

基于软件在环的智能驾驶仿真测试方法研究^①

黄志凌，李志杰，吕豪，勾天，肖迪
重庆长安汽车股份有限公司

【摘要】 仿真测试因场景泛化灵活、测试效率高、测试成本低、可实现真实交互式测试等优势，可有效加速测试，缩短开发周期，加快智能驾驶产品迭代速度，已成为智能驾驶功能测试评价不可或缺的重要环节。受限于算力性能等因素，当前仿真测试过程人工依赖性仍较高，仍存在脚本代码生成效率不高、转换过程繁琐等问题。本文提出了一种基于软件在环的智能驾驶仿真测试方法。首先对测试需求文档进行转置。然后对生成的执行任务书和步骤文档进行校验。最后基于 Pytest 测试框架和 Jinja2 模板引擎，将步骤文档转换为脚本代码。对比分析结果显示，相较于 ECU-TEST，本文提出的测试方法实施步骤简单，生成脚本时间仅需 0.1%，效率更高。

【关键词】 智能驾驶，仿真测试，Pytest，脚本转换

Research on Intelligent Driving Simulation Testing Method Based on Software in the Loop

Huang Zhiling, Li Zhijie, Lyu Hao, Gou Tian, Xiao Di
Chongqing Changan Automobile Co., Ltd.

Abstract: Due to the advantages of flexible scenario generalization, high testing efficiency, low testing cost, and the ability to achieve real interactive testing, simulation testing can effectively accelerate testing, shorten development cycles, and accelerate the iteration speed of autonomous driving products. It has become an indispensable and important link in the evaluation of intelligent driving function testing. Due to factors such as computational power performance, the current simulation testing process still relies heavily on manual labor, and there are still issues such as low efficiency in script code generation and cumbersome conversion processes. This article proposes a software in the loop based intelligent driving simulation testing method. Firstly, transpose the testing requirements document. Then verify the generated execution task book and step documents. Finally, based on the Python testing framework and the Jinja2 template engine, convert the step document into script code. The comparative analysis results show that compared to ECU-TEST, the testing method proposed in this article has simple implementation steps, only requires 0.1% to generate scripts, and is more efficient.

Key words: intelligent driving simulation, simulation testing, pytest, script transform

引言

智能驾驶技术被认为是人工智能技术在汽车与交通领域中最为典型的应用^[1-4]。《中国制造 2025》中明确指出，车辆智能驾驶系统可降低交通事故的发生概率，最大限度保障驾乘人员的安全^[5]。系统测试作为智能驾驶开发流程中的重要一环，在各大车企的 V 型开发流程中占据重要地位，是整车测试前的重要步骤之一^[6]。

智能驾驶系统大规模投放前的系统测试有助于企业规避可能发生的安全风险，降低后期的召回风险^[7]。随着智能驾驶等级的提升，智能驾驶系统面临海量场景测试挑战。仿真测试因效率高、成本低等优势，已成为测试验证系统安全的重要支柱之一，有利于弥补实车测试的体量不足^[8]。当前仿真测试通常以数学建模为基础，通过智能驾驶应用场景的数字化还原，可建立一个趋近于真实世界的模型^[9]。通过对模型输出仿真测试数据的分析，实现智能驾驶系统的测试验证。智能驾驶仿真测试方法包括模型在环（Model-in-the-Loop, MIL）、软件在环（Software-in-the-Loop, SIL）、硬件在环（Hardware-in-the-Loop, HIL），驾驶员在环（Driver-

in-the-Loop, DIL）等^[10]。首先，通过 MIL 测试，集成车辆动力学、传感器、驾驶员和交通环境等多种模型，优化算法和控制参数，以满足期望的功能需求和性能指标；然后，对控制模型进行代码编译，通过 SIL 开展测试验证；最后，基于实际硬件设备，通过 HIL 开展测试验证^[11]。各技术优缺点对比见表 1。

表 1 测试方法优缺点对比

方法	优点	缺点
VIL	1. 真实物理部件多 2. 保真度高	1. 成本较高 2. 效率一般
HIL	1. 硬件组合灵活 2. 可验证整车逻辑与控制器间时序	1. 执行器不全，整车性能无法测试 2. 造价高昂，无法大规模部署
SIL	1. 纯软环境测试效率高 2. 可加速实现超实时仿真	1. 无法覆盖硬件性能测试 2. 数字化建模前期投入高
MIL	1. 可介入时间最早 2. 测试及问题分析方便	1. 模型复杂度越高效率越低

相较于其他测试方法，SIL 测试开发更为成熟，研究与应用更为广泛。SIL 测试可实现超实时加速仿真，可加快百万千米级仿真测试的达成。SIL 仿真测试方法针对智能驾驶系统的功能逻辑测试包含测试案例设计、场景搭建、测试脚本编写三部分^[12]。尤文冰^[13]、徐舒^[14]等人的研究，实现了场景搭建的自动化，但并未考虑测试案例和测试脚本的自动化生成。马勤政^[15]在云平台中引入自动化测试脚本与自动生成测试案例技术，提高了自动化测试的覆盖范围，但未对测试案例开展校验，转换准确性有限。

伴随智能驾驶系统等级提升，测试案例、测试场景不断细化，测试脚本数量呈爆发式增长。目前，针对测试案例和测试脚本自动化生成的相关研究尚不全面，主要表现为，缺少对转换过程的自动化校验和相关日志输出、转换过程依赖国外软件、自动化代码生成过程繁琐等问题，导致测试效率低下。

为进一步提升智能驾驶系统仿真测试效率，本文提出了一种基于软件在环的智能驾驶仿真测试技术方法。以 Python 开发平台为基础，自动生成仿真测试案例，并对案例信息和步骤文档进行匹配校验，最后基于 Pytest 测试框架和 Jinja2 模板，将自然语言转换为测试脚本代码。

本文所提方法生成的测试脚本具有良好的可移植性，能够根据不同仿真软件的脚本框架进行修改和适配。与传统自动化测试软件 ECU-TEST 的对比分析结果表明，本文所提方法可有效缩短算法验证周期、提高编写效率、降低人力资源投入，部分替代国外软件在智能驾驶系统开发中的应用，具有显著的研究意义与应用价值。

1 基于软件在环的测试方法

针对脚本代码转换过程繁琐，测试效率低等问题，本文提出了一种基于软件在环的测试方法。该方法以 Python 平台为基础，可从需求文档中提取关键信息生成步骤文档；然后对生成的步骤文档进行自动化校验，检测所生成测试案例是否准确；接着构建测试案例模板；最后，基于生成的步骤文档与测试脚本框架，将生成的测试案例文字描述转化生成为测试脚本代码。整体测试方法的系统结构如图 1 所示。

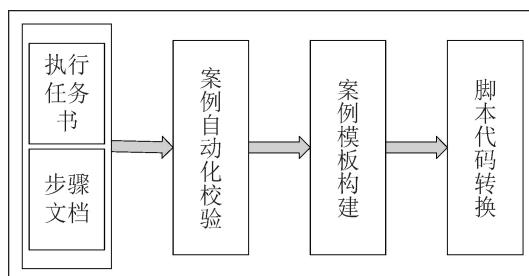


图 1 基于软件在环的仿真测试系统结构图

1.1 测试执行任务书和步骤文档生成

本文采用关键词识别的方式，提取需求文档信息并生成测试执行任务书。关键词包括需求 ID、用例 ID、测试概述、测试条件、执行操作、判断条件、测试优先级等七项指标。将提取的信息储存于数据表格。利用 Python 进行表格数据读取，提取测试案例的文字信息，分别以不同功能命名，生

成测试执行任务书。按照产品技术规范指定的模板，将案例号从大到小、功能由易到难排序，生成步骤文档。生成的执行任务书和步骤文档分别如图 2a、图 2b 所示。



图 2 测试所生成文档

转置文档生成同时，记录转置过程的错误日志，并通过自定义函数，实现对有效案例与无效案例的区分。软件界面如图 3 所示。



图 3 软件执行界面

1.2 测试案例自动化校验

为确保所生成的测试执行任务书的准确性和有效性，对所生成的测试步骤文档和需求文档进行校验。校验流程图如图 4 所示。

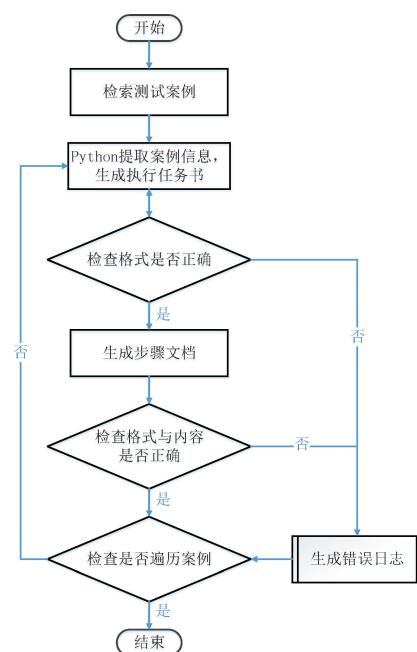


图 4 自动化校验流程

首先通过 Python 工具检索测试案例，并根据测试案例生成执行任务书；然后确认执行任务书格式是否正确；随后

生成步骤文档，利用正则表达式匹配校验步骤文档信息。最后检查是否遍历所有步骤信息，同时输出各步骤错误日志。

1.2.1 案例信息校验

根据提取的测试案例信息，将其拆分为步骤文档表格。包括案例 ID、测试概述、测试条件、执行操作、判断条件、观测接口，以及输入接口等。运用 Python 中的 Pandas 分析库，将表格中的信息存储到字典中。随后采用正则表达式匹配测试步骤，进行格式检查，同时记录校验过程。其运行时序如图 5 所示。

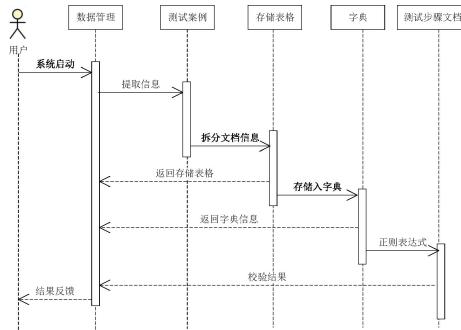


图 5 转置步骤文档信息校验

1.2.2 案例步骤校验

步骤文档的匹配校验以标点符号作为标识符。将测试步骤的格式转变为特定格式，与前期保存在字典的信息比较。检查步骤编号是否缺失或增加，并记录输出日志。

1.3 测试案例模板构建

本文采用 Pytest 测试框架和 Jinja2 模板语言进行用例代码转换，可提高代码的统一性和可读性。基于 Pytest 测试框架和 Jinja2 模板语言，将信号引用和步骤顺序递进过程作为固定结构，来构建用例代码的转化模板；随后，将对应的信号名与参数值写入模板中。在代码转化过程中，测试条件、执行操作、判断条件等信息需要按照既定顺序逐步进行。其流程如图 6 所示。



图 6 模板构建流程

1.4 测试脚本代码转换

测试脚本代码转换过程分为重组、分割、回传三个部分。重组将存储的信息按照数字逻辑顺序依次排序重组，存放于对应的列表中，并通过相关函数实现调用。分割则运用 Python 中的正则匹配函数，单独存储信号名与参数值作为参数化变量。回传将存储的参数化变量按对应顺序写入搭建好的测试案例模板中。其代码时序如图 7 所示。

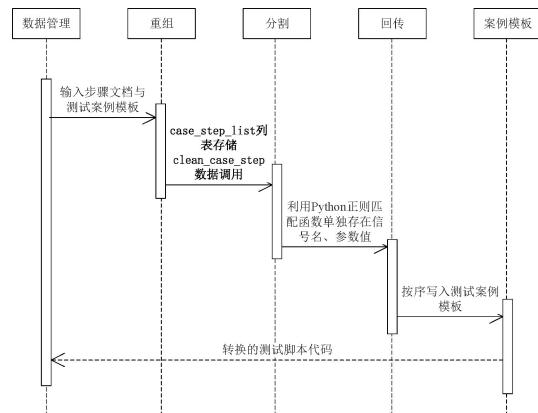


图 7 脚本代码转换

最后，基于开发者增删改查的需要，将代码转换过程中的步骤，通过日志形式输出。

2 基于 ECU-TEST 的测试方法

ECU-TEST 由德国 TraceTronic 公司开发，常用于系统测试验证。ECU-TEST 软件底层采用 Python 语言编写，可支持 Python 脚本调用。其功能较多，精度较高，支持在线升级，支持多操作系统，在智能驾驶系统开发中应用广泛。ECU-TEST 的使用界面如图 8 所示。

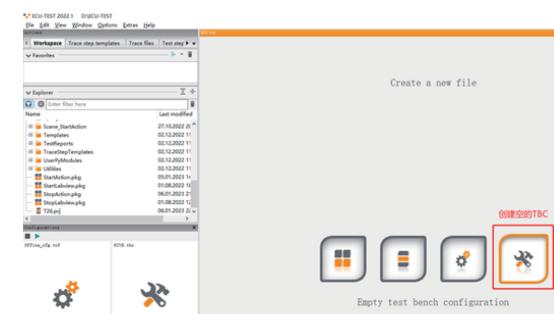


图 8 ECU-TEST 使用界面

ECU-TEST 测试软件测试步骤如下。

- 1) 创建测试套件。将多个测试用例进行组合，创建为一个测试套件，形成一个较为完整的测试计划，用于测试完整功能。
- 2) 编写测试用例。自定义用例属性，对系统功能进行输入、输出、逻辑控制等操作。
- 3) 配置测试环境。配置选择模型、版本和固件版本等信息。
- 4) 运行测试用例。选择运行测试套件中的测试用例。

运行测试用例前，需指定测试用例的输入、输出、控制逻辑等参数。

5) 分析测试结果。自动输出分析测试结果，并为用户生成测试报告。

ECU-TEST 在其使用过程需手动配置信息，过程繁琐，工作量较大。其次，软件的许可费用昂贵，易受国际形势影响，难以保证长期稳定使用。

3 系统测试

3.1 测试准备

为验证本文的方法的效率优势，采用包含 10000 个测试用例的实际项目做测试对比。测试中采用同一测试人员作为试验者，以减少个体差异带来的测试误差。测试人员能同时熟练应用 ECU-TEST 和本文的方法，每天固定工作时间为 8 小时。

3.2 转换时间测试

效率最直观的衡量标准为转换时间。本文分别对测试用例转换时间和后续测试用例更改情况下的转换时间进行对比。

3.2.1 转换时间测试

针对转换时间测试，本文设置的测试用例数量为 100、500、1000、1500、2000、2500 个。分别对本文方法和 ECU-TEST 方法的转换时间进行对比。其对比结果如图 9 所示。

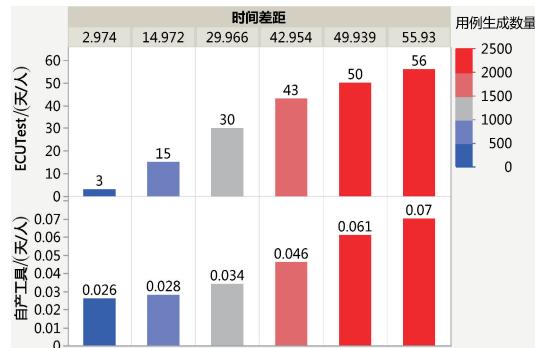


图 9 转换时间对比

由上图可知，本文所用方法的转换时间最少为 0.026 天，最多为 0.07 天。而对比方案中，其转换时间均在 72h 以上，最长为 448h。本文方法的转换时间仅为 ECU-TEST 方法的 0.1%。

3.2.2 变更用例后转换时间测试

测试用例受前期开发和后期实车测试的影响，会不断修改。因此，测试本文方法的可维护性，具有一定的实际价值。在前期测试基础上，分别设置更改 100、500、1000、1500、2000、2500 个测试用例。其转换时间对比结果如图 10 所示。

由上图可知，本文所用方法，在变更用例后的转换时间依然大幅低于对比方法。其转换时间最高仅为 ECU-TEST 方法的 1.5%。本文方法的转换时间和变更用例后的转换时间均优于 ECU-TEST。

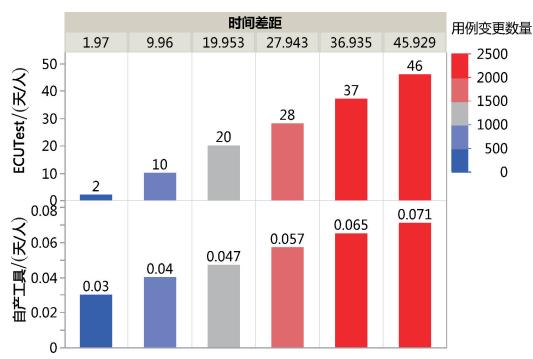


图 10 变更用例后转换时间对比

3.3 准确率测试

准确率作为测试的重要指标，能够客观反映测试方法性能。为反映测试方法准确率的变化情况，共设置 10000 个测试用例，以 100 为坐标间隔，查询并记录转换用例的准确率。两种方式的准确率如图 11 所示。

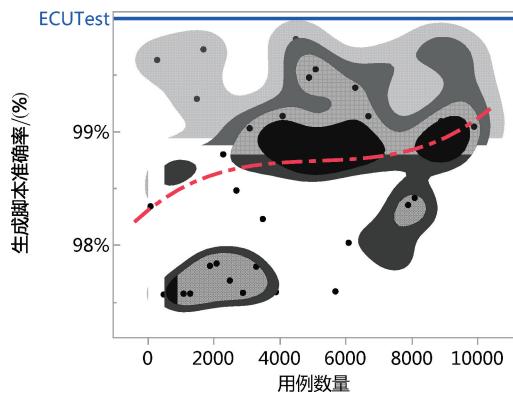


图 11 准确率对比

由上图可知，ECU-TEST 测试方法因采用人工输入，在输入之初就可实现检查，因此其准确率基本为 100%，且波动较小。而本文方法其准确率基本在 97%-100% 之间波动。随着测试用例数量的增加，准确率基本稳定在 99%。

3.4 稳定性测试

保证软件的稳定运行是技术人员的首要任务。在转换测试中记录中央处理器（CPU）使用率和内存占用率，其结果如图 12 所示。

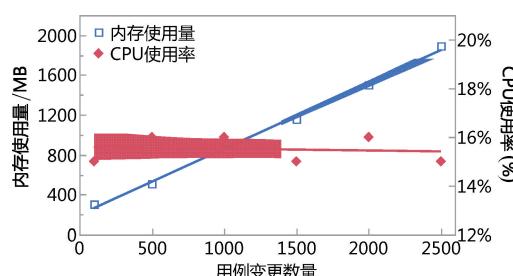


图 12 CPU 使用率和内存占用率

由上图可知，本文方法的内存占用率随测试用例的增

加,从303MB增至1891MB,CPU占用率基本不变,维持在15.5%左右。本文方法受内存影响较大,其内存占用随用例数量增加保持线性增加。

4 结论

本文提出了一种基于软件在环的智能驾驶仿真测试方法,以Python开发平台为基础,对需求文档进行转置,生成执行任务书和步骤文档,实现执行任务书和步骤文档的自动化校验。最后将所生成的步骤文档转换为测试脚本代码,并与ECU-TEST软件进行了对比分析。具体结论如下。

参 考 文 献

- [1] BENGLER K, DIETMAYER K, FARBER B, et al. Three Decades of Driver Assistance Systems: Review and Future Perspectives [J]. IEEE Intelligent Transportation Systems Magazine, 2014, 6 (4): 6-22.
- [2] 中国人工智能学会. 中国人工智能系列白皮书——智能驾驶 2020 [R/OL]. (2020-10-01) [2023-07-01]. <http://www.caai.cn/index.php?s=/home/article/detail/id/2333.html>
- [3] 洪峰. 智能汽车交通车辆的综合认知及其虚拟测试方法研究 [D]. 长春: 吉林大学, 2018.
- [4] 孙剑, 黄润涵, 李霖, 等. 智能汽车环境感知与规划决策一体化仿真测试平台 [J]. 系统仿真学报, 2020, 32 (2): 236-246.
- [5] 国家制造强国建设战略咨询委员会, 中国工程院战略咨询中心. 《中国制造 2025》重点领域技术创新绿皮书——技术路线图 [M]. 北京: 电子工业出版社, 2016.
- [6] 杨健. 基于汽车电气系统开发测试和生产验证的 VU 模型分析 [J]. 传动技术, 2019, 33 (1): 12-16
- [7] 段剑犁. 面向智能驾驶系统的自动化仿真测试评价技术研究 [D]. 重庆: 重庆大学, 2020. du. 2020. 000169.
- [8] 冯亚杰. 基于刚柔耦合仿真的汽车横臂疲劳寿命分析 [D]. 长春: 吉林大学, 2014.
- [9] 刘涛, 周忠贺, 迟霆. 某车自动紧急制动系统可靠性分析 [J]. 汽车实用技术, 2022, 47 (24): 33-39.
- [10] 康诚, 严欣, 唐晓峰. 智能网联汽车自动驾驶仿真测试技术研究综述 [J]. 时代汽车, 2022, 394 (22): 4-6.
- [11] 姚朝华, 张伟. 智能驾驶汽车测试与评价方法分析 [J]. 北京汽车, 2022, 238 (1): 9-11, 36, 46.
- [12] 勾天, 吕豪, 唐诚成, 等. 自动驾驶仿真测试脚本的自动化生成方法、装置、生成设备及存储介质: [P]. 2022-12-02.
- [13] 尤文冰, 马桂香, 唐惟胜, 等. 一种智能驾驶软/硬件在环自动化仿真测试系统 CN114896176B [P]. 2022-11-04.
- [14] 徐舒. 智能驾驶的自动化仿真测试系统及相关设备: CN115061386A [P]. 2022-09-16.
- [15] 马勤政, 徐中伟, 梅萌. 基于 Kubernetes 的列控系统测试容器云平台设计 [J]. 计算机技术与发展, 2021, 31 (6): 52-58.

1) 测试效率大幅提升。本文方法能自动生成脚本代码,并记录转换日志。其转换所需时间仅为 ECU-TEST 软件的 0.1%。

2) 降低了成本投入。在实际工程中,本文方法所需的测试准备时间可缩短至 0.5 天,时间成本缩短 95%,人力及设备成本投入降低 92%。本文方法极大地缩短了项目开发验证周期。

3) 准确率可继续提升。本文方法相比 ECU-TEST 准确率降低了 1%~3%,后续可进一步提升准确率。但与所节约的成本相比,本文的测试效率仍有大幅提升。