Testing an agriculture robot in virtual crop fields

Hélène Waeselynck,
Joint work with: Clément Robert, Thierry Sotiropoulous, Jérémie Guiochet, Simon Vernhes

CPSE Labs

TrustMeIA
July 5, 2019
Toulouse France

Validation of autonomous robots

- Autonomous robots = with decisional capability
  - ✓ Have to accomplish missions in diverse and previously unknown environments
- Mostly validated by field testing
  - ✓ Costly
  - ✓ Risky in case of misbehavior
- Intensive testing in **virtual** worlds?

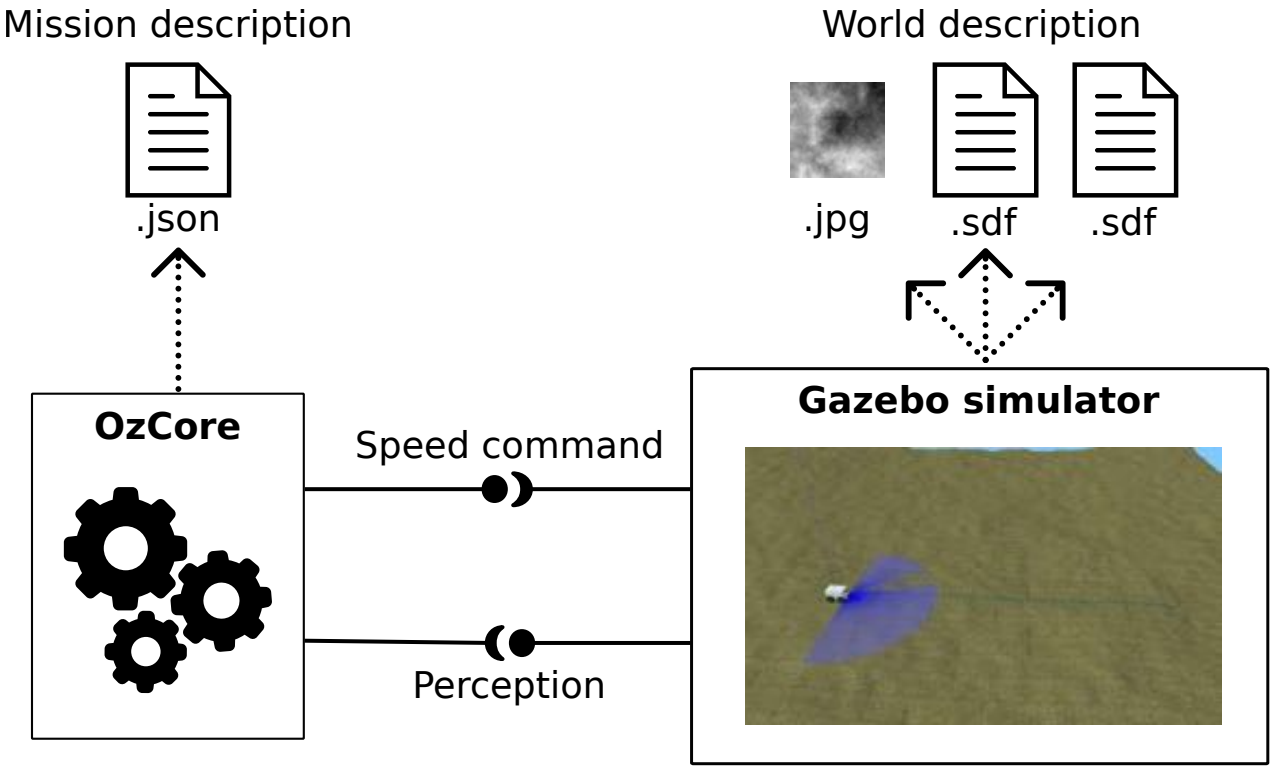
Outline

- The Oz case study
- Design of simulation-based testing
  - Defining and generating virtual worlds & missions
  - Test oracle
- Results and comparison with the field tests
- Conclusion

OZ: an agriculture robot

- Developed and commercialized by Naïo Technologies
- Weeding missions
- Perception: LIDAR 2D, two cameras
- Software in C, C++: 151 KLOC

Gazebo-based simulator



- Software-in-the-loop configuration
- Focus on testing the autonomous navigation
- Performance issues → low-fidelity simulation
  - ✓ Simplified physics (interaction wheels/ground)
  - ✓ Small-scale crop fields

Experimentation

@Naïo:

- Light pre-validation in simulation
  - ✓ Nominal case exemplifying a non trivial mission

  - ✓ Manual oracle (visual check)
- 5 test sessions in the field (each session half a day, 1-2 hours of testing)

@LAAS:

- Intensive simulation-based tests
  - ✓ 80 randomly-generated cases x 5 runs per case (=400 runs)
  - ✓ Range of values of parameters: expected to represent reasonable operating conditions
  - ✓ Automated oracle
  - ✓ Intended to represent reasonable requirements, not too demanding
- No information about the faults found by Naïo

RQ1: issues revealed in each case?

RQ2: practical recommendations for simulation-based testing?

Outline

- The Oz case study
- **Design of simulation-based testing**
  - **Defining and generating virtual worlds & missions**
  - **Test oracle**
- Results and comparison with the field tests
- Conclusion

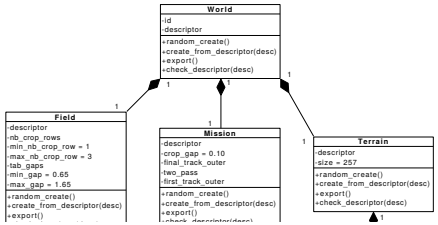
World & mission Models

- Defining test input domain? World (and mission) models are first class citizens
- Manual production of the worlds would be tedious → procedural content generation techniques (cf. video games)

A procedurally-generated world (Minecraft game)

- Principle: randomized generation controlled by a few high-level parameters (the world model parameters)

Randomized generation



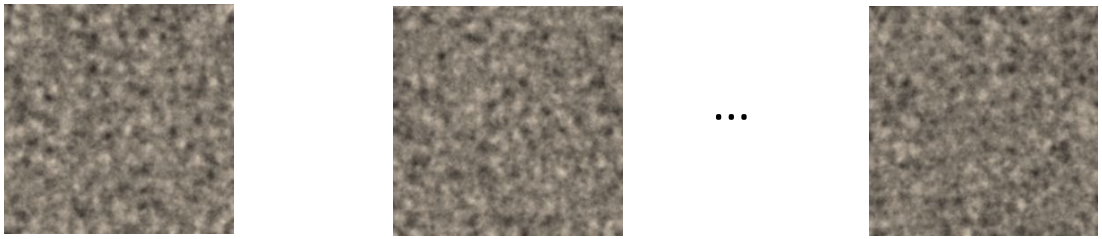
World model = structured view of world elements with their parameters (15)

UML class diagram +
attribute grammar (constraints)

Genotype = set of chosen parameter values

Valid words of the grammar

World_1 World_2 ... World_k ...



Phenotype = world content generated from the parameters

Note: the format of the generated content depends on the interfacing with the simulator

Oracle problem

- Behavior = continuous perception/decision/action
- Indeterminism
- Mission Failure ≠ Fail verdict
  - ✓ An autonomous system is allowed not to suceed 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 a previous study of navigation bugs helped in the identification of a richer set of abnormal behavior patterns to detect

1. **Requirements attached to mission phases**
2. **Thresholds related to robot movement**
3. **Catastrophic events**
4. **Requirements attached to error reports**
5. **Perception requirements**

Oz: properties to check

Mission Phases	**P1**	**U-turn in 5-7 maneuvers**
	P2	*Robot maintains reference distance to the crop row*
	P3	**Sequence of weeded rows is correct**
Movement thresholds	*P4*	*Velocity < Vmax*
Catastrophic events	**P5**	**No collision**
	P6	**Robot does not go outside of the crop field**
Perception	*P7*	*Self-localization with a certain precision*
Error reports	**P8**	**Stopping distance < dmax after reporting an error**

P2, P7: performance-related properties, should not yield a fail verdict

P4: transient violations due to low-fidelity simulation (engine braking force ignored)

Outline

- The Oz case study
- Design of simulation-based testing
  - Defining and generating virtual worlds & missions
  - Test oracle
- **Results and comparison with the field tests**
- Conclusion

Comparison approach

@LAAS (400 runs):

- 48% of the runs had a fail verdict
- 4 out of 5 properties could be violated
  - P1: U-turn in 5-7 maneuvers
  - P3: sequence of weeded rows
  - P5: collision with vegetables or red stakes
  - P6: outside of the crop field
  - P8: Stop after reporting an error
- Detailed analysis of the failed scenarios

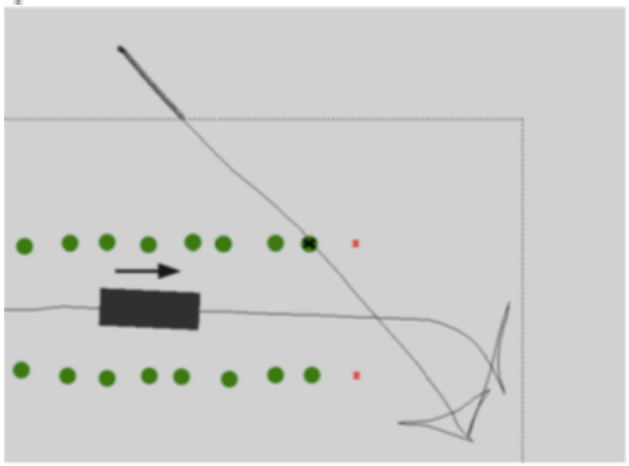
@Naïo (5 field test sessions):

- 23 navigation failures were reported during the field tests
- Diagnosis of the software issues causing the failures

Same issues revealed?

Confirmed issues

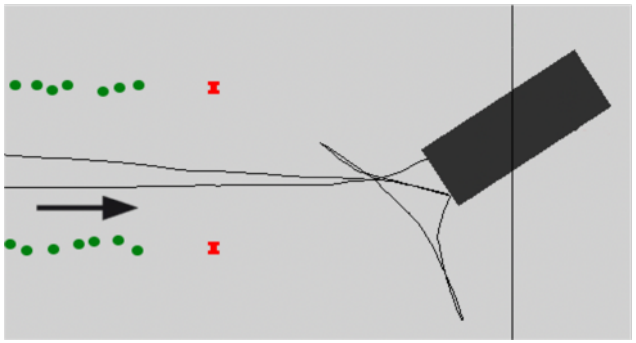
Issues	Field tests	Simul. tests
I1 - U-turn functionality	✓	✓
I2 - Space margin for U-turn	–	✓
I3 - Heuristics for transient perception losses	✓	✓
I4 - Processing of red stake images	✓	–
I5 - Alignment at the beginning of a row	✓	(✓) (with P2')
I6 - Skidding/odometry	✓	–



- Major issue, causing most of the failures (65% field, 75% simul.)
  - collisions, entrance of wrong row, escape trajectory
- The U-turn functionality had to be entirely re-developed

Confirmed issues

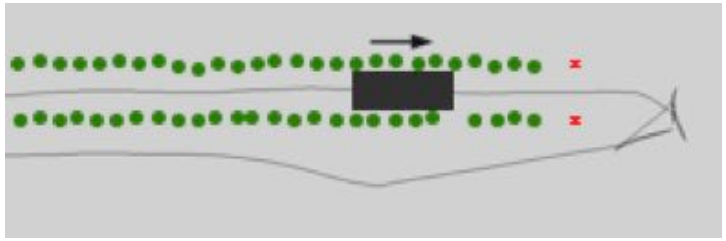
Issues	Field tests	Simul. tests
I1 - U-turn functionality	✓	✓
I2 - Space margin for U-turn	–	✓
I3 - Heuristics for transient perception losses	✓	✓
I4 - Processing of red stake images	✓	–
I5 - Alignment at the beginning of a row	✓	(✓) (with P2')
I6 - Skidding/odometry	✓	–



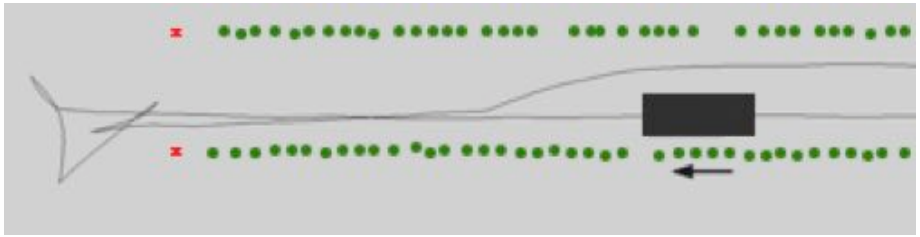
- Not revealed by the field tests (not noticed?)
- Conditions of use of Oz were revised to provision more space for the U-turn

Confirmed issues

Issues	Field tests	Simul. tests
I1 - U-turn functionality	✓	✓
I2 - Space margin for U-turn	–	✓
I3 - Heuristics for transient perception losses	✓	✓
I4 - Processing of red stake images	✓	–
I5 - Alignment at the beginning of a row	✓	(✓) (with P2')
I6 - Skidding/odometry	✓	–



Observed both in the field and in simulation



In simulation only: other misalignment cases

Confirmed issues

Issues	Field tests	Simul. tests
I1 - U-turn functionality	✓	✓
I2 - Space margin for U-turn	–	✓
I3 - Heuristics for transient perception losses	✓	✓
I4 - Processing of red stake images	✓	–
I5 - Alignment at the beginning of a row	✓	(✓) (with P2')
I6 - Skidding/odometry	✓	–

- Intensive simulation-based testing is effective
  - ✓ Finds real issues (causing 87% of the failures observed by field testing)
  - ✓ Is helpful to show the different failure cases induced by a given issue
  - ✓ Even uncovers a new issue

Confirmed issues (missed in simulation)

Issues	Field tests	Simul. tests
I1 - U-turn functionality	✓	✓
I2 - Space margin for U-turn	–	✓
I3 - Heuristics for transient perception losses	✓	✓
I4 - Processing of red stake images	✓	–
I5 - Alignment at the beginning of a row	✓	(✓) (with P2')
I6 - Skidding/odometry	✓	–

- Issues missed by the simulation-based tests:
  - ✓ I4 - the simulated images are too clear and crisp compared to the real ones (suggests that visual hazards should be added)
  - ✓ I6 – the simulation is not accurate wrt skidding, slippage or sliding

Spurious failures (in simulation only)

- Spurious P8 violations

P8	Stopping distance < dmax after reporting an error

Bug in the simulator (simulation of the stop)

- Spurious P4 violations

P4	Velocity < Vmax

Transient overspeed due to low-fidelity simulation (engine braking force ignored), does not correspond to a real behavior

Outline

- The Oz case study
- Design of simulation-based testing
  - Defining and generating virtual worlds & missions
  - Test oracle
- Results and comparison with the field tests
- **Conclusion**

Conclusion (RQ1: issues revealed)

- Many navigation bugs do not require a high physical fidelity (see also a previous study with another robot*)

*** Can robot navigation bugs be found in simulation? An exploratory study**, QRS 2017.

- Intensive (rather than light) pre-validation in simulation is effective
  - ✓ Promising in order to alleviate the costly field tests

- But the simulation may also introduce spurious failures
  - ✓ Bugs in the simulation code
  - ✓ Unrealistic behavior that would not occur in real world

- Naïo's strategy has evolved
  - ✓ Lighter simulation platform (simplified wrt the Gazebo-based one, easier to maintain)
  - ✓ Set of diverse test cases

Conclusion (RQ2: recommendations)

- Test generation and oracle: design for evolvavibility
  - ✓ Hard to specify
  - ✓ To be continuously improved as more experience is gained on the system (e.g., from the field tests)

- Generation
  - ✓ Well-structured world model to accommodate the addition/removal/modification of elements
  - ✓ Must also accommodate constraints on the generation parameters

- Oracle
  - ✓ Set of error detectors, each focused on a property
  - ✓ Five broad classes of properties to check
  - ✓ Separate data recording (online) and data analysis (offline)
  - ✓ Detectors can be added/revised/removed without having to re-execute the tests