Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Automated Test Generation for Robot Self-Examination

Salman Omar Sohail

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

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

January 2021

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Salman Omar Sohail

Abstract

Acknowledgements

Contents

List of Figures

List of Tables

List of Abbreviations

Acronym	**Full Form**
AST	Adaptive Stress Testing's
CEM	Cross-Entropy Method
CNN	Convolutional neural network
DANN	Domain–Adversarial Neural Networks
DRL	Deep Reinforcement Learning
F-RCNN	Faster Region-based Convolution Neural Networks
HMM	Hidden Markov Model
HRI	Human-Robot Interaction
HSR	Human Support Robot
LRCN	Long Term Recurrent Convolutional Network
LSTM	Long Short Term Memory
MC	Monte Carlo
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
ML	Machine Learning
NN	Neural Network
RL	Reinforcement Learning
RNN	Recurrent Neural Network

1

Introduction

Autonomous robots are robots that are capable of automatically and independently performing tasks [54]. The tasks they perform are dependent on the application and context. A few examples would be that in the manufacturing industry, robots perform tasks such as assembly, welding, machining, etc. In the aerospace industry, robots perform tasks like painting, drilling, and assembly. Similarly, in the domestic sector, robots are being used as automated vacuum cleaners, cooks, and as health care aides.

For the past five years, countries like the USA, China, Germany, and South-Korea have steadily increased their purchase and deployment of autonomous robots. In 2020 alone, there has been a 10% increase in robot purchases [23]. As more and more robots are being employed industries, several questions come to mind, such as, are these robots safe? How susceptible are these robots to faults and failures? If these robots do have faults and failures, is there an impact on the safety of human lives, and if not, is there an economic impact? An answer to these questions may be found in a survey conducted from 2007 to 2013 on robot-assisted surgery [61]. This survey informs us that, from the 1.7 million robot-assisted surgeries performed, 144 patients died, and 1,391 people were injured. Unfortunately, the robots were the direct cause of some of the deaths due to malfunction, wrong movement, electrical failure, etc. In this survey, the critical point that we would like to highlight is not the 99.52% success rate but the cost of the missing 0.48%.

The most reliable way that users (industrialists, researchers, and hobbyists) have adopted to prevent faults in autonomous robots, is by intensively testing them. The faults they test, range from faulty actuators and sensors to wrong decisions given by the task planner. However, testing consumes a lot of time and resources, but as indicated in [61], it is a very vital step. To speed up and perform exhaustive testing, users have started to use simulators to test their robots for faults. However, performing a few tests with a robot is insufficient at most times. This concept holds true both for field tests and simulated tests. A report [8] states that when the ExoMars rover crash-landed in 2016, the malfunction was reproducible in their simulator when provided with the same input data. This report demonstrates that if proper test generation approaches were applied to the ExoMars using a simulator, then, the crash-landing could have been avoided. So testing alone is not enough. The right kind of testing is required; it has to be both expansive and exhaustive.

Researchers are working on new standard testing methodologies and systems for robots [15] [17] [1], however, as aforementioned, the right kinds of tests are required. 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 [17].

Our 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 [24]; these faults can range from faulty actuators and sensors to wrong decisions given by the task planner. Faults that occur in robotic manipulators are especially deadly as indicated by [61].

The most reliable way to prevent these faults is by intensively testing the robot. This process of testing is time and resource intensive. To accelerate this process for safer and more dependable robots, we propose an R&D project in which we enable the user's robots to examine themselves for 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.

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 developing the base infrastructure which may then be adapted for use by machine learning algorithms to enable the robot to learn from failed scenarios and apply corrected behavior.

1.2 Challenges and Difficulties

After a comprehensive review of the state of the art, we have identified a number of challenges within the domain of robotics pertaining to testing in simulations. We enumerated the top 5 as follows:

1. It is difficult to generate complex and valid test case scenarios for robots especially for dynamic environments [24].

2. It is more difficult to assess robotic faults when they perform complex tasks due to dynamic environments and spatial-temporal information of objects [48][62].

3. Dynamic-agents introduce a new paradigm of complexity and faults for robots [17].

4. Robots that are tested in simulators are susceptible to faults that may occur only in simulation and not in reality [4].

5. Current testing methodologies that are in use are neither expansive nor exhaustive both for field testing and simulated testing [4].

1.3 Problem Formulation

This R&D proposal addresses the problem of how to expansively test robotic-navigation as well as robotic-manipulators interaction with the environment to identify faults and enable the robot to learn from these faults. We address challenges 1, 2, and 5 from §1.2. We will be answering the following research questions:

R1 How to generate extensive and varied simulated scenarios for robots?

R2 How to qualitatively test robotic interaction with the environment?

R3 How to test a robot when it performs complex tasks in complex-scenarios?

We break our research questions into sub-problems as follows:

1.3.1 Scenario-Generation

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 [24]. Developing and implementing this module will ease the workload and save time for users who want to test their robots in different scenarios.

1.3.2 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, additionally when a robotic action does fail, more often than not, it is due to simple errors in robotic software [62]. 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.

1.3.3 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 and complex due to factors like the environment and spatial-temporal information of objects [48]. Complex-scenario-tests will be utilizing a series of action-tests to evaluate whether the overall given task has been successfully achieved.

1.3.4 Behavior Adaptability

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 (reinforcement learning) to learn from failed scenarios and apply the corrected behavior for the next time.

2

State of the Art

In this chapter, we present a meta-analysis of literature concerning automated test generation. The objective of this section is to shed light on the innovative and core approaches used by academics and industrialists for testing automated systems. To achieve this objective, we have covered the most relevant and influential research papers pertaining to automated test generation.

2.1 Automated Test Generation for Robot Self-Examination

With the rapid integration of autonomous robots into the industries and households, there is an ever-growing requirement for heightened safety and dependability from these robots. To uphold these requirements users (researchers, designers, and industrialists) vigorously test and validate their systems before deployment. Currently, the most common approach employed for the verification and validation of autonomous systems are done in the real world, however, there has been a shift geared towards simulation due to the attractive features it provides (e.g. scalability, cheapness, speed, and safety) [57]. To further expedite the process of verification and validation, researchers have begun automating tests for autonomous robots. We formally define automatic test generation for robots as: *The process of automatically generating a set of test cases to assesses (A) a robot's task performance. (B) a robot's software. (C) a robot's interface between the software and the hardware.* A good example of automatic test generation is given by [38] in which they test the software as well as the hardware components of a robot using automated unit tests, system tests, and integration tests. More advanced work for automated testing that are similar to this, are given by [35], [9], [4], [59], [52] etc. A quick overview of the general issues faced by researchers who develop automated test suites for their systems are as follows:

1. Robots in a simulator generally require a substantial amount of time for defining and designing a scenario. This as mentioned in §1.3.1 is because each scenario for a robot has to be hand-crafted and when the need arises to test different scenarios, each of those scenarios have to be manually altered as well [24].

2. Expansive and exhaustive testing for robots is a complex challenge [4], mainly because not only do tests have to cover the software aspects of the robot but also the overall behavior and decisions that a robot makes.

3. Detecting faults and failures in the performance of a robot is difficult due to factors like the environment and spatial-temporal information of objects [48].

Once proper approaches and frameworks have been developed to resolve these issues, consumers of both in commercial and residential sectors will have much more safer and dependable robots. As for the industries that develop these robots, they will require less time and effort in testing and validating their robots. The consequence of insufficient research in the development of automated test frameworks may lead to costly and deadly failures similar to that of the Mars rover incident [8].

We will now discuss some of the prominent works that are focused on developing and testing autonomous systems in simulations. To accomplish this, we have divided the central theme of autonomous testing into three sub-themes. The first sub-theme is scenario generation, the second sub-theme is action-tests, and finally, the third sub-theme is complex-scenario-tests. The first sub-theme 'scenario generation' is focused on the development of test case scenarios for autonomous systems (e.g. development of dynamic environments in simulation). The second sub-theme 'actions-tests' are focused on testing primitive actions of an autonomous system (e.g. grasping action). The third sub-theme 'complex-scenario-tests' is focused on advanced tests that assess the overall behavior of an autonomous system when performing a set of primitive action tests. For each work, we will investigate the objective, motivation, proposed approach, test cases, results, contributions, limitations, and relevance.

2.2 Scenario-Generation

A scenario is classically defined as the execution of tasks performed by an agent to achieve an objective [65]. Scenario generation refers to the creation of tasks for an agent either in the real world or in simulation. In our context, we define scenario generation as the process of setting up an environment and assigning tasks to an agent in a simulation. In this section, we investigate the state-of-the-art methodologies employed for simulated scenario generation for autonomous systems, in addition to past relevant literature. The objective is to gain meaningful insights and possibly integrate it into our development strategy.

We start with [62] in which Timperley and fellow co-authors address whether simulation software is sufficient enough to express reality in a way that bugs within the software of an autonomous system can be identified. The key argument for the investigation of simulation is to quickly and effectively identify all types of bugs within the robotic software. This is achieved by coupling simulation with automated testing. The overarching goal of the authors is to develop techniques for automated testing of robotic software to decrease the cost of robot deployment by identifying bugs in robotic software during simulations. The authors hypothesize that the investigation of the history of bugs that may occur in open source robotic simulation software leads to a deeper insight into the root issue of bugs. Once the general root issues are identified, techniques and approaches can be developed to address these issues. These techniques and approaches can then be applied to identify identical problems and issues in different robotic software. Several datasets that contain robotic bugs exist, however, none can be utilized to reproduce the faults within the simulation. The authors worked on analyzing the 228 commits that were made to correct bugs for the ARDUPILOT project (autonomous navigation robotic software). The objective of the analysis was to uncover the underlying issues with which could be detected and triggered by simulation.

The methodology employed by the authors to identify the bug fixes from the 29,000+ commits in the ARDUPILOT project, is the use of GITPYTHON to mine bugs. The authors used a filter script in the version history to identify all relevant bug fixes using a set criterion as mentioned within the research paper. After the identification of the bugs. Each commit was manually analyzed based on a set number of criteria such as, is it triggered by hardware, does it occur due to another event, etc. The result of the research paper is as follows 10/228 bugs were caused due to hardware, 13/228 required a previous event to trigger the bug, 9/228 bugs occur due to input by the user while 219/228 occur due to pre-programmed routines and ground control systems, 20/228 bugs occurs in initialization, 10/228 occur in during fail-safe and recovery, 81/228 bugs depend on a specific static, dynamic or both static and dynamic configuration, 22/228 bugs occur due to a specific environment and finally 23/228 bugs are harmless, while the rest are not. The key point they identified was that simulation bugs did not depend on the environment or hardware. Instead, the bugs depended on fixed-configuration and inputs of the system. The contributions of this paper are that it provides support for the claim that simulations suffice as valid and effective platforms for testing and verifying faults and bugs within autonomous systems. Research papers like [53] refer to reinforce their claim of validating testing and verification of autonomous systems based on result analysis. The deficits of this research paper are that the techniques presented by this research paper are insufficient [53]. Where [53] presents a software fault localization oracle for identifying bugs in a control-cyber physical system. It highlights that although this research papers framework (test generation for bug identification) [62] has its place and benefits in test oracles, yet, it is insufficient for autonomous navigation and that their approach covers this deficit. Another deficit is mentioned by the authors themselves is that their work focuses on the result of a single software simulation, hence its scope is limited. However, the majority of users use ARDUPILOT and Gazebo as software simulators, hence, this work is not redundant and the results are still viable according to the authors.

Akin to the research of [62], Sotiropoulos and co-authors present a research paper [57] that aims to perform a qualitative exploratory study on navigation software to identify bugs. The motivation of their work is to come up with a test selection and test oracle for autonomous robots. The test selection mechanism utilized is a sub-space in the simulator, in which the characteristics and decisions generated by a robot performing a task are tested by a test oracle. The authors hypothesize that the reproduction of bugs does not require sophisticated physical phenomena to be replicated within a simulator. The contribution, if this hypothesis is proven to be true, will mean that low-fidelity simulators are sufficient enough for testing. This hypothesis is broken down into the following sub-problems:

- Reproducibility of bugs in navigation software via activators.
- Necessary activators required for triggering bugs in either the world environment or the task being performed by the bot.
- Necessary activators required for triggering bugs in either the robot and navigation algorithm.
- Based on the analysis, which oracles may be considered?

To test this hypothesis, the authors perform the analysis for each of the software P3D, LibP3D, DTM, and POM. All combined have 356 commits and the analysis was performed using a form that relates

to 6 aspects of the commit. Aspect number 1 is where is the location of the fault, aspect 2 is, what type of fault is it, aspect 3 is, failure, aspect 4 is, the time required to fix, and finally aspect 6 is, is it reproducible. The validation of aspect 6 was done by inserting the identified bug in the latest version of the software. The result of the analysis was 33 critical bugs were found, of which most of them were related to incorrect data assignments, un-freed dynamic memory allocations, incorrect algorithm implementations, and interface implementation. The result of the analysis was that all but one bug was reproducible in a low-fidelity, which occurred due to the mechanical vibration in the robot and it requires a high-end simulator to capture this behavior. A parameter that seemed to be quite vital was the inertial parameter. Overall, the result was that most bugs will be detected in low-fidelity simulators with the addition of certain parameters like inertia. The key in all reproducibility was the input configuration parameters. Based on the results, the authors recommended some of the suggested property tests:

- Requirements for missions: This will check certain parameters like alignment required for reaching the final destination or the quality of the alignment.
- Threshold: This will check the speed of the robot and prevent excessive turning of more than 180 degrees.
- Catastrophic events: Checks if the robot has gotten itself into a position it cannot recover from, like falling into a hole.
- Error reports: Stops the robot when an error is detected. Perception requirement: Detection of correctly perceived positions.

The contribution of this research paper is that it provides and in depth-view of the triggers and effects of bugs in navigation software. Additionally, it suggests intuitive verification and validation methods; it highlights the importance of the configuration files and archiving these files. The discrepancies are that the data analysis focuses only on academic material and not on the industrial material. Furthermore, it concentrates on known and resolved bugs and not the difficult and unknown bugs. The analysis is based on non-formal field test procedures. Due to the use of manual analysis, some bugs may have been left out and fault injection used for reproducing the error was done on the latest version instead of the original deprecated version. The key findings of the paper are that complex simulators are not required for the reproduction of most bugs, one-third of all bugs identified were non-domain related and they can be resolved by tools like Valgrind, additionally, most bugs were diverse and could only be reproduced with a concrete and precise reference.

With sufficient drive for using simulators as a fault detection mechanism, we move on to existing research of scenario generation with simulators. We start with [50] in which Park and co-authors propose a hybrid approach for multiple-event scenario generation intended for autonomous vehicle systems using two deep learning approaches (Faster region-based convolution neural networks F-RCNN and Convolution neural network CNN). The motivation of their work is that despite the abundance of research on deep neural networks, most lack the capability of tracking multiple events in a single video session and the ones that can do it are primitive and are not advanced. Due to which there is a great benefit in an approach

that utilizes computer vision that analyzes the real world and using deep learning generates a wide range of test scenarios without manually modeling them. The approach used by Park for the Multi-event-based scenario generation is as follows: The initial step is to excerpt the main event from a video using F-RCNN. The excerpted event is analyzed by using a long term recurrent convolution network (LRCN) which is a classifier. This classifier generates the self-driving scenarios that will be used as an input for the simulator. The objects that are interacting with others are classified as a single event i.e. objects that have their bounding box touching each other are integrated by an algorithm into a single event. The video-based learning was done using Keras on 725 videos and the simulation was conducted by using the author's unity-based application for testing autonomous vehicles. From the analyzed 725 videos, 23 classes were created which included working of vehicles, people, animals as well as how they interacted with each other. During the experiment, once the high-level features were extracted, the next step was to pass them on to the LRCN classifier. LRCN classifies the data utilizing the convolution neural network CNN that extracts a set of features as well as learning them with long-short term memory LSTM. This is repeatedly done until a scenario is created from the multiple events that have been processed which is then ported into the simulator. Evaluation was performed using cross-validation comparing the actual values with the classified values. 600 videos were used for training and 125 were used for testing and the accuracy of the implemented framework (LRCN inception v3 + Full area) was 95% which is an improvement on the standalone LRCN and LRCN full area used networks which in similar conditions had an accuracy of 78% and 80% respectively. The contribution of this research paper is the successful development and implementation of a framework that performs analysis of real-world cases using deep neural networks; based on that develops testing and training scenarios for autonomous vehicles. Additionally, proposing a methodology for dealing with the deficit of single event analysis by using the developed framework for tracking multiple objects in an event enabling the ability to produce complex and advanced test case scenarios. The deficiency in this work is that generated complex scenarios lack robustness and require ample time to model.

Attempting to solve a similar issue are Medrano and co-authors [42] in which instead of multiple-event scenario generation they seek to generate coherent scenarios for autonomous systems. They [42] propose 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 work 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 [41], 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 our R&D comes into play.

As can be seen, working with a simulator for the generation of a coherent scenario is difficult which is why Majumdar and co-authors [41] present 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 a 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.

An interesting work by Wang and co-authors [64] present an approach for testing sensor simulation using shader programming in combination with LiDAR and Radar sensors, furthermore, they investigate the viability of sensor simulation. The motivation of their research is to accelerate the development of simulated sensor technology. The proposed approaches use the Velodyne and IBEO Lux sensors. A software interface then takes the sensor readings from these sensors and maps it into a radio intensity grid map (via sampler3D software), The data acquired is then used for shading. The result of their research work is that compared to the ray tracing tech., shaders can produce better results when there are fewer amount of lasers to work with. Although, it should be noted that there are significant computational costs associated with both ray tracing and shaders, hence a deeper investigation may be required when more powerful computational resources are available in the future. The contributions of their work are the proposition and demonstration of simulated LiDAR sensor tech. with shaders as well as a comparative review of ray tracing and shaders in terms of simulated sensors. The limitations of are that it suffers the

same issues as all simulated radar technologies which is the inability to effectively simulate realism as well as computational limitations and the requirement of highly detailed CAD models for the entire system. The work they present is quite unique as we have come across very few literature that focuses on the verification and validation of sensors in a simulation.

Abey and co-authors [2] present an approach for the automation of testing and validation of self-driving autonomous vehicles in a multi-agent system using reinforcement learning (RL) in conjunction with simulators. The main motivation of their research is to assist in increasing the reliability of autonomous vehicles by creating sundry and effective scenarios for testing. The proposed methodology revolves around using 'Bayesian Optimization' (BO) for the optimal path parameters selection of pedestrians and automobiles to create stimulating adversarial scenarios. The optimal parameters are acquired using reinforcement learning. Unreal Engine 4 has been used as the simulation engine and CARLA has been used for evaluating the collision events. The assessment of the proposed approach has proven to be effective in reducing the number of accidents in the simulator; it is much more easier for generating scenarios as compared to hand-crafting them. The BO approach as compared to Cross-Entropy and Randomized parameter approaches has been observed to produce more fruitful results in terms of reduced accidents and the effect of these accidents. The contribution of this research paper is that it provides a novel approach for generating automated testing scenarios for autonomous vehicles. The discrepancies are the inability to scale the simulation due to hardware limitation, the failure to involve the use of normal multi-agent scenarios (this paper produces automated tests only for adversarial cases), and finally the capability to select parameters keeping in view the entire relevant state space.

Kim and co-authors [31] present an approach for testing industrial-grade autonomous systems by the use of simulation. The motivation of their work is to tackle the issue of the growing complexity and restrictions faced by industrial cyber-physical systems (CPS). The proposed approach revolves around developing a specialized test bench for the CPS in which all possible test cases are covered by employing the cause-effect graphing technique. The novelty in their work is the focus on the integration of legality and environment when designing the test bench. Test case studies were performed on different drones with the use of the proposed approach. A noteworthy point was the integration of legal restrictions like the amount of time the drone was allowed to fly, the maximum and minimum altitude allowed, etc. The evaluation criteria for these test case studies were through a comparison of the generated test report with manual observation of the actual scenario. The results of the project were observed to be successful in terms of the defined criteria. The contribution of this research paper is the inclusion and focus of legality when developing tests for an industrial CPS. The limitation of this work is that the simulation data does not translating well into reality (i.e. the simulation data is not completely reliable).

Bousmalis and co-authors [10] present the effectiveness of the mutual effect of domain adaptation and simulation with a monocular vision approach. In simple terms, the authors evaluate whether the integration of monocular RGB images from a simulation enhances the grasping capability of a robotic manipulator using a Convolutional Neural Networks (CNN) like domain–adversarial neural networks (DANN). The motivation of their works is to enhance and overcome the difficulty of utilizing an autonomous robotic manipulator to grasp sundry objects. The approach used by this paper is grounded on the pixel-level

DANN, whereas pixel-level domain adaptation simply means that a generator function is learned in which the source domain is passed to the target domain in the first level of the neural network i.e. the input layer. This altered and adapted model is named as GraspGAN. The methodology is based on predicting the grasping action of the robot. The grasping is done using visual data from monocular RGB images and the relative position of the end-effector. The learning of the DANN is founded on spatially segregated visual images. The prediction probability is achieved by the cross-entropy method (CEM). The visual images obtained are from both the simulation as well as the real-world data. The results of their work clearly shows that the introduction of simulated data substantially enhances real-world vision-based grasping actions. Additionally, it is observed that the objects selected for training are not required to be real models based on the evaluation.

The contributions of their work are as follows:

1. A detailed assessment of simulation to real-world transfer.
2. Adapted and improvement of monocular RGB images.
3. An original effective demo of simulation to reality for monocular vision-based grasping.

The deficit of their research is that the images used from the simulation are altered to make them seem closer to real images. Moreover, this work is based solely on monocular RGB images.

Fermont and co-authors [18] present an original software language for the development and examination of perception systems that employ machine learning techniques. This language is based on probabilistic models and is termed as a SCENIC and the strength of it lies in its ability to generate specialized features in a scenario by applying constraints using a probability distribution. A test case was studied in which SCENIC analyzed and enhanced the accuracy of a deep neural network perception system for an autonomous vehicle via simulated data. The contribution of their research is the performance leap (as compared to the current state-of-the-art) of car detection with a fixed training data set of sim data. In addition to this contribution, it also provides a dynamic platform for generating scenarios for autonomous vehicles.

Hauer and co-authors [22] propose a statistically guaranteed methodology to check whether all the scenarios that an autonomous vehicle can encounter with the provided data has been tested or not. The motivation of their work is to ensure sound testing of autonomous vehicles in all possible scenarios. The working of the presented approach is based on using known types of scenarios to generate alternate scenarios using the Monte Carlos approach. The goal is to view each type of scenario at least once. After generating a sufficient number of scenarios, they are clustered either by using machine learning techniques or manually. The clusters are then used to generate a histogram which is then used to calculate a probability distribution. This procedure is iterated and if no new clusters are obtained, then based on the defined confidence interval, it is said the list of scenarios is complete. The test cases studied in this research paper are high-way driving scenarios and city driving scenarios. From the result of the test cases, a 99% scenario coverage is guaranteed. The contribution of their research is the proposition and demonstration of obtaining coverage for all of the driving scenarios based on the provided information. The issue with their research is the assumption of independence between different clusters of driving

scenarios. Furthermore, their work is highly dependent on the provided information meaning that a mountain driving scenario cannot be used to discover open-country driving scenarios. summary

2.3 Action-tests

We define action-test as a framework for testing the actions performed by an autonomous system, in our case being a Toyota human support robot (HSR). The test framework is grounded upon fundamental testing concepts like unit tests [21], action-driven tests [37] as well as advanced techniques such as property-based testing [40]. In this section, we investigate the state-of-the-art methodologies employed for testing the actions of autonomous systems. The objective is as with scenario generation, to gain a meaningful understanding of cutting-edge test practices used by fellow researchers and incorporate it into our development strategy.

Kunze and co-authors [35] present an implemented approach for predicting the effects of robotic manipulators actions based on partially simulated actions. The motivation for their work is to enhance the success rate of the tasks performed by a robotic manipulator; avoiding the heavy penalty of computation and time. The proposed approach is to formulate any given task into a parameterized simulation scenario. When a simulated scenario is initiated, all data acquired is stored. The data that is stored contains the states and time of all entities within the simulation. Alongside this process, a separate domain knowledge base is defined that details task-related information. The task-related information contains the physics for each entity (i.e. a 3D modeled egg will contain additional information like mass, viscosity, fragility, shape, etc.) as well as information between the different objects. To start the process, the simulated scenarios are initiated with varied parameters, which are logged as aforementioned. The logged data structures are then interpreted into first-order representations that are segregated by time-steps. These first-order representations are called timelines. A PROLOG then goes through the timelines and selects the action to be executed. For the test case scenarios, the authors used the PR2 robot developed by Willo Garage [12]. The test object investigated were eggs and liquids. Some of the task case scenarios were for the robot to successfully pick up and hold an egg without breaking it, breaking an egg against another object, picking up a liquid vessel, and pouring its contents into another liquid vessel, and a few more similar scenarios. Positive results were achieved for most of the test case scenarios, however, some cases required further extensions and tuning to be successful. The contribution of their research work is that it introduces a hybrid approach (i.e. using both first-order representation and parameterized physics) for predicting the actions of a robotic manipulator. The limitations of their work are the high amount of workload required to define parameterized physics models and environments. Also, it depends on the realistic simulation to acquire feasible results. Nonetheless, their research paper provides valuable insight on overcoming limitations in first-order representation when it comes to naïve physics.

Bihlmaier and co-authors [9] propose robot unit tests (RUT) as a means to automatically test robots in a simulated environment. The focus of the paper is on presenting a novel testing methodology. It is founded upon two successful prior testing methodologies that are as follows: The first of the two test methodologies that are discussed talks about test-driven development (TDD) in which tests are written before the actual code. The second test methodology discussed continuous integration (CI) which talks

about tests that provides a high degree of coverage to the automated software. The motivation of this research paper is that the crux of enhancing a software system is through testing which is often missing in robotic software. For the demonstration of the methodology employed by the paper, a simple robot (Sir) is used. Sir uses the information acquired through a camera to feed an object recognition software to identify objects. A scenario that is used for testing the RUT methodology is as follows: Sir has to move to a certain landmark, in which Sir re-orients itself to be directly in front of a red button which it has to press. The unit test case is testing the ability of Sir to press a button, this was done by creating the necessary work-space, executing the action and finally checking the final pose of the robotic manipulator.RUT follows the traditional regulations set by the software community. Each developed RUT should be independent of all other components and test units and itself should not affect anything else. The contribution of this paper is that the implementation of this testing methodology is expected to significantly increase the quality of robots as well as their dependability. Additionally, RUT has been designed in such a way that it tests all components of the robot except the very low-level hardware which includes the interface, internal calculations (of kinematics of the joints), visual computation etc. It is a necessary and basic step that will be used as a foundation for this R&D. The deficits of this research paper are that generation of 3D models through the use of meshes in URDF and SDF requires significant effort and is vital for simulators. Drafting these models requires an extensive amount of time depending on the complexity. An additional drawback is that despite it being able to run multiple iterative tests to check the robot, the tests run for the robotic controller is done as a black box and is non-adaptive which is why researchers [25] have attempted to deal with this by using spiking neuronal networks to connect with the robotic controllers. Another glaring deficit pointed out by [16] is that RUT is simply not enough to verify complex and sophisticated systems; especially in the context of interactions with dynamic environments.

Araiza and co-authors [4] investigate the performance of belief-desire-intention-agents (BDI) against the traditional methods used by the human-robot-interaction (HRI) domain, additionally, the scalability for effective tests using BDI models in HRI domains is examined as well. The motivation for the author's work is that robots that interact with humans require thorough verification and validation of their software, however, HRI introduces a new dimension of complexity and challenge for robotic software developers; hence this research paper. The traditional method for verification includes testing, model checking, and theorem proving. Each has its trade-offs. Typically, a robot is deemed safe when all exhaustive possibilities are checked using an abstract model. Evolved methods of this include running the robotic code in a simulated environment with real robots (also known as hardware-in-the-loop HiL) and using a simulated environment with a simulated robot also known as (human-in-the-loop). The authors evaluate multiple testing methodologies namely pseudo-random sampling, constraint solving, and model-searching. They use two test case studies for the evaluation. The first test case is of a table assembly. The robot is tasked with providing a human with components of a table and it has to be careful of the in judgment of giving the components to the human in a timely manner. The humans voice will act as the catalyst for starting this procedure. Some factors that the robot is programmed to be aware of are the location of the leg, the pressure sensor, and tracking the humans head position. The second test case study is the robot acting as a home care assistant. The task of this robot is to assist disabled people. It will perform high-level actions

like bringing food to a table, cleaning a table, and other similar tasks. Again, similar to the previous case studies, the human will activate it through voice command. An additional quirk to this case study is that the robot will also be constantly scanning in its environment to ensure that it does not collide with anything in this case being a dog which is commonly found with disabled people. The result of the case studies is that the BDI agent using model-based test generation performs significantly better than the other traditional method. The contributions of this paper are that they provide an efficient and innovative testing technique as mentioned by [58]. Also, the robotic software coverage provided by implementing the methodologies of this research paper is quite pronounced as indicated by [46]. The deficits of this research paper are that the chosen test case scenarios lack in complexity as the author's have mentioned this themselves. The scenarios were lack luster-ed to such an extent that any testing models would have generated sufficient results for code coverage. This is a significant issue and warrants a re-run with a different test case scenario. Additionally, [24] highlights that this research paper lacks autonomy and the decision making has to be handcrafted, moreover, this type of method can only be implemented when the user has full access to the model.

Takaya and co-authors [59] aim to develop a dependable robot capable of assisting elderly people and handicapped people. They attempt to achieve this goal by expanding the knowledge of fellow researchers in the Robotic Operating System (ROS) and Gazebo. The motivation for using ROS and Gazebo is that despite there being many simulation software for robotic systems, yet, few match the seamless workings between ROS and Gazebo. Their research work presents a step by step design on how to utilize ROS and Gazebo for simulating mobile robots. Additionally, it provides a tutorial on how to build 2D and 3D worlds in Gazebo and how to simulate mobile robots within those created worlds. It describes in-depth each of the important modules used within ROS and Gazebo. For demonstration purposes, it uses the PeopleBot as well as the Pioneer 3-DX robots in the simulation. As for environments, the Willow Garage and the authors home lab have been used. The robots utilized for the experimental analysis have been created in a very precise manner; their properties have been meticulously calculated and implemented to match the real-world data. For the experimental test case scenario, the robots have been fitted with LIDAR sensors, web cameras, odometry systems, and a laptop. Due to Gazebo not having a plugin for sonar, the authors modified the laser plugin to work as a sonar plugin. The experimental setup is that Gazebo is used as the simulation environment and ROS is used as the control system. The robot is tasked to reach a point selected by the user in Gazebo. It has the aforementioned sensors. Additionally, to navigate around obstacles and mapping its trajectory, it uses ROS's native library Navigation Stack that builds a cost map for the path traversed, moreover, it uses the same library as for local navigation. The result from both test cases showed no deviation in the behavior of the robot. The only thing of note was that the Navigation stack had to be adjusted due to the communication speed difference in the simulation and the real world. The contribution of this paper is that it presents a tutorial on how to build 2D and 3D worlds in Gazebo and how to simulate mobile robots within those created worlds. Additionally, it presents a technique for building 3D maps using the built-in ROS packages. In terms of the software used, the virtual test-bed environment provided by Gazebo and ROS allows for diverse testing as well as a beneficial platform for validating autonomous systems thus making it suitable for different research papers

like [11]. Where the research paper published by [11] focuses on the Hardware in the Loop (HiL) method for testing and validating autonomous vehicles. As this paper discusses ROS, Gazebo and its uses, very few pieces of literature exist that criticize or point out the deficit of this paper, and the main deficits lie within the use of the software itself. The environment and robots built-in Gazebo is not flexible. They are suited to general robots and do not go beyond the general bounds of grounded robots. Another issue is their communication tools are not effective for unmanned vehicles (UAV). The deficiency does not lie only with the communication but also with the control module which is also incompatible with Gazebo [70]. Where the research paper [70] aims to develop and further progress autonomous flying agents' vision system. Additionally, they present their simulated environment which is tuned for flying autonomous agents. Another deficit for [59] is that although Gazebo is a very useful simulating tool, it does not efficiently and effectively simulate multi-robot system due to which a need has arisen for a simulator the provides the same ease of use and intuitive features as Gazebo and also effectively simulate and process multi-robot systems [56].

Araiza and co-authors [3] investigate whether Belief-Desire-Intention (BDI) agents can be utilized as a framework to appropriately capture the interaction between humans and robots, and if so, is it possible that they can be used to automatically create tests and be enhanced using Machine learning techniques for instance reinforcement learning (RL). The motivation of their work is to test, simulated realistic robots with an automated testing system to allow for smarter and safer human-robot interactions. The success of their research greatly enhances the safety of autonomous robots and leads to easier acceptance by the masses for the integration of robots. The proposed model will be used to test multi-agent systems. The interaction of the agent with the environment and the interaction of the agent with other agents will also be tested. The focus of the test will be on the belief, desire agent's decision making. The testing for testing will be done by altering the beliefs of the agents and by testing it against handcrafted and coverage techniques to evaluate the benefit, drawback, and feasibility. To achieve this goal, the authors utilized algorithms like Q-learning and RL-learning. For the experimental setup three methods i.e. manual, random, and reinforcement belief selectors were used as BDI models for one test case scenario and were compared to pseudo-random abstract test models. The test case that has been used to test the BDI agent framework is a cooperative table manufacturing scenario. A simulation is provided in Gazebo, in which a human is assisted by a robot (BERT2). The human issues voice commands to which the robot has to automatically pick and provide components of table pieces to the human when the human is perceived as 'ready' to receive them. The result of the evaluation is that the BDI handcrafted models cover a large part of code with a minimum number of tests, the second one is BDI pseudo-random model which required an additional 30 tests, and finally, the RL BDI required an additional 80 tests for maximum code coverage. The pseudo-random abstract models that were used as the base comparison fell quite short and were 20% short of code coverage compared to the rest of the BDI abstract models. The RL BDI model is the best because no parameters were given nor constraints, allowing it to automatically reach peak performance on its own albeit takes a bit of extra time. The contribution of this research paper is that it presents an elegant solution to the problem of testing complex robots. New and innovative ways are required for online testing and validating of these robots in dynamic environments [7] and this research paper [3] is

exactly that. The drawbacks of this research paper is that it has not focused on testing a multitude of things like quality of the approach used, time of the approach used, specification, constraint coverage, etc. [6]. Another critique by [39] is that heuristics used for Q-learning does not provide enough coverage for faults due to which they proposed a fragility system in place of the heuristics.

Arora and co-authors present [6] a qualitative and comparative analysis of test generation techniques, specifically for regression test generation and multi-agent-based software tests. This survey was conducted on 9781 research papers, out of which 115 were selected. The period that this survey covers is from 2000-2016. It includes international conference papers, journals, theses, and workshop seminars. The motivation for this research paper is to increase software quality development in terms of cost, time, and efficiency. The result of their work will aid researchers in filtering out the most suitable approaches and techniques required for enhancing their software. The approach used by the authors for the selection of literature is based on Kitchenham and Brereton criterion [33]. Key terms such as 'software test generation' were used to extract relevant papers from different sources such as IEEE, Springer, ACM, and Elsevier. The criteria for selection, have been based on a set of research questions that were created in a way to determine the relevance of the research paper in question e.g. what approach has been used, who has referenced it and how has this study evolved? The result of their work demonstrates the advantages and disadvantages of Model-based testing, Structural testing, Functional testing, Formal specification-based testing, Mutation-based testing, and Random testing. A point that is highlighted is agents that utilized the Belief-Desire-Intention framework for testing regression have proven to require less time and effort. The contribution of this research work is that it provides a structured and valuable literature review that allows fellow researchers to quickly and easily analyze and distinguish their required testing methodology. The drawback is as mentioned by the authors themselves, that is the faulty selection of the keywords while filtering out relevant papers. More precisely, the issue is that some research papers may have not been associated with the correct keywords but were contextually relevant. However, as a literature review paper, this survey provides a valuable source of pertinent information.

Duyson and co-authors [55] present an interface that fuses two simulators for testing and validating autonomous vehicles; optimized algorithms for testing autonomous vehicles control, planning, and perception. The motivation of their work is to enhance the dependability of autonomous vehicles in diverse traffic scenarios and environments. The proposed approach works as follows: A perfect run in a simulation is conducted with the assumption of no error, noise, or range limitation of the sensors. This is the base comparison scenario. Then a test case is simulated, in which parameters for planning are optimized via a machine learning approach using this surreal base comparison scenario. To introduce realism a second simulator provides accurate information on the dynamics of the vehicle like the friction of tires, sensor readings of the environment, etc. The testing occurs in the updates acquired from the learned data in a closed-loop fashion. Four test cases are studied that are adaptive cruise control, green wave tech., parking, and finally double lane change. The platform used is Siemens 'Simcenter Amesim' and 'Simcenter Prescan'. The result of the test cases were positive and demonstrated viable planning paths and trajectories. The contributions of this research paper is the development of a co-simulation interface framework as well as techniques for optimizing algorithms that utilize machine learning for planning, control, and perception.

The limitations are that the test cases were very specialized. For validation and verification, more tests would be required in diverse scenarios. Moreover, the presented testing and validation system seemed to be inflexible and targeted to a specific set of scenarios.

Mullins and co-authors [47] present a novel approach for the selection and sortation of simulated test case scenarios for autonomous systems utilizing adaptive sampling and unsupervised clustering techniques respectively. The motivation of their work is mainly to improve the robustness of an autonomous system in dynamic environments. The requirement of improved robustness stems from the exponential growth of parameters in dynamic environments. Hence, to curb this computational strain, the proposed approach is a closed-loop system in which the unstable performance of autonomous systems in different simulated scenarios are excerpted using adaptive sampling (Gaussian Process Regression (GPE)). These scenarios are then clustered and sorted using density-based clustering techniques. The performance of the autonomous system is based on certain parameters like time taken and path selected. For adaptive sampling, it initially searches for a test scenario and runs it using GPE. GPE provides samples and information on the performance of the autonomous system. An identification system then separates these performance modes using the density-based clustering technique. Test cases that have been evaluated were math test functions as well as an autonomous underwater sea vehicle. The result of these test cases was that in comparison with the current state-of-the-art (SOBOL, LOLA, LH) the proposed GPE approach demonstrated significant improvement in terms of searching capability for performance modes both for the math functions as well as the autonomous underwater sea vehicle. The contribution of this research paper is the proposition and demonstration of a powerful search algorithm for identifying relevant test cases. The limitations of this paper are dealing with the curse of dimensionality and scalability. However, this research paper provides valuable insight into the development of techniques for the selection of appropriate test case scenarios.

Santos and co-authors [52] present an approach for automatically testing the robot operating system (ROS) using property-based testing. The motivation of their work is to ensure that the robotic software is adequately tested and verified to prevent faults from being discovered during run time in safety-critical tasks. The proposed approach tests ROS by treating ROS as a black-box system. It provides a set of parametrized inputs to the black-box system and analyzes whether the results conform to the expected output. The expected outputs are defined when modeling the system behavior. The tools used are ROS, hypothesis, and HAROS. The test cases focus on the verification of ROS nodes with a fictitious robot and the results of these test cases proved fruitful as in that it repeatedly discovered faults within the system (e.g. when a negative velocity message is published, the robot is assumed to be colliding with an object and the implemented hypothesis framework was consistently able to find nominal messages to trigger this event which in this case was to send a message drop for a singular wheel). The contribution of this research is the successful implementation and demonstration of property-based testing in ROS. The limitation of this research is that it requires a significant amount of time to run each test case due to the rigid but necessary framework of ROS. This type of testing also requires powerful hardware to run the tests and most importantly the power of testing is determined by the defined properties of the system.

A short summary (placeholder)

2.4 Complex-scenario-tests

Complex-scenario-testing is a process in which an autonomous agent is tested over several chained scenarios. The incentive behind complex-scenario-testing is to identify potential failures in an agent when it performs a series of tasks due to reasons such as increased complexity and operational time. In this section, we investigate up-to-date methodologies applied for testing the autonomous agents in complex scenarios with the hope of assimilating it into our development strategy.

Mosen and co-authors [48] present a methodology 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 authors 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.

Echeverria and co-authors [14] present a software application 'modular open robot simulation engine' (MORSE) that is capable of simulating complex robot scenarios using a real-time physics engine. The motivation of this software application is to aid researchers as well as roboticists in developing and testing their robots in a simulated environment with a powerful rendering engine (Blender). The key feature that makes MORSE stand apart from the rest is as follows:

1. The rendering engine allows for visualization and calculation of various complex environments.
2. The software-in-the-loop (SIL) for testing robots.
3. The flexibility of middle-ware configuration and integration.

MORSE can utilize all the regular robotic libraries like the robot operating system (ROS) and the Bullet physics library. Alongside the usage of these libraries, it can use middle-ware software which allows for

a high degree of control and flexibility within the simulation i.e. to create custom configuration code. MORSE is based primarily on python but C and C++ language can be used for faster processing with an additional interface. MORSE has proved to be a useful platform for many researchers, specifically for the aerial medium. Researchers have used MORSE in their research projects an example being ONERA labs [27] using MORSE for the evaluation and validation of UAV vehicles. The limitation of this software application is that existing robots created in other platforms (like Gazebo) are not readily transferable to this platform. Additionally, most of the robotics community have given preference to the Gazebo platform due to its features and user base for ground-based robots.

Koren and [34] presents a modified version of the Adaptive Stress Testing's (AST) approach. It utilizes the Markov Decision Process (MDP) to determine failures in a system and the MDP itself uses Deep Reinforcement Learning (DRL) for the estimation of failure and the environmental response. The motivation of this work is to increase the safety and reliability of autonomous vehicles in critical scenarios e.g. scenarios involving a collision with pedestrians. The framework known as Adaptive Stress Testing (AST) is specifically designed to identify scenarios that are most probable to cause failure in a system. It uses a Markov Decision Process (MDP). It continuously provides input information to the system when used in a simulator, until a failure has been detected which is then conveyed along with the environmental response. MDP is typically solved by Deep Reinforcement Learning (DRL) and Monte Carlo Tree Search (MCTS) techniques, however, DRL is preferred due to efficiency. The deficits of using DRL is that it has high dependencies on the initial condition and the observed simulation state that stores all the internal state variables required for defining simulation. For the simulation state, systems are large and complex, therefore, to overcome this hurdle, researchers typically treat the simulator as a black box. This prevents the projection of the exact history of the actions. As for the initial condition limitations, the DRL has to be re-run for each initial condition and it has to be discretized. As of yet, there does not exist a way to continuously provide initial conditions for this type of DRL problem due to the high dimensionality and the vast variation that comes with slightly different variations in the input. The modification presented by the authors is to change the policy architecture of DRL to a Recurrent Neural Network (RNN). The selection of RNN is due to the two distinct advantages that it provides.

1. RNN is developed in such a way that they require sequenced data. This removes the need for the simulation state because it will be maintained within the internal hidden layers of the RNN.

2. Certain RNN can learn patterns that are dependent on time. This allows initial conditions to be generalized for different initial states.

The test case scenarios for this is a simulated scenario through which an autonomous vehicle navigates with various pedestrians. The evaluation for this research paper is achieved by comparing the modified DRL with the existing DRL as well as the Monte Carlo Tree Search (MCTS) techniques. The result of this work was that the proposed architecture outperforms other methods in failure detection by converging to a failure scenario in a few amounts of iterations. The contributions of this research paper is the increase of efficiency and effectiveness of an existing AST architecture. The benefit of this modification is the easy generalization with different initial conditions. The deficits of this work is that although the proposed

method detects failures within the simulation quicker, it does not perform better than existing methods in detecting collisions. Additionally, [29] highlights the methodology applied by this research paper to low dimensional data does not translate well into higher dimensional data. Moreover, [26] mentions that the presented work cannot be applied to cases with multiple behaviors.

Castro and co-authors [17] propose the use of continuous integration (CI) using the model-view-controller (MVC) design alongside simulators that do not have any graphical user interface (GUI). The motivation of this work is, firstly, to aid in the quick, seamless, and coherent integration of different software modules within a robotic system (e.g. object recognition with a robot locomotion), and secondly, is to run the simulator faster than in real-time. This speed-up will allow machine learning algorithms and multi-agent systems to operate quicker. As existing simulator platforms do not in a true sense provide GUI free simulation, the authors have designed their package that utilizes the 'Swift' [36] and 'GTK' [60] platforms. They follow the MVC design pattern for development and have opted for 'Jenkins' [30] as the platform that will continuously integrate independent software modules. For testing, their system uses 'Logic-labelled finite state machine' (LLFSMS). The test cases that they have run in their package are on the 'RoboCup Standard Platform League robots' [32]. The result of their work was the successful validation of some of the behaviors of the soccer bots. The contribution of their research is the practical implementation and demonstration of a CI with a GUI'less simulator. The limitations as mentioned by the authors are that, when the headless simulator interfaces with the model of the robot and its sensors, it faces substantial complexity and complications. This is further amplified when attempting to apply unit tests on them. Another limitation of stripping the GUI is that although the performance speed is enhanced, but there is a loss in the visual data; at times more often than not, it is quicker to identify whether something is wrong with a system when a visual interface is provided. To conclude this research paper provides valuable information on LLFSMS and their integration with simulators which may be a potential approach used for this research and development project.

Winkler and co-authors [67] present a hybrid software system that allows a robot to perform actions based on experiences. It is hybrid in the sense that it utilizes a heuristic search to identify the task parameters and builds up a knowledge base. It then extracts useful data from the knowledge base (i.e. robot memories) and applies machine learning algorithms (e.g. Multivariate Gaussian Distribution (MGD), K-means) to select the best action. The motivation of their work is to enhance and expand the robot's ability to perform tasks that have the same general task outline. The success of their work will increase the robot's performance across the board for similar tasks. The test cases conducted were done using the PR2 robot in which it had to reorient itself to pick an object utilizing previous experiences in a simulated kitchen. The robotic memories were captured by a software called 'SemRec' [66]. The results of the test cases were positive and normally showed the ideal positioning of the robot for picking up an object. The contributions of this work is the increment of data that can be used to decide the best course of action (i.e. previous researchers only used the position of the Cartesian coordinate x and y to determine a good action, however, this method allows for extra parameters to be involved in the determination of a good action by the use of multivariate Gaussian distributions; in this case, θ has been the additional parameter). The limitations of the work is the issue of learning bad behaviors, the high requirement of data for feasible

results, and the testing of the proposed methodology on varied robots and platforms.

Gross and co-authors [20] present a novel approach for assessing the navigation description in domestic environments, moreover, it proposes a set of mandated requirements for domestic robots that are to aid senior citizens based on the results of a 3-year research project (SERvice RObotics for Gesundheits Health Assitance (SERROGA)). The focus of their research project was to study medical robot assistants interaction with the elderly in their private residences along with investigating the factors that could enable these medical robot assistants to have a long-term cooperative and friendly interactions. Tying to the focus of this project, the source of motivation was to provide the elderly with a companion that could support them both mentally and physically as well as provide them with some medical services. The cases studied were conducted over 12 apartments with 9 elderly people along with some project staff members. The test case studies discovered that the overall experience of the users (elderly folk) was positive, however, some time was required by them to become familiar with the robot. The common disappointment factor among the users was due to the limited robot assistance and services. A point to note is that though the users were comfortable around the robot, they all kept an eye on it due to the possibility of un-wanted action. As for the proposed approach for tackling various domestic environment navigation descriptions, it has been achieved by defining certain geometric parameters of the workspace that the robot was to operate in e.g. the operational zone, the free zone, clearance between objects, etc. This descriptive approach enables an easy way for benchmarking and comparison. In conclusion, this research paper provides an invaluable insight into the future of human-robot interaction (HRI). Also, it demonstrates the influence of certain sets of features that will be required for the integration of robots into society.

Mitrevski and co-authors [44] present a hybrid fault detection model that uses symbolic as well as geometric models to generate simulated scenarios based on a fault, apply a learning model, and excerpt the best action possible. The symbolic models represent the configuration of objects as well as the relation of the objects to each other, both in terms of similarity and location in the environment (e.g. a cup might be more similar to a mug and can be used as a generalized model). The geometric models utilize geometric distribution sampling (GSM) to select an appropriate execution pose from a probability distribution based on the template configuration provided by the symbolic model. With the combination of both, a system is created that can be utilized for the implementation of a learning model that can learn the best execution action. The motivation of their work is to assist service robots that perform wrong actions due to external events. Four case studies have been investigated with various scenarios. Three case studies were conducted on a Care-O-bot and one case study was conducted on a KUKA youbot. Some of the case studies focused on actions like picking and placing objects from a table while other case studies focused on placing objects on top of each other. The result of these test cases showed an increase in the successful action execution. The contributions of this work is that it proposes and demonstrates a powerful approach for dealing with external faults in robots. The limitations of this work are that it lacks flexibility (i.e. for each test case, the initial parameters and objects are required to be known and are currently incompatible with unknown objects and undefined parameters), moreover, it has a high threshold of complexity and difficulty in creating templates that can be generalized using the symbolic model.

Gambi and co-authors [19] present an approach to generate autonomous driving scenarios from police reports that are then used as a test-bench for autonomous vehicles. The motivation behind their work is to enhance the safety of autonomous vehicles by concentrating tests on common accident scenarios recorded by the police. The proposed approach works by excerpting relevant information from the police reports (by the use of natural language processing (NLP)) and using the obtained information to generates trajectories for the vehicles. A simulation then runs the planned trajectory. Based on the simulation, test cases are generated to see if alternative results can be obtained without crashing. 27 reports were used as test case studies and on average it took 22 seconds to generate and run the scenario compared to an average of 2 hours when done manually. The result of the case studies was that they were able to produce highly accurate driving accident scenarios from the police reports and discover ways for autonomous vehicles to avoid such accidents. The limitation of this work is that it is highly dependent on sufficient and relevant information provided within the report (e.g. some countries have a different method for recording accidents, so although a scenario would be generated, it may not conform to the real scenario.)

Mitrevski and co-authors [45] present a semantic model that defines conditions under which an action of a robot may be successful along with a Gaussian probability model that measures the success factor of that action. The motivation of their work is to provide robots with the ability to reason for selecting the best action-model because reasoning will allow for easier fault diagnosis in addition to allowing the robot to learn from experience. Test case studies were performed with the Toyota HSR bot [69]. These test case studies were aimed at learning tasks for grasping and pulling objects. The evaluation of these tasks was performed by a human observation and the results of these studies demonstrated an improved success rate when performing both actions. The contribution of this research is the novel hybrid model for robot action-execution. It creates a foundation for each action of the robot and in a sense meaning behind each action. The limitation of their work is that is of note is the issue of fine tuning the learning rate of the robot's action model and the time the required to recover from being excessively penalized.

A summary (placeholder)

2.4 Relevant Tutorial Papers

Qian and co-authors [51] present a research paper that provides the basic yet vital information required by the average user to set up and use both ROS and Gazebo. The motivation of the authors is that due to the growth and increased dependency on simulators, simulators have had a positive impact on robotic users both time-wise and monetarily. This research paper [51] explains in extreme detail the setup and running of Gazebo with a robotic manipulator. It also indicates the simple issues that can be encountered in the process. This research paper is simply an introductory tutorial for robotic researchers. The contribution of this research paper is that it provides a clear and precise description and methodology for the use of ROS and Gazebo. This research has benefited several researchers such as [13], [49] and [68]. Where [13] benefits from the approach used for generation and modification of URDF files, [49] credits the authors for the precise definition and explanation of the ROS nodes and communication tools, and [68] utilized the description and methodology presented in this research paper to develop a robot that could bend archwires for orthodontic applications. The shortcomings of this research paper is that the

design and implementation of using ROS and Gazebo lacked the depth required to properly demonstrate the capability of the system due to which [43] published a research paper similar in nature to this [51] research paper in an attempt to fill in the gap.

This research paper [63] presents a joint controller system for an open-source robot known as the 'OpenDog'. The objective of this research paper is to provide an easy step by step development process of 'Unified Robot Description Format' (URDF) for complex legged robots. The motivation of this work is to expand and extend current robotic research towards legged robots. The reason being is that legged robots are more flexible and useful in traversing difficult terrain, especially in disaster case scenarios. The approach for the development of this project by the authors is straight forward. They utilized an open-source CAD model of the 'OpenDog' (Solidworks (SW) was used as the CAD editor for the project) to generate a URDF model. They accomplished this by exporting the CAD model using an SW to URDF converter after which they adjusted the inertia, center of mass, and links of the robot model manually. They then tested the robot by developing custom slider controllers for each of the individual kinematic links of the robot. The results of their work is the successful and open-source implementation of the 'OpenDog' robot. The contribution of their work can be summarized as a quick tutorial for designing URDF models for legged robots. The limitations of their research paper are that it fails to explain the techniques and practices employed in its implementation e.g. implementation of assigning the center of mass values.

3

Development Strategy

3.1 Proposed Approach

Keeping in mind the state-of-the-art review, 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:

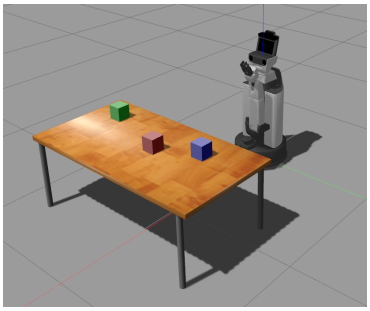

Figure 3.1: Lucy in a simulated world (Gazebo) with three cubes placed on a table [28].

3.1.1 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 [5]. 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.

3.1.2 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 [52]. 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 in this R&D, we will be focusing on implementing the action tests for picking action and placing action.

3.1.3 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).

3.1.4 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 two folds, the first is to obtain a human readable report and the second is to utilize a machine learning algorithm (reinforcement learning) to asses the report and learn from good and bad examples. The time frame of this project may not allow us to reach the second goal; nevertheless, it is part of our intended objective.

3.2 Use Cases

Following are the use cases by which we intend to test Lucy with.

3.2.1 Navigation in a lab

In this use case, Lucy will be tasked with navigation to different locations assigned by a test generator.The test will consist of some property-test such as during the navigation, has any object in the world been moved or disturbed, is the robot in the location where it believes itself to be, and some legal tests such as has the robot left the prescribed safety zone, has it come to close to X-person etc.

3.2.2 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.

3.2.3 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.

3.2.4 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.

3.3 Scenario Generation

Axis-aligned bounding box + image

3.4 Scenario Generation for Robots

3.5 Testing

3.5.1 Typical Robotic Testing Practices

3.5.2 Metamorphic Relations

3.5.3 Coupon Collector's Problem

[22]

3.5.4 Legality tests?

[31]

3.5.5 Property-Based Testing

To our knowledge, only 1 other author [52] has implemented property-based testing for robots and that too for testing the ROS. behavior Finding the counter example

3.5.6 Simulators

gazebo stage player morse USARSim UDK V-Rep Webots

3.6 Scientific process

see [52] for how to make the images

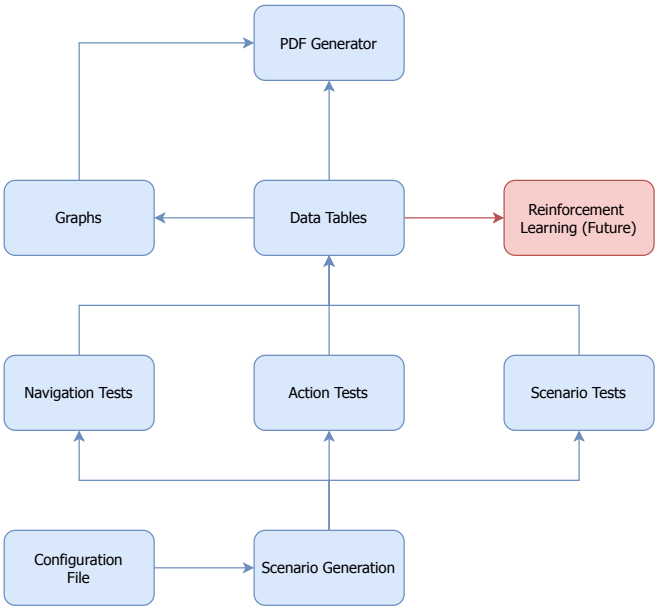

Figure 3.2: The overview design for the development of the automatic-test-generator for Lucy.

3.6.1 Defining System behaviour

3.7 Evaluations

3.7.1 Evaluating the Random Scenario Generation

3.7.2 Evaluating the Action-tests

One of the first steps was taken for automating test generation using property-based testing was done by [52].

3.7.3 Evaluating the Complex-Scenario-Tests

3.8 Standards adhered to

3.9 Experimental Design

3.9.1 Design criteria

3.9.2 Possible shortcomings

3.9.3 Possible errors

3.9.4 Issues expected

the errors we expect are number one version conflict of libraries. Outdated libraries.

4

Project Development

imageplaceholderflowchartifscenariogenerationrobot > tableboundbox > placementvalues > exisitingobjectcol validplacement

4.1 Experimental Setup

4.2 Software Requirements

4.2.1 Applications

- Simulation Environment - Gazebo
- Robot Application - Robot Operating System (ROS)
- Object Design - CAD software

4.2.2 Libraries

use [20] table style here for libraries and hardware

- Numpy
- Pandas

4.3 Hardware Requirements

4.4 Scenario-Generation

(Rough thoughts) for the scenario generation, the first thing we did was to setup and run the package provided by toyota for the HSR robot. All the instructions for the setup and installation were provided clearly within the documentation of the toytoa HSR. A point to note that the section 7 and 6 from the hsr manual provides the most useful information within the documentation. Once the package was setup and completely functional, we also setup the b-it bots ropod package as well as the gym_HSR package. As the ropod package has been on the works more recently its packages especially pertaining to

ROS kinetic worked on the newer version and was not compatible with the older version. For the gym HSR this also had issues with version conflicts were mostly related to the python version and certain key libraries being updated and having depreciated functions. Both packages however provided valuable insight and inspiration for the method of approach for designing the random scenario generator. only compatible with stl files as bounding boxes are created manually.

we developed "12 (change as more models are made)" 'Computer Aided Design' (CAD) models using CAD. Each model has been developed with accurate dimension following standard engineering practice. After the development of each model, it has been converted into high-density meshes which were then exported as .stl files.

For the project we developed a ROS package with custom worlds and models which was then integrated with the existing Toyota HSR package, MAS domestic package and MAS HSR package.

For the model, we used the 'Standard Database Format' (SDF); we utilized the created .stl files as the models. Each sdf has been assigned with meshes as regular simple polygons are not sufficient for adequate testing for the test case scenarios we have like for picking a glass. For each model we assigned the optimal physical property values i.e. mass and inertia.

Once the entire package has been setup specifically the world file, the model file and the launch file that activates hsr package and Toyota package. We setup a configuration file that allows for the selection of different models that will be spawned, as well as the number of models that can be spawned. The spawning of the randomized model which has been termed as *Dynamic Models* is currently based on the static position of a model that has been spawned in with the world file. This static model has been termed as *Static Model*. We developed the software as in to accommodate easy extension of additional *Static Models*. This means that we have the ability to select any model as a spawning ground for other objects.

The *Dynamic Models* are placed within the bounding box of which they has been assigned to. The bounding boxes created have been computed by extracting the minimum and maximum values of the Cartesian coordinates i.e. x, y, and z. To add a safety tolerance the *Dynamic Models* are spawned within 80% of the actual maximum x, y, and z coordinates of the *Static Models*. As the dynamic models are spawned, each of their attributes are stored and before the next *Dynamic Model* is spawned, it checks whether it collides with previously spawned models using the **Axis-Aligned Bounding Box** approach.

4.5 Navigation-tests

For the navigation tests, we developed a new unit that logs all the information of the scenario at a given moment. The idea is that we can store temporal differences in data by using the logs or more accurately snapshots of the current state of the world. This module will also be used on the picking and placing tests. The snapshot information is stored in a csv file. During the test in navigation we import and use the pytest for unit test as we will be calling the tests using py tests and hypothesis to generate strategies for the tests where applicable.For navigation we plan to generate some property test for each navigation run, some issues is the time sync which is a hardware limitation More test that we plan to incorporate are legality tests and scenario tests which will be the end goal.

4.6 Picking-tests

4.7 Placing-tests

4.8 Scenario-tests

4.9 Report Generation

4.9.1 Lucy

Lucy descriptive picture

4.9.2 Gazebo Models pictures

renders of blender

4.9.3 ROS Services pictures

Demonstration of used nodes.

4.9.4 Model Drafts pictures

drafts of some of the images

4.9.4 Inertia and Mass pictures

table for mass and inertia

4.9.5 Meshing pictures

mesh pictures.

4.9.6 Collision Detection pictures

collison zone

4.9.7 Model Placement pictures

graph of computation time for 5 - 500 objects object placement.

4.10 Action-tests

navigation test case of robot goals similar to [20]

4.10.1 Mapping

4.11 Complex-scenario-tests

4.12 Behavior-Adaptability

5

Evaluation

Implementation and measurements.

5.1 Navigation

evaluation can be that whether the test catches the mistake or you can compare it to existing test approaches give prefernce to the first one

5.2 regression test to ensure same results.

5.3 Picking an object from the table

5.3.1 Statistical Analysis

5.3.2 Observations

5.4 Placing an object from a cluttered table

5.4.1 Statistical Analysis

5.4.2 Observations

5.5 Picking and placing an object on the same table

5.5.1 Statistical Analysis

5.5.2 Observations

5.6 Results

5.7 Result Findings

5.7.1 Usecase 1

describe result findings

5.7.2 Usecase 2

5.7.3 Usecase 3

..

5.8 Discussion

6

Conclusions

6.1 Discussion

This R&D proposal addresses the problem of how to expansively test robotic-navigation as well as robotic-manipulators interaction with the environment to identify faults and enable the robot to learn from these faults. We address challenges 1, 2, and 5 from §1.2. We will be answering the following research questions:

R1 How to generate extensive and varied simulated scenarios for robots?

R2 How to qualitatively test robotic interaction with the environment?

R3 How to test a robot when it performs complex tasks in complex-scenarios?

6.2 Conclusion

6.3 Threats to validity

6.4 Contributions

6.5 Lessons learned

6.6 Future work

A

Design Details

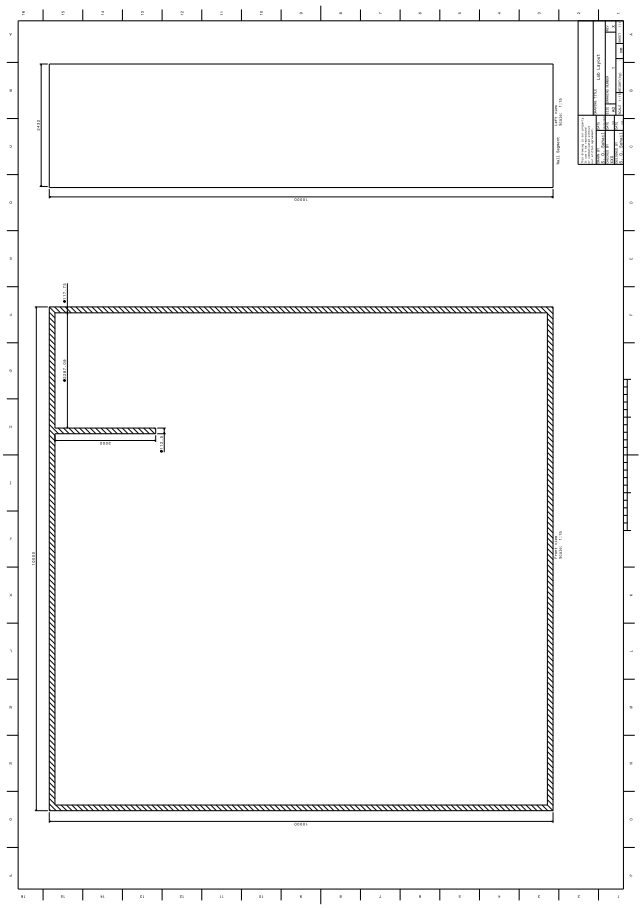

Figure A.1: The layout pf the simulated lab in which Lucy performed her tasks.

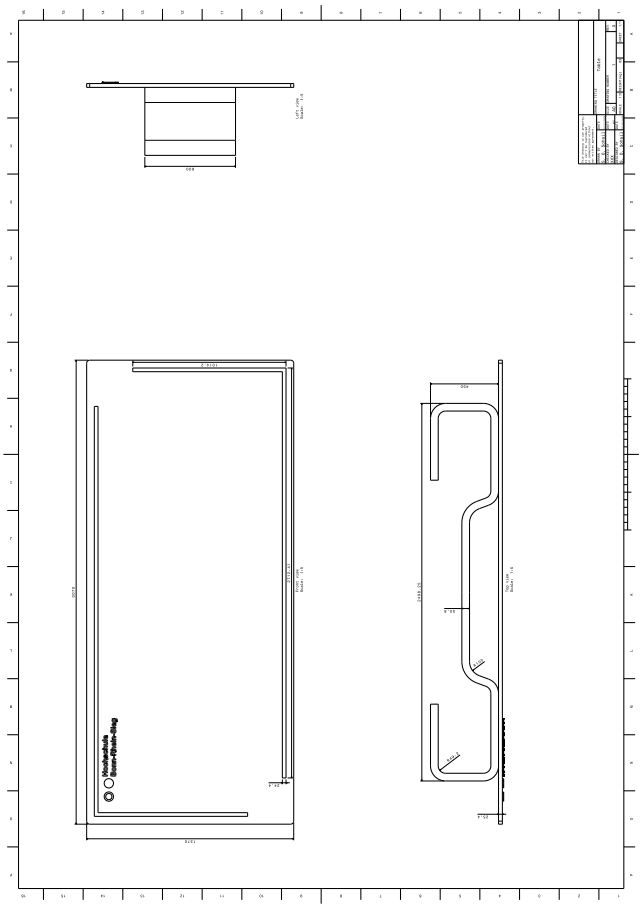

Figure A.2: The design draft of a table used in the scenario generation.

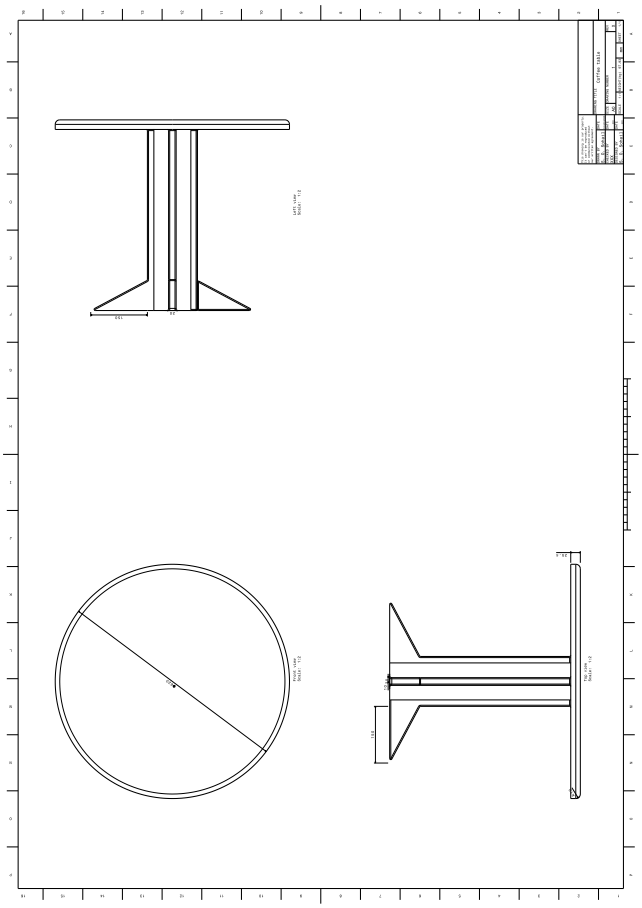

Figure A.3: The design draft of a coffee table used in the scenario generation.

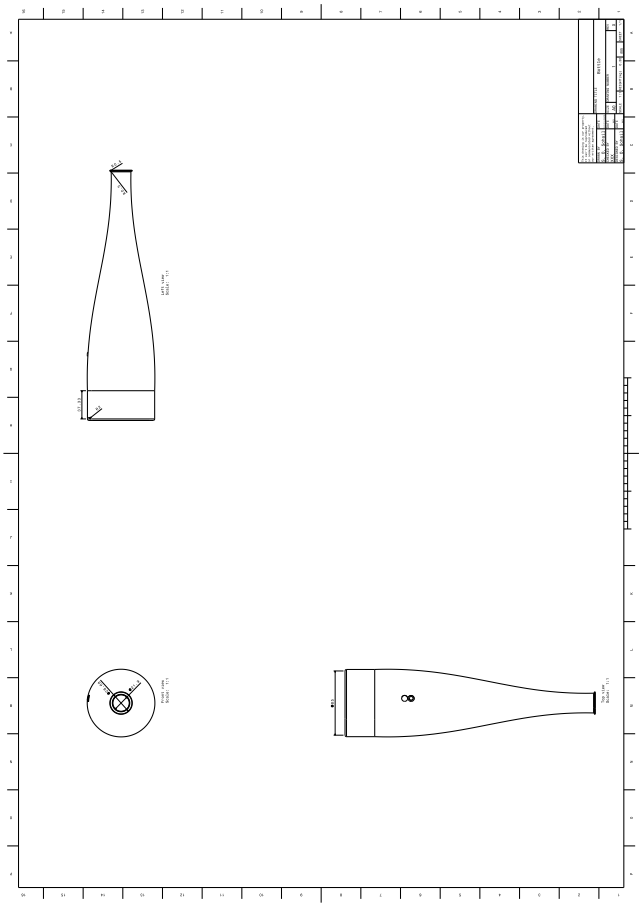

Figure A.4: The design draft of a bottle used in the scenario generation.

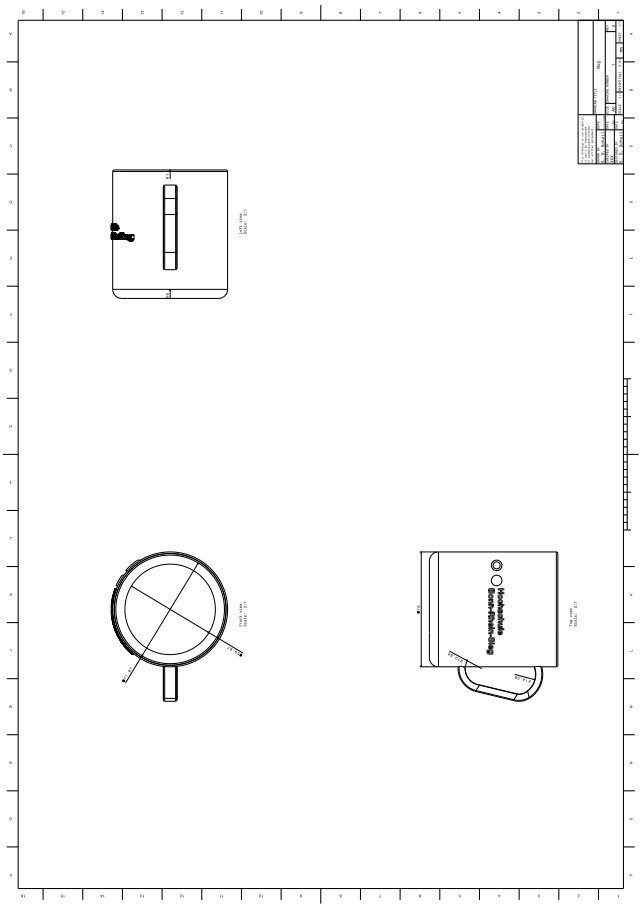

Figure A.5: The design draft of a mug used in the scenario generation.

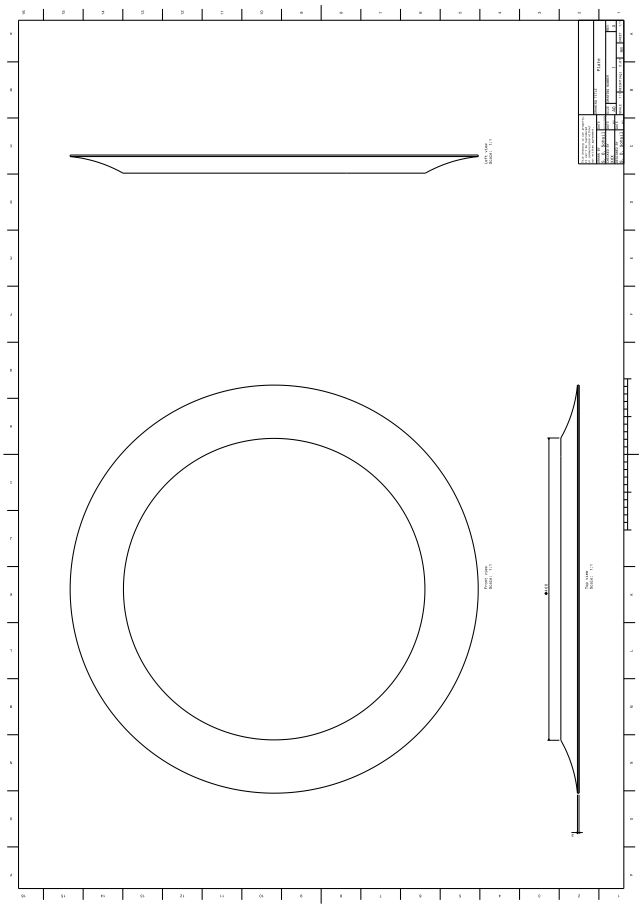

Figure A.6: The design draft of a plate used in the scenario generation.

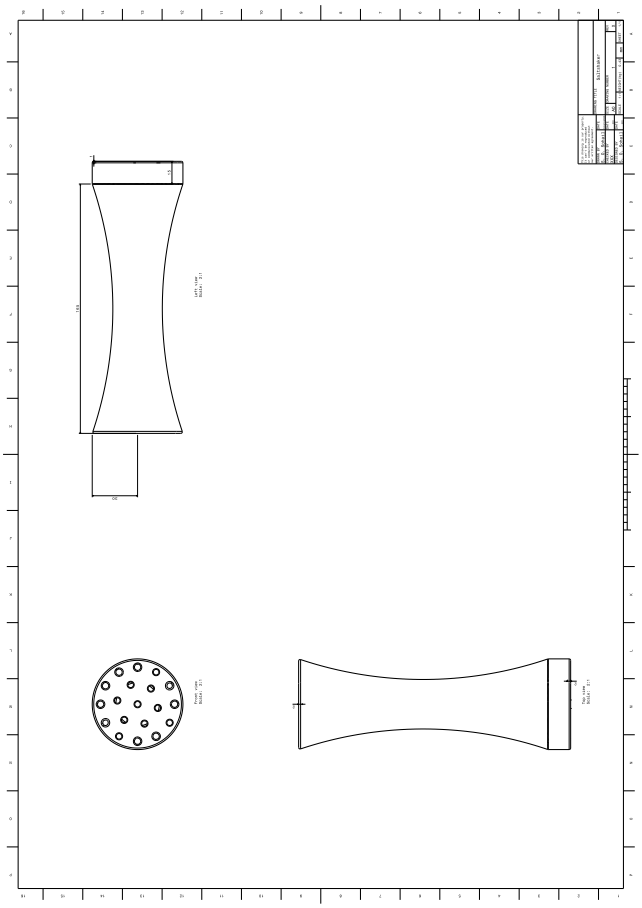

Figure A.7: The design draft of a salt shaker used in the scenario generation.

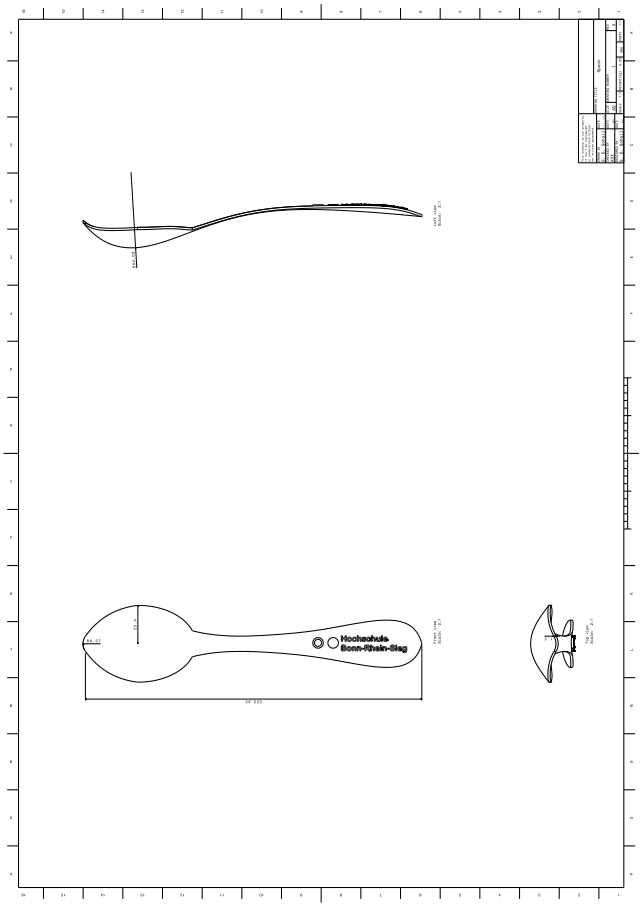

Figure A.8: The design draft of a spoon used in the scenario generation.

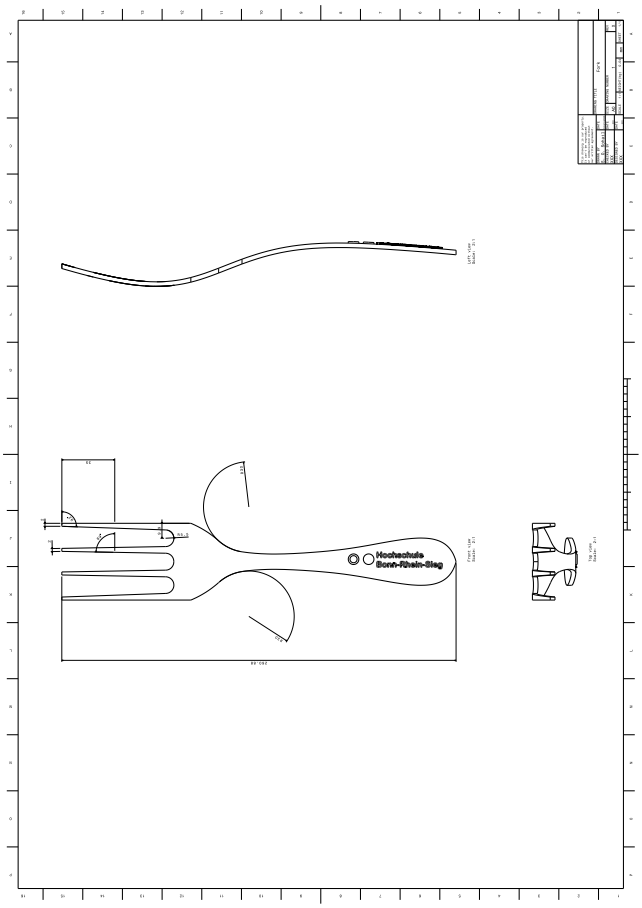

Figure A.9: The design draft of a fork used in the scenario generation.

B

Parameters

Your second chapter appendix

References

[1] A. Bihlmaier and F. Stein and H. Wörnz. 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.

[2] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek. Generating Adversarial Driving Scenarios in High-Fidelity Simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277, 2019.

[3] D. Araiza-Illan, A. G. Pipe, and K. Eder. Intelligent Agent-Based Stimulation for Testing Robotic Software in Human-Robot Interactions. In *Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering*, MORSE '16, page 9–16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342599. doi: 10.1145/3022099.3022101. URL `https://doi.org/10.1145/3022099.3022101`.

[4] D. Araiza-Illan, A. G. Pipe, and K. Eder. Model-based Test Generation for Robotic Software: Automata versus Belief-Desire-Intention Agents. *CoRR*, abs/1609.08439, 2016. URL `http://arxiv.org/abs/1609.08439`.

[5] J. Arnold and R. 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.

[6] P. Arora and R. Bhatia. A Systematic Review of Agent-Based Test Case Generation for Regression Testing. *Arabian Journal for Science and Engineering*, 43:1–24, 08 2017. doi: 10.1007/s13369-017-2796-4.

[7] D. Banerjee and K. Yu. Robotic Arm-Based Face Recognition Software Test Automation. *IEEE Access*, 6:37858–37868, 2018.

[8] M. Bauer. *Schiaparelli landing investigation makes progress.* 2016. URL `https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Exploration/ExoMars/Schiaparelli_landing_investigation_makes_progress`.

[9] A. Bihlmaier and H. 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.

[10] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250, 2018.

[11] Y. Chen, S. Chen, T. Zhang, S. Zhang, and N. Zheng. Autonomous Vehicle Testing and Validation Platform: Integrated Simulation System with Hardware in the Loop*. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 949–956, 2018.

[12] S. Cousins. Willow Garage Retrospective [ROS Topics]. *IEEE Robotics Automation Magazine*, 21(1): 16–20, 2014.

[13] H. Deng, J. Xiong, and Z. Xia. Mobile manipulation task simulation using ROS with MoveIt. In *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 612–616, 2017.

[14] G. Echeverria, S. Lemaignan, A. Degroote, S. Lacroix, M. Karg, P. Koch, C. Lesire, and S. Stinckwich. Simulating Complex Robotic Scenarios with MORSE. volume 7628, 11 2012. doi: 10.1007/978-3-642-34327-8_20.

[15] J. Ernits, E. Halling, G. Kanter, and J. 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.

[16] V. Estivill-Castro, R. Hexel, and J. Stover. Modeling, Validation, and Continuous Integration of Software Behaviours for Embedded Systems. *2015 IEEE European Modelling Symposium (EMS)*, pages 89–95, 2015.

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

[18] D. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. Sangiovanni-Vincentelli, and S. Seshia. Scenic: A Language for Scenario Specification and Scene Generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 63–78, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367127. doi: 10.1145/3314221.3314633. URL `https://doi.org/10.1145/3314221.3314633`.

[19] A. Gambi, T. Huynh, and G. Fraser. Automatically Reconstructing Car Crashes from Police Reports for Testing Self-Driving Cars. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 290–291, 2019.

[20] H. Gross, S. Mueller, C. Schroeter, M. Volkhardt, A. Scheidig, K. Debes, K. Richter, and N. Doering. Robot companion for domestic health assistance: Implementation, test and case study under everyday conditions in private apartments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5992–5999, 2015.

[21] P. Hamill. *Unit Test Frameworks: Tools for High-Quality Software Development*. 02 2009. ISBN 9780596552817.

[22] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner. Did We Test All Scenarios for Automated and Autonomous Driving Systems? In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2950–2955, 2019.

[23] C. Heer. *Industrial Robots: Robot Investment Reaches Record 16.5 billion USD*. The new World Robotics. 2019. URL `https://ifr.org/ifr-press-releases/news/robot-investment-reaches-record-16.5-billion-usd`.

[24] A. Hentout, A. Mustapha, A. Maoudj, and I. 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.

[25] G. Hinkel, H. Groenda, S. Krach, L. Vannucci, O. Denninger, N. Cauli, S. Ulbrich, A. Roennau, E. Falotico, M. Gewaltig, A. Knoll, R. Dillmann, C. Laschi, and R. Reussner. A Framework for Coupled Simulations of Robots and Spiking Neuronal Networks. *Journal of Intelligent & Robotic Systems*, 09 2016. doi: 10.1007/s10846-016-0412-6.

[26] X. Huang, S. McGill, J. DeCastro, B. Williams, L. Fletcher, J. Leonard, and G. Rosman. Diversity-Aware Vehicle Motion Prediction via Latent Semantic Sampling. 11 2019.

[27] C. Hung. Onera (the french aerospace lab), 1946. URL `https://www.onera.fr/en`.

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

[29] K. Julian, R. Lee, and M. Kochenderfer. Validation of Image-Based Neural Network Controllers through Adaptive Stress Testing. 03 2020.

[30] K. Kawaguchi. Jenkins, 2011. URL `https://www.jenkins.io`.

[31] J. Kim, S. Chon, and J. Park. Suggestion of testing method for industrial level cyber-physical system in complex environment. In *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 148–152, 2019.

[32] H. Kitano. RoboCup Standard Platform League, 2008. URL `https://spl.robocup.org`.

[33] B. Kitchenham and P. Brereton. A Systematic Review of Systematic Review Process Research in Software Engineering. *Inf. Softw. Technol.*, 55(12):2049–2075, December 2013. ISSN 0950-5849. doi: 10.1016/j.infsof.2013.07.010. URL `https://doi.org/10.1016/j.infsof.2013.07.010`.

[34] M. Koren and M. Kochenderfer. Efficient Autonomy Validation in Simulation with Adaptive Stress Testing. pages 4178–4183, 10 2019. doi: 10.1109/ITSC.2019.8917403.

[35] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz. Simulation-Based Temporal Projection of Everyday Robot Object Manipulation. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, page 107–114, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0982657153.

[36] C. Lattner. Homepage — SWIFT - The global provider of secure financial messaging services, 2014. URL `https://www.swift.com/`.

[37] Li Feng and Sheng Zhuang. Action-driven automation test framework for Graphical User Interface (GUI) software testing. In *2007 IEEE Autotestcon*, pages 22–27, 2007. doi: 10.1109/AUTEST.2007.4374197.

[38] J. Lim, S. Song, J. Son, Y. Tan, H. Park, and H. Kim. An Automated Test Method for Robot Platform and Its Components. *International Journal of Software Engineering and Its Applications*, 4, 01 2010.

[39] T. Ma, S. Ali, T. Yue, and M. Elaasar. Fragility-Oriented Testing with Model Execution and Reinforcement Learning. In Nina Yevtushenko, Ana Rosa Cavalli, and Hüsnü Yenigün, editors, *Testing Software and Systems*, pages 3–20, Cham, 2017. Springer International Publishing. ISBN 978-3-319-67549-7.

[40] D. MacIver, Z. Hatfield-Dodds, and M. Contributors. Hypothesis: A new approach to property-based testing. *Journal of Open Source Software*, 4:1891, 11 2019. doi: 10.21105/joss.01891.

[41] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey. Paracosm: A Language and Tool for Testing Autonomous Driving Systems. 02 2019.

[42] C. Medrano-Berumen and M. Akbaş. Abstract Simulation Scenario Generation for Autonomous Vehicle Verification. 04 2019.

[43] R. K. Megalingam, R. Chinta, S. Sreekanth, and A. Raj. ROS based Autonomous Indoor Navigation Simulation Using SLAM Algorithm. 11 2019.

[44] A. Mitrevski, A. Kuestenmacher, S. Thoduka, and P. G. Plöger. Improving the reliability of service robots in the presence of external faults by learning action execution models. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4256–4263, 2017.

[45] A. Mitrevski, P. G.Plöger, and G. Lakemeyer. Representation and Experience-Based Learning of Explainable Models for Robot Action Execution, 2020.

[46] G. Mullins. *ADAPTIVE SAMPLING METHODS FOR TESTING AUTONOMOUS SYSTEMS, school=University of Maryland.* PhD thesis, 2018.

[47] G. Mullins, P. Stankiewicz, and S. Gupta. Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles. pages 1443–1450, 05 2017. doi: 10.1109/ICRA.2017.7989173.

[48] 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.

[49] Y. P. Nugraha, H. M.Ridlwan, M. I. Riansyah, and B. R. Trilaksono. Autonomous Tracking of Hexacopter on Moving Mobile Robot Using Gazebo ROS Simulation. In *Proceedings of the 9th International Conference on Machine Learning and Computing*, ICMLC 2017, page 498–501, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348171. doi: 10.1145/3055635.3056657. URL https://doi.org/10.1145/3055635.3056657.

[50] Park, Wen, Sung, and Cho. Multiple Event-Based Simulation Scenario Generation Approach for Autonomous Vehicle Smart Sensors and Devices. *Sensors*, 19(20):4456, Oct 2019. doi: 10.3390/s19204456. URL http://dx.doi.org/10.3390/s19204456.

[51] W. Qian, Z. Xia, J. Xiong, Y. Gan, Y. Guo, S. Weng, H. Deng, Y. Hu, and J. Zhang. Manipulation Task Simulation using ROS and Gazebo. 12 2014. doi: 10.1109/ROBIO.2014.7090732.

[52] A. Santos, A. Cunha, and N. 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.

[53] L. Sartori. Simulation-Based Testing to Improve Safety of Autonomous Robots. pages 104–107, 10 2019. doi: 10.1109/ISSREW.2019.00053.

[54] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2nd edition, 2011. ISBN 0262015358.

[55] T. D. Son, L. Awatsu, J. Hubrechts, A. Bhave, and H. V. Auweraer. A simulation-based testing and validation framework for ADAS development. 11 2017.

[56] A. Soni and H. Hu. A multi-robot simulator for the evaluation of formation control algorithms. In *2019 11th Computer Science and Electronic Engineering (CEEC)*, pages 79–84, 2019.

[57] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand. Can robot navigation bugs be found in simulation? An exploratory study. In IEEE Computer Society Conference Publishing Services (CPS), editor, *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS2017)*, page 10p., Prague, Czech Republic, July 2017. URL https://hal.archives-ouvertes.fr/hal-01534235.

[58] Z. Tahir and R. Alexander. Coverage based testing for V&V and Safety Assurance of Self-driving Autonomous Vehicle : A Systematic Literature Review. 01 2020. doi: 10.1109/AITEST49225.2020.00011.

[59] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache. Simulation environment for mobile robots testing using ROS and Gazebo. In *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 96–101, 2016.

[60] T. Team. The GTK Project - A free and open-source cross-platform widget toolkit, 1988. URL `https://www.gtk.org/`.

[61] L. Thomson. *Robot surgeons kill 144 patients, hurt 1,391, malfunction 8,061 times*. 2015. URL `https://www.theregister.com/2015/07/21/robot_surgery_kills_americans/`.

[62] C. S. Timperley, A. Afzal, D. S. Katz, J. M. Hernandez, and C. Le Goues. Crashing Simulated Planes is Cheap: Can Simulation Detect Robotics Bugs Early? In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, pages 331–342, 2018.

[63] P. Vyavahare, S. Jayaprakash, and K. Bharatia. Construction of URDF model based on open source robot dog using Gazebo and ROS. pages 1–5, 03 2019. doi: 10.1109/ICASET.2019.8714265.

[64] S. Wang, S. Heinrich, M. Wang, and R. Rojas. Shader-based sensor simulation for autonomous car testing. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 224–229, 2012.

[65] M. Wen, J. Park, and K. Cho. A scenario generation pipeline for autonomous vehicle simulators. *Human-centric Computing and Information Sciences*, 10, 12 2020. doi: 10.1186/s13673-020-00231-z.

[66] J. Winkler, M. Tenorth, A. K. Bozcuoglu, and M. Beetz. CRAM Memories for Robots Performing Everyday Manipulation Activities. 2014.

[67] J. Winkler, A. K. Bozcuoğlu, M. Pomarlan, and M. Beetz. Task Parametrization through Multi-Modal Analysis of Robot Experiences. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 1754–1756, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[68] Z. Xia, H. Deng, S. Weng, Y. Gan, J. Xiongand, and H. Wang. Development of a robotic system for orthodontic archwire bending. pages 730–735, 05 2016. doi: 10.1109/ICRA.2016.7487200.

[69] T. Yamamoto, K. Terada, A. Ochiaiand F. Saito, Y. Asahara, and K. Murase. Development of Human Support Robot as the research platform of a domestic mobile manipulator, 2019.

[70] P. Śmigielski, M. Raczyński, and Ł. Gosek. Visual simulator for MavLink-protocol-based UAV, applied for search and analyze task. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1177–1185, 2017.