Testing robot navigation in virtual worlds

Hélène Waeselynck,
Joint work with: Jérémie Guiochet, Félix Ingrand, Clément Robert and Thierry Sotiropoulous
About LAAS

Laboratory for Analysis and Architecture of Systems
Associated with the University of Toulouse

- > 700 persons
- 8 Departments

  - Crucial Computing
  - Networks & Comm.
  - Robotics
  - Decision and Optimization
  - Energy Management
  - Nanomat’s, Nanofuidics and Smart Systems
  - Micro Nano Bio Techs
  - Microwaves and optics

**TSF Team: Dependable Computing and Fault tolerance**
Our work on dependable autonomous systems

- Risk analysis and safety argumentation
- Active Safety Monitoring
- Testing in virtual worlds (in collaboration with the RIS Team of the Robotics Department)
Testing robot navigation

- Autonomous robots = with decisional capability
  - E.g., ability to navigate in diverse and previously unknown environments

- Mostly validated by field testing
  - Costly
  - Risky in case of misbehavior

- Intensive testing in virtual worlds?

Inputs
- World
- Mission

Virtual world ≠ real world!

SIMULATOR

Input model of worlds and missions?

Test oracle? (No ground truth about the decisions to take)
Two case studies (outdoor navigation)

**Mana**
- Used for outdoor experiments by LAAS researchers in robotics
- Generic navigation missions
- Path planning based on NASA’s GESTALT algorithm for Mars exploration rovers
- 35 KLOC including 3D mapping, localisation, path planning
- MORSE simulator (based on the Blender game engine)

**Oz**
- Agricultural robots developed and commercialized by Naïo Technologies
- Weeding missions
- Proprietary and mission-specific software
- 151 KLOC (also including modules we do not test: control of weeding tools & user interface)
- Gazebo simulator
Outline

- Reproducibility of bugs in low-fidelity simulation
- Defining and generating virtual worlds & missions
- Test control via generation parameters
- Oracle problem
Research Question

RQ1: Can robot navigation bugs be reproduced in low-fidelity simulation?

In-depth analysis of the triggers and effects of bugs affecting the Mana navigation software

- No inertia
- No reaction between wheels and ground
- No slippery areas

Baseline = low-fidelity simulation of the physics with MORSE (used by the developers for prototyping)
Dataset of bugs

- 2005-2015 history of the code, focus on the core navigation modules and libraries (17.5 KLOC out of 35 KLOC)
- Manual extraction of bugs from the analysis of commits
  - Code diffs + comments entered by the authors of the commits

<table>
<thead>
<tr>
<th>Module</th>
<th>Commits</th>
<th>Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3D</td>
<td>69</td>
<td>12</td>
</tr>
<tr>
<td>LibP3D</td>
<td>154</td>
<td>14</td>
</tr>
<tr>
<td>DTM</td>
<td>50</td>
<td>3</td>
</tr>
<tr>
<td>POM</td>
<td>83</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>356</strong></td>
<td><strong>33</strong></td>
</tr>
</tbody>
</table>

- Note: mature software (mostly developed before 2005), sporadic usage for experiments when required by projects, gaps in the archiving process

- Most of the bugs were found by outdoor experiments
Reproducibility results

<table>
<thead>
<tr>
<th>Not reproducible</th>
<th>Reproducible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
</tr>
</tbody>
</table>

- Only one bug is judged as not reproducible (mechanical vibration during spot turn)
- Useful to add inertia to the baseline test platform – still low-fidelity, but abrupt braking no longer stops the robot immediately
- Reproducibility results are encouraging:
  - Low-fidelity simulation is cheaper: computational resources, effort to develop the robot avatar
  - Low-fidelity is more tractable: performance issues with existing simulators (both MORSE and Gazebo)
Outline

- Reproducibility of bugs in low-fidelity simulation
- Defining and generating virtual worlds & missions
- Test control via generation parameters
- Oracle problem
World & mission Models

- Navigation: no explicit interaction protocol with the environment, but a continuous perception/decision/action loop
- Defining test input domain? World (and missions) models are first class citizens

Use case(s) -> World model

Class attributes = generation parameters
Class constructors = generation functions
Oz: richer world model

31 generation parameters, but a varying number of instances
E.g., if there are N crop rows, we must select N-1 inter-row distances

Grammar-based approach to manage the parameters
Descriptor: genotype of a world or a world element
Operations to create, parse, manipulate, and check descriptors

Descriptors can have wildcards
R (= any random world)
\[ f+0.0+0-R-R \] (flat terrain, any mission and field)
\[ R-R-3+R+R+R \] (any terrain and mission, field with 3 crop rows)

\[
\langle \text{World} \rangle ::= \langle \text{terrain} \rangle \ "-" \langle \text{mission} \rangle \ "-" \langle \text{field} \rangle \\
\langle \text{terrain} \rangle ::= \langle \text{terrain_type} \rangle \ "+" \langle \text{magnitude} \rangle \ "+" \langle \text{seed} \rangle \\
\langle \text{field} \rangle ::= \langle \text{nb_crop_row} \rangle \ "+" \langle \text{gap_list} \rangle \ "+" \langle \text{crop_row_list} \rangle \ "+" \langle \text{disturbing_element_list} \rangle
\]
Outline

- Reproducibility of bugs in low-fidelity simulation
- Defining and generating virtual worlds & missions
- Test control via generation parameters
- Oracle problem
A navigation software is not deterministic!

 mana: same world, same mission, but 5 different trajectories

 Oz: trajectories are more constrained, but still the test verdict varies

 30% of the test cases do not yield a consistent verdict for 5 repeated runs

- Search-based testing
  - Fitness value for one run is meaningless
  - Relevance of neighborhood operators?
- Control of dynamic test situations, e.g., involving mobile obstacles?
Controlling the difficulty of missions

- Problem: calibration of test generation parameters to control the difficulty level (e.g., easy, challenging, very difficult)

- Experimental approach
  - Generate several test cases (world + mission) for a given parameter setting
  - Have repeated runs for each test case
  - Measure difficulty: mission success rate, effort to accomplish the mission (duration, detours)
  - Aggregate results into difficulty levels (clustering)

- Example: parameter = percentage of obstruction for Mana missions

Trees (small obstacles)

<table>
<thead>
<tr>
<th>Easy</th>
<th>Challenging</th>
<th>Very difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;5%</td>
<td>5-7%</td>
<td>&gt;7%</td>
</tr>
</tbody>
</table>

Buildings (large obstacles): Easy up to 20%

- Note: indeterminism observed at all difficulty levels, no clear evolution pattern
Outline

- Reproducibility of bugs in low-fidelity simulation
- Defining and generating virtual worlds & missions
- Test control via generation parameters
- **Oracle problem**
No ground truth for decisional functions

- Mission Failure ≠ Fail verdict
  - An autonomous system is allowed not to succeed in a mission!
  - How to determine whether the mission fail reveals an abnormal behavior?

- The test oracle often merely detects catastrophic events (e.g., collisions)

- Feedback from the Mana bugs study was insightful to improve the situation
Mana Failures

- Infinite spot turn
- Failure to align to the target destination point
- Jerks in angular speed commands
- Robot does not immediately stop after detecting an error
- The robot arrives successfully at destination but considers itself as blocked
- The robot brakes too late when arriving at destination
- The speed commands are not refreshed and retain their value forever
- P3D does not start
- Execution crash
- Unexpected mission failure
- The robot goes round and round in circles until time-out
- The robot falls into a hole
- The robot has an absurd trajectory

High diversity of effects!
Oracle procedure?

- Collecting raw data: less an issue than interpreting it!
- High diversity of failures -> many checkers focusing on fine-grained properties
  - **Requirements attached to mission phases** (e.g., initial bad alignment to the destination)
  - **Thresholds related to robot movement** (e.g., maximal variation of speed commands)
  - **Catastrophic events** (e.g., collision)
  - **Requirements attached to error reports** (e.g., stop immediately after reporting an error)
  - **Perception requirements** (e.g., difference between perceived and real location, maximal percentage of unknown areas in the perceived map)
- Regression testing to detect other performance-related issues

The five classes of properties proved relevant for Oz as well
Conclusion

- Many navigation bugs do not require high physical fidelity
- World & mission models are first class citizens
  - Use cases
  - Well-structured approach (e.g., UML + grammar)
- A navigation software exhibits non-determinism
  - Repeated runs
  - Difficulty for strategies using previous tests to produce new ones
- Effects of bugs are diverse
  - Multiple fine-grained error detectors
  - Five broad classes of properties to check

Can robot navigation bugs be found in simulation? An exploratory study, QRS 2017.
Testing the Oz robot: in preparation...