but often used in FAIL cases

"VERY SMALL" Y often appears in danger cases

Applying the same technique by discretization by fuzzy sets

e.g., x=0.3 → x={ 0.2 – "VERY SMALL", 0.8 – "SMALL" }

[ICECCS'19]

TOC

- Preliminary
- Testing and Debugging a Path Planner
- <u>Future Perspectives</u>

Ongoing Direction: Comprehensiveness

- Testing so far was to detect (and fix) problems

➡ Testing to give a certain level of assurance

  - Scenario coverage and risk evaluation: combining with scenario analysis and probabilities (likelihood)

  - Whitebox coverage criteria:
    showing we tested "all of significant behaviors"

  ➡ "Weight coverage" definition and search-based input generation
    (e.g., "uncomfortable behavior activated by the tests?")

[ICECCS'20]

GAUSS Aspects? (1)

- Adaptive?
  - Generally, "emergent behavior" is avoided as it is difficult to give safety assurance
  - However, we tried a self-adaptive path planner
  - ⇨ Switch between sets of weight values investigated through the testing phase

⬇

Use of adaptation in testing: once a collision is detected, the self-adaptive path planner avoids similar ones by adaptively changing the weights

⇨ Continue the search to collect diverse collision cases

GAUSS Aspects? (2)

- Unplanned Systems of Systems?

  (Maybe also said “multi-agent systems”)

➡ Future work: very essential aspect in autonomous driving

  - Other cars and pedestrians are autonomous and may respond to behavior of the ego-car, leading to unexpected emergent behavior as the whole
  - We need sophisticated “models” and simulators

Summary

- Our experience of testing and debugging a path planning software
  - Difficulties in the open world: non-specifiable and unclear boundaries of valid/invalid or correct/incorrect
  - Power of techniques investigated for software programs, transferred to the continuous, fuzzy world

Thanks to the JST-supported project and Mazda!

References

- Testing – detection of “avoidable” collisions
  - Alessandro Calò et al., Generating Avoidable Collision Scenarios for Testing Autonomous Driving Systems, ICST 2020 Industry Track
- Debugging – automated repair
  - Alessandro Calò et al., Simultaneously Searching and Solving Multiple Avoidable Collisions for Testing Autonomous Driving Systems, GECCO 2020
- Debugging - explanation
  - Xiao-Yi Zhang et al., Investigating the Configurations of an Industrial Path Planner in Terms of Collision Avoidance, ISSRE 2020 PER
  - Xiaoyi Zhang et al., Assessing the Relation Between Hazards and Variability in Automotive Systems, ICECCS 2019
- Others
  - Thomas Laurent et al., Achieving Weight Coverage for an Autonomous Driving System with Search-based Test Generation, ICECCS 2020 (to appear)
  - Kun Liu et al., Leveraging Test Logs for Building a Self-Adaptive Path Planner, SEAMS 2010 NIER