Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project Proposal

Automated Test Generation for Robot Self-Examination

Salman Omar Sohail

Supervised by

Prof. Dr. Paul G. Plöger
Prof. Dr. Nico Hochgeschwender
Alex Mitrevski

May 15, 2020

1 Introduction

Humanity has entered the era of cyber-physical-systems in which robotics is at the heart of it all.

During the last 60 years robots have been continuously evolving and have expanded beyond the scope of industrial manufacturing which was its original purpose [7].

Robots are now being extensively used in domestic applications, agriculture, civic departments, etc.

Now more than ever there has been a sharp increase in the implementation of robots across different domains which mandates the increase of testing, verification, and validation is done in systems to ensure dependability.

In the past, tests in robotic software were coded only when necessary and without structure, and if it did follow any structure, it commonly followed the standard software engineering practices but in the recent decade, things have changed, specifically that now, independent testing methodologies exclusively for robotics have been developed like the robotic unit tests [2].

As new standard testing methodologies and systems are developed for robots [5] [6] [3], it must be kept in mind that importing practices from software engineering is simply not enough. Robots are complex and sophisticated machinery and testing them and their interaction with their environment requires additional verification and validation [6].

This R&D aims to facilitate testing of complex robotic systems by automatically generating a set of simulated test case scenarios in which a robot assesses its performance. The primary tasks that will be assessed are of robotic-manipulation (e.g. testing whether a cup has been successfully grasped in a generated scenario). The test case scenarios will be generated by the stochastic yet plausible placement of different objects like a cup in a random pose within a simulated environment. These generated test case scenarios will vary in complexity (e.g. one scenario might chain multiple tasks together for testing). The core tools that will be utilized include the 'Robot Operating System' (ROS) and 'Gazebo'. As for the actual tests, actions-tests will be implemented which means that it will check if certain actions have been completed (e.g. checking if the robot has picked a cup).

1.1 Motivation

One of the greatest challenges faced by users that wish to use a fully autonomous robot is the prevention of faults that disrupt the robot from accomplishing its task [10]; these faults can range from faulty actuators and sensors to wrong decisions given by the task planner.

The most reliable way to prevent these faults is by intensively testing the robot which consumes a lot of time and resources.

This R&D aims to enable the user's robots to examine themselves for these faults in a variety of test case scenarios that range from grasping actions to the execution of complex scenarios automatically; all within a simulated environment saving time and resources.

Additional benefits include exhaustive testing of the robots in a large variety of scenarios, testing the robots with different objects and environments which may not be accessible, testing edge cases and finally providing the training data required by machine learning algorithms to enables the robot to learn from failed scenarios and apply corrected behavior.

1.2 Problem Statement

This R&D proposal addresses the problem of how to expansively test a robotic-manipulators interaction with the environment to identify faults and enable the robot to learn from these faults. The problem is broken down into sub-problems as follows:

1. **Generating scenarios for the robotic manipulator:** The issue is that to test a robotic manipulator or robots in a simulator generally requires a significant amount of time. This is because each scenario for a robot is hand-crafted and when the need arises to test different scenarios, each of those scenarios have to be manually hand-crafted as well. Developing and implementing this module will ease the workload and save time for users who want to test their robots in different scenarios.

2. **Action-tests:** Whenever a robot acts, like grasping a cup or placing a bottle, it is desirable to ensure that it has actually performed that action,

this is where action tests come in. Action tests will be checking whether an action task has been successfully implemented or not. The current plan is to implement action tests for two use cases, first is the pick-action, and second is the place-action.

3. **Complex-scenario-tests:** When a robot performs a series of tasks in a complex environment, (e.g. like picking a cup and placing a cup from a cluttered table), then testing and detecting faults in it becomes significantly more difficult. Complex-scenario-tests will be utilizing a series of action-tests to evaluate whether the overall given task has been successfully achieved.

4. **Generating reports for machine learning algorithms that can be used for behavior correction:** The issue is that robots cannot learn from experience, due to which users have to hard code certain behaviors to prevent the same mistakes from recurring. This module can help users by generating reports from action-tests and complex-scenario tests that can then be used by machine learning algorithms (re-enforcement learning) to learn from failed scenarios and apply the corrected behavior for the next time.

2 Related Work

In this section we will review three highly relevant literature in which the researchers have used simulations to test their autonomous agents.

Fast temporal projection using accurate physics-based geometric reasoning [15]

The target of this research paper is to identify a set of valid positions and orientations for a mobile robot's base while it uses its manipulators for picking and placing objects by considering predictions from simulated events.

The approach being used by the paper is to create a projection of the future which is then analyzed to see if there is a violation that may disrupt the current solution. The projection for the future is done by a lightweight simulation. The projection generates a timeline for each of the different world states. From the

generated world scenarios if one solution is better than another, it is taken as the current solution. Each failure is analyzed and recorded to match with future references. This is done by selecting samples from a probability distribution. The environment is defined by a 3D representation. An inference engine goes through various predicates to attempt to find a legal solution.

The contributions of this research paper are that it presents a hybrid approach that utilizes physics-based reasoning and inference-based reasoning to generate a timeline for detecting faults in a robots planning model. This enables accurate and effective positioning of a robotic base while the manipulator arm picks and places objects. Additionally, it utilizes the generated timelines for path-planning and to create key positions for the robotic-manipulators trajectory.

Its shortcomings are that it is computationally expensive and requires a world database. This research paper is very similar to what we plan to achieve with our R&D. The key differences are that the approach used by this research paper has been developed specifically to detect faults in the planning model of a robot, additionally, it does not simulate the complete trajectory of the robotic manipulator but only the key positions. The proposed R&D will be developed not only for testing the planning model but detecting the point of failure in the robotic component as well, moreover, we plan to simulate the complete trajectory of the robotic manipulator as it can reveal additional points of failure.

Paracosm: A Language and Tool for Testing Autonomous Driving Systems [13]

This paper presents a programming language PARACOSM that creates realistic environment simulation based on the Unity physics engine for testing autonomous driving systems. PARACOSM provides support for dynamically changing test case scenarios and environments. It generates meshes that can be read by a car's camera for test cases. It utilizes the quasi-Monte Carlo method to ensure continuous parameter creation with pure randomness. PARACOSM is based on reactive objects i.e. reactive objects store geometric and graphical characteristics of an object. Each reactive object has its separate input stream, these streams provide the value of the environment object through sensors. Complex assemblies

are created by the assembly of smaller simpler objects. This engine has a key feature that allows for the detection of collision amongst the generated objects.

PARACOSM can do quantitative as well as qualitative analysis ensuring that because a solution is reached, it does not mean the autonomous driving system has passed the test. Test case scenarios are built both for static and dynamic reactive objects by the configuration of the input streams and variation of parameters. Test case scenarios using single deep neural network, history-oriented gradients, and (you only look once: YOLO) neural networks have shown promising results for PARACOSM.

The contributions of this paper are that it provides a programmable simulation framework that depicts the real world extensively for autonomous driving systems, additionally, it displays the coverage of its system by creating intricate test scenarios for the autonomous driving agent to go through.

The drawbacks are that no mechanic computes the deformation of the vehicle after the collision and the test case is simply treated as a failure. This research paper provides a strong guideline for the development of dynamic scenarios that will be utilized in this R&D.

Abstract Simulation Scenario Generation for Autonomous Vehicle Verification [14]

This paper proposes a system that creates simulation scenarios to test the decision-making capability of autonomous vehicles through approaches of hardware verification. The framework uses a language that takes a series of inputs and makes a logical driving scenario while rejecting illegitimate simulation scenarios. The modules within the framework of this paper utilize inputs to generate the required random scenario models in which the autonomous vehicle performs its assigned task and its decisions are recorded. From the recorded data of the simulation, certain key data are additionally recorded which include safety as well as scenario definition. The scenario models are created using a semantic language defined by the author which expresses certain parameters such as road generation, road types, parameter values, etc. The developed language is used as input and generates tokens that have additional attributes (e.g. B > Bend Road > attributes radius,

degrees). The collection of these tokens is then used to generate an XML file that keeps track of all the components within the simulation that are to take place. A validation is performed for checking inconsistencies and preventing invalid routes; this allows for a solid scenario formation and prevents wasting time during the simulation.

The contribution of this research paper can be summarized, as the development of a semantic language for defining scenes in a simulation and developing an approach for checking the plausibility of a generated scenario. Similar to [13], this paper opens up yet another approach for the generation of simulated scenarios and unlike previous research papers, it talks about testing and verification.

The drawback is that unlike autonomous vehicles, robots with manipulators have an additional layer of complexity and are much more challenging, so, the scenarios and testing required are different which is where this R&D comes into play.

Additional Literature

Other highly relevant works that use simulation for testing autonomous agents are [17], [12], [8], [4], and [9].

3 Proposal Approach

We plan to design and implement our proposed modules (i.e. random scenario generator, action-tester, and complex scenario tester) in a virtual replica of the Toyota Human Support Robot(HSR). We will be using Gazebo as our simulation environment and ROS as our application interface with the robot. Our planned approach for each module is as follows:

3.1 Random Scenario Generator

The objective of the random scenario generator is to automatically generate scenarios by placing objects in different positions and orientations. The starting point for how we plan to achieve this is by initially expanding the existing 3D model

database. This will be done to allow for more variety of test case scenarios. The next step is to design and implement a parameter randomizer to create diverse scenarios. For this we may look and use techniques similar to [1]. A challenge that can be expected is to place the objects in a plausible manner (e.g.a spoon next to a coffee cup). A possible idea is that the random scenario generator module is linked with the ontology which defines the relationship between different objects.

3.2 Action-Tests

The action tests will be developed to assess whether a robot has completed an action that it was assigned (e.g. picking a cup). This will be implemented in a very similar manner to property-based tests. Our work may closely follow the approach used by [16]. These tests will be developed for Lucy the Toyota(HSR) robot. The ideal situation is that we can implement an action test for every action but this R&D, we will be focusing on implementing the action tests for picking action and placing action.

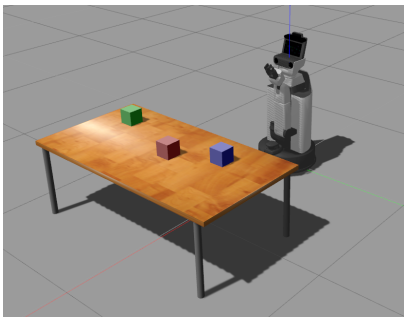

Figure 1: Lucy in a simulated world (Gazebo) with three cubes placed on a table [11].

3.3 Complex-Scenario-Tests

The objective of complex scenario tests is to assess whether our robot Lucy can perform complex tasks successfully. We have a two-step plan for this module. Step-I is to increase the number of objects generated by the random test case scenario generator (e.g. instead of placing only one object on the table with a random pose, the random scenario generator will place multiple objects on the table with different poses creating a cluttered table). Step-II is to then to let Lucy perform a series of tasks like picking and placing a bottle from the cluttered table and assessing whether she has successfully achieved her overall task. To increase the complexity of step-I, we may even add objects in a complex manner (i.e. a spoon inside of a glass).

3.4 Generating reports for machine learning algorithms that can be used for behavior correction

This module aims to generate a report after Lucy has completed a task or a series of tasks. The generated report will be based on the action-tests as well as the complex-scenario-tests. Additionally, it will provide information on the environment and objects. The goal of this report is to utilize a machine learning algorithm(re-enforcement learning) to asses the report and learn from good and bad examples. The time frame of this project may not allow us to reach this point; nevertheless, it is part of our intended objective.

3.5 Use Cases

These are the use cases by which we intend to test Lucy:

1. **Picking an object from the table:** In this use case, Lucy will be tasked to pick an object. The object will be placed as prescribed by the random scenario generator module. The object placed will vary, it may be any type of item like a spoon or a cup. The action tests will assess whether Lucy has picked up the object or not. This will be the initial basic use case.

2. **Picking an object from a cluttered table:** In this use case, Lucy will be tasked to pick an object from a cluttered table. The objects will be placed as prescribed by the random scenario generator module. The objects will be different from each other. Similar to the initial use case, the action tests will assess whether Lucy has picked up the object or not.

3. **Picking and placing an object on the same table:** In this use case, Lucy will be tasked to pick and place an object. The object will be placed by the random scenario generator module. The object placed will vary, it may be any number of items like a spoon or a cup. The action tests will assess whether Lucy has picked up the object or not. The complex scenario test will assess whether the complete task has been completed. This will be the advanced use case.

3.6 Tools and Environment

- Device Under Test(DUT) - Toyota's Human Support Robot(HSR)
- Simulation Environment - Gazebo
- Robot Application - Robot Operating System (ROS)

4 Project Plan

4.1 Work Packages

The bare minimum will include the following packages:

WP1 Literature Review

This work package will aim to extensively search for related literature focused on simulation generation and robotic testing.

T1.1 Search for papers related to the project.

In this task the aim will be to search for papers related to the on simulation generation and robotic testing and what types of approaches they used.

T1.2 Analysis of reference list.

In this task we will go through the reference list and shortlist all the relevant research papers.

T1.3 Analysis of state of the art.

In this task we will go through the reference list and analyze all the state of the art.

WP2 Development package.

This work package aims to provide the required working components for the random scenario generator, action-tests, and complex scenario tests.

T2.1 Random scenario generator.

In this task we will design and implement a random scenario generator in which the robot is required to perform a set assignment in a simulated environment (gazebo).

T2.2 Action-tests.

In this task we will design and implement an action test in which the actions of the robotic manipulator will be tested.

T2.3 Complex scenario tests.

In this task we will design and implement complex scenario tests in which several action tests will be performed by the robot in the randomly generated scenario.

WP3 Project Report.

This work package involves writing the project report. This work package will be scheduled in parallel to all the previous work packages.

4.2 Milestones

M1 Literature search and Analysis - 31/Jul/2020

M2 Design and implementation of the random scenario generator - 15/Sep/2020

M3 Design and implementation of the action testing - 15/Oct/2020

M4 Design and implementation of the complex scenario tester - 15/Dec/2020

M5 Report submission - 15/Jan/2021

4.3 Project Schedule

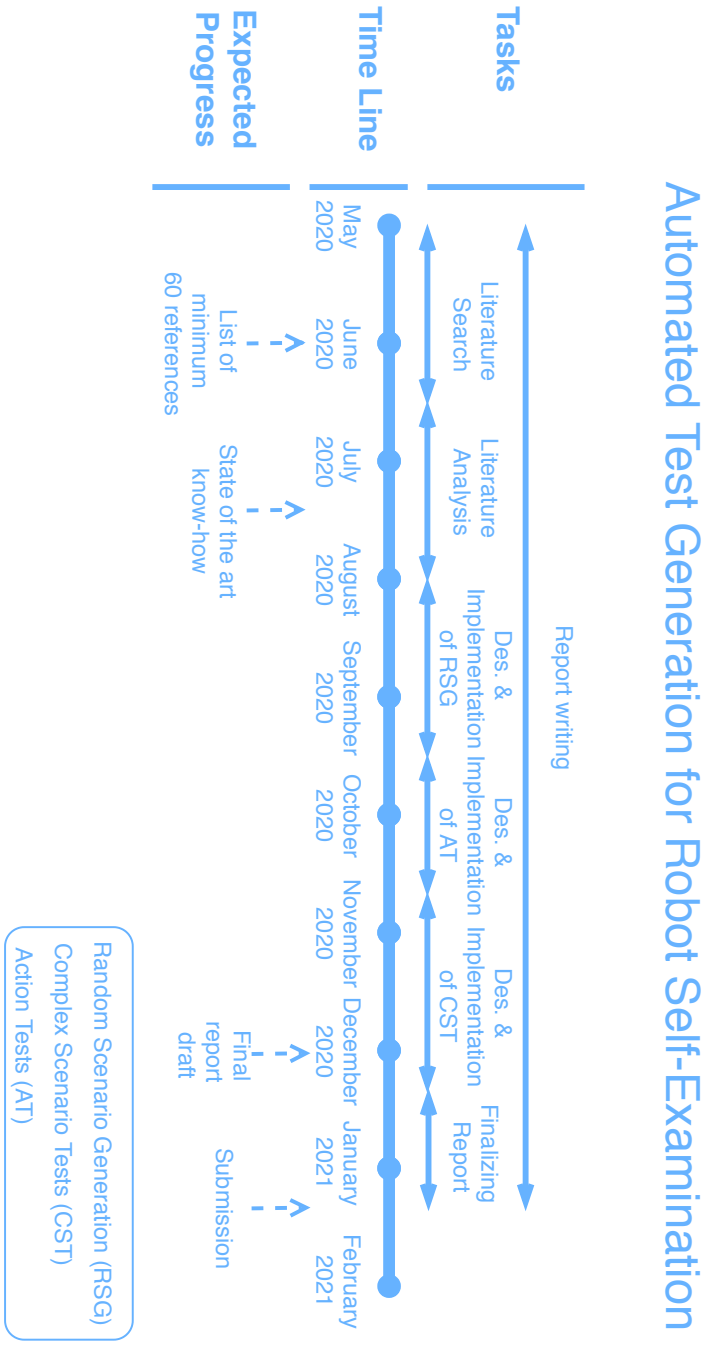

Figure 2: Project Schedule

4.4 Deliverables

Minimum Viable

- A survey on the related literature
- A detailed analysis of the state of the art.
- A fully implemented random scenario generator.

Expected

- Design and implementation of at least one action-test.
- Design and implementation of complex scenario tests.

Desired

- Generating reports from action-tests and complex-scenario-tests for each task that the robot has performed which can then be used by machine learning algorithms to teach the robot.

References

[1] James Arnold and Rob Alexander. Testing autonomous robot control software using procedural content generation. In Friedemann Bitsch, Jérémie Guiochet, and Mohamed Kaâniche, editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8153 LNCS, pages 33–44, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 9783642407925. doi: 10.1007/978-3-642-40793-2_4.

[2] Andreas Bihlmaier and Heinz Wörn. Robot unit testing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8810, pages 255–266, 2014.

[3] Andreas Bihlmaier, Fabian Stein, and Heinz Wörn. Towards a generic BRDF/BTF measurement system: Improving visual realism in robot simulators using robots. In *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies, EmergiTech 2016*, pages 410–416, 2016. ISBN 9781509007066. doi: 10.1109/EmergiTech.2016.7737376.

[4] Gilberto Echeverria, Séverin Lemaignan, Arnaud Degroote, Simon Lacroix, Michael Karg, Pierrick Koch, Charles Lesire, and Serge Stinckwich. Simulating complex robotic scenarios with MORSE. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7628 LNAI, pages 197–208, 2012. ISBN 9783642343261. doi: 10.1007/978-3-642-34327-8_20.

[5] Juhan Ernits, Evelin Halling, Gert Kanter, and Jüri Vain. Model-based integration testing of ROS packages: A mobile robot case study. In *2015 European Conference on Mobile Robots, ECMR 2015 - Proceedings*, pages 1–7, 2015. ISBN 9781467391634. doi: 10.1109/ECMR.2015.7324210.

[6] Vladimir Estivill-Castro, René Hexel, and Carl Lusty. Continuous integration for testing full robotic behaviours in a GUI-stripped simulation. In *CEUR Workshop Proceedings*, volume 2245, pages 453–464, 2018.

[7] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez De Santos, and Manuel Armada. The evolution of robotics research. *IEEE Robotics and Automation Magazine*, 14(1):90–103, 2007. ISSN 10709932. doi: 10.1109/MRA.2007.339608.

[8] Patrice Godefroid, Nils Klarlund, and Koushik Sen. DART: Directed automated random testing. In *ACM SIGPLAN Notices*, volume 40, pages 213–223, 2005. doi: 10.1145/1064978.1065036.

[9] Bhargav S. Gulavani, Thomas A. Henzinger, Yamini Kannan, Aditya V. Nori, and Sriram K. Rajamani. SYNERGY: A new algorithm for property checking. In *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 117–127, 2006. ISBN 1595934685. doi: 10.1145/1181775.1181790.

[10] Abdelfetah Hentout, Aouache Mustapha, Abderraouf Maoudj, and Isma Akli. Key challenges and open issues of industrial collaborative robotics. In *RO-MAN 2018: Workshop on Human-Robot Interaction: From Service to Industry*, number August, 2018.

[11] Chia-Man Hung. ascane/gym-gazebo-hsr, 2018. URL `https://github.com/ascane/gym-gazebo-hsr/tree/master/images`.

[12] Jongwoo Kim, Joel M. Esposito, and Vijay Kumar. Sampling-based algorithm for testing and validating robot controllers. *International Journal of Robotics Research*, 25(12):1257–1272, 2006. ISSN 02783649. doi: 10.1177/0278364906072513.

[13] Rupak Majumdar, Aman Mathur, Marcus Pirron, Laura Stegner, and Damien Zufferey. Paracosm: A language and tool for testing autonomous driving systems. 02 2019.

[14] Christopher Medrano-Berumen and Mustafa Akbaş. Abstract simulation scenario generation for autonomous vehicle verification. 04 2019.

[15] L. Mösenlechner and M. Beetz. Fast temporal projection using accurate physics-based geometric reasoning. In *2013 IEEE International Conference on Robotics and Automation*, pages 1821–1827, 2013.

[16] André Santos, Alcino Cunha, and Nuno Macedo. Property-based testing for the robot operating system. In *A-TEST 2018 - Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, Co-located with FSE 2018*, A-TEST 2018, pages 56–62, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360531. doi: 10.1145/3278186.3278195. URL `https://doi.org/10.1145/3278186.3278195`.

[17] Kenta Takaya, Toshinori Asai, Valeri Kroumov, and Florentin Smarandache. Simulation environment for mobile robots testing using ROS and Gazebo. In *2016 20th International Conference on System Theory, Control and Computing, ICSTCC 2016 - Joint Conference of SINTES 20, SACCS 16, SIMSIS 20 - Proceedings*, pages 96–101, 2016. ISBN 9781509027200. doi: 10.1109/ICSTCC.2016.7790647.