软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(6):1867−1881 [doi: 10.13328/j.cnki.jos.006254]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向 ROS 的差分模糊测试方法*

王 颖[1,2], 王冰青[1,2], 关 永[1,3], 李晓娟[1,2], 王 瑞[1,2]

[1](首都师范大学 信息工程学院,北京 100048)

[2](轻型工业机器人与安全验证北京市重点实验室(首都师范大学),北京 100048)

[3](电子系统可靠性与数理交叉学科国家国际科技合作示范型基地(首都师范大学),北京 100048)

通讯作者: 关永, E-mail: guanyong@cnu edu.cn

摘 要: 机器人操作系统(robot operating system,简称 ROS)是一种广泛应用于机器人开发的开源系统,它可以为开发者提供硬件抽象、设备驱动、库函数、可视化、消息传递和软件包管理等诸多功能,应用前景广阔.ROS 集成了可以实现不同功能的功能包,例如定位绘图、行动规划、感知、模拟等等,但其中可能存在一些漏洞,破坏整个机器人系统的安全性和可靠性.提出了一种差分模糊测试方法对 ROS 不同版本的功能包进行测试,找出其中的漏洞.该方法包括测试用例生成和差分模糊测试执行两个模块:首先,对于输入文件进行加载、处理,并基于策略生成的方法生成测试用例文件;其次,节点间使用话题通信机制实现通信,使用上一模块生成的测试用例文件作为统一的模糊输入,对 ROS 不同版本的功能包进行差分模糊测试;接着,对测试结果中的不一致输出进行差异计算并评估,符合评估指标的种子将被保留并反馈给用例生成模块循环生成测试用例,有效提高了种子质量及代码覆盖率;最后分析不一致输出原因,找出漏洞.将该方法应用在机器人坐标转换的实验中,实现对不同参考系下坐标转换的功能包 TF 和 TF2 的测试.最终的实验结果表明:与 TF2 相比,TF 在功能实现上更加准确.TF2 实现坐标旋转变换的函数存在漏洞.

关键词: 差分模糊测试;机器人操作系统;功能包;漏洞检测;可靠性

中图法分类号: TP311

中文引用格式: 王颖,王冰青,关永,李晓娟,王瑞.面向 ROS 的差分模糊测试方法.软件学报,2021,32(6):1867−1881. http://www.jos.org.cn/1000-9825/6254.htm

英文引用格式: Wang Y, Wang BQ, Guan Y, Li XJ, Wang R. Differential fuzz testing of robot operating system. Ruan Jian Xue Bao/Journal of Software, 2021,32(6):1867−1881 (in Chinese). http://www.jos.org.cn/1000-9825/6254.htm

Differential Fuzz Testing of Robot Operating System

WANG Ying[1,2], WANG Bing-Qing[1,2], GUAN Yong[1,3], LI Xiao-Juan[1,2], WANG Rui[1,2]

[1](College of Information Engineering, Capital Normal University, Beijing 100048, China)

[2](Beijing Key Laboratory of Light Industrial Robot and Safety Verification (Capital Normal University), Beijing 100048, China)

[3](National International Science and Technology Cooperation Demonstration Base of Interdisciplinary of Electronic System Reliability and Mathematics (Capital Normal University), Beijing 100048, China)

Abstract: Robot operating system (ROS) is an open source system widely used in Robot development. It can provide developers with hardware abstraction, device driver, library function, visualization, messaging, software package management, and other functions, which

* 基金项目: 国家重点研发计划(2019YFB1309900); 国家自然科学基金(61877040); 首都师范大学交叉研究院项目(19530012005); 上海控安开放课题

Foundation item: National Key R&D Plan of China (2019YFB1309900); National Natural Science Foundation of China (61877040); Cross Research Institute of Capital Normal University (19530012005); Shanghai Security Open Project

本文由“形式化方法与应用”专题特约编辑姜宇副教授推荐.

收稿时间: 2020-08-31; 修改时间: 2020-10-26; 采用时间: 2020-12-19; jos 在线出版时间: 2021-02-07

has an important and broad application prospect. ROS integrates various software packages that can realize different functions, such as positioning drawing, action planning, perception, simulation, etc. However, some vulnerabilities may damage the overall safety and reliability of robot system directly. In this study, an ROS oriented fuzzing method is proposed to test different versions of ROS packages and find out the vulnerabilities. The proposed approach includes two modules: Test cases generation and differential fuzz testing execution. Firstly, load and process the input file, and generate the test cases file based on the strategy's generation. Secondly, communication among nodes is achieved using topic communication mechanism, and the test case files are used as the inputs to carry out differential fuzz testing on the ROS packages. Then, the inconsistent outputs in the test results are calculated, and the seed meet the evaluation indicators are reserved and fed back to the test case generation module to generate test cases, it will improve seed quality and code coverage effectively. Finally, analyze the cause of inconsistent output and find out the vulnerability. This method is applied in the experiment of robot coordinate transformation, testing the packages TF and TF2 that realize coordinate transformation under different reference frames. Final experiment results show that TF is more accurate in function implementation compared with TF2, and there are vulnerabilities in the function of TF2 to realize coordinate rotation transformation.

Key words: differential fuzz testing; ROS; packages; vulnerabilities detection; reliability

Robot Operating System(以下简称 ROS)是 Willow Garage 公司于 2010 年发布的机器人软件平台,能为异质计算机集群提供类似操作系统的功能,适合于机器人的大规模研发.ROS 提供了包括硬件抽象、底层设备控制、常用函数的实现、进程间消息传递以及包管理等操作系统应有的服务,也提供用于获取、编译、编写和跨计算机运行代码所需的工具和库函数,在代码重用、模块化、对等设计、较少依赖编程语言和开放源代码方面具有优势[1].随着工业和技术的发展,机器人越来越多地应用于医疗、军事、无人机和太空探索,因此,机器人开发的安全性及准确性越来越成为大家关注的焦点[2].其中,很多集成于 ROS 的软件包,例如定位绘图、行动规划、感知、模拟等等都发挥着重要作用.因此,每一个功能的精准实现,都直接影响着机器人的整体开发[3].

近年来,机器人涉及无人机、自动驾驶、医疗等多个安全性极强的领域,机器人开发的可靠性和安全性问题也成为了热点研究问题.ROS 设计灵活使用简单,包含丰富的功能设计包,比如机械臂运动规划、机器人定位导航、仿真、可视化工具、坐标变换等,都与机器人开发的重要环节息息相关.因此,ROS 成为了机器人开发领域比较受欢迎的开发系统.但关于对测试 ROS 系统开发的安全性和可靠性,国内外研究大致主要包括对 ROS 中的协议、节点间通信进行形式化验证[4]以及进行一些功能包的静态测试,如使用 Coverity 等静态分析[5]工具对 ros_comm,actionlib 等 ROS 中重要的通信功能包以及核心功能库进行静态测试,并发现了缓冲区溢出、整数溢出等多个危险漏洞,其中,已有 10 个漏洞被 CVE[6]数据库收录.静态分析作为一种传统的软件测试方法[7],主要指不运行被测程序本身,通过分析或检查源程序的语法、结构、过程、接口等来检查程序的正确性;对需求规格、软件设计说明书、源程序做结构分析、流程图分析、符号执行来找错.静态方法通过程序静态特性的分析,找出欠缺和可疑之处,例如不匹配的参数、不适当的循环嵌套和分支嵌套、不允许的递归、未使用过的变量、空指针的引用等,其结果可用于进一步的查错,并为测试用例选取提供指导,是保证系统安全性和可靠性的传统手段.但静态分析这种静态测试的方法常常会产生误报,耗时较长,在一定程度上暴露了传统方法的缺陷,也促使研究出一种更为有效的漏洞挖掘方法.

模糊测试技术[8]是近几年软件测试领域的焦点技术,其原理主要是通过提供非预期的输入并监视异常结果来发现软件故障.它不关心被测试目标的内部实现,而是利用构造畸形的输入数据引发被测试目标产生异常,从而发现相应的安全漏洞[9].一直以来,模糊测试技术被广泛应用于操作系统[10]、数据库[11]、Web 应用程序和服务器[12]、区块链等等;模糊测试[13]不需要程序源码、不受限于被测系统的内部实现细节和复杂程度、不用关心被测对象的实现语言等,所以它可以针对各类格式的文件进行测试[14].与传统的测试方法不同,模糊测试不受限于被测系统的内部实现细节和复杂程度,不需要程序的源代码即可发现问题,省时省力且误报率低.因此,使用模糊测试是对 ROS 系统可靠性保障[15]的一种高效便捷的测试方法.但 ROS 不同于其他系统,它有自己独特的节点、消息、主题等计算图级来处理数据以及一种分布式的通信机制[16]来实现节点间通信,完成消息的传递和功能的实现,还有自己的文件系统级,这些对于我们的测试工作都是很大的挑战.而且,在已知的 ROS 中存在由不同语言或不同算法库实现的不同版本的软件包,却可以实现相同功能.针对 ROS 的这种特殊性,我们对其

使用差分模糊测试[17].即连续提供无效、意外或随机数据作为具有相同功能的多个程序的输入,然后监视这些程序以捕获某些输入上的“不同行为”,这样我们可能会在某些程序中发现错误.

对此,针对 ROS 所集成的各个功能包,本文提出了一种面向 ROS 的差分模糊测试方法:通过搭建差分模糊测试的框架,对 ROS 所集成的各个功能包进行模糊测试,从而验证软件包内各项功能在实际应用中的准确性.首先为 ROS 建立模糊测试框架,模糊器根据不同要求生成测试用例;接着,将差分测试的思想融入到框架中,使用统一的测试用例文件作为模糊数输入,节点间实现通信,执行测试;核心思想是不断为不同的通信模式生成测试用例,以便在执行结果之间找到尽可能多的不一致.最后,我们将该方法应用在机器人坐标转换的实验中,对实现不同参考系下坐标转换的功能包 TF 和 TF2 进行测试.通过引导性模糊测试,成功地在实验过程中触发了 3120 组用例的不一致执行输出,并通过分析找出了触发这些不一致的原因.在本文中,我们主要做了以下贡献.

(1) 我们设计了一种面向 ROS 的差分模糊测试方法,以有效揭示 ROS 中不同版本功能包的差异和漏洞;在该方法中介绍了用于 ROS 的差异信息的评估指标,并设计了用于基础模块的相关算法;

(2) 我们使用该方法来测试 ROS 中影响意义广泛的功能包 TF 和 TF2,检测到许多不一致和安全问题;通过分析原因判断出 TF 较 TF2 更为准确.

本文第 1 节介绍 ROS 和模糊测试相关的基础知识,为本文提到面向 ROS 的差分模糊测试方法的研究提供理论基础.第 2 节对方法进行整体概述,介绍测试用例的生成方法、测试模块的执行以及评价指标的制定.第 3 节是提出方法进的具体实现.第 4 节对该方法进行实验评估,将该方法应用在机器人坐标转换的实验中进行评估并通过分析得出结果.第 5 节进行本文总结及对未来工作的展望.

1 背景知识

1.1 ROS

ROS 是一个用于编写机器人软件的灵活框架,它集成了大量的工具、库、协议,提供了类似操作系统所提供的功能,包括硬件抽象描述、底层驱动程序管理、共用功能的执行、程序间的消息传递、程序发行包管理,可以极大简化繁杂多样的机器人平台下的复杂任务创建与稳定行为控制.

如图 1 所示,ROS 的系统架构设计分为 3 层:操作系统层、中间层、应用层.ROS 的操作系统层依托于 Linux 系统运行,它并不是一个传统意义上的操作系统,无法像 Windows,Linux 一样直接运行在计算机硬件之上.但 Linux 是一个通用系统,并没有针对机器人开发提供特殊的中间件,所以 ROS 在中间层做了大量的工作.中间层主要负责包括 ROS 的通信系统及客户端库,其中最为重要的就是基于 TCPROS/UDPROS 的通信系统,这是基于 TCP/UDP 网络所做的再次封装.通信系统使用发布/订阅、客户端/服务器等模型,实现多种通信机制的数据传输.应用层主要运行一个节点管理器以及集成多个功能包实现相关功能,管理器负责管理整个系统的正常运行,功能包内的模块以节点为单位运行,以 ROS 标准的输入/输出作为接口实现复用,极大地提高了开发效率.

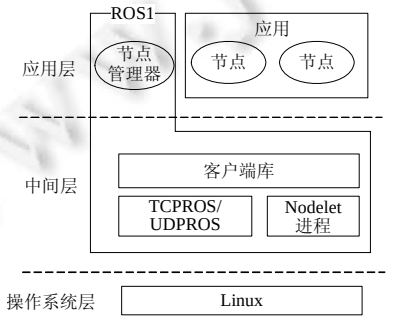


Fig.1 ROS Architecture

图 1 ROS 系统架构

基于该类型的系统架构,ROS 建立了一个分布式通信机制,为用户提供多进程之间的通信服务,主要包括话题通信机制、服务通信机制以及参数管理机制.因此,ROS 程序的运行区别于其他的程序运行,节点运行、消息传递、功能实现都离不开节点间通信,而特殊的分布式通信机制对我们的测试方法也是一个大的挑战.本方法中使用话题通信机制实现节点间通信及话题消息的传递,话题通信机制模型如图 2 所示.话题通信机制主要包括话题发布者、话题订阅者及节点管理器这 3 个角色:节点管理器充当中介角色,管理了所有发布者和订阅者,它们通信前都要把自己的地址、发布或订阅的话题注册到节点管理器当中;发布者节点负责发布话题,订阅者负责订阅话题,二者通信前把自己的地址、发布或订阅的话题注册到节点管理器当中,当发布的话题和订阅的话题能够匹配到时,双方即可进行通信连接.当发布者发布的消息被订阅者准确接收时,节点间可以相互传递话题消息,如与机器人在坐标系下的位姿信息、坐标变换关系等等,通信就被建立起来了.

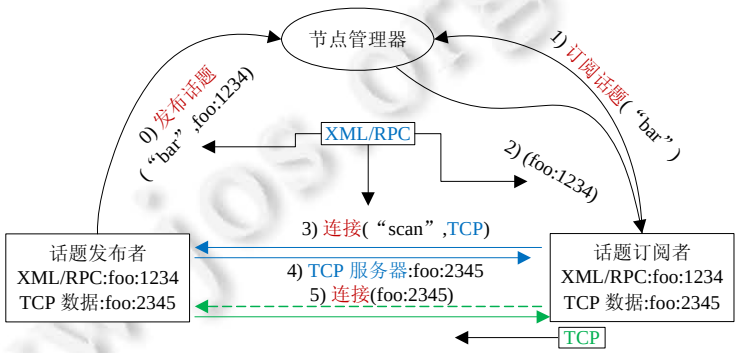


Fig.2 Topic communication model

图 2 话题通信机制模型

1.2 模糊测试

模糊测试作为一种软件测试技术,其核心思想是:自动或半自动地生成随机数据输入到一个程序中,并监视程序异常,如崩溃、断言失败,以发现可能的程序错误,比如内存泄漏,常常用于检测软件或计算机系统的安全漏洞.传统的模糊测试主要包括识别目标系统、确定输入、生成模糊数据、使用模糊数据执行测试、监控系统的行为以及记录缺陷等 6 个核心步骤;通过使用模糊测试技术,可以提高应用程序的健壮性和安全性.但是仅靠模糊测试无法全面了解整个安全威胁或 bug,而且需要大量的时间,所以现在众多研究者将模糊测试与黑箱测试、Beta 测试[18]和其他调试方法一起使用,不仅可以提高测试的覆盖率,还可以节省大量的时间.差分测试有两种情况:一种情况是分析同一程序上执行不同输入的差异,另一种情况是分析多个程序或变体上执行相同输入的差异.针对 ROS 的通信机制的特殊性以及功能包的版本问题,本文我们提出差分测试与模糊测试结合方法,为 ROS 建立模糊测试框架,根据不同要求生成测试用例;接着,将差分测试的思想融入到框架中,节点间使用话题通信机制传递消息、实现相应功能.该方法的核心思想是:不断为不同的通信模式生成测试用例,以便在执行结果之间找到尽可能多的不一致,最终发现漏洞.

2 方法概述

在本文中,基于 ROS 的三层系统架构,我们提出一种面向 ROS 的差分模糊测试方法,主要对应用层(图 1)中各节点进行测试.本节我们将介绍该方法具体的工作流程.被测模块主要是对于机器人开发环节有重要意义的各版本功能包.差分模糊测试的概念较为简单,主要是将差分测试的框架与模糊测试的流程结合在一起,即:连续提供无效、意外或随机数据作为具有多个相同功能的多个程序的输入,监视这些程序来捕捉执行结果间尽可能多的“不一致行为”,这样可能会在程序中发现漏洞.该方法的概述在图 3 中给出,它主要由两个部分组成,即测试用例的生成模块和差分模糊测试执行模块.参数由 CLI 接口输入到用例生成模块,该模块负责对输入数据的处理和种子生成,主要是基于种子策略生成的方法生成所需的测试用例文件;测试用例生成后进入执行模块进

行测试.执行模块的设计与ROS中间层的通信机制相结合,节点间使用话题通信机制传递消息;该模块还负责差异信息的比较以及差分模糊测试的执行,对于测试执行后产生的信息,对其进行评估筛选,保留下的种子再次按策略生成测试文件,进而提高种子质量.我们还会在第 2.3 节介绍用于该方法的评估指标的设定.

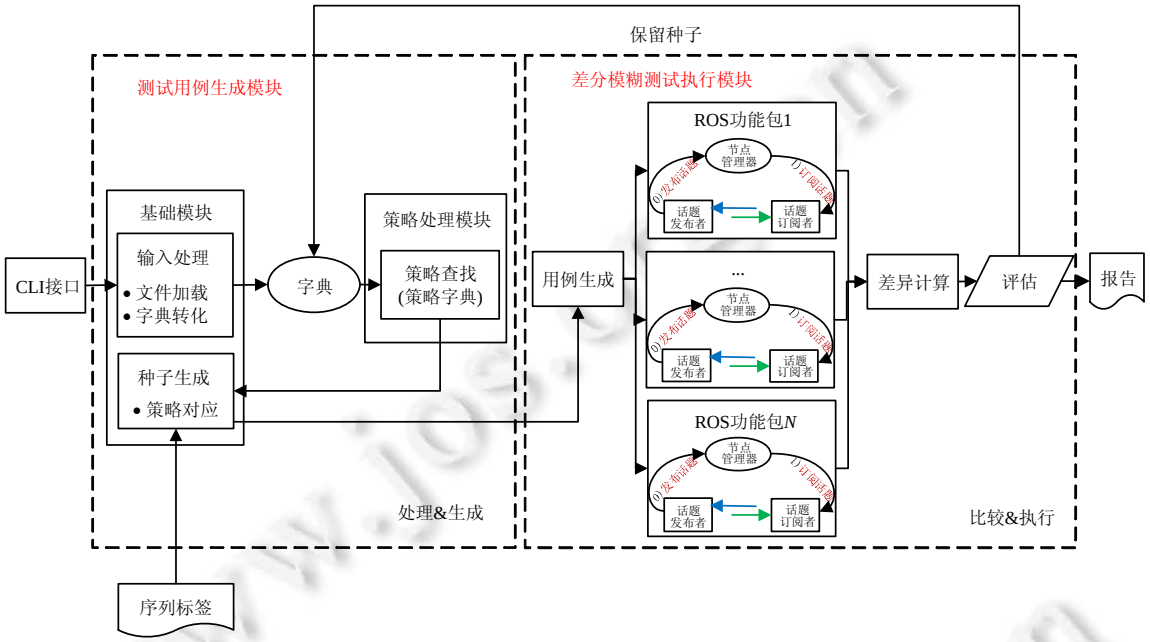


Fig.3 Overview of a differential fuzz testing in ROS

图 3 面向 ROS 的差分模糊测试方法设计图

2.1 测试用例生成

模糊测试中,测试用例可以从头开始生成,也可以从现有的测试用例中变异,因此有基于变异和基于策略生成两种常见的测试用例生成方法:基于变异的生成方法是通过变异已知的测试用例来创建新的测试用例;基于策略生成的方法会从头开始生成新的测试用例,需要知道输入文件的结构,否则它们只会生成随机的字节.在本文提出的测试方法中,我们使用基于策略生成的方法.

测试用例生成模块主要分为基础模块及策略处理模块两个子模块,完成种子的处理及被测用例的生成.如图 3 所示:该模块的输入是由 CLI 接口传递来的界面参数,输出是测试用例;CLI 接口简单来说就是一个命令行输入接口,接收的参数是我们需要的测试用例的消息类型及话题名称;生成的测试用例经种子生成环节进行了序列标识;每一组种子都获取了唯一的序列标号,以便于后续步骤的回溯和分析.首先,基础模块的种子处理部分接收 CLI 接口的输入,对输入进行一系列初步处理后,形成字典文件传递给策略处理模块;策略处理模块使用多个简单策略,如数组、时间戳、字符串等以及组合复杂策略来指导种子生成过程,并将相应策略返回给基础模块进行种子生成,实现策略处理模块与基础模块的策略对应过程,最终为差分模糊测试的具体执行环节生成测试用例文件.

2.2 差分模糊测试执行

差分模糊测试执行模块分为差分测试的执行以及测试结果差异对比及评估的比较两部分.对于 ROS 功能包来说,节点是最小的可执行程序,因此,我们需要以节点为单位建立模型,节点间通过分布式通信机制传递消息实现通信,最终实现相应功能.在本模块,我们首先建立被测模型,模型包括角色、内容、规则等;从上一模块接收测试用例文件后,运行节点,实现消息的传递实现相应功能,结束后得到相应的差异信息;接着,差异信息根据设定好的评估指标进行计算并比较输出结果;最后,符合评估结果的种子将被保留,该部分高质量的种子通过序

列标识回溯到种子文件,继续根据对应策略生成测试用例完成测试过程;当最终执行输出出现不一致时,该模块将记录异常,我们对异常原因进行分析得出报告.

2.3 评估指标制定

为了提高种子质量,我们设立了一定的评估指标.由于多数机器人机械臂的开发与运动和坐标系息息相关,但无论是运动规划还是坐标系转化都将涉及到数据的可靠性与实用性,基于此,我们制定了数据优化和差值合理两个指标.

2.3.1 数据优化

测试用例生成模块可以根据测试需求,即 ROS 中的所有消息类型依照策略生成尽可能多的数据对应文件作为测试用例;该部分文件的生成多数不考虑数据在实际开发应用过程中的合理性;数据优化指标的制定即为测试模块祛除不符合实际应用的数据,如 nan,inf,-inf 等过大或过小的数据,保留有意义的实验数据以此达到数据优化的目的.实现过程中,在差异信息比较时插入条件判断,符合该条件的用例才有可能成为能被保留下的高质量用例.

2.3.2 差值合理

不同程序对数据精度的处理可能存在差异,为了避免这种差异对结果的影响,我们设定差值合理指标;将经过差分模糊测试得出的数据差值进行对比,设计条件判断语句,如差异过小则忽略不计,即两者差值小于 10^{-5} 的数据,我们认为不是异常数据;若达到一定差异值则该部分用例符合该评估指标.对于该指标中阈值的设置也是所依据的;计算机里面的数字是由二进制保存的,在计算机内部有些数字不能准确的保存,于是就保存了一个最靠近的数字.计算机表示浮点数(float 或 double 类型)都有一个精度限制,对于超出了精度限制的浮点数,计算机会把它们的精度之外的小数部分截断.Python 3.5 之后,比较两个 float 是否相等不能仅仅依靠全等来判断;PEP485 提案中提出可以使用近似相等的方法,传入需要比较的两个数和可以容忍的小值,当他们两者的差值小于容忍值时,就可以认为它们相等,这个容忍值一般为 10^{-5}.

基于上述两个指标,我们进一步定义了差异信息的评估系统.测试后得出的差异数据进行数据评估,同时符合两个指标的数据证明其种子质量较高;该部分数据将通过标记序列回溯找到种子文件,并将其按上一模块字典保存的格式规则进行保存,此时字典内数据阈值将被更新;种子生成模块将按更新后的阈值范围继续生成测试用例,进一步执行测试过程.经过评估后的种子质量将远高于经过初始变异的种子,也将进一步提高实验效率以及实验结果的准确性.

3 整体设计

在本节我们将主要介绍图 3 中的关键组成部分以及各部分具体都是如何实现的.

3.1 基础模块

基础模块是整个模糊过程的初始模块,负责完成对输入的处理及最终测试用例文件的生成,主要由输入处理和种子生成两部分组成.

3.1.1 输入处理

输入处理分为文件加载及字典转化两个步骤.ROS 在文件的存储及处理上有自己特殊的文件系统级,不同功能组件分别放在不同的文件夹中,运行期间会根据实际的需要再将它们重新组织在一起.CLI 接口主要负责接收由界面传来的参数,并传递给输入处理过程;数据处理算法如图 4 所示,首先通过加载消息类型导入 ROS 的.msg 格式消息文件,文件声明了 ROS 消息的各个域,可以用来生成基于不同编程语言的承载消息的源代码.接下来创建字典存储函数,将上一步加载进来的参数保存为类型字典;即可以存储消息类型的基本属性,如类型是否复杂、是否有父模块、是否是数组类型,如果是,数组大小、数据阈值范围等信息.

Algorithm 1. Input Data Process.

```
Input : msg_type ← The message type of CLI;
Output : type_dict{·} ← A dictionary including basic properties of input parameter.
function ROS_TYPE_TO_DICT
    module = importlib.import module(msg_type + '.msg')
    ros_msg_loader(module)
    if msg_type = 'array' then
        type_regexp = {⟨module⟩⟨array⟩⟨array_size⟩⟨data_threshold⟩}
        type_match = type_regexp.match(msg_type)
    else if msy_type = 'complex' then
        type_regexp = {⟨complex⟩⟨type⟩⟨type_size⟩⟨data_threshold⟩}
        type_match = type_regexp.match(msg_type)
    else
        type_regexp = {⟨module⟩⟨type⟩⟨type_size⟩⟨data_threshold⟩}
        type_match = type_regexp.match(msg_type)
    end if
    type_dict = type_match
end function
```

Fig.4 Algorithm of input data process

图 4 输入数据处理算法设计图

3.1.2 种子生成

种子生成主要是目标文件按策略处理模块返回的策略进行种子生成,产生测试用例的过程.如图 5 中算法 2 所示,首先创建策略映射函数,将上一模块返回的 ROS 消息字段映射到策略处理模块,并按策略处理模块定义好的具体策略指导种子生成,如后文图 6 所示.模糊实现过程中调用 *map_ros_types*(·)函数生成随机的测试用例,在 *@setting*(·)装饰器中通过 *max_examples* 控制随机数的个数.除此之外,完整的测试用例生成之前,文件中的每组数据使用全局变量添加唯一的序列标识,方便后续过程的种子分析及回溯.

Algorithm 2. Seed Generation Process.

```
Input : ros_clsss ← The ros class to be fuzzed;
        msg_type ← The message type of CLI;
Output : test_data.json ← The generated test case file.
function MAP_ROS_TYPES
    strategy_dict = {·}
    slots_full = list(zip(ros_class_slot, ros_class.slot_types))
    for s_name, s_type in slots_full do
        type_dict = ros_type_to_dict(s_type)
        strategy_dicts[s_name] = type_dict['type']
    end for
    msg = map_ros_types(msg_type)
    test_data.json = json.dumps(msg.x, mag.y, msg.z)
end function
```

Fig.5 Algorithm of seed generation

图 5 种子生成算法设计图

3.2 策略处理模块

策略处理模块基于 Python 中 hypothesis 策略库的 st 模块定义了多个简单策略,如数组、时间戳、字符串等以及组合复杂策略来指导种子生成过程,如定义_Time 策略可以指导 time 类型的种子生成,st.text 策略可以指导字符串类型的种子生成等.

该模块的输入是来自基础模块提供的字典,字典中包含需要变异的消息类型的名称、类型等字段.策略模块将已定义好的策略函数与消息类型进行一一对应,最终将具体策略返回给基础模块由种子生成过程(表1).ROS 中常用的消息类型有很多种,既有像字符串、数组这样的简单类型,也有像 PointStamped 这样包含坐标系位置及时间戳信息的复杂类型.因此,对于 ROS 中复杂消息类型,在该模块中将其策略定义为组合策略,即:当遇到复杂的消息类型时,我们将其对应消息加载函数 *ros_type_to_dict*(·)(图 4),将其加载为字典;再根据字典内的类型属性分别进行简单的策略对应,最终组合在一起并返回一个与复杂消息类型相对应的组合策略.

Table 1 Strategies mapping

表 1 策略对应

消息类型	策略字典对应函数	返回策略
string	*string*(·)	st.text
time	*time*(·)	_Time
duration	*Duration*(·)	_Duration
array	*array*(·)	st.lists
complex	*ros_type_to_dict*(·)	组合策略,按返回字典内容依次生成并组合

3.3 差分模糊测试执行模块

该模块主要是基于模糊循环以及差分框架的具体执行,分为差分模糊测试与结果比较分析两个部分;ROS 的运行依赖于节点的运行与节点间的通信,节点间实现通信是功能包正常功能实现的前提,也是该模块设计中的重要环节.本模块首先建立被测模型,模型包括角色、内容、规则等,建立模型实现广播节点与监听节点间的消息传递;模型中,节点使用话题通信机制传递话题消息,实现节点间基本通信.然后,以上一模块生成的测试用例作为被测部分的输入执行差分测试并得出差异信息,例如,图 6 为一个 JSON 格式的测试用例文件,该测试用例主要是坐标点在坐标系下的坐标值,可以提供给 TF 能包现坐标转换;接着对所得信息按设定好的评估指标进行评估,符合评估指标的种子将被保留进行种子循环生成;最后分析异常原因得出结论报告.


```
{"pointX": 2.220446049250313e-16, "pointY": -0.0, "pointZ": -1.0263818935397904e+146,"sequence":1}
{"pointX": -2.2250738585072014e-308, "pointY": -8.447882830094597e-265, "pointZ": 1e-05,"sequence":2}
{"pointX": 1.175494351e-38, "pointY": 65837.0, "pointZ": 2.289124364299345e-220,"sequence":3}
{"pointX": 4134536477.0, "pointY": 0.0, "pointZ": 4.1925505395493554e+224,"sequence":4}
{"pointX": 0.5, "pointY": 0.0, "pointZ": -2.541120133171423e+16,"sequence":5}
{"pointX": -1.9, "pointY": -1.9, "pointZ": 1.7976931348623157e+308,"sequence":6}
{"pointX": 1.7170682878232216e-214, "pointY": -0.0, "pointZ": 0.0,"sequence":7}
{"pointX": 2.419495290638118e+18, "pointY": 2.419495290638118e+18, "pointZ": 2.419495290638118e+18,"sequence":8}
{"pointX": 0.0, "pointY": 3.402823466e+38, "pointZ": 0.0,"sequence":9}
{"pointX": 2.2217144590000614e-257, "pointY": 2.2217144590000614e-257, "pointZ": 1.3471761022986548e+16,"sequence":10}
{"pointX": 0.0, "pointY": 0.0, "pointZ": -2.355444623753133e+16,"sequence":11}
{"pointX": 1.6752846531840462e-08, "pointY": 1.6752846531840462e-08, "pointZ": 5.581469672229002e+16,"sequence":12}
{"pointX": 0.0, "pointY": 0.5, "pointZ": 2.00001,"sequence":13}
{"pointX": 0.0, "pointY": -0.3333333333333333, "pointZ": 0.0,"sequence":14}
{"pointX": -4065120256786626.0, "pointY": -3.2676322719483536e+176, "pointZ": -1.2153879510478483e+151,"sequence":15}
{"pointX": -1.5, "pointY": -10000000.0, "pointZ": 1.5,"sequence":16}
{"pointX": 1.5, "pointY": -536870912.0, "pointZ": 1.7170682878223774e-214,"sequence":17}
{"pointX": 2.730300405501133e+16, "pointY": 0.3333333333333333, "pointZ": 2.543209201338633e+16,"sequence":18}
```


Fig.6 An example of test cases file

图 6 测试用例举例

4 实验评估

本节将详细介绍实验细节,我们将该方法应用在可以帮助我们简化坐标系间的转换工作的功能包 TF 上.

在机器人环境的搭建中,坐标系是一个非常重要的概念,TF 则是 ROS 为用户提供的与其息息相关的功能包.它使用一种树型数据结构,根据时间缓冲并维护多个参考系之间的坐标变换关系,可以帮助用户在任意时间,将点、向量等数据的坐标,在两个参考系中完成坐标变换.TF 和 TF2 是被开发者广泛应用的两个不同版本,二者可以实现相同功能;对于开发者来说,功能实现的准确性及可靠性将直接影响开发过程.TF 和 TF2 可以帮助我们处理各种数值计算的细节,而数值计算的准确性将直接影响坐标的转换结果.因此我们对 TF 和 TF2 应用差

分模糊测试方法,通过对差异信息的分析来验证两个功能包的准确性.在实验过程我们还将回答以下两个问题:(1) 该方法是否能保证生成高质量的种子?(2) 我们能否通过该方法找到 ROS 功能包中的问题?

4.1 数据和环境设置

所有实验在操作系统为 Ubuntu 16.04 的计算机上执行.为了实验的公平性,实验环境我们都选择 ROS Kinetic 版本.ROS 的发行版本指 ROS 软件包的版本,其与 Linux 的发行版本(如 Ubuntu)的概念类似.推出 ROS 发行版本的目的在于使开发人员可以使用相对稳定的代码库,直到其准备好将所有内容进行版本升级为止.因此,每个发行版本推出后,ROS 开发者通常仅对这一版本的 bug 进行修复,同时提供少量针对核心软件包的改进.而 Ubuntu 16.04 可以支持 ROS Kinetic 的运行.

4.2 评估指标制定

该指标主要由差分模糊测试执行模块制定,包括数据优化与差值合理两部分,只有同时满足两个指标的种子才能被保留;接下来我们将详细说明两个指标在实验中的应用.

4.2.1 数据优化

数据优化指标的制定即为测试模块去除不符合实际应用的数据,如 nan,inf,-inf 等过大、过小或无法表示的数据,保留对实际应用有意义的实验数据,以此达到数据优化的目的.代码过程如图 7 所示:*i* 是预先定义的全局变量,用来为每组用例做序列标识;当一轮测试用例生成后,全局变量 *i* 将按顺序对每组用例进行“打标”;对于生成的测试数据中过大过小或不可描述的数据,我们使用 python 中 math 模块的函数 *math.isnan*(·)及 *math.isinf*(·)来进行判断;若为该种类型的数据,我们将执行 *i*–1 过程,即删除该组序列标识,此时的“标记”将直接赋予下一组符合条件的种子.

```
if math.isnan(msg.point.x) or math.isnan(msg.point.y) or math.isnan(msg.point.z) or
math.isinf(msg.point.x) or math.isinf(msg.point.y) or math.isinf(msg.point.z):
    i=i-1
else:
    json_file.write(json_str)
    json_file.write('\n')
```

Fig.7 Data optimization for *if* statement

图 7 数据优化条件语句

4.2.2 差值合理

不同程序对数据精度的处理可能存在差异,为了避免这种差异对实验结果的影响,我们设定差值合理指标;对于经过差分模糊测试得出的数据差值进行对比,如差异过小则忽略不计,即根据浮点型数据特征,对于两者差值小于 10^{-5} 的数据,我们认为不是异常数据,代码过程如图 8 所示.换句话说,差值在阈值范围内的种子并不能真正触发出不一致结果,因此对于这部分数据我们不必进行手动原因分析,种子也无需保留.我们也对其进行了实验验证,对于转换后产生的两组数据我们使用“全等”及“差值小于容忍值”两种方式进行对比,结果如图 9 所示;两种方式对于两组数据是否一致的判断完全不同,其实数据间的差值过小可以忽略不计,甚至有一些是由于本来应该相等的两个浮点数由于计算机内部表示的原因可能略有微小的误差,这时用“全等”就会认为它们不等,造成“假不一致”的情况.

```
if abs(result1.point.x-result2.point.x)>1e-5 or abs(result1.point.y-result2.point.y)>1e-5
or abs(result1.point.z-result2.point.z)>1e-5
    data.append(json_data)
else:
    pass
```

Fig.8 Difference reasonable for *if* statement

图 8 差值合理条件语句

```
[INFO] [1604908319.138007]: tf1转化之后的坐标: x=-32713.0976352181,y=57134.6113324374,z=0.0000000000
[INFO] [1604908319.139514]: tf2转化之后的数据: x=-32713.0976352181,y=57134.6113324374,z=0.0000000000
[INFO] [1604908319.140046]: 使用1e-5判断的结果: 一致
[INFO] [1604908319.140467]: 使用! =判断的结果:  不一致
[INFO] [1604908319.313755]: tf1转化之后的坐标: x=3588030053.8174605370,y=2054369054.6117179394,z=4192550539549
90331711274766071081555210438084850486439913200072610468528812819717607044574356686871637617197830731975284598
[INFO] [1604908319.314316]: tf2转化之后的数据: x=3588030053.8174600601,y=2054369054.6117177010,z=4192550539549
90331711274766071081555210438084850486439913200072610468528812819717607044574356686871637617197830731975284598
[INFO] [1604908319.315570]: 使用1e-5判断的结果: 一致
[INFO] [1604908319.316389]: 使用! =判断的结果:  不一致
[INFO] [1604908319.513318]: tf1转化之后的坐标: x=0.4339095898,y=0.2484400689,z=-25411201331714232.0000000000
[INFO] [1604908319.514869]: tf2转化之后的数据: x=0.4339095898,y=0.2484400689,z=-25411201331714232.0000000000
[INFO] [1604908319.515390]: 使用1e-5判断的结果: 一致
[INFO] [1604908319.515802]: 使用! =判断的结果:  不一致
```

Fig.9 An example of difference comparison method

图 9 差值对比方式举例

在 3 天内,我们共使用该方法生成了 3 654 组测试用例;对于最终输出产生不一致的原因,我们也对其进行了分析.关于评价指标的设定是否能保证生成高质量的种子,我们也对此进行了实验.我们分别在设定评估指标和没有任何评估指标的情况下,使用该方法进行测试,并对输出不一致的触发情况进行了统计和比较分析,如表 2 所示:经过评估指标后的种子质量有了明显提高,能触发更多的不一致输出.这也回答了我们前面提出的第 1 个问题,即该方法确实可以生成高质量的种子.

Table 2 Output inconsistent data

表 2 输出不一致数据统计

是否使用评估指标	触发不一致输出用例数量(组)	未触发不一致输出用例数量(组)	不一致输出比例(%)
是	3 120	534	85.37
否	2 066	1 588	56.52

4.3 实验流程

关于实验,我们首先确定被测目标.ROS 在应用层为机器人开发者集成了大量的功能包,例如定位绘图、行动规划、感知、模拟等等;其中,坐标系是一个非常重要的概念,TF 则是 ROS 为用户提供的与其息息相关的功能包;它使用一种树型数据结构,根据时间缓冲并维护多个参考系之间的坐标变换关系,可以帮助用户在任意时间,将点、向量等数据的坐标,在两个参考系中完成坐标变换.换句话说,TF 可以帮助我们简化坐标系间的转换工作,更好地实现机器人开发.使用 TF 功能主要包括广播和监听两个环节:广播过程可以向系统中广播参考系之间的坐标变换关系;系统中更可能会存在多个不同部分的 TF 变换广播,每个广播都可以直接将参考系变换关系直接插入 TF 树中,不需要再进行同步.监听过程则接收并缓存系统中发布的所有参考系变换,并从中查询所需要的参考系变换.TF 和 TF2 作为同时存在的两个功能包版本,其应用的准确性也将直接影响机器人功能开发.本文采用差分模糊测试的方法验证 TF 在 ROS 应用中的准确性.我们将差分测试的框架与模糊测试相结合,测试两个不同版本功能包在应用时的准确性.

接下来建立被测模型,如图 10 所示.

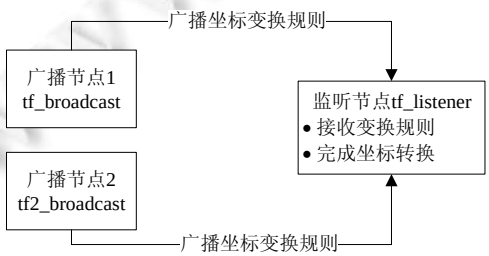


Fig.10 Tested model

图 10 被测模型

在该模型中,我们首先创建实现 TF,TF2 广播功能的广播节点 1(tf_broadcast)和广播节点 2(tf2_broadcast),在两个广播节点中定义相同的变换规则,如设置相同的平移距离和相等的欧拉角旋转角度并完成广播;同时,我们创建一个监听节点 tf_listener 实现 TF 与 TF2 的监听功能,节点同时接收来自两个广播节点广播的消息话题并完成相应的坐标转换.因此,确定被测模块的输入数据为空间坐标系下任意坐标值.具体实验流程如下.

(1) 确定测试目标,生成测试用例

基于上述分析及模型的建立,我们对 TF 和 TF2 使用面向 ROS 的差分模糊测试方法:第 1 步执行测试用例生成模块,确定生成 Pointstamped 类型消息数据作为测试用例;CLI 接口接收参数并传递给基础模块的输入处理,该模块对于输入进行文件加载及按复杂的消息类型的格式完成字典存储;接下来,策略处理模块根据类型字典在策略字典中为其匹配对应策略,并反馈回上一模块;最后,按对应策略完成种子生成,产生 JSON 文件格式的测试用例将被传递给下一模块.

(2) 执行差分模糊测试

下一步为执行模块:首先,在被测模块中使用话题通信机制,使各个节点可以实现通信;被测模块不断接收测试用例作为输入,两个广播器充当话题发布者的角色,利用广播功能发布坐标变换规则,当前位姿信息等话题,监听节点即为订阅者来订阅话题并进行坐标转化,不断得出两组转化后的坐标结果.将每组数据进行差异信息结算并进入评估环节,对于不符合评估结果的数据不予保留,对于符合评估指标的数据我们进行保留并根据种子序列标识回溯到种子文件并将其保存到上一模块的字典中,进而根据保留的高质量种子阈值更新字典内的阈值,继续种子生成环节以生成更多高质量种子.最后结合记录的差异信息,具体分析输出不一致的原因.

4.4 实验结果分析

在实验中,我们共生成了 3 654 组测试用例,统计如图 11 所示:大多数生成的数据范围都在 10^4 以上,这部分数据大约超过了生成总数的 80%.其中,大部分原因是经过评估过程后,我们缩小了生成用例的数值范围,提高了种子质量.也就是说,在测试用例文件中,这 80%的数据更容易触发不一致输出.

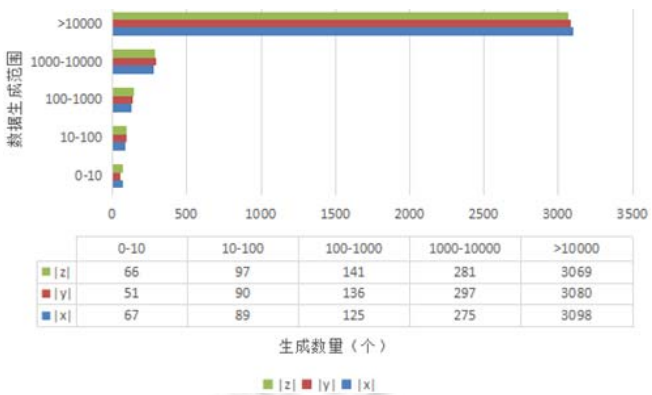


Fig.11 Graph of data generation

图 11 生成数据统计图

接下来,我们将对所触发的不一致结果进行统计及分析.

如图 12、表 3 所示.分析数据我们得知:在生成的测试用例中,大约 85.37%的用例可以触发不一致输出.其中:差值范围在 10^{-6}~10^{-4} 之间的数据大约占整体的 48.92%,这部分数据结果差异较小;但大约有超过 30%的数据差异范围超过了 10^{-2},在一定程度上说明坐标在转换过程中的不准确性对结果确实造成了一定影响.

对于差异造成的原因,我们进行进一步的分析:首先,我们明确 TF 和 TF2 实现的功能是坐标转化.换句话说,就是把一个点在某个坐标系的描述,变换成在另外一个坐标系下的描述.而坐标变换的方式有平移和旋转两种.我们设置通信过程中两个广播节点广播坐标的变换规则,这里的规则即我们为其设置平移和旋转的参数.我们

为两个广播节点设计了以下 3 种方式:

(1) 设置相同的平移参数,设置旋转参数为 0;

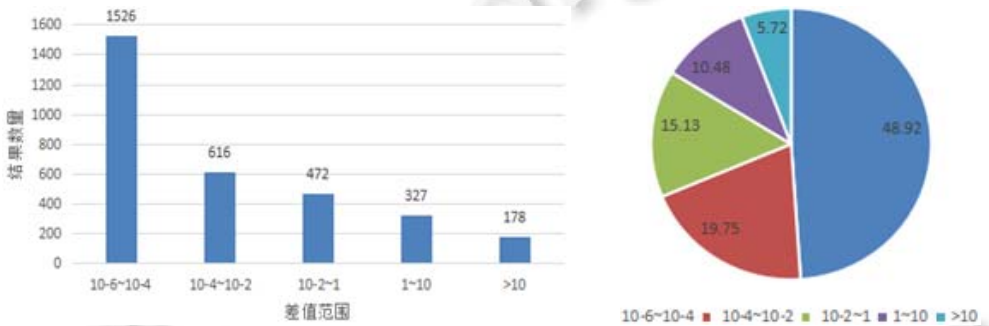
(2) 设置相同的平移参数、旋转参数;

(3) 设置相同的旋转参数,设置平移参数为 0.

Table 3 Data difference

表 3 数据差值统计

差值范围	结果数量(组)	综合占比(%)
10^{-6}~10^{-4}	1 526	48.92
10^{-4}~10^{-2}	616	19.75
10^{-2}~1	472	15.13
1~10	327	10.48
>10	178	5.72



(a) 差值范围统计 (b) 差异结果统计

Fig.12 Graph of difference range

图 12 差值范围统计图

在实验过程结束我们发现:平移参数的设定并不会产生差异数据,而旋转参数的设置会对其产生影响.也就是说,无论我们为其设计相同的旋转和平移,还是只设计旋转不设计任何平移,都会有差异数据的产生.因此,我们回顾旋转参数设置及坐标转化的源代码进一步分析原因.

在 TF 和 TF2 中,我们分别使用 *tf.transformations.quaternion_from_euler*(*roll*,*pitch*,*yaw*)以及 *tf_conversions.transformations.quaternion_from_euler*(*roll*,*pitch*,*yaw*)这两个函数来实现旋转变换,其中,*roll*,*pitch*,*yaw* 为欧拉旋转参数.数据的平移和旋转过程实际上是通过矩阵来完成的,在线性空间里,一个矩阵不但可以描述在同一个基下把一个点运动到另一个点的线性变换,还可以描述一个基在另外一个基下的表示,也可以表示一个基到另一个基的线性变换.平移需要涉及矩阵的加减,而旋转则与矩阵相乘息息相关.TF 为我们定义函数简化计算过程,但实际上,通过函数计算得出的结果并不准确.对于实验过程中产生的 3 119 组不一致结果,我们对其中 977 组差值范围大于 10^{-2} 的数据进行了分析,统计结果如表 4 所示.每组差值范围中计算结果与 TF 转换得出结果更接近的约为 80%.

Table 4 Analysis of conversion results

表 4 转换结果分析统计

差值范围	结果数量(组)	与 TF 结果相近	与 TF2 结果相近	TF 综合占比(%)
10^{-2}~1	472	379	93	80.30
1~10	327	261	66	79.81
>10	178	145	33	81.46

接下来,我们以其中一组数据举例来说明我们是如何对其进行进一步计算及详细的分析对比的.

以图 13 的一组实验结果为例,经过相同变换规则转化后的数据出现了不一致情况,我们分析实现转化的过

程发现,每组数据都通过平移旋转函数设置了相同的参数值,但经上述实验,我们得知平移设置不会影响结果,因此,造成差异的原因在于旋转的设置上.我们提到过关于旋转 TF 及 TF2 分别使用 *tf.transformations.quaternion_from_euler*(*roll*,*pitch*,*yaw*)及 *tf_conversions.transformations.quaternion_from_euler*(*roll*,*pitch*,*yaw*)这两个函数来实现,两个函数能否真正定义相同的坐标变换也就成为了关键问题.

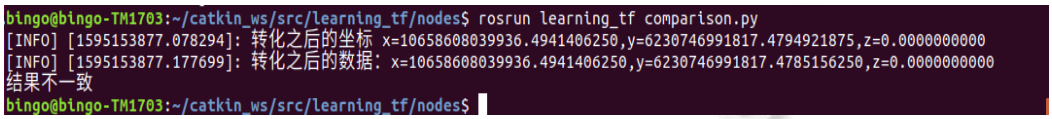


Fig.13 An example of experimental results

图 13 实验结果举例

函数的功能是简化变换的计算过程,但在实现过程中,输入的参数值都会由欧拉角值转化为四元数,最后转化为旋转矩阵进行计算.我们将函数内部欧拉角转化为四元数最后转化为旋转矩阵的结果输出,发现相同的欧拉角值通过两个函数可以转化为相同的旋转矩阵,但前提是保持旋转矩阵内的数据精度设置.根据数据类型,我们得出现在的旋转矩阵内的数据精度与浮点型保持一致,均为小数点后 8 位.也就是说,我们看到的结果也许只是精度受限.为了验证猜想,接下来我们提高转换矩阵的输出精度,最终发现:当我们提到精度到 10^{-16},两个旋转矩阵就会出现明显差异.而两个转换函数在实现坐标转换功能的过程中会涉及到旋转矩阵的加法及乘法,这就造成了误差的累积,最终导致坐标不能实现准确转换.

虽然我们找到了不一致产生的原因,但这种不一致是否会对我们的实际应用产生影响?回答当然是肯定的.因为界面简洁使用方便等原因,现在很多开发者推荐使用 TF2 而“弃用”TF,但通过测试,对于两个功能包在功能实现上的准确性还是有问题的.为了验证两组不一致的实验结果哪一个更加准确,我们使用精度提高后的旋转矩阵数据手动计算多组实验数据,如图 13 内的结果是将相同的原始种子数据通过设置旋转参数为(0,0,1),使用两个功能包实现转换后得出的.我们使用高精度矩阵计算转换,得出的值与图中数据对比发现:我们计算的出的结果与使用 TF 最终得出的结果差异约为 10^{-5},但与 TF2 最终得出的结果差异约为 10^{-2}.也就是说,使用 TF 的结果更加准确.

TF 所实现的坐标转换功能对 ROS 的开发和应用都有着重要意义,而作为一个集成开发平台,ROS 中还包含很多其他重要功能的功能包.我们的方法可以面向整个 ROS 系统,在具体应用过程中,只需要功能包提供 CLI 接口参数即数据消息类型,我们就可以使用该方法对其生成相应的测试用例文件并执行差分模糊测试.至此,我们也回答了本节开始提出的第 2 个问题,我们确实能通过该方法找到 ROS 功能包中的问题.

5 总结展望

本文提出一种面向 ROS 的差分模糊测试方法,通过搭建差分模糊测试的框架,对 ROS 节点间通信进行模糊测试,从而验证软件包内各项功能在实际应用中的准确性.在该方法中,我们主要设计了测试用例生成及差分模糊测试执行模块两个部分,通过用例的生成和差分模糊测试的执行得出差异信息并进行异常记录;除此之外,为了提高种子质量,我们还设计了两个评估指标,同时符合两个评估指标的种子将被保留至测试用例生成模块.对于输出的不一致结果,我们通过手动分析异常原因,得出最终报告.我们将该方法应用在机器人坐标转换的实验中,对实现不同参考系下坐标转换的功能包 TF 和 TF2 进行测试.最终,通过实验及结果对比分析得出:与 TF2 相比,TF 在功能实现即坐标的转换上更加准确.

虽然我们使用差分模糊测试的方法发现了 ROS 功能包中问题所在,但在方法的设计上还存在可以优化的地方,例如测试用例的生成可以加入变异策略、使用策略变异与策略生成结合的方法提高种子质量、增加代码覆盖率,这也是我们下一步想要完善的地方.

References:

[1] Zhang JW, Zhang LW, Hu Y. Open Source Robot Operating System-ROS. Beijing: Science Press, 2012. 1–4 (in Chinese).

[2] Quigley M, Gerkey BP, Conley K, Faust J, Ng AY. ROS: An open-source robot operating system. In: Proc. of the ICRA Workshop on Open Source Software. Piscataway: IEEE, 2009. 1–6.

[3] Liu RJ, Wang F, Zhang Q. ROS based trajectory planning of robotic arm. Navigation, Positioning and Timing, 2016,3(6):16−24 (in Chinese with English abstract). [doi: 10.19306/j.cnki.2095-8110.2016.06.016]

[4] Bertolino A. Software testing research: Achievements, challenges, dreams. In: Matthew D, James H, eds. Proc. of the Future of Software Engineering (FOSE 2007). IEEE, 2007. 85−103. [doi: 10.1109/FOSE.2007.25]

[5] Chen W. Research on vulnerability detection technology in static analysis of Java Programs [MS. Thesis]. Beijing: Beijing University of Posts and Telecommunications, 2018 (in Chinese with English abstract).

[6] Zhu YF, Li LJ. Security vulnerability and discussion on CVE standard. In: Proc. of the 19th National Conf. on New Computer Technology and Computer Education. Chengdu: Southwest Jiaotong University Press, 2008. 577−580 (in Chinese with English abstract).

[7] Dai LC, Zhang YR, Li FZ. A brief analysis of traditional software testing methods. Science and Technology Wind, 2011,(16): 136−137 (in Chinese with English abstract). [doi: 10.3969/j.issn.1671-7341.2011.16.118]

[8] Sutton M, Greene A, Amini P. Fuzzing: Brute Force Vulnerability Discovery. Beijing: Machinery Industry Press, 2007. 29−41.

[9] Li H, Zhang C, Yang X. Survey of OS kernel fuzzing. Journal of Chinese Computer Systems, 2019,40(9):1994−1999 (in Chinese with English abstract). [doi: CNKI:SUN:XXWX.0.2019-09-033]

[10] Mao DD, Ye XJ, Xie Feng, *et al*. Automata based database management system vulnerability mining. In: Proc. of the Information Security Vulnerability Analysis and Risk Assessment Conf. Beijing: China Information Security Assessment Center, Tsinghua University. 2011. 98−108 (in Chinese with English abstract).

[11] Da XW, Wang XC, Chen ZH. Web application vulnerability mining method based on improved fuzzy testing. Computers and Modernization, 2016,(8):100−104 (in Chinese with English abstract). [doi: 10.3969/j.issn.1006-2475.2016.08.021]

[12] Fu ML, Wu LF, Hong Z. Research on intelligent contract security vulnerability mining technology. Computer Applications, 2019, 39(7):1959−1966 (in Chinese with English abstract). [doi: 10,11772/j. issn. 1001-9081.2019010082]

[13] Gao J, Xu YW, Jiang Y, Liu Z, Chang WL, Jiao X, Sun JG. EM-Fuzz: Augmented firmware fuzzing via memory checking. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems. Circuits Syst., 2020,39(11):3420−3432.

[14] Fu Y, Ren M, Ma F. EVMFuzzer: Detect EVM vulnerabilities via fuzz testing. In: Marlon D, Dietmar P, eds. Proc. of the 2019 27th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. New York: Association for Computing Machinery, 2019. 1110−1114. [doi: 10.1145/3338906.3341175]

[15] Sun Y, Zhang S. Research and implementation of key technologies of service robots based on ROS. Software, 2019,40(11): 186−190, 194 (in Chinese with English abstract). [doi: 10.3969/j.issn.1003-6970.2019.11.042]

[16] Bai CC, Cheng WY, Guo HT. Brief analysis of open source ROS robot operating system. Science and Information Technology, 2019,(36):2 (in Chinese with English abstract).

[17] Guo J, Jiang Y, Zhao Y, Chen Q, Sun JG. DLFuzz: Differential fuzzing testing of deep learning systems. In: Proc. of the 26th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (FSE). New York: Association for Computing Machinery, 2018. 4–9. [doi: 10.1145/3236024.3264835]

[18] Dolan RJ, Matthews JM. Maximizing the utility of customer product testing: Beta test design and management. Journal of Product Innovation Management, 2010,10(4):318−330. [doi: 10.1111/1540-5885.1040318]

附中文参考文献:

[1] 张建伟,张立伟,胡颖.开源机器人操作系统——ROS.北京:科学出版社,2012.1−4.

[3] 刘汝佳,王芳,张强.基于 ROS 的机械臂轨迹规划研究.导航定位与授时,2016,3(6):16−24. [doi: 10.19306/j.cnki.2095-8110.2016.06.016]

[5] 陈玮.Java 程序静态分析中的漏洞检测技术研究[硕士学位论文].北京:北京邮电大学,2018.

[6] 朱一枫,李玲娟.安全漏洞问题及 CVE 标准的探讨.见:第 19 届全国计算机新科技与计算机教育学术大会论文集.成都:西南交通大学出版社,2008.577−580.

[7] 戴凌宸,张朕荣,黎丰泽.传统的软件测试方法浅析.科技风,2011,(16):136–137. [doi: 10.3969/j.issn.1671-7341.2011.16.118]

[9] 李贺,张超,杨鑫.操作系统内核模糊测试技术综述.小型微型计算机系统,2019,40(9):1994–1999. [doi: CNKI:SUN:XXWX.0.2019-09-033]

[10] 冒冬冬,叶晓俊,谢丰.基于自动机的数据库管理系统漏洞挖掘.信息安全漏洞分析与风险评估大会论文集.北京:中国信息安全测评中心,清华大学,2011.98–108.

[11] 达小文,王晓程,陈志浩.基于改进模糊测试的 Web 应用漏洞挖掘方法.计算机与现代化,2016,(8):100–104. [doi: 10.3969/j.issn.1006-2475.2016.08.021]

[12] 付梦琳,吴礼发,洪征.智能合约安全漏洞挖掘技术研究.计算机应用,2019,39(7):1959–1966. [doi: 10,11772/j.issn.1001-9081.2019010082]

[15] 孙弋,张松.基于 ROS 的服务机器人关键技术研究与实现.软件,2019,40(11):186–190,194. [doi: 10.3969/j.issn.1003-6970.2019.11.042]

[16] 白冲冲,程文雅,郭洪佚.浅析开源 ROS 机器人操作系统.科学与信息化,2019,(36):2.

王颖(1996－),女,硕士,CCF 学生会员,主要研究领域为软件安全验证,模糊测试.

王冰青(1997－),女,硕士,CCF 学生会员,主要研究领域为软件安全验证,模糊测试.

关永(1966－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为形式化验证,高可靠嵌入式系统,机器人.

李晓娟(1968－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为嵌入式系统形式建模与验证,网络协议分析.

王瑞(1981－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为形式化方法,软件安全验证.