First insights into testing autonomous robot in virtual worlds

Clément Robert
LAAS-CNRS, Toulouse, France
Email: crobert@laas.fr

***Abstract*—The capability of decisional autonomous systems has expended significantly in recent years. The failure of such a system can result in a catastrophic event. The variety of their tasks implies expensive and laborious test campaign. Along with actual computing power, simulation software seems mature enough to carry out test. Nevertheless, there is a no current method to select test. This work aims to provide a first step toward systematic testing of autonomous robot by exploring a test environment generation method and the oracle problem**

***Index Terms*—simulation-based testing; autonomous systems; safety; domain specific defects.**

I. Introduction

The development of decisional autonomous systems now makes it possible to perform tasks without human supervision for extremely varied environments. However, the failures of these systems can have unacceptable consequences for the mission, performance or reliability. When developing these systems, a major challenge is defining and carrying out tests. Indeed, uncertainties related to the environment and perception algorithms, combined with the possibilities of execution context, lead to an infinite field of test inputs (e.g., terrain, visibility conditions, obstacles, etc.). In most cases, robots are tested by deploying them in the real world under very limited experimental conditions. In order to explore more operational situations, and for obvious reasons of safety and cost, there is the possibility of carrying out these tests in simulation in virtual worlds. The MORSE infrastructure developed at LAAS [1], as well as other platforms such as GAZEBO [2], are designed to enable such simulations in robotics. However, there is not yet a systematic approach to select the worlds and situations to be tested, and to implement them automatically on simulation platforms. The aim of this work is to propose such a testing approach in the context of decisional autonomous systems. More precisely, the thesis will focus on testing the basic services of an autonomous system, taking the example of the navigation service of a mobile robot. The test input domain is then a world space in which the system is likely to evolve, and a set of mission configurations.

The structure of this paper is as follows. After a brief overview of related work (Section II), we introduce our approach in Section III. Then, we present the preliminary work in Section IV. Finally, we describe future direction in Section V.

II. Related Work

The testing of autonomous systems has started to spark interest from the testing community. Test selection strategies have been investigated, based on an abstract model of test situations, that describes the involved entities, their relationships and some interaction patterns. The authors of [3] use UML (Unified Modeling Language) to specify a metamodel of entities and a set of interaction scenarios. The approach is applied to a vacuum cleaner robot, using meta-heuristic search techniques to generate abstract test data from the models. The work of [4] defines several types of mid-air collision situations, and uses them to guide the evolutionary testing of a drone collision avoidance algorithm. The same authors extend their work to find challenging situations with a Genetic-Algorithm-based approach for a UAV collision avoidance system [5].

All these approaches have a very simplified view of the simulated environment and do not call for sophisticated simulation means. To the best of our knowledge, the work of [6] is the only one to consider complete virtual world environments, in the framework of 2D simulations. An interesting contribution of this work is to establish a connection with world content generation techniques used in the domain of video games [7], which we believe is a promising direction of research. Work along these lines has recently started at LAAS.

The authors of [8] investigate the possibility of using test campaigns in simulation to find bugs. They analyzed bugs in an academic software for outdoor robot navigation and seek what triggers them. Among the 33 bugs found during the in depth analysis of code commits, only 1 requires a high fidelity level to be replicated. The bug they did not replicate was related to mechanical vibration during rotation which is a complex physical event that does not occur in the simulation. Their work shows that most of the bugs are indeed replicable in low fidelity simulation. An important result is that some bugs required very specific configurations of the environment, mission and robot status to be triggered. The study also highlights the difficulty of defining an oracle due to the diversity of misbehavior patterns. They introduce a classification of properties that could provide a method to define an oracle.

The same authors work on the definition of the difficulty level of a generated test environment in [9]. They experimen-

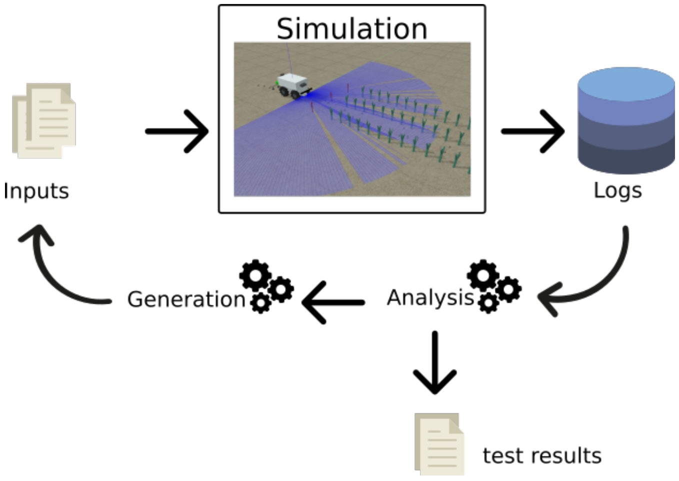


Fig. 1. Conceptual view of the testing loop

tally study the difficulty evolution as they change the test environment generation parameters. The simulated world consists in a bumpy terrain with 2 different obstacle sizes referenced as trees or buildings. The world is procedurally generated from two abstract parameters: smoothness and obstruction rate. The robot task is to reach the opposite corner of the square map. The difficulty levels are defined a posteriori by a clustering algorithm on some relevant information about the run such as the success rate or the mission time. Their results show that it is possible to coarsely control the difficulty level from the generation parameters. They also analyzed the evolution of the indeterminism as the difficulty level increases, and found out that there is no clear relation between the difficulty level and the indeterminism ”intensity”.

III. Proposed approach

To be able to consider a large sample of test situations, it is proposed to implement techniques derived from the procedural generation of worlds as in [7], used especially in the creation of scenes for video games, but also to add uncertainties on the environmental perception by the robot, or stressful conditions (e.g., moving obstacles, slippery ground). A first work was carried out at LAAS, and allowed to better identify the technological and methodological difficulties in getting such an approach. However, exploration has been restricted to a subset of world generation attributes, and the link between test criteria and the world synthesis has not been deepened. The contribution of this doctoral thesis will address the issues of modeling the space of worlds, including stressful characteristics for the service under test, and the use of test criteria to guide the synthesis and selection of worlds and missions. The design of the testing framework will also include identification of data to be collected during simulations to establish measures such as code coverage, or non-regression against previous versions of the system. The approach will be demonstrated on a real robotic software case. To do so, the real robot software is used, the sensors and the actuators are simulated to interface with the simulation environment.

Our approach consists in a loop to generate testing environment with a better potential of triggering bugs (Figure 1). We can see the world generator parameters as the dimensions of the possible test environment space. From that standpoint, we can now transform our problem in an optimization problem in which we try to maximize the potential of the testing environment. This method faces the following challenges:

- Define the input domain.
- Generate world and missions within the input domain.
- Automatically analyze the tests results.

A. Input domain definition

The input control parameters are the information needed to generate a world and a mission in this world. The definition of the parameters is fundamental because they should describe the key characteristics of the generated worlds and missions. Selecting the right parameters is a crucial challenge in our work. If the parameters are insufficient, it could result in a failure to trigger some bugs, and in the impossibility to detect the related effects. On the other hand, the more parameters we specify the more complex the world generator will be. A large number of parameters allows a better control in the world generation by adding details or generation rules. But, as we increase the number of parameters, the possible world space dimensions grow, making impossible a wide exploration of the possible world space. In the Naïo case study (Section III-D) we have up to 31 different parameters.

It is also interesting to work on the modeling of dynamic elements such as mobile objects or special events triggered by a specific situation (for instance: more noise on the sensor in one part of the world). This will be a future direction to extend our work.

B. Test generation

Autonomous systems are as various as the tasks they are made for (space exploration robot, driverless cars, human interactive robot). All of them are exposed to the same validation issues. The companies that use simulation to prototype their product generally run a set of empirically chosen test cases. The lack of automated methods for simulation based testing is a real hurdle to the validation process. One of the goals of this research is to synthesize test cases from the input domain definition and a set of test criteria.

The test criteria can be the coverage of code, of inputs subdomains and situations of interest. The main problem is the coverage control from high level generation parameters. Our solutions are generate-and-test approaches involving randomness. The search space is infinite therefore it is impossible to undertake a brute force method. We propose to use an iterative approach using metaheuristics. A metaheuristic approach by definition does not guarantee that an optimal solution can be found but can be practically effective. Furthermore, we consider stochastic optimization so the solution depends on the randomly generated variables. This approach raises an open question: Is it possible to converge towards a solution regarding the complexity of the search space? For example, if

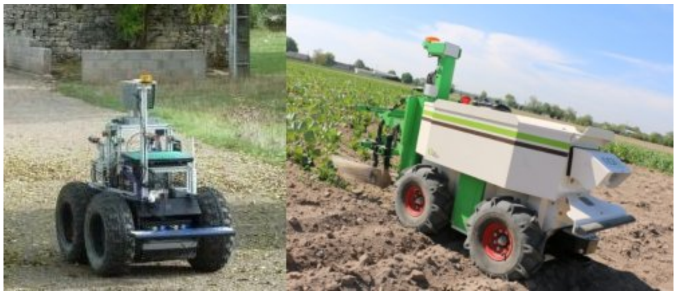


Fig. 2. Mana from LAAS (left) and Oz from Naïo Technologies (right)

we use GAs, which mutation or cross-over operators should we apply on the generation parameters?

Last but not least, indeterminism and experimentation time (up to 5 minutes for one run) are big constraints. We may have to run the same test more than once to evaluate the fitness, which implies a limitation on the number of iterations.

C. Oracle

The diversity of misbehaviour patterns makes the oracle problem very challenging. [9] advise trying to devise as many error detectors as possible, each focusing on a simple property. Their property classification is as follow:

- requirements attached to mission phases
- thresholds related to robot movement
- catastrophic events
- requirements attached to error reports
- perception requirements

This classification, derived from the analysis of an exemplary robot navigation software, will need to be validated on other case studies. The set of checkers can be enhanced as more experience is gained on the target system. Nevertheless it may be impossible to have a complete error coverage, especially for performance related issues (not caught by the detectors). Our future work will explore whether the concept of difficulty levels in ([9] can offer a partial solution, in the framework of regression testing.

D. Case Studies

To validate our approach we must confront it to case studies as diverse as possible. Two case studies are available for this work (Figure 2):

- The first case study is the Mana robot (a Segway RMP 400 platform) navigation service used in in [8] and [9]. The navigation software is part of the OpenRobots software repository, which includes software mostly developed at LAAS for the study and design of various kinds of robotic platforms. The offered path planning service is an academic implementation of NASA's GESTALT algorithm for Mars exploration rovers [10]. This software was used for all outdoor field experiments conducted by LAAS researchers in various collaborative projects between 2005 and 2015. The test platform is available on the MORSE simulator.
- The RAST (Risk Analysis and Simulation Testing for agricultural Robots) experiment from the CPSE Labs project (Cyber-Physical Systems Engineering) consists in a study of safety and validation issues on an agricultural robot from Naïo Technologies. Naïo Technologies is a company that sells agricultural robots designed to weed, hoe and assist during harvesting. The Oz robot is an autonomous weeding robot. With some parameters such as the crop width, the robot finds its way between crop rows and performs its task. The simulation uses the ROS robotic middleware and Gazebo simulator.

The complementarity of those 2 case studies allows us to experiment our approach on both academic and industrial robots, compare MORSE and GAZEBO for testing purposes and to confront our approach to 2 different navigation missions (generic missions to reach a destination point versus specific missions along crop rows).

IV. Preliminary Work

The different parts of the testing loop (Figure 1) raise numerous implementation problems. This section gives an overview of our solution for the Naïo case study.

The inputs are defined by all the parameters needed to procedurally generate a world and a mission. Procedural generation is a content creation method mainly used in video games to reduce file size or to create unique contents. Such a method allows us to algorithmically generate large data with respect to some initial rules. The rules constrain the definition domain of every test environment variable and the invariant patterns. For instance, an algorithm to procedurally produce a tree has the size of the tree and the number of leaves as variables and has the general shape of a tree as the invariant pattern. The randomness of such a method is fundamental to explore the possible test environments defined by the rules and get a wide range of test cases.

We choose to generate and store the inputs using a formal grammar. This method allows us to easily generate world descriptors that respect the rules that we define. In addition, a lot of work already exists on grammar-based testing, which can be used to iterate in the loop. Figure 3 and 4 shows how inputs are generated using the formal grammar. We first define a hierarchical view of world elements to produce, which gives us the non-terminals and terminals of the grammar. For example, a world contains a field, which in turn contains rows of vegetables.

The limited number of rules do not require tools as yacc or Bison to generate the parser. We decide to implement the world as a class structure (Figure 3). We distribute the parser, the generation function and the checker in their corresponding classes in such a way that a call to any world class function recursively call every component. The implementation is highly expandable (it was effortless to add new static world elements during the design stage).

We define a few mutation operators on the world descriptor that allow us to generate new worlds and missions from previous ones. Our first test criterion is the collision situation

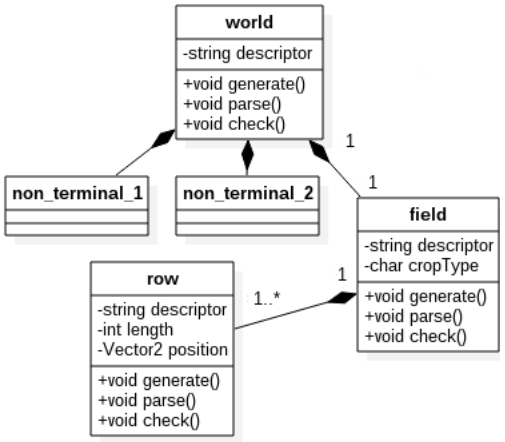
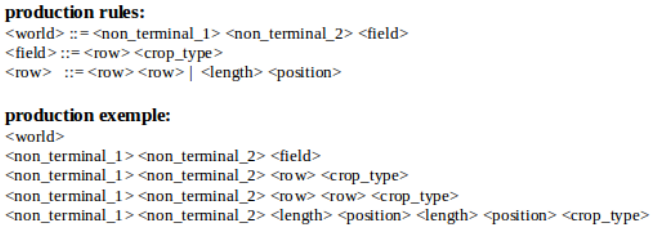


Fig. 3. UML diagram of the world



```
production rules:
<world> ::= <non_terminal_1> <non_terminal_2> <field>
<field> ::= <row> <crop_type>
<row>   ::= <row> <row> | <length> <position>

production exemple:
<world>
<non_terminal_1> <non_terminal_2> <field>
<non_terminal_1> <non_terminal_2> <row> <crop_type>
<non_terminal_1> <non_terminal_2> <row> <row> <crop_type>
<non_terminal_1> <non_terminal_2> <length> <position> <length> <position> <crop_type>
```

Fig. 4. Grammar example (BNF)

coverage (the robot collides with crops). Hence, the fitness is the robots smallest distance to a crop during a run. So far, we are running experiments to evaluate the feasibility of our approach. Those preparatory experiments aim to assess indeterminism effect on the fitness by doing multiple runs in a same world and by generating multiple worlds from the same descriptor. In the Oz case study, a few runs seem to be enough to erase the indeterminism effect. On the other hand, the indeterminism has a bigger impact on the Mana case study as the mission involves more decision from the robot. The last preliminary experiment is the evaluation of the fitness shape in the world space by doing small mutations. A chaotic shape would make an iterative search inefficient and would reveal a wrong choice of fitness and/or mutation operators. In the Oz case study, the fitness appears to evolve in a coherent shape in the simple test configurations and chaotically in the harder ones.

V. Future Direction

As presented, the research work is still at an early stage. The first contribution of this thesis will be to extend the testing loop of Figure 1 and to work on the feedback processing (trying various method to generate new inputs from the result of the previous one).

Year 1: (starting the 1st of October 2017)

- Study of different metaheuristic approaches (random search, hill climbing, genetic algorithms) in order to find specific configurations that trigger collision faults.
- Evaluation of the oracle performance on Naïo's case study (based on the oracle properties in Section III-C).
- Work on the modeling of situations and on its incorporation to the grammar.

Year 2:

- Further research on metaheuristic approach with other objectives than collision.
- Formal definition of selection criteria based on situation and world coverage.
- Development of the testing tools required for the new criteria (new observation data, new generation parameters, new metaheuristic operators, dynamic elements activated while a mission is running).

Year 3:

- Experimental comparison of the various test criterion (derived from the second year work) and some code coverage criteria.
- Finalization of the testing tools.
- Opening on regression testing.
- Conclusion on the test approach.

Acknowledgement

This work was supported in part by the EU CPSE Labs project funded by the H2020 program under grant agreement No 644400. We thank Hélène Waeselynck for her suggestions and advice during the preparation of this paper.

References

[1] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: Morse," in *IEEE International Conference on Robotics and Automation ICRA*, pp. 46–51, 2011.

[2] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154, 2004.

[3] Z. Micskei, Z. Szatmári, J. Oláh, and I. Majzik, "A concept for testing robustness and safety of the context-aware behaviour of autonomous systems," in *Agent and Multi-Agent Systems. Technologies and Applications*, pp. 504–513, Springer, 2012.

[4] X. Zou, R. Alexander, and J. McDermid, "Safety validation of sense and avoid algorithms using simulation and evolutionary search," in *Computer Safety, Reliability, and Security*, pp. 33–48, Springer, 2014.

[5] X. Zou, R. Alexander, and J. McDermid, "On the validation of a uav collision avoidance system developed by model-based optimization: Challenges and a tentative partial solution," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshop*, pp. 192–199, 2016.

[6] J. Arnold and R. Alexander, "Testing autonomous robot control software using procedural content generation," in *Computer Safety, Reliability, and Security*, pp. 33–44, Springer, 2013.

[7] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[8] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand, "Can robot navigation bugs be found in simulation? an exploratory study," in *IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2017.

[9] T. Sotiropoulos, J. Guiochet, F. Ingrand, and H. Weaselynck, "Virtual worlds for testing robot navigation: a study on the difficulty level," in *IEEE 12th European on Dependable Computing Conference EDCC*, pp. 153–160, 2016.

[10] J. J. Biesiadecki and M. W. Maimone, "The mars exploration rover surface mobility flight software driving ambition," in *IEEE Aerospace Conference*, pp. 15–pp, 2006.