Software Risk and Autonomy

NASA Autonomy Workshop
Oct 11, 2018
Overview

- **Control & Planning safety**
  - Breaking robots for fun and profit

- **Perception safety**
  - It’s a bird. It’s a plane.
  - It’s ... what the heck is that?

- **Edge cases**
  - Back to breaking robots for fun and profit
Carnegie Mellon University Faculty, staff, students
Off-campus Robotics Institute facility
Before Autonomy Software Safety

- The Big Red Button era

© 2018 Philip Koopman
APD (Autonomous Platform Demonstrator)

TARGET
GVW: 8,500 kg
TARGET SPEED: 80 km/hr

Safety critical speed limit enforcement
Traditional Validation Meets Machine Learning

- Use traditional software safety where you can

..BUT..

- Machine Learning (inductive training)
  - No requirements
    - Training data is difficult to validate
  - No design insight
    - Generally inscrutable; prone to gaming and brittleness
Safety Envelope Approach to ML Deployment

- Specify unsafe regions
- Specify safe regions
  - Under-approximate to simplify
- Trigger system safety response upon transition to unsafe region
Architecting A Safety Envelope System

“Doer” subsystem
- Implements normal, untrusted functionality

“Checker” subsystem – Traditional SW
- Implements failsafes (safety functions)

Checker entirely responsible for safety
- Doer can be at low Safety Integrity Level
- Checker must be at higher SIL

(Also known as a “safety bag” approach)
Robustness Testing

**ASTAA: Automated Stress Testing of Autonomy Architectures**
- Key idea: combination of exceptional & normal inputs to an interface

**Example: Ground Vehicle network**
- Test Injector
  - Selectively modifies CAN messages on the fly
  - Modification based on data type information
- Invariant monitor
  - Reads messages for invariant evaluation
  - “Checker” invariant monitor detects failures

**Commercial tool build-out:**
- **Edge Case Research Switchboard**
  (software & hardware interface testing)
Robustness Test + Monitor → ASTAA

Automated Stress-Testing for Autonomy Architectures

Test Specification and Execution Overview

- ASTAA Test Spec (XML)
  - Interface Definition
    - Ports & Protocols
    - Message Dictionary
    - Message Frame 1
    - Parameter 1 (int)
    - Parameter 2 (bounded int)
- Constructors & Destructors
- Invariants Definition
  - Mode State Machines
  - Invariant List
  - Inv1: Parent < Plimit
  - Inv2: ~Condition
- Invariant Failure Destructors

- Exceptions Database
- Test Generator
  - Message Types
  - Type Exceptions
- Test Command Sequences
- Test Cases (XML)
- ASTAA Test Runner
  - Invariant Monitors
  - Test Injectors
  - Orchestrator
  - GUI
- Module Manager
  - System Under Test
  - Safety Monitor (optional)

Define ASTAA Test Specification (guided manual process)
Execute Test Generator (automated process)
Execute Test Cases with Test Runner (automated process)

© 2018 Philip Koopman
Researchers evaluated 150 bugs from 11 distinct projects over 4 years [ICSE 2018]

From "RIOT Expanded Technical Brief, NAVAIR Public Release- 2016-842 'Approved for Public Release; distribution is unlimited'.
Robustness Testing Finds Problems

- Improper handling of floating-point numbers:
  - Inf, NaN, limited precision
- Array indexing and allocation:
  - Images, point clouds, etc...
  - Segmentation faults due to arrays that are too small
  - Many forms of buffer overflow with complex data types
  - Large arrays and memory exhaustion
- Time:
  - Time flowing backwards, jumps
  - Not rejecting stale data
- Problems handling dynamic state:
  - For example, lists of perceived objects or command trajectories
  - Race conditions permit improper insertion or removal of items
  - Garbage collection causes crashes or hangs
Non-Machine Learning Robustness Lessons

- Protect your robots from data assumptions
  - Don’t trust that your configuration is valid
  - Time is not always monotonic
  - Semantically redundant field mismatches

- Floats and NaNs useful but dangerous
  - Do not use floats as iterators
  - NaNs propagate

- Plan for the system to fail
  - Nodes should not fail silent
  - Good logging is invaluable

- Common sense?
  - (Not so common it turns out)

Send of “infinity” floating point joint angle causes unsafe wind-milling
Validation of an Autonomous Vehicle Pipeline

- **SENSORS**
  - Perception
    - Machine Learning Based Approaches
      - Randomized & Heuristic Algorithms
      - Run-Time Safety Envelopes
      - Doer/Checker Architecture
    - ???

- **PLANNING**
  - Control Systems
    - Control Software Validation
    - Doer/Checker Architecture

- **TRAJECTORY EXECUTION**
  - Autonomy Interface To Vehicle
    - Traditional Software Validation

- **VEHICLE CONTROL**

Perception presents a uniquely difficult assurance challenge.
Brute Force Road Testing

- If 100M miles/critical mishap...
  - Test 3x–10x longer than mishap rate
    → Need 1 Billion miles of testing

- That’s ~25 round trips on every road in the world
  - With fewer than 10 critical mishaps...
Brute Force AV Validation: Public Road Testing

- Good for identifying “easy” cases
  - Expensive and potentially dangerous
Did We Learn The Right Lesson from Tempe?

■ NOT: Blame the victim
  ● Pedestrian in road is expected

■ NOT: Blame the technology
  ● Immature technology under test
    – Failures are expected!

■ NOT: Blame the driver
  ● A solo driver drop-out is expected

■ The real AV testing lesson:
  ➔ Ensure safety driver is engaged ➙
  ● Safety argument: Driver alert; time to respond; disengagement works

© 2018 Philip Koopman
Can Safety Driver React In Time?

**Safety Driver Tasks:**
- Mental model of “normal” AV
- Detect abnormal AV behavior
- React & recover if needed

**Example: obstructed lane**
- Does driver know when to take over?
- Can driver brake in time?
  - Or is sudden lane change necessary?

**Example: two-way traffic**
- What if AV commands sudden left turn into traffic?
**Closed Course Testing**

- Safer, but expensive
  - Not scalable
  - Only tests things you have thought of!

© 2018 Philip Koopman
Highly scalable; less expensive
- Scalable; need to manage fidelity vs. cost
- Only tests things you have thought of!

Simulation


Udacity

Apollo

http://bit.ly/2toFdeT
What About Edge Cases?

- You should expect the extreme, weird, unusual
  - Unusual road obstacles
  - Extreme weather
  - Strange behaviors

- Edge Case are surprises
  - You won’t see these in testing
    ➔ Edge cases are the stuff you didn’t think of!
Just A Few Edge Cases

- Unusual road obstacles & obstacles
- Extreme weather
- Strange behaviors

https://dailym.ai/2K7kNS8
https://goo.gl/J3SSyu
https://en.wikipedia.org/wiki/Magic_Roundabout_(Swindon)
Why Edge Cases Matter

Where will you be after 1 Billion miles of validation testing?

Assume 1 Million miles between unsafe “surprises”

- Example #1:
  100 “surprises” @ 100M miles / surprise
  - All surprises seen about 10 times during testing
  - With luck, all bugs are fixed

- Example #2:
  100,000 “surprises” @ 100B miles / surprise
  - Only 1% of surprises seen during 1B mile testing
  - Bug fixes give no real improvement (1.01M miles / surprise)

https://goo.gl/3dzguf
The Real World: Heavy Tail Distribution(?)

- **Common Things Seen In Testing**
  - Random Independent Arrival Rate (exponential)
  - Power Law Arrival Rate (80/20 rule) (Heavy Tail Distribution)

- **Edge Cases Not Seen In Testing**
  - Many *Different*, Infrequent Scenarios
  - Total Area is the same!
The Heavy Tail Testing Ceiling

TIME BETWEEN EVENTS (LOG SCALE)

# OF TRAINING MILES (LOG SCALE)

DEPLOYMENT SAFETY GOAL

HEAVY-TAIL CEILING

FAULT INJECTED TRAINING

HERE THERE BE DRAGONS!
(UNKNOWNABLE UNKNOWNS)

UNIQUE SURPRISES
(MOSTLY UNSEEN IN TRAINING)

“HARD SCENARIO” WEIGHTED TRAINING

RARE EVENTS
(SEEN ONCE IN TRAINING)

BRUTE FORCE TRAINING

UNUSUAL EVENTS

EVERYDAY EVENTS

© 2018 Philip Koopman
Malicious Image Attacks Reveal Brittleness

QuocNet:

AlexNet:


https://goo.gl/5sKnZV
ML Is Brittle To Environment Changes

Sensor data corruption experiments

Synthetic Equipment Faults

Defocus & haze are similarly a significant issue

Exploring the response of a DNN to environmental perturbations from “Robustness Testing for Perception Systems,” RIOT Project, NREC, DIST-A.
What We’re Learning With Hologram

A scalable way to test & train on Edge Cases

Your fleet and your data lake

Hologram cluster tests your CNN

Hologram cluster trains your CNN

Your CNN becomes more robust
Context-Dependent Perception Failures

- Perception failures are often context-dependent
  - False positives and false negatives are both a problem
  - *This is an active research area ... technology still in development*

Will this pass a “vision test” for bicyclists?
Ways To Improve AV Safety

- **More safety transparency**
  - Independent safety assessments
  - Industry collaboration on safety

- **Minimum performance standards**
  - Share data on scenarios and obstacles
  - Safety for on-road testing (driver & vehicle)

- **Autonomy software safety standards**
  - Traditional software safety ... **PLUS** ...
  - Dealing with uncertainty and brittleness
  - Data collection and feedback on field failures


Thanks!