UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Formalization and Runtime Verification of Invariants for Robotic Systems

Ricardo Jorge Dias Cordeiro

Mestrado em Engenharia Informática
Especialização em Interação e Conhecimento

Dissertação orientada por:
Alcides Miguel Cachulo Aguiar Fonseca
Christopher Steven Timperley

2022

Acknowledgments

I would like to thank my coordinator, Prof. Alcides Fonseca, for the exceptional way of teaching not only through the making of my thesis but also throughout all my academic course.

My coordinator Prof. Chris Timperley for taking his time to help me in this chapter of his life where he had to take care of his baby.

My upperclassman Paulo and Catarina for all the advice and help.

All my friends that spent their time with me, know that somehow you helped me through this process.

All my family, in particular my grandparents.

My little brother.

In the end and more importantly my mother for taking care of me all my life and giving me the opportunity to follow this path.

"Dreams breathe life into men, and can cage them in suffering. Men live and die by their dreams, but long after they've been abandoned, they still smolder deep in men's hearts. Some see nothing more than life and death. They are dead! For they have no dreams."

Kentaro Miura in Berserk

Resumo

A Robótica tem uma grande influência na sociedade atual, ao ponto que a falha em algum sistema robótico que seja crucial pode impactar o modo em como nós vivemos, se, por exemplo, um carro autónomo provocar a morte de algum passageiro devido a um defeito, futuros e atuais utilizadores deste modelo irão certamente ficar apreensivos em relação à sua utilização. Assegurar que robôs reproduzam um comportamento correto pode assim salvar bastante dinheiro em estragos ou até mesmo as nossas vidas.

As práticas atuais em relação a testes de sistemas robóticos são vastas e envolvem métodos como simulações, verificação de “logs”, ou testagem em campo, frequentemente, um denominador comum entre estas práticas é a necessidade de um humano pessoalmente analisar e determinar se o comportamento de um sistema robótico é o correto. A automatização deste tipo de análise poderia não só aliviar o trabalho de técnicos especializados, facilitando assim a realização de testes, mas também possivelmente permitir a execução massiva de testes em paralelo que podem potencialmente detetar falhas no comportamento do sistema robótico que de outra maneira não seriam identificados devido a erros humanos ou à falta de tempo.

Apesar de existir alguma literatura relacionada com esta investigação, de uma maneira geral a automatização no campo da deteção de erros ou criação de invariantes continua a não ser adotada, pelo que o estudo apresentado nesta tese é justificado.

Esta dissertação visa assim explorar o problema da automatização na deteção de erros comportamentais em robôs num ambiente de simulação, introduzindo uma linguagem de domínio específico direcionada a especificar as propriedades de sistemas robóticos em relação ao seu ambiente, assim como a geração de “software” de monitorização capaz de detetar a transgressão destas propriedades.

A linguagem de domínio específico necessita de expressar requisitos de determinados estados ou eventos durante a simulação, desta maneira precisa de apresentar determinadas características. Palavras-chave para representar relações temporais de ou entre objetos, como, por exemplo, o robô ”nunca”, ou ”eventualmente”o robô. Referências a estados anteriores da simulação, como, por exemplo, a velocidade do robô está sempre a aumentar, ou seja, é sempre maior que no estado anterior. Atalhos para ser possível referir certas características de ou entre objetos, como, por exemplo, a ”posição”, ”velocidade”ou ”distância”de ou entre robôs.

A linguagem de domínio específico também assume que o sistema robótico irá ser executado por meio da framework ROS (Robot Operating System), que é amplamente utilizada para investigação e na indústria da robótica. A arquitetura do ROS engloba características como

"publish-subscribe" entre "tópicos"e tipos de mensagem, estas características são tidas em conta e foram integradas no desenvolvimento da linguagem.

O "software" de monitorização gerado refere-se a um ficheiro python que correrá sobre a framework ROS. A geração deste ficheiro assume também que a monitorização será feita no simulador Gazebo, isto porque para obter dados como a posição ou velocidade absoluta de um robô durante a simulação é necessário aceder a "tópicos"ROS específicos que na geração do ficheiro de monitorização estão "hardcoded". A geração de um ficheiro capaz de executar a monitorização significa que esta pode ser executada independente de um sistema robótico, permitindo assim a automatização da monitorização a respeito de vários objetos e as suas relações.

Resultados mostram que é possível expressar propriedades temporais e posicionais de e entre robôs e o seu ambiente com o suporte da linguagem de domínio específico. O trabalho mostra também que é possível automatizar a monitorização da violação de alguns tipos de comportamentos esperados de robôs em relação ao seu estado ou determinados eventos que ocorrem durante uma simulação.

Evaluation ?

possiveis problemas, e futuro - proof of language or that it works - better information on the errors - frequency of checking the properties can be modified in some circunstances to not check at every iteration - alargar a outros simuladores - integration with other tools liek scenario generation

Palavras-chave: Robótica, Linguagem de domínio, Deteção de erros, Automação, Monitorização

Abstract

Robotics has a big influence in today's society, so much that a potential failure in a robot may have extraordinary costs, not only financial but can also cost lives.

Current practices in robot testing are vast and involve such methods as simulations, log checking, or field tests. The frequent common denominator between these practices is the need for human visualization to determine the correctness of a given behavior. Automating this analysis could not only relieve this burden from a high-skilled engineer but also allow for massive parallel executions of tests, that could potentially detect behavioral faults in the robotic system that would otherwise not be found due to human error or lack of time.

For my thesis, I have developed a domain-specific language to specify the properties of robotic systems in ROS. Specifications written by developers in this language can be compiled to a monitor ROS module, that will detect violations of those properties. I have used this language to express the temporal and positional properties of robots, and we have automated the monitoring of some behavioral violations of robots in relation to their state or events during a simulation.

Evaluation ?

Keywords: Robotics, Domain-specific language, Error detection, Automation, Monitoring

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Robot arms in car assembly lines, autonomous vacuum cleaners, or cat-like robots to carry food in a restaurant, robotics already have a great impact on our current society. Due to their broad practicality, the quality of software used by robots should be of extreme importance to us.

Robot software as well as the techniques used to test their quality are very field-specific and different from the techniques employed in traditional Software Engineering, mainly because robots are meant to interact with the real world. Automatic tests are barely used in robotics due to multiple factors: cost, complexity, and hardware integration, among others [4].

The goal of this thesis is to remove the need for humans to manually inspect the correctness of robot behavior based on visual inspection, through the study of a possible solution for automation in the testing of robotic systems.

1.1 Motivation

Today, robots are vastly used industrially (medicine, agriculture, etc.) or leisurely (contests, personal use, etc.). The tendency is for robot usage to keep growing at a global level. Robot tasks tend to be repetitive or rather specific, but the robot software tends to be quite different from conventional software. The Cyber-Physical systems of robots are non-deterministic and unreliable, mainly because robots interact directly with the real world. A sensor can return imprecise values since the environment itself can be very hard to predict. As a result, verifying whether a task or movement is correct can be hard for a system to conceive.

Current practices in testing robot software involve, field testing, simulation testing, logs checking, among others. The common denominator among these is that they require a human to analyze the behavior of the robot to determine whether the behavior is correct. Studying possible options for viable automation of tests in robotic systems could lead to an opening on its usage in research and the industry. Allowing for multiple parallel executions of tests not depending on human visualization could improve the quality of current and future robot software.

1.2 Problem Statement

The multiple challenges in robot testing have an influence on planning how to test a robot because there are tradeoffs among choices. While simulation-based tests are a promising approach for automation there is still distrust in the precision and validity of the results. This means that, despite being dangerous and sometimes expensive, real-life robot testing is still the main choice. Both in real-world testing or simulations, human supervising will most likely still be necessary. This is because identifying if a robot fulfills an expected behavior is very hard for the robot itself. For this reason, automatic tests in the robotics field are hardly reliable and hard to implement. The resulting product is a lack of quality in the software across projects. In short, right now in the field is manually costly to identify test scenarios and identify if the robot does what we want.

1.3 Objectives

This work has the objective of showing the potential of automatic tests in robotics and of simplifying their execution. With this in mind, we propose a mechanism that monitors a subset of the components of the robot during or after tests execution. These components aren't arbitrary but are defined with the help of a descriptive high-level language. Not only the components but the test scenario should be described in this language. With the description of this language, one should be able to detect and orchestrate relevant robot components associated with the testing scenario. The language should allow describing a robot property in a simple and intuitive way. This language will need to be supported by a compiler. The compiler should translate the language to a monitoring mechanism. In this way, if a robot doesn't follow the properties defined by the language, either during execution or a log analysis, the compiler will detect an anomaly in the normal behavior of the robot.

1.4 Contributions

The expected contributions of this thesis are below enumerated.

1. Definition of a descriptive high-level language to specify robots properties.
2. Implementation of compiler for the language that can be used for monitoring.
3. Evaluation of the expressive capability of the solution in real-world examples.

1.5 Structure of the document

The document is organized as follows:

- Section 1...
- Section 2...
- Section 3...

Chapter 2

Background & Related Work

In this chapter as a background to the work, the Robot Operating System (ROS) is intruduced. Has for related work, Gazebo is intruduced as the simulation software to be used and GzScenic as a language that allows to specify testing scenarios.

ROS is an open-source framework with a vast collection of libraries and tools that help build robot software. ROS runs on Linux Ubuntu and provides an abstraction between hardware and software. ROS was built with the purpose of cross-collaboration, there are packages for almost everything and no need to reinvent the wheel. ROS is the most widely used tool for writing robot software. Companies like Sony, LG, Rapyuta Robotics, etc., rely on ROS to deliver their products [2].

Robot simulation is an essential tool for testing robots behavior. Gazebo started with the idea of a high-fidelity simulator to simulate robots in outdoor environments under varied conditions. Today it offers the ability to simulate numerous robots in complex distinct environments. Gazebo is an open-source 3D simulator that supports sensors simulation and actuators control under different physics engines [1].

Scenic is a domain-specific language used to describe scenarios. It is a probabilistic programming language and it helps design the cyber-physical systems used in a simulation. Scenic big potential is allowing the randomization of specific scenarios delimited by the language. Although this is true Scenic as a problem, it is designed to support only a specific number of simulators, mainly vehicle simulation [5].

With this in mind, GzScenic was built. The GzScenic tool allows the generation of this type of scenarios in the Gazebo simulator, which is the most popular general-purpose simulator [3].

Chapter 3

Proposed Approach

The proposed approach consists initially in creating a domain-specific language. The language will serve as a way to describe the properties of a robot. For instance, if our robot is an autonomous car navigating on the road, one property could be that the robot stops at stop signs. To describe robot properties, we also need the description of the testing scenario. In the above example, the ”road” and ”stop sign” should be defined in the language as part of the testing scenario, without it there would be no way to describe the above property effectively. To describe the scenario itself we can use GzScenic [3] in order to take advantage of the arbitrary creation of multiple scenario possibilities. This language will then be composed of a new domain-specific language in association with the already established GzScenic language.

Next in the approach, there is a need to build a compiler for the proposed language. The compiler should be able to interpret a property in the language and be able to identify the components of the robot necessary to monitor the said property. The monitorization could take place either during runtime or after using log files. Taking the above example into account, our compiler should have the information of which component of the robot is responsible for the car position as well as the position of the stop sign, it can then monitor the component and check if the property has been broken.

The language should be of high level in the sense that it should be intuitive to the writer. With this approach, the person doing the robot testing shouldn't need so much in-depth knowledge about the robot to perform a test. This is because of the writing simplicity of the language and the removal of the manual labor side of personally monitoring the robot.

The final scheme of the tool proposed is represented in the below diagram.

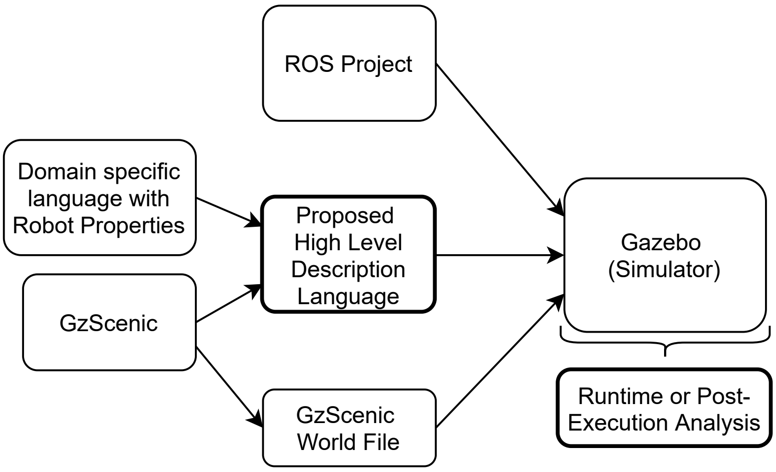

Figure 3.1: Tool for monitoring robot properties.

Appendix A

Bibliography

[1] http://gazebosim.org/.

[2] https://www.ros.org/.

[3] Afsoon Afzal. Gzscenic: Automatic scene generation for gazebo simulator. https://arxiv.org/pdf/2104.08625.pdf, .

[4] Afsoon Afzal. A study on challenges of testing robotic systems. .

[5] Daniel J. Fremont, Edward Kim, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Scenic: A language for scenario specification. https://scenic-lang.readthedocs.io/en/latest/index.html.