南京大學

研 究 生 毕 业 论 文
（申请工程硕士学位）

论 文 题 目 自动驾驶软件的激光雷达模糊测试系统设计与实现

作 者 姓 名 张晓波

学科、专业名称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 陈振宇 教授，冯洋 助理研究员

2021 年 **5** 月 **20** 日

学　　号：**MF1932245**

论文答辩日期：**2021** 年 **05** 月 **20** 日

指 导 教 师：　　（签字）

The Design and Implementation of Lidar Fuzzing System for Autonomous Driving Systems

by

Xiaobo Zhang

Supervised by

Professor **Zhenyu Chen**

Research Associate **Yang Feng**

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
MASTER OF ENGINEERING
in
Software Engineering

Software Institute
Nanjing University

May 20, 2021

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：自动驾驶软件的激光雷达模糊测试系统设计与实现

工程硕士（软件工程领域）专业 2019 级硕士生姓名：张晓波

指导教师（姓名、职称）：陈振宇 教授，冯洋 助理研究员

摘　要

激光雷达是高级别自动驾驶重要的环境感知传感器，能产生在信息表达上比二维数据更有优势的三维点云数据。随着激光雷达传感器在自动驾驶车辆上逐渐普及，激光雷达的安全性和稳定性也受到越来越多的关注。特别是在自动驾驶软件方面，由于激光雷达在目标检测上采用了深度学习技术，输入域空间难以覆盖所有可能场景，在安全性上可能存在漏洞。因此，如何有效对自动驾驶软件的激光雷达模型进行充分测试成为研究的关键点。

模糊测试作为被广泛使用的质量保证与漏洞发现技术，主要通过生成大量非预期输入来对目标程序进行测试。在机器学习系统中，模糊测试能大幅提高测试的效率和覆盖率，因此采用模糊测试来提高自动驾驶软件的激光雷达测试充分性是可行的。然而，点云数据的数据特征跟图片和文字相比差异较大，因而需要针对自动驾驶激光雷达点云数据设计新的模糊测试变异算子。本文通过分析自动驾驶激光雷达点云数据的数量、坐标以及强度等特征，同时结合实际场景信息，设计了一系列针对自动驾驶激光雷达点云数据的模糊测试变异算子。随后，基于 Spring Boot 框架实现了一个针对自动驾驶软件的激光雷达模糊测试系统。通过该模糊测试系统，用户可以上传测试数据集以及待测模型，然后选择变异算子并配置测试任务参数，从而执行测试任务，系统将基于设计的点云特殊变异算子生成扰动测试用例，并对测试模型进行自动化部署及测试，并生成测试报告。另外，该系统还提供点云可视化等功能，具备较强的可用性。

本文基于系统的需求分析及概要设计对系统进行功能测试，以保证系统的可用性。紧接着基于点云目标检测网络 PointPillars 对本系统进行实证研究，进一步验证本文提出的点云变异算子以及自动驾驶软件的激光雷达模糊测试的有效性。实验结果表明，本文的点云特殊变异算子生成的扰动测试用例是真实有效的，且本文的模糊测试能够揭露自动驾驶激光雷达模型存在的问题。本文开

发的模糊测试系统能有效对自动驾驶软件激光雷达进行测试，提高测试的充分性，对于提高自动驾驶软件激光雷达的安全性有一定的积极意义。

关键词： 激光雷达，自动驾驶，模糊测试，点云变异算子

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of Lidar Fuzzing System for Autonomous Driving Systems

SPECIALIZATION: Software Engineering

POSTGRADUATE: Xiaobo Zhang

MENTOR: Professor **Zhenyu Chen** Research Associate **Yang Feng**

Abstract

LiDAR is an important environmental sensing sensor for high-level autonomous driving, which can generate three-dimensional point cloud data that is more advantageous than two-dimensional data in terms of information expression. With the increasing popularity of LiDAR in autonomous vehicles, the safety and stability of LiDAR have also received more and more attention. Especially in terms of autopilot software, because LiDAR uses deep learning technology for target detection, it is difficult for the input domain space to cover all possible scenarios, and there may be loopholes in security. Therefore, how to effectively test the LiDAR model of autonomous driving software has become a key point of research.

Fuzzing is a widely used quality assurance and vulnerability discovery technique, which tests the target program by generating a large number of unexpected inputs. In the machine learning system, fuzzing can greatly improve the efficiency and coverage of the test, so it is feasible to use fuzzing to improve the adequacy of LiDAR test of autopilot software. However, the data characteristics of point cloud data are quite different from those of pictures and text. Therefore, it is necessary to design some new fuzzing mutation operators for the point cloud data of autonomous driving LiDAR. In this thesis, by analyzing the characteristics of the number, coordinates and intensity of the point cloud data of autonomous driving LiDAR, combined with the actual scene information, a series of fuzzing mutation operators for the point cloud data of autonomous driving LiDAR are designed. Subsequently, based on the Spring Boot framework, a LiDAR fuzzing system for autonomous driving software was implemented. Through this system, users can upload test data sets and test model, and then select mutation operators

and configure test task parameters to execute the test tasks. The system will generate disturbance test cases based on the designed point cloud special mutation operator, and automatically deploy and test the test model, and generate test reports. In addition, the system also provides point cloud visualization functions, with strong usability.

This thesis conducts functional tests on the system based on the system's demand analysis and outline design to ensure the usability of the system. Then, based on the Point Cloud Target Detection Network, POINTPILLARS, an empirical research on this system is carried out to further verify the effectiveness of the point cloud mutation operator and the LiDAR fuzzing for autopilot software proposed in this thesis. The experimental results show that the perturbation test cases generated by the point cloud special mutation operator in this paper are real and effective, and the fuzzing test method proposed in this paper can reveal the problems of the autonomous driving lidar model. The fuzzing system developed in this thesis can effectively test the autopilot software LiDAR, improve the adequacy of the test, and have certain positive significance for improving the safety of the autopilot software LiDAR.

keywords: Android Testing, Automated Testing, Mutation Testing

目　　录

图 目 录

表 目 录

第一章　引言

1.1　研究背景及意义

近年来，自动驾驶受到越来越多的关注，自动驾驶汽车在提高生产率和改善交通情况等方面具有巨大潜力。随着自动驾驶汽车的发展，其安全性成为公众最关心的问题，因此确保自动驾驶车辆的安全可靠对于其在市场上取得成功至关重要。测试是自动驾驶软件开发的一个重要方面，是确保自动驾驶软件驱动的车辆安全性的主要方式。由于自动驾驶软件在环境感知上融合多种传感器数据且大量使用深度学习技术，输入域极其庞大，仅通过实景路测与仿真环境测试获取感知数据对自动驾驶软件系统进行测试，很难在资源有限的情况下保障测试的完备性与充分性，因此，有必要对自动驾驶软件的单一种类环境感知模块进行针对性测试。

在自动驾驶车辆上，激光雷达（LiDAR）传感器在所有环境传感器中扮演着越来越重要的角色。一方面，激光雷达传感器能够提供较为直接且精准的距离测量，其距离测试精度要高于毫米波雷达，另一方面，激光雷达传感器产生的三维点云数据可以用于障碍物的聚类和识别，并且算力要求比图像低高，扫描频率高，实时性好 [1]。然而，激光雷达的环境适应性较差，容易受大雾、暴雨或沙暴等不良天气影响。随着激光雷达传感器越来越多被安装到自动驾驶车辆上，激光雷达的安全性和稳定性也受到越来越多的关注，如何对自动驾驶激光雷达进行有效全面的测试成为研究的关键点。

模糊测试（Fuzzing）作为被广泛使用的质量保证与漏洞发现技术，主要通过向目标应用程序生成大量的正常和非正常输入，并尝试将生成的输入提供给目标应用程序，同时监视目标应用程序的执行状态来检测异常 [2]。相比其他测试技术，模糊测试容易部署，具备优秀的可扩展性和适用性，而且可以在缺乏源代码的情况下执行。此外，模糊测试是在目标应用程序实际运行过程中进行测试的，具备较高的准确性。更重要的是，模糊测试并不需要学习大量的目标应用程序知识，因此可以很容易地扩展到大规模应用程序。特别在机器学习系统中，模糊测试能大大提高测试的效率和覆盖率。而在自动驾驶软件系统中，

目标检测、分类和决策等阶段均使用了机器学习卷积神经网络（CNN）算法。例如，在对激光雷达点云数据的目标检测过程中使用了机器学习中的神经网络算法。因此，将模糊测试应用到对自动驾驶软件激光雷达的质量保证中是可取的。

但目前模糊测试在自动驾驶系统激光雷达中的应用还存在诸多限制，首先，传统模糊测试所使用的传统字节级别的变异技术对点云数据这种具有高结构化的输入类型影响较小，无法对巨大的输入域空间进行充分覆盖。另外，点云数据是包含噪声的非均匀稀疏坐标点，具有无序、颜色信息不明显、易受到干扰、信号易衰减等特点，其数据特征跟图片和文字相比差异较大，如何针对这些激光雷达传感器产生的点云数据建立自动化数据生成策略也是一种很大的挑战。

针对以上问题，为了有效提高自动驾驶软件激光雷达测试的充分性，我们拟研究针对自动驾驶软件系统的激光雷达模糊测试技术，并基于此构建一个针对自动驾驶软件的激光雷达模糊测试系统。我们首先研究针对自动驾驶软件激光雷达的模糊测试技术，测试用例采用基于变异的模糊生成，根据相关资料及研究设计针对自动驾驶场景的激光雷达点云数据变异算子。变异算子以用户输入的测试数据集为基础，根据预定义的变异规则来生成扰动数据作为测试用例。生成的测试用例将被模糊测试系统用于对测试模型进行测试评估，最后生成测试评估报告，以此来实现针对自动驾驶软件激光雷达的模糊测试，提高自动驾驶软件激光雷达测试的充分性。

1.2 国内外研究现状

1.2.1 激光雷达点云数据生成

目前针对自动驾驶软件的激光雷达测试相关研究主要集中在激光雷达点云数据生成领域。在激光雷达点云数据生成领域，自动驾驶软件系统中大多数基于 LiDAR 的感知任务，如语义分割和可驾驶区域检测，都需要大量的点云数据来进行训练和验证。获取大量训练和验证点云数据的方式主要有对激光雷达获取的点云数据进行标注以及对带标签点云数据集进行扩增两种。

对于第一种方式，由于手工标注点云数据是十分昂贵的，为了方便手工标注过程，人们在交互式标注方面做了大量的工作。其中，对于室内场景和室外驾驶场景的三维点云，已经提出了许多标注方法 [3]，但许多方法在标注过程中

需要大量使用人力进行标注，而很少使用计算机进行辅助标注，因此效率低下且代价昂贵。为了提高标注效率，Veit 等人 [4] 提出利用增强人机交互的方法来辅助标注以提高标注效率。而在 [5] 中，Wong 等人系统地提出了室内 RGBD 场景的标注建议，并由用户进行交互校正或细化。紧接着，Boyko 等人 [6] 提出了一种用于在三维激光雷达扫描中标记对象的组标注方法，以进一步提供更快的交互式标记率。进一步，Kapoor 等人 [7] 在标注过程中引入了主动学习方法来训练所需交互较少的分类器，但最终仍然需要用户逐个与示例进行交互。

对于第二种方式，在对带标签点云数据的扩增方面，Groueix 等人 [8] 提出的 AtlasNet 可以学习生成三维形状曲面，由可学习的参数化结合组成，将一组二维正方形转换为三维曲面。AtlasNet 中的自动编码器部分是基于 PointNet [9] 的编码器构建而成的。另外，Qi 等人 [10] 提出了一种将自动编码器与 GAN 相结合的新模型，该模型可以生成合理的点云，但需要先训练一个自动编码器，然后同时训练发生器和鉴别器，这需要花费太多的时间。本文提出的模糊测试是基于变异的模糊测试，基于规则建模来设计变异算子，设计的自动驾驶激光雷达点云数据变异算子主要通过对带标签点云数据进行变异来生成扰动数据作为测试用例。

1.2.2 模糊测试

模糊测试最开始是威斯康星大学的 Barton Miller 在 1990 年代提出用来测试 UNIX 系统中应用程序健壮性的一项测试技术。模糊测试的关键是模糊测试用例的生成方法，模糊测试方法的有效性很大程度依赖于测试用例的生成效率。用于生成模糊测试用例的工具被称为模糊器，模糊器主要有基于生成的和基于变异的两大类 [11]。本节从基于生成的模糊测试以及基于变异的模糊测试两方面来介绍模糊测试。

其中基于生成的模糊测试主要思想是：首先测试人员必须对目标测试程序的输入协议结构进行学习和理解，然后根据对协议的理解构造出基本符合协议要求的测试用例。例如，cross_fuzz [12] 和 DOMfuzz [13] 可以生成随机的文档对象模型（DOM）对象。jsfuzz [13] 基于自己的语法模型生成随机但语法正确的 JavaScript 代码。QuickFuzz [14] 利用现有的 Haskell 库来描述生成测试用例时的文件格式。一些网络协议模糊器，如 Frankencerts [15]、TLS-Attacker [16]、tlsfuzzer [17] 和 llfuzzer [18]，是用特定网络协议（如 TLS 和 NFC）的模型设计的。Dewey 等人 [19] 提出了一种方法，通过利用约束逻辑编程来生成语法

正确且语义多样的测试用例。LangFuzz [20] 通过解析作为输入的一组种子来生成代码片段。然后它随机组合片段，并将种子与片段一起突变以生成测试用例。因为它提供了语法，所以它总是产生语法正确的代码。LangFuzz 应用于 JavaScript 和 PHP。BlendFuzz [21] 基于与 LangFuzz 类似的思想，但目标是 XML 和正则表达式解析器。基于生成思想所构造出来的测试用例一般能够对目标测试程序进行更深层的测试。

而相对而言，基于变异的模糊测试更容易部署和实施，与基于生成的模糊测试不同，基于变异的模糊测试在测试开始时需要有一定数量的正常样本数据，模糊器将对这些样本数据内容进行随机变换。基于变异的模糊测试是否有效的关键就在于选取的样本数据是否足够多且分布较广，一般来说，由于并没有深入学习目标测试程序的协议结构，基于变异的模糊测试生成测试用例的有效性要稍微低于基于生成的模糊测试。基于变异的模糊测试算法的关键是学习已存在的数据模型，并基于对已有数据的分析来生成随机数据做为测试用例。变异的方法有很多，例如，位翻转是许多变异模糊器常用的技术 [22]，其中一些模糊器只是简单地翻转固定数量的位，而一些模糊器则确定要随机翻转的位的数量。AFL [22] 和 honggfuzz [23] 则使用算术变异，将选定的字节序列视为整数，并对该值执行简单的算术运算，然后使用计算出的值来替换选定的字节序列。基于块的变异方法 [22] [23] [24] 则将随机生成的块插入种子的随机位置，或者从种子中删除随机选择的块，或者使用随机值替换随机选择的块。一些模糊器则使用一组具有潜在重要语义权重的预定义值，例如 0 或 −1，并格式化字符串以进行变异。例如，AFL 和 honggfuzz 在对整数进行变异时使用 0、-1 和 1 等值。Radamsa [24] 使用 Unicode 字符串。考虑到目前国内外针对激光雷达模型的模糊测试技术较为匮乏，并且本文的模糊测试系统针对的激光雷达点云数据较为复杂，直接生成测试用例的难度较大且可用性难以保证，因此本文提出的模糊测试是基于变异的模糊测试。

1.3 本文主要工作

本文设计并实现了一个针对自动驾驶软件激光雷达的模糊测试系统，主要工作如下：

在方法上，本文将模糊测试的思想引入到自动驾驶软件激光雷达的测试中去，通过模糊测试来评估自动驾驶软件激光雷达目标检测模型的可靠性。首

先，本文通过对已有的模糊测试系统以及自动驾驶软件激光雷达的分析，提出一个针对自动驾驶软件的激光雷达模糊测试框架，来提高自动驾驶软件激光雷达的可靠性，最终提高自动驾驶的安全性。然后，为了确保模糊测试能更全面的覆盖自动驾驶激光雷达点云数据的特性，本文从点云数量、点云坐标和点云强度等方面入手，针对自动驾驶软件激光雷达设计了一组针对性的变异算子。最后，介绍了本文使用的测试评估指标，以对测试结果进行分析计算。

在系统上，本文在理论研究的基础上，实现了一个针对自动驾驶软件的激光雷达模糊测试系统，该测试系统实现了一系列针对自动驾驶场景的激光雷达点云数据变异算子，可以高效地执行模糊测试用例生成，并提供任务配置、报告生成以及点云数据可视化等服务。此外，该系统还提供一个简洁易用的 web 图形用户交互界面。以上这些特征，使得该系统成为自动驾驶软件激光雷达测试人员的理想测试工具。

在测试与实验上，本文首先基于系统的需求分析对系统进行功能测试，以保证系统的可用性。紧接着基于 3D 点云目标检测网络 PointPillars 对本系统进行实证研究，进一步验证本文提出的针对自动驾驶软件的激光雷达模糊测试技术的有效性，并对以下两个问题进行探索：（1）本文提出的变异算子是否有效？（2）本文提出的自动驾驶软件激光雷达模糊测试是否有效？实验结果表明，本文提出的变异算子生成的测试用例与原始数据大部分是语义相同的，是有效的，并且本文提出的模糊测试能够揭露模型存在的问题，提高模型质量。

1.4 本文组织架构

本文共分为六个章节，组织结构如下：

第一章，引言。首先介绍本文研究背景并阐述该研究的重要意义，其次介绍了激光雷达点云数据生成和模糊测试的国内外研究现状，并针对存在的问题说明本文的主要工作并总结本文贡献，最后给出本文的组织结构。

第二章，相关研究工作。首先介绍了自动驾驶系统的基本概念，其次对自动驾驶激光雷达相关研究进行详细叙述，紧接着对模糊测试的基本概念进行介绍，最后详细阐述模糊测试用例生成技术，为后文模糊测试变异算子的设计做铺垫。最后对本文使用到的工具和框架进行介绍。

第三章，自动驾驶软件的激光雷达模糊测试。首先从测试流程的角度入手，介绍了自动驾驶软件的激光雷达模糊测试框架。然后介绍了自动驾驶激光

雷达点云数据的特性，并从点云数据变异算子的角度入手，针对点云数量、点云坐标与点云强度三个方面设计针对自动驾驶激光雷达模糊测试的变异算子。最后详细介绍了本文使用的测试评估指标。

第四章，需求分析与概要设计。首先根据已有知识储备和研究现状对系统进行涉众和用例分析，其次，以此为基础确定功能与非功能需求，再次对系统进行概要设计并给出项目的整体架构，最后进行架构模型设计以及数据库持久化设计。

第五章，详细设计与实现。主要介绍针对自动驾驶软件的激光雷达模糊测试系统的详细设计与实现。根据第三章的需求分析，将本系统划分为测试任务配置、测试用例生成、测试报告生成以及点云可视化四大模块，并对每个模块进行详细设计，给出每个模块的详细设计、核心类设计、流程设计以及关键代码。

第六章，系统测试与实验评估。首先说明系统的测试环境，然后对系统进行功能测试。紧接着给出本文的研究问题，其次介绍实验对象以及评估指标，然后详细介绍了针对实验问题进行的两组实验设置，最后对实验结果进行研究分析并得出结论。

第七章，总结与展望。总结实现系统与完成论文期间所做的工作，并展望项目的后续研究工作。

第二章　相关研究工作

本章首先介绍了自动驾驶系统的基本概念，其次对自动驾驶激光雷达相关研究进行详细叙述，紧接着对模糊测试的基本概念进行介绍，并介绍了模糊测试用例的生成技术，为后文的模糊测试用例变异算子的设计做铺垫。最后介绍本文所实现的针对自动驾驶软件的激光雷达模糊测试系统所使用到的一些工具与框架。

2.1　自动驾驶与激光雷达

2.1.1　自动驾驶系统

自动驾驶汽车想要实现对车辆的智能化控制，需要使用自动驾驶系统（ADS）技术来替代人类来实现对车辆的控制以及对周围环境（如车辆、行人和非机动车等）的监控 [25]。ADS 是一个复杂的系统，现代 ADS 架构主要由传感器层和六个基本模块组成 [26]。

传感器层：传感器层对于 ADS 系统十分重要，其主要作用是预处理传感器输入数据并过滤传感器噪声。ADS 实现了对多种传感器的支持，如惯性测量单元、全球定位系统、超声波雷达、摄像机、毫米波雷达以及激光雷达等，每一种类型的传感器都各有优劣，通过多种传感器的相互补充来实现对周围环境的充分感知从 [27]。

感知模块：感知模块主要用于读取传感器层处理后的数据，并利用计算机视觉和深度学习技术检测交通环境中的静态物体（如道路标线、路牌和障碍物）和动态物体（如汽车、行人和非机动车）。目标检测算法基于来自单个传感器的传感器数据执行诸如分割、分类和聚类等任务，然后使用扩展卡尔曼滤波等融合技术 [28]，对数据进行合并，生成最终的目标轨迹列表。本文的模糊测试主要针对的就是自动驾驶软件系统的激光点云感知模块的算法模型。

定位模块：定位模块负责提供自动驾驶汽车的位置信息。它能够融合不同传感器的多组输入数据，整合所有传感器优点来准确定位自动驾驶车辆，主要

通过从 GPS、IMU 和 LiDAR 传感器收集输入数据来实现的。

预测模块：预测模块主要负责对感知模块检测到的对象的行为进行研究和预测，并生成有关对象的基本信息，例如它们的位置、方向、速度和加速度，然后使用这些数据生成具有这些对象概率的预测轨迹 [27]。

路由模块：路由模块根据自动驾驶车辆的当前位置和目的地生成高级导航信息。模块、通道和道路的输出基于高清地图进行计算。

规划模块：在复动态环境下规划自动驾驶车辆是一件困难的事情，规划模块根据自动驾驶车辆的起点和终点生成导航计划，并使用定位模块和预测模块的输出数据计算自动驾驶车辆的安全（即无碰撞）驾驶轨迹，使用完全确定模型去搜索所有可能的路径并利用代价函数来确定最佳路径。

控制模块：控制模块以计划的轨迹为基础，来生成对车辆的控制指令（例如驱动、制动、转向等）传递到 CAN 总线，CAN 总线将指令传递给自动驾驶车辆的机械系统，最终对车辆进行实际操控。

2.1.2 自动驾驶激光雷达

自动驾驶系统涉及了多种复杂的技术，其中包括高精度地图、实时定位以及目标障碍物检测等，而这些技术的完善都需要激光雷达的参与。激光雷达技术具有很高的探测精度，在过去几十年发展迅速。激光雷达通常被用于测距、探测和识别物体的工作上 [29] [30] [31]，随着近些年自动驾驶的快速发展，LiDAR 高探测精度等特点也使得它成为自动驾驶汽车中的重要传感器。LiDAR 可以对物体周围的一圈进行 360° 的无死角扫描，并高频率从环境中采集高清晰度的点云数据，这使得它成为车辆上感知系统的首选。LiDAR 的原理是首先通过向目标物体发射激光，然后根据接收激光来计算反射时间间隔最终确定物体的实际距离，并且通过结合激光发射的角度，还能推导出物体的具体位置信息。另外，由于 LiDAR 波长低于传统雷达，因而反射率高，精度一般能达到厘米级。自动驾驶车辆使用 LiDAR 以一定的角速度匀速转动，并不间断地发出激光，同时收集反射点的信息，以最终得到全面的环境信息。激光雷达在接收反射激光的同时也会记录该反射点发生的时间和水平角度，由于激光雷达中的每个激光发射器都有固定的编号和垂直角度，因此，根据这些信息就可以计算出所有反射点的具体坐标。激光雷达通过旋转一周，并将收集到的所有反射点坐标作为一个集合就形成了点云（Point Cloud）。鉴于激光雷达技术是高级别自动驾驶和机器人技术感知周围环境的关键，越来越多的人投入到对 LiDAR 的研

究中去。

激光雷达的环境适应性较差，容易受大雾、暴雨或沙暴等不良天气影响，因此，研究环境对 LiDAR 的影响也十分重要。Filgueira 等人 [32] 为了定量的分析降雨对激光雷达产生的影响，使用 Velodyne VLP-16 型激光雷达在特定路段进行测试，测试路段包括带有建筑物信息的金属标牌、混凝土墙、石材立面、窗户、交通标志、反光表面、沥青路面以及垂直反光标志等物体，这些测试物体均在 100 米的范围以内，并且测试是在真实的降雨条件下进行。测试结果表明，雨水强度的变化不仅影响材料表面，还影响激光路径，各目标的回波强度和探测点数均随降雨的增加而减小。随着雨强的增加，探测点的数量急剧减少，它们从 164 个的点（不下雨时）下降到 4 个点（7.2L/m2*h）。激光雷达在不同场景下的回波强度和探测点数随降雨强度的变化曲线图被记录下来。Hadj-Bachir 等人 [33] 建立了激光雷达模型，使用 Pro-SiVIC 软件模拟了激光雷达在晴天、雾、雨等天气条件下的表现，方便有效地代替真实的激光雷达测试活动来测量它们的性能。该模型考虑了物质的相互作用及反射率和激光脉冲的传播以及能量的衰减，模拟了激光雷达在轻雾、中雾、浓雾、小雨、中雨、大雨下的表现，得出了在不同天气条件下激光雷达信噪比随距离的变化曲线。Goodin 等人 [34] 使用雷达功率方程以及消光系数方程建立了一个模型，该模型使用激光雷达传感器的简单参数化来预测降雨引起的反射强度和射程减小。模型除了考虑降低激光雷达回波的强度外，也考虑了雨水的存在也会给测距带来噪声。Goodin 等人将该模型集成到基于物理的汽车自动驾驶模拟器中，并进行了仿真实验，评估了降雨对障碍物检测性能的影响，最终得出激光雷达射程、激光雷达扫描点数、障碍物探测范围随降雨的曲线图。同时他们也定量的模拟测试了降雨对自动驾驶系统的影响。通过研究不良天气对 LiDAR 的影响，有助于我们对激光雷达点云数据变异算子的设计。

2.2 模糊测试

2.2.1 模糊测试概述

模糊测试是一种安全性测试的方法，从概念上来说，模糊测试为目标应用程序生成大量正常和非正常的输入，然后将生成的输入作为测试用例尝试馈送到目标程序并监督其执行状态来判断异常 [2]。模糊测试相比其他测试技术更容易部署和实施，同时具备良好的可扩展性和适用性，能够在缺乏源代码的情况

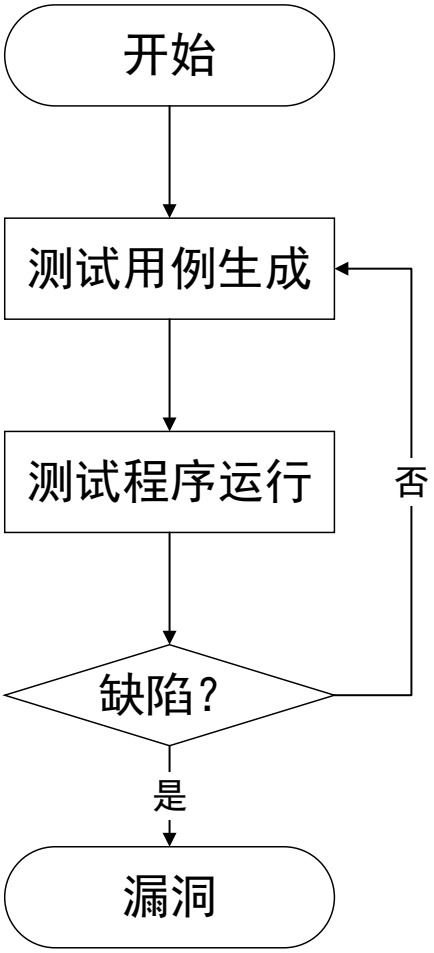

图 2-1: 模糊测试流程图

下依旧有效。此外，模糊测试是在目标应用程序实际运行时执行的，因此具备良好的准确性。而且，模糊测试无需对目标应用程序进行深入了解，并且可以轻松扩展到大型应用程序。

模糊测试的工作流程主要包括四个主要阶段，即测试用例生成阶段，测试用例运行阶段，程序状态监视阶段以及测试结果分析阶段。如图 2-1所示，为模糊测试的流程图。

测试用例生成阶段。测试用例的生成是模糊测试中最重要的一个阶段。模糊测试根据预定义规则生成目标应用程序的大量输入，即测试用例，生成的测试用例质量将直接决定模糊测试的效果。一方面，生成的测试用例应尽可能满足目标程序的输入格式要求，以免因格式错误而被程序拒绝。另一方面，生成的测试用例应该尽可能进行分解，以便在程序出现异常时定位错误原因。根据被测程序的不同，生成的测试用例是多种类的，例如格式不同的文件、二进制

文件、网络通信数据或图片等，如何生成有效的测试用例一直是模糊测试面临的主要挑战。一般而言，模糊测试有两种模糊器，分别为基于生成的模糊器和基于变异的模糊器 [35]。对于基于生成的模糊器，需要目标应用程序详细的相关输入知识，模糊器通常会提供一个预定义文件格式的配置文件，并根据配置文件生成测试用例。由于拥有目标应用程序的相关知识，基于生成的模糊器一般能轻松通过程序验证，并有较大概率测试到程序的深层次代码。但是，如果缺少相关文档资料，基于生成的模糊器生成测试用例将是一项艰巨的工作，此时，基于变异的模糊器更适合。对于基于变异的模糊器，需要先提供大量有效输入，然后通过对这些输入进行变异来进行测试用例的生成。

测试用例运行阶段。用例生成阶段生成的测试用例将被输入到目标应用程序中，紧接着模糊测试程序将自启动并驱动目标应用程序去运行测试用例。在测试用例执行前，测试人员能够配置目标应用程序的启动以及测试完成条件，并预定义测试任务的参数。

程序状态监视阶段。在模糊测试执行期间，一般需要对被测程序进行监督来获取其执行测试用例的结果，同时监督其出现的异常或崩溃。

测试结果分析阶段。一般来说，模糊测试程序需要将测试结果进行分析整理，以便测试人员根据测试结果尝试确定被测程序违规行为的位置和根本原因。

本文的自动驾驶软件激光雷达模糊测试系统架构就是根据该工作流程进行设计的，通过对模糊测试技术的研究，有助于我们更好的设计针对自动驾驶软件激光雷达的模糊测试系统。

2.2.2 模糊测试用例生成技术

对于模糊测试技术来说，测试用例的质量是决定测试效率和有效性的最重要因素。一方面，高质量测试用例可以探索到目标程序更多的执行状态，并在较短的时间内覆盖更多的代码。另一方面，高质量测试用例可以针对目标程序潜在的易受攻击位置，并更快地发现程序错误。因此，如何生成良好的测试用例是一个重要的问题。

在传统程序领域，模糊测试用例的生成技术已经较为成熟。例如，Rawat 等人 [36] 提出一种具有应用程序意识的灰盒模糊器 Vuzzer，它将静态和动态分析相结合。对模糊测试的输入进行变异以生成测试用例的过程主要涉及两个关键问题：在哪里进行变异以及用于变异的值。具体而言，通过静态分析技术

和动态污点分析技术，Vuzzer 可以提取瞬时值和其他影响控制流及其分支的信息。通过具备收集值的变异以及在公认位置的突变，Vuzzer 能够生成更可能满足分支判断条件并通过输入验证的测试用例。但是，Vuzzer 仍然可能无法通过像基于哈希的校验和等类型的验证，并且由于 Vuzzer 的污点分析和主模糊循环等模块均基于 Pin [37]，与 AFL 相比，测试速度相对较慢。Wang 等人 [38] 提出一种数据驱动的种子生成解决方案 Skyfire。Skfire 从爬取的输入中学习概率上下文相关语法（PCSG）知识，并利用学到的知识生成结构合理的测试用例。实验表明，相比 AFL，Skyfire 生成的测试用例能够覆盖更多的代码，并且发现了更多的漏洞。这项工作也进一步证明，测试用例的质量是影响模糊测试效率和有效性的重要因素。随着机器学习技术的发展和广泛使用，一些研究尝试使用机器学习技术来辅助测试用例的生成。Godefroid 等人 [39] 使用基于神经网络的统计机器学习技术自动生成测试用例。首先，利用机器学习技术从大规模有效输入中学习输入格式，然后根据所学知识来指导测试用例的生成。他们在 Edge 浏览器的 PDF 解析器上展示了他们的模糊生成技术的具体过程。虽然该实验最终没有取得满意的结果，但这也提供了一个很好的思路。另外，Rajpal 等人 [40] 也利用神经网络学习之前的模糊测试用例，最终寻找到输入文件中要进行变异的字节。

针对传统应用程序的模糊器，如 AFL，主要通过对测试输入进行字节翻转、块替换、输入文件交叉等方式来生成测试用例。然而这些传统方式生成的测试用例在针对深度神经网络（DNN）的模糊测试中通常是没有意义的。因此，如何在增加变异的可变性和产生有意义的输入之间取得平衡是一个挑战。一方面，如果变异产生的变化很小，生成的测试用例可能原始数据没什么区别，虽然这种测试用例也可能有一定的意义，但是会导致模糊测试出现失败用例的几率就比较低。另一方面，如果变异产生的变化很大，可能出现大量的失败用例，远远脱离实际的失败用例更加没有意义。因此，如何针对深度神经网络生成有效的模糊测试用例是一个主要挑战。在这方面已经取得了一些成果，例如：Pei 等人提出的 DeepXplore [41] 则利用神经元覆盖率来指导测试用例的生成。DeepXplore 在进行测试用例的生成过程中，每次测试的迭代计算扰动时都随机选择 1 个未激活的神经元并提高 DNN 模型在该神经元的激活输出值，同时每次迭代计算扰动时也可以选择最大化多个未激活神经元的激活输出值。为了模拟不同的自动驾驶场景（例如图像在不同照明条件下的效果、摄像头被单矩阵遮挡、多矩阵遮挡摄像头模拟污垢覆盖效果等），DeepXplore 在进行测

试迭代的过程中对原始图像的变异进行了一些领域约束，以避免在变异生成的过程中图像语义变化过大。但是，DeepXplore 生成的场景的真实性是存疑的，并且生成效率可能较低。本文针对的是自动驾驶软件激光雷达的模糊测试，属于针对深度神经网络的模糊测试。虽然现有研究缺乏针对激光雷达的模糊测试用例生成技术，但通过对其他领域的深度神经网络模糊测试用例生成技术的研究，有助于本文针对自动驾驶激光雷达点云数据变异算子的设计。

2.3 工具框架

2.3.1 Spring Boot

Spring 框架 [42] 目前最流行的开源 Java/J2EE 应用开发框架，主要提供构建 Web 应用的全功能 MVC 模块。Spring 是最流行的 Java 框架，其特性支持从简单 Web 到复杂企业应用程序的高效开发。Spring 框架的核心概念包括：控制反转（Inversion of Control，IoC）、依赖注入（Dependency Injection，DI）、面向切面编程（Aspect Oriented Programming，AOP）以及 Java 持久性 API（Java Persistence API，JPA）。IoC 是一个通用的概念，在这个概念中，流的控制是反向的：程序员不再控制程序的流，而是由外部源（框架、服务、其他组件）来控制它。DI 是一种 IoC 形式的模式。AOP 基于改进模块性和代码结构的概念。JPA 负责将对象状态映射到数据库列，以及如何跨对象发出查询。

Spring Boot 是由 Pivotal 团队开发的基于 Spring 的全新框架 [43]，旨在简化 Spring 应用程序开发，帮助开发者快速搭建一个 web 容器，提高程序开发效率。Spring Boot 本质上依旧是基于 Web 开发流程来管理 Java 组件的生命周期，如图 2-2所示为 SpringBoot 框架的架构图，主要包含 Data Access、Web、MVC、AOP、Spring Core 以及 Test 等基本模块。其中，Spring Core 是 Spring Boot 的核心模块，其中的 Core 子模块提供 Spring 核心内容，Bean 子模块则负责 Bean 对象的创建和管理以及依赖注入等功能。

Spring Boot 提供了自动配置服务，将应用程序配置为标准 Spring 应用，进行通用 Spring Bean 对象的自动装配。此外，Spring Boot 还支持命令行解释器，用于启用应用程序控制台。Spring Boot 为所有 Spring 开发提供了一个非常快速和广泛可访问的入门体验，并提供了许多非功能性的特性，这些特性是许多大型项目所共有的（例如嵌入式服务器、安全性、运行状况检查、外部化配置），而且完全没有代码生成，也不需要 XML 配置。基于 Spring Boot 的以上

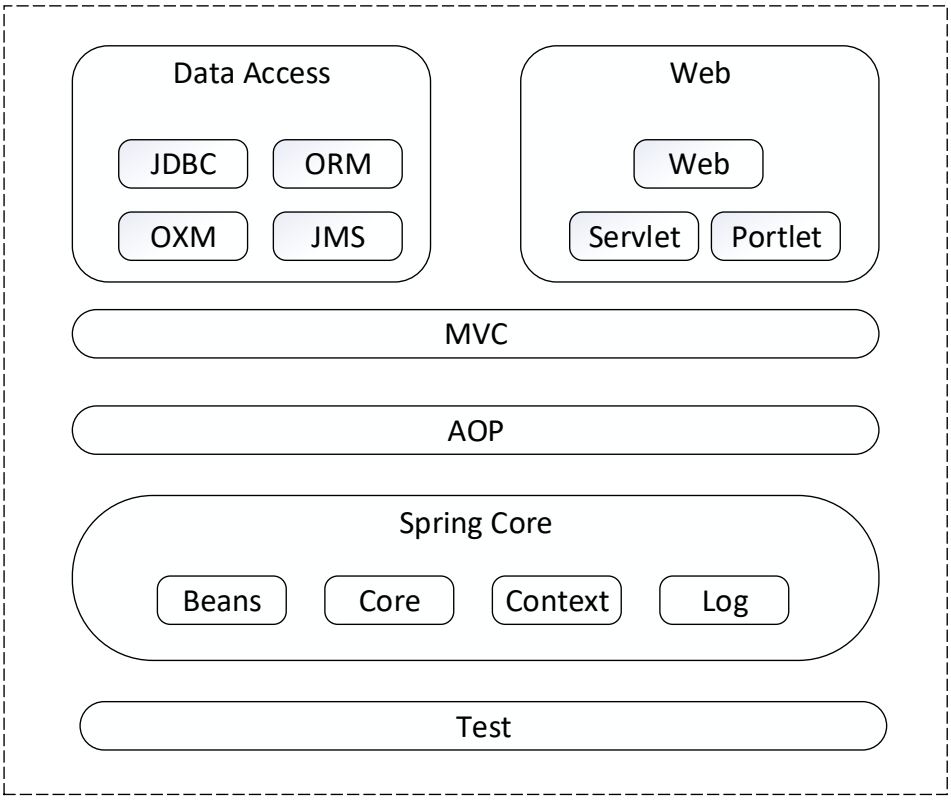

图 2-2: Spring Boot 架构图

特性，本文使用 Spring Boot 实现对自动驾驶软件激光雷达模糊测试系统的快速搭建以及统一管理。利用该框架，可以显著提高本文的模糊测试系统的开发效率和迭代速度，并方便后续的升级维护。

2.3.2 Docker

Docker 是一项开源的容器引擎技术 [44]，它使用 Go 语言进行开发，它运行应用程序，并使流程更易于开发、分发。Docker 中构建的应用程序与所有支持的依赖项一起打包到称为容器的标准表单中。这些容器在操作系统内核上以隔离的方式运行。环境配置是软件开发工作中最为冗余和耗时的部分，如果软件应用需要的各种依赖以及环境变量设置错误，应用通常无法正常运行，这对于不熟悉服务器环境的开发人员来说是一大难题。因此，Docker 被提出用于解决应用开发的环境与部署问题，Docker 提供了将应用程序部署到容器中时实现自动化的工具，开发者可以在 Docker 容器中嵌入所有的运行和开发环境。Docker 的设计方法是提供一个快速、轻量级的环境，在这个环境中，代码可以高效地运行。

Docker 允许以尽可能快的速度测试代码并将其部署到生产环境中。用户只需在目标环境上安装 Docker，然后拉取所需的镜像文件并运行，即可为该文件创建一个虚拟容器，保障其正常运行。无论系统环境是 Window、Linux 或是 Mac，开发人员都可以利用 Docker 高效地部署应用程序。Docker 解决了运行环境和配置问题，为系统的部署和运行提供了一种新的方式，同时也方便持续集成和发布，解决了传统软件开发中繁琐的环境配置耗费大量时间的问题，让开发团队可以专注于业务功能开发。Docker 因高性能、可移植性、可扩展性和快速交付等特性受到广大开发团队喜爱，因此，本系统也采用 Docker 容器实现开发环境的快速搭建以及对运行环境进行一站式打包。

2.3.3 Point Cloud Library

点云库（Point Cloud Library，PCL） [45] 是以前人点云相关研究为基础建立起来的大型跨平台开源 C++ 编程库，PCL 基于 Boost、Eigen、FLANN、VTK、CUDA、OpenNI 和 Qhull 等第三方库构建，是一个完全模板化的、现代的 C++ 点云库，用于 3D 点云处理。

从算法的角度来看，PCL 是指结合多种 3D 处理算法，对点云数据进行操作，包括：滤波（filter）、特征提取（features）、曲面重建（surface）、分割（segmentation）、配准（registration）以及可视化（visualization）等，每一组算法都是通过基类定义的，基类试图集成整个管道中使用的所有公共功能，从而保持实际算法的实现紧凑和干净。基于 PCL 集成的代码和算法，我们可以较为方便的对点云数据进行处理。自动驾驶激光雷达传感器产生的数据为与 2D 图像不同的 3D 点云数据，需要一种高效处理点云的机制。基于以上叙述，本文使用 PCL 来实现对点云的高效处理。

2.4 本章小结

本章对自动驾驶、激光雷达以及模糊测试进行了介绍，并介绍使用到的相关工具框架。介绍自动驾驶和激光雷达相关技术时，首先介绍自动驾驶的基础知识，说明了自动驾驶系统的基本组成和特征，然后介绍了自动驾驶激光雷达的相关研究。紧接着，简要概括了模糊测试的基础概念并展示了模糊测试的一般流程图，然后对模糊测试的一般流程进行简要介绍，最后对模糊测试的用例生成的相关技术研究进行概括。介绍模糊测试用例生成技术的相关研究时，通

过分析传统程序和机器学习程序的相关研究，为后续针对自动驾驶软件的激光雷达点云数据变异算子的设计打下基础。在介绍相关工具框架时，首先介绍了Spring Boot 框架的概念以及架构，然后对 Docker 技术进行介绍，最后介绍了PCL 点云库的相关研究。

第三章　自动驾驶软件的激光雷达模糊测试

本章主要介绍本文针对自动驾驶软件的激光雷达模糊测试技术相关研究工作。首先从模糊测试流程入手，详细介绍自动驾驶软件激光雷达模糊测试的流程。然后介绍了自动驾驶激光雷达点云的主要数据特征，然后针对这些特征设计针对自动驾驶场景的激光雷达点云数据变异算子并给出每个变异算子的详细介绍。最后详细介绍了模糊测试结果的评估指标，用于对测试结果进行统计分析。

3.1　自动驾驶软件激光雷达模糊测试流程

本文研究了传统模糊测试以及自动驾驶激光雷达的数据特征，在此基础上提出自动驾驶软件的激光雷达模糊测试流程。如图 3-1所示，该模糊测试框架的输入为待测模型 (Model Under Test, MUT) 和测试数据集 (Test Data Set, TDS), 最终输出为一份测试报告。下面对测试流程的关键步骤进行详细描述：

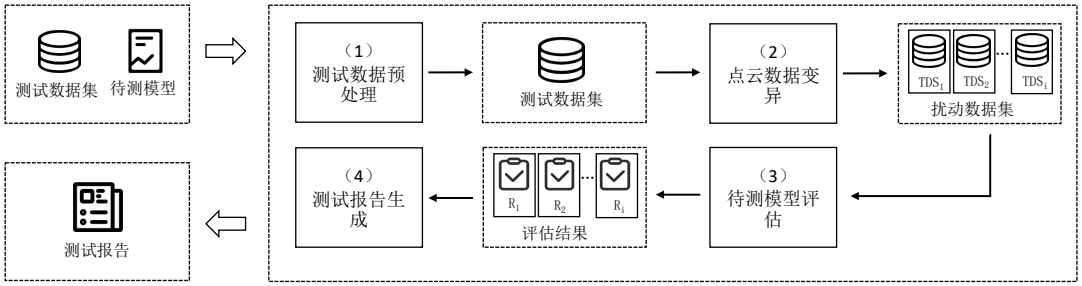

图 3-1: 自动驾驶软件的激光雷达模糊测试流程

(1) 测试数据预处理。首先验证用户上传的 TDS 的格式是否正确，若不是 PCD 格式，则将点云数据文件解析为 PCD 格式，以形成统一格式方便对点云数据进行处理。本系统使用 PCL 点云处理库对点云数据进行处理，使用 PCD

格式存储点云数据方便 PCL 库对点云数据进行统一处理。验证通过后将 TDS 保存在指定路径下等待下一阶段的处理，并将 TDS 的信息持久化到数据库中。

(2) 点云数据变异。根据预先设定的点云变异算子对 TDS 进行变异，一个 TDS 将根据用户选择的不同变异算子生成多个的扰动变异 TDS 作为测试用例。将变异后的扰动 TDS 和输入的 MUT 进行部署，等待下一阶段的处理。

(3) 待测模型评估。当变异后的 TDS 和 MUT 被成功部署后，即可对被测系统执行模糊测试。对于 MUT，系统记录其评估的最终结果。对于 TDS，在使用其对 MUT 进行评估前，需要先运行数据用例创建命令将其转化为最终输入的测试用例。对于给定的 MUT，使用扰动 TDS 进行评估，得到评估的结果并存储起来等待下一阶段使用。

(4) 测试报告生成。在每个扰动 TDS 测试完成后，可以得到在该扰动 TDS 下，MUT 测试评估的结果 AP 值（Average Precision）。通过对 AP 值的分析，并最终综合所有变异算子下的评估结果 AP 值，可以得到本次测试任务的最终结果，并输出测试报告。

3.2 自动驾驶激光雷达点云数据变异算子

本节围绕 3.2.1中总结的自动驾驶激光雷达点云数据特征来设计模糊测试变异算子。为了使变异算子能涵盖更丰富的异常情况以找到意想不到的缺陷，本文分别针对点云数量、坐标和强度三个方面分别设计针对自动驾驶场景的激光雷达点云数据特殊变异算子。

3.2.1 自动驾驶激光雷达点云数据特征

激光雷达点云数据包含丰富的信息特征，如三维坐标信息、回波强度、回波次数、RGB、GPS 时间、扫描角度、扫描方向等 [46]。本文通过阅读相关文档，结合本文模糊测试系统所需领域知识，从数量、坐标、强度、格式四个方面总结了在自动驾驶情景中，激光雷达点云数据的主要信息特征，具体如下：

1. 点云数量。点云的数量由激光雷达的线数以及周围环境决定。线数越多产生的点云数更多，另外，在不同的环境影响下，例如雨雪雾等天气下，激光雷达点云的采样率不同，会导致部分点丢失并产生一些离群噪点。
2. 点云坐标。激光雷达传感器产生的点云数据通常会输出在笛卡尔坐标系下，由于点云数据是三维数据，通常包含 XYZ 坐标，通过 XYZ 坐标可直

接定位到点云中各个点的位置，点云的其他信息以及对点云的加工处理，都是以点云的坐标为基础。

3. 点云强度。点云强度信息通过采集反射激光的强度来获取，是反映反射点的激光雷达反射强度的一种测量指标。反射点不同的表面材质以及粗糙度等都会对于激光雷达的反射程度产生影响，通过强度信息可以把不同的物体区分开来。例如，通过反射激光不同的强度来区分红绿灯、汽车、植被和行人等，另外，由于噪声的强度也有特征，所以点云强度在数据预处理的去噪阶段也非常有用。
4. 存储格式。点云数据格式有多种，包括 pts，LAS，xyz，PCD，txt 和 bin 等，不同格式特点不同，例如 PCD 格式统一且可读性强，而 bin 格式读写效率较高。由于本文使用 PCL 库对点云数据进行处理，且考虑系统的可扩展性对格式统一的要求，所以在对点云数据的处理时，会转化为 PCD 文件格式进行处理。而在进行模型测试时，再根据具体模型的输入要求转化为指定格式。

3.2.2 点云数量变异算子

通过研究分析近年来发生自动驾驶安全事故，自动驾驶激光雷达产生的噪点可能对自动驾驶造成致命错误。本文根据自动驾驶激光雷达点云的数量特性设计了 6 个针对自动驾驶激光雷达点云数据的特殊变异算子。

(1) 均匀噪声添加 (Uniform Noise Addition, UNA)

噪声在激光雷达点云数据中是十分常见的，很多自动驾驶软件系统会在进行目标检测之前对点云数据进行去噪处理，但实时去噪可能并不完全，所以自动驾驶软件系统的激光雷达模块必须要有足够的鲁棒性，在点云数据存在噪声的情况下仍能正常运作。而事实上，对于一些情况下，看起来随机的无关紧要的点云噪声，可能会使自动驾驶软件系统的激光雷达出现错误。Zhou 等人在 [47] 中证明，即使是在感兴趣区域（ROI）外添加的随机噪声，也可能会导致自动驾驶激光雷达目标检测出现错误，使得 ROI 中的某些目标不再被激光雷达目标检测模型检测到。通过在点云数据中添加随机的噪声，能够在保持点云数据语义不变的情况下，获得点云变异数据，提高模糊测试的覆盖度。

UNA 变异方法主要通过在点云数据中添加随机均匀分布的噪声来实现。在具体实现中，首先从点云数据中均匀取一定数量的随机点，由于随机添加的噪声值不能跟原有点云数据偏离太大，因此噪声要以随机选取的点作为基点，通

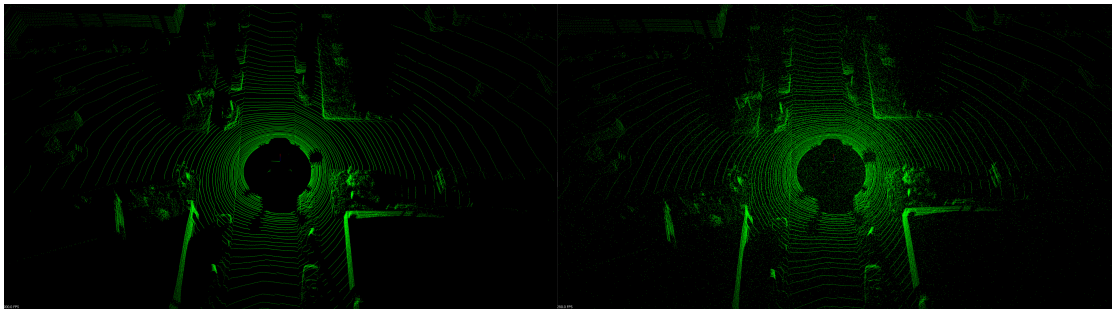

图 3-2: UNA 变异算子效果图

过复制基点的数据再进行小幅度的随机修改来获得噪声，最后将噪声添加到点云数据中去，在完成所有噪声的添加后，即可得到一份添加了随机噪声的自动驾驶激光雷达点云数据。如图 3-2所示，左边为添加噪声前的原始点云数据可视化演示图，右边为添加了噪声后的点云数据可视化演示图。

(2) 均匀点云删除 (Uniform Point Cloud Deletion, UPD)

在实际驾驶场景下，由于激光雷达发出的激光受到周围环境影响，可能出现缺失的情况，例如，在受到强光、下雨或特殊材质的影响下，会导致点云缺失或者点云强度大幅度下降。在下雨的情况下，激光雷达传感器产生的点云数据会出现不同程度的采样缺失和点云强度下降，并且随着雨量的增大，采样缺失和点云强度下降越明显。虽然部分自动驾驶软件系统也会对点云数据进行补全处理，但实时进行点云数据补全不仅消耗算力，也可能补全不完整。因此，自动驾驶软件的激光雷达目标检测模型必须要对部分点云的缺失有一定的容错性，确保随机的点云的缺失不会使自动驾驶软件系统的激光雷达出现致命错误。

UPD 变异方法主要通过在点云数据中随机删除一定数量的点云来实现。如图 3-3所示，左边为原始点云数据可视化演示图，右边为随机删除一定数量点云后的点云数据可视化演示图。

(3) 高斯噪声添加 (Normal Distribution Noise Addition, NNA)

在实际驾驶场景下，激光雷达会受到雪天或雾霾天的影响，使得自动驾驶激光雷达产生的点云数据在汽车周围形成密集的噪声，并随着离车辆越远，噪声越少，使得噪声沿着离汽车距离呈现高斯分布。这种情况下，自动驾驶软件系统的激光雷达模块必须要有足够的鲁棒性，在点云数据存在高斯分布噪声的情况下仍能正常运作。基于以上情形分析，我们设计了高斯分布噪点添加的模糊测试变异算子。

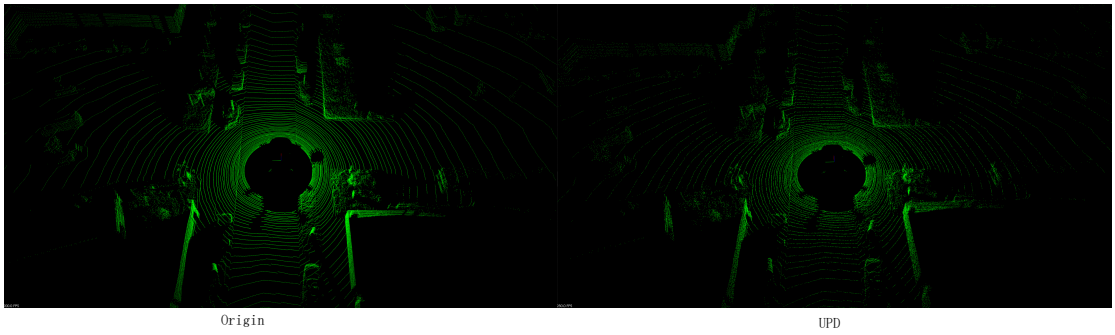

图 3-3: UPD 变异算子效果图

NNA 变异方法主要通过在点云数据中根据离汽车激光雷达传感器的距离呈高斯分布添加噪声来实现。在具体实现中，首先随机设置要增加的噪声数量，然后通过高斯分布随机数生成器分别生成噪声的 xyz 坐标值，然后将该噪声加入到点云数据中，在完成所有噪声的添加后，即可得到一份添加了高斯分布噪声的自动驾驶激光雷达点云数据。如图 3-4所示，左边为添加噪声前的原始点云数据可视化演示图，右边为添加了噪声后的点云数据可视化演示图。

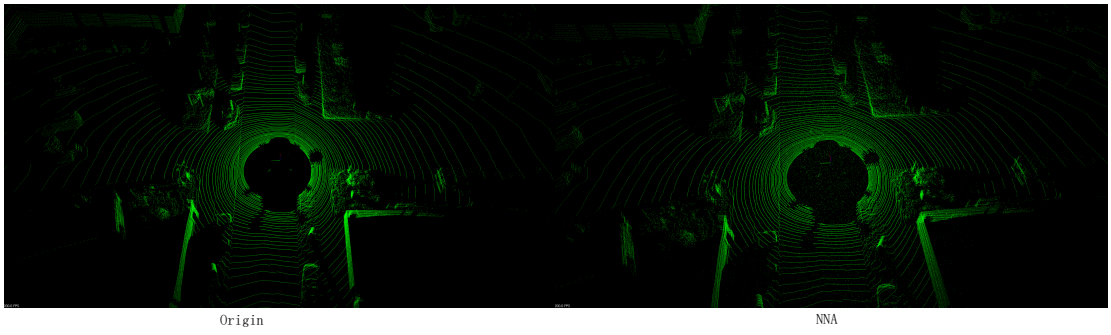

图 3-4: NNA 变异算子效果图

(4) 区域噪声添加 (Random Region Noise Addition, RRNA)

在实际驾驶场景下，一些情况如突然某一区域出现大量粉尘，可能会使自动驾驶激光雷达在特定区域产生大量噪声，为了保证自动驾驶软件系统的激光雷达模块的鲁棒性，自动驾驶软件系统激光雷达模块必须能处理特定区域的噪声，避免其产生致命错误。通过选取随机区域并向其中添加噪声，即可得到变异的点云数据，用来提高模糊测试的覆盖度。

RRNA 变异方法主要通过在点云数据中取随机区域添加随机分布的噪点来实现。在具体实现中，首先获取点云数据各个坐标的最值，然后在最值范围内取随机值划分区域，接下来在划分的区域内添加随机噪声，在完成所有噪声的

添加后，即可得到一份在随机区域添加了随机噪声的自动驾驶激光雷达点云数据。如图 3-5所示，左边为添加噪声前的原始点云数据可视化演示图，右边为添加了噪声后的点云数据可视化演示图。

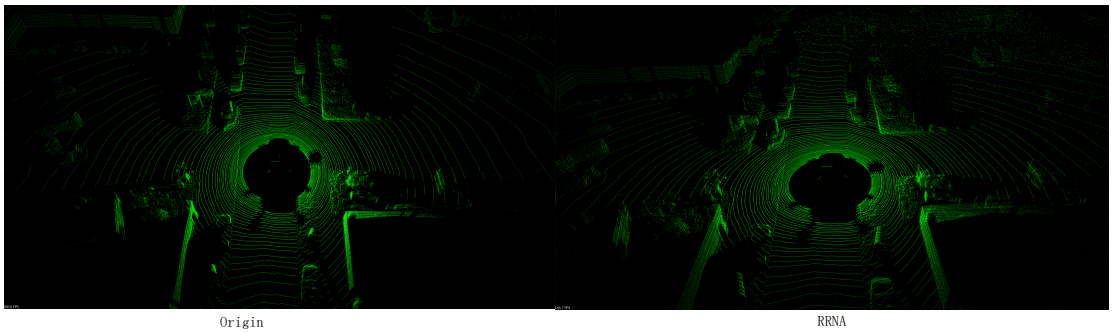

图 3-5: RRNA 变异算子效果图

(5) 区域点云删除 (Random Region Point Cloud Deletion, RRPD)

在实际驾驶场景下，由于激光雷达发出的激光受到周围环境影响，可能出现缺失的情况，例如，在反射面为积水或其他特殊材料时，可能会导致特定区域的点云缺失或者点云强度大幅度下降。因此，自动驾驶软件系统的激光雷达模块必须要对特定区域点云的缺失有一定的容错性，确保部分区域点云的缺失不会使自动驾驶软件系统的激光雷达出现致命错误。通过对随机区域的点云进行删减，得到变异点云数据，提高模糊测试的覆盖度。

RRPD 变异方法主要通过在点云数据中取随机区域删除点云来实现。在具体实现中，首先获取点云数据各个坐标的最值，然后在最值范围内通过随机数生成器取随机值划分区域，接下来在划分的区域内随机删除点云，即可获得变异后的点云数据。如图 3-6所示，左边为原始点云数据可视化演示图，右边为随机删除区域点云后的点云数据可视化演示图。

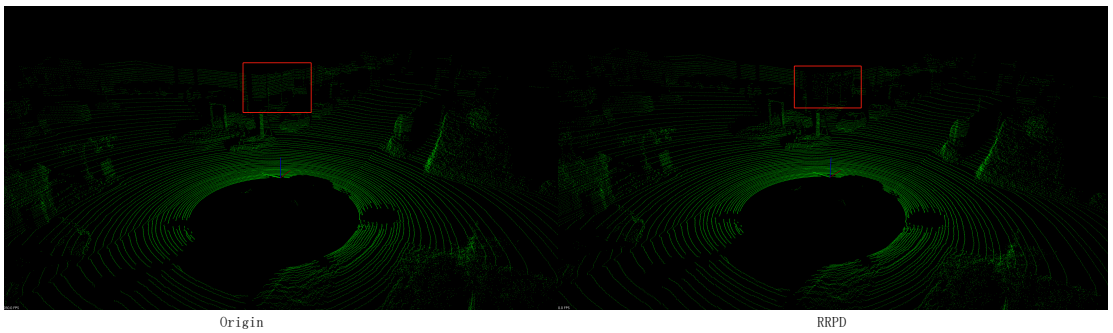

图 3-6: RRPD 变异算子效果图

3.2.3 点云坐标变异算子

(1) 随机坐标轴旋转 (Random Axis Rotation, RAR)

在实际驾驶场景中，车辆的方向变化都会导致激光雷达点云数据的坐标发生变化，点云数据包含丰富的信息特征，如 XYZ 坐标信息，其中，在 PCD 文件中 Z 轴表示三维空间中物体的高低位置。由于汽车行驶过程中，发生方向变化的时候一般是 Z 轴发生变化，因此，本文目前只考虑沿着 Z 轴进行点云坐标旋转。通过将点云数据绕 Z 轴进行随机角度的旋转，得到点云变异数据，提高模糊测试的覆盖度。

RAR 变异方法的实现，首先通过随机数生成器产生旋转度数，旋转的度数范围为 0 到 180 度，然后对点云数据进行旋转，得到旋转后的坐标，最后将原始坐标修改为旋转后的坐标值，同时需要对标签文件进行修改，以保证评估的正确性。在对点云数据进行旋转时，首先需要定义一个 3 阶的单位矩阵，然后将输入的点云数据看作一个 3*N 的二维矩阵，乘上相应的 3 阶单位矩阵后即可得到旋转后的点云数据。如图 3-7所示，左边为原始点云数据可视化演示图，右边为随机旋转 Z 轴后的点云数据可视化演示图。可以看到，旋转后的点云数据除了坐标轴角度的变化外，其余信息与原始坐标轴无异。

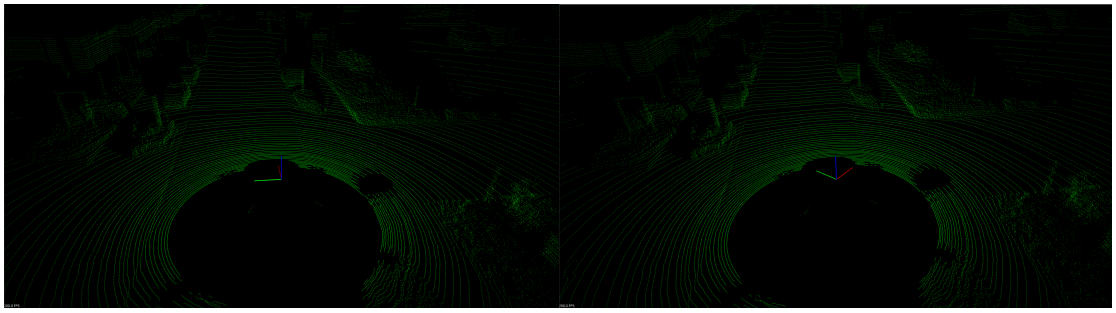

图 3-7: RAR 变异算子效果图

3.2.4 点云强度变异算子

(1) 均匀强度修改 (Uniform Intensity Modification, UIM)

在实际驾驶场景中，激光雷达的回波反射强度会受环境的影响而产生波动。Filgueira 等人 [48] 通过研究降雨对激光雷达测量的影响，发现激光反射强度与降雨强度相关，降雨强度越大，激光反射强度越低。点云强度在一定程度上可以反映目标物体的材质和可信度，不同的点云强度会对自动驾驶软件系统

的激光雷达目标检测结果产生影响，所以自动驾驶软件系统的激光雷达模块必须保证在点云强度发生变化的情况下仍能正常运作。通过对随机点云的强度进行修改，得到点云强度变异数据，提高模糊测试的覆盖度。

UIM 变异方法的实现，首先通过随机数生成器产生要修改的点云数量，然后循环在点云数据中随机选取点，对这些点的强度按照进行随机修改。为了防止修改后的点云强度值与实际值差距过大，新的点云强度值将在原点云强度值的基础上进行随机比例的增强或衰减。

(2) 区域强度修改 (Region Intensity Modification, RIM)

在实际驾驶场景中，突然出现的局部环境变化，如路面洒水车喷出的水汽，可能会使得该区域内的点云强度出现变化。为了测试特定区域内点云强度的整体变化对自动驾驶软件激光雷达模块产生影响，我们设计了点云强度的随机区域修改变异算子来覆盖这种场景，提高模糊测试的覆盖度。

RIM 变异方法主要通过在点云数据中取随机区域修改区域中点云的强度来实现。首先在点云中随机选取一块区域，随机区域的选取参照 RRNA 或 RRPD，然后使用随机数生成器产生点云强度的变化比例，最后循环点云数据中所有的点，判断该点是否在选定的区域中，若是则按照统一的变化比例对该点云的强度进行修改。

3.2.5 点云变异算子总结

表 3-1: 自动驾驶激光雷达点云特殊变异算子

类型	变异算子	描述
点云数量	UNA	均匀噪声添加
	UPD	均匀点云删除
	NNA	高斯噪声添加
	RRNA	区域噪声添加
	RRPD	区域点云删除
点云坐标	RAR	随机坐标轴旋转
点云强度	UIM	均匀强度修改
	RIM	区域强度修改

综上所述，如表 3-1所示为本文提出的 8 个自动驾驶激光雷达点云特殊变

异算子，其中 5 个与激光雷达点云的数量相关，1 个与激光雷达点云的坐标特性相关，还有 2 个与激光雷达点云的强度相关。每个变异算子都对应自动驾驶激光雷达点云数据的特性，并与真实驾驶场景相联系，运用这些变异算子可以在点云数据中模拟真实驾驶场景可能出现的缺陷。

3.3 评估指标

在执行测试评估任务时，需要按照一定的评估指标对评估结果进行计算，以向用户提供全面的测试评估报告。我们将以目标检测领域常用的精度均值（Average precision，*AP*）作为评估指标。

在介绍 *AP* 之前，首先要定义一个测试标准，用来衡量模型的准确度。交并比（Intersection over union，*IoU*）主要用来衡量预测区域和实际区域的重叠程度，一般用来测量目标检测中检测相应物体的准确度。*IoU* 是目标检测模型常用的一个测量标准，只要是输出的结果一个预测边界框（bounding box）的任务都可以用 *IoU* 来进行测量。*IoU* 的计算公式如 (3-1) 所示，其中，*Area of Overlap* 代表预测边界框和实际边界框的交集面积，*Area of Union* 代表预测边界框和实际边界框的并集面积。在目标检测任务中，如果我们模型输出的预测边界框与我们人工标注的实际边界框框的 *IoU* 值大于某个阈值时（通常为 0.5 或 0.7）即认为我们的模型输出了正确的。

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3-1}$$

表 3-2: 预测结果混淆矩阵表

真实情况	预测正例	预测反例
正例	TP（真正例）	FN（假反例）
反例	FP（假正例）	TN（真反例）

在目标检测领域，准确率（*Precision*）主要用来表示模型检测出来的目标中判定正确的概率。而召回率（*Recall*）主要表示数据集所有正样本中被模型检测出来的比例。如表 3-2所示为目标检测预测结果的混淆矩阵，表格展示了模型预测结果和真实结果的所有可能组合。通过表 3-2的结果组合定义，我们

可以定义 *Precision* 和 *Recall* 的计算公式。*Precision* 计算公式如 (3-2) 所示，*Recall* 的计算公式如 (3-3) 所示。但是，准确率和召回率是一队矛盾的度量。一般来说，准确率高时召回率偏低，而召回率高时则准确率偏低。因此，为了直观展示模型的准确率和召回率，一般以准确率为纵轴、召回率为横轴作图，得到 *Precision-Recall* 曲线，简称“PR 曲线”。

$$Precision = \frac{TP}{TP + FP} \tag{3-2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3-3}$$

PR 曲线虽然能直观展示模型的准确率和召回率，但如果两个模型的性能较为接近时，通过 PR 曲线不太容易比较两个模型的优劣，因此，还需要设计一些性能度量来对模型性能进行准确计算。精度均值（*AP*）表示目标检测模型检测结果的平均精准率，常用来度量目标检测模型的性能。具体来说，AP 的计算方式就是对检测结果的 PR 曲线求积分。精度均值的计算公式如 (3-4) 所示，其中 $p(r)$ 表示评估结果的 PR 函数。精度均值相比于 PR 曲线，能够更加准确直观的反映目标检测模型的质量。另外，平均精度均值（*mAP*）则是对所有的精度均值取平均值，从而直观反映模型的整体质量。平均精度均值的计算公式如 (3-5) 所示，其中 N 表示目标类型总数，AP_i 表示类型 i 的检测精度均值。

$$AP = \int_0^1 p(r)dr \tag{3-4}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{3-5}$$

3.4 本章小结

本章主要介绍了自动驾驶软件激光雷达模糊测试技术的研究工作。首先为自动驾驶软件系统激光雷达设计了完整的模糊测试流程，为后文的需求分析和

概要设计打下基础。然后从点云数量、坐标、强度和存储格式四个方面介绍自动驾驶激光雷达点云数据的主要特征，紧接着根据这些特征从点云数量变异算子、点云坐标变异算子和点云强度变异算子三种类型共设计了 8 个针对自动驾驶激光雷达点云数据的变异算子，用于生成点云扰动测试用例。最后，详细介绍了本文使用的测试评估指标。

第四章　需求分析与概要设计

本章介绍针对自动驾驶软件的激光雷达模糊测试系统的需求分析与概要设计。首先从用例分析、功能性需求和非功能性需求对系统进行需求分析，随后通过系统架构、4+1 视图以及持久化设计三方面进行系统概要设计。

4.1　需求分析

4.1.1　用例分析

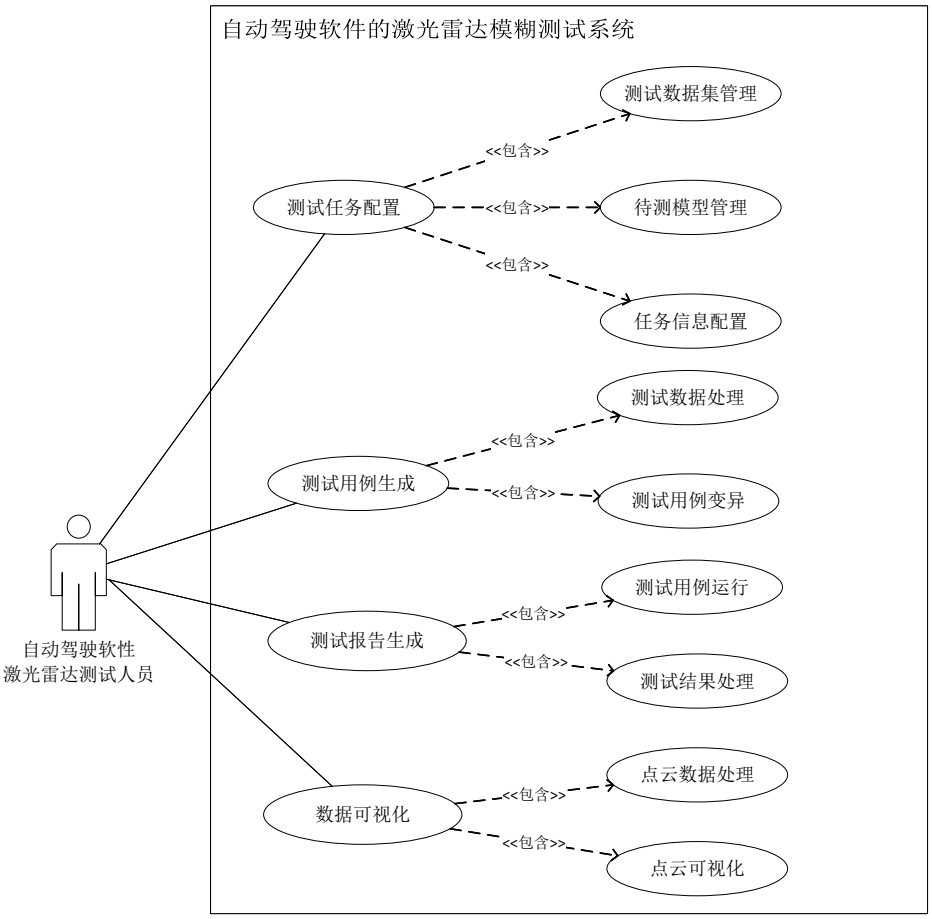

图 4-1: 系统用例图

如图 4-1所示为自动驾驶软件的激光雷达模糊测试系统用例图。本系统作为一个针对自动驾驶软件的激光雷达模糊测试工具，用户一般为自动驾驶软件激光雷达模型测试人员，根据需求，我们首先将系统分为测试任务配置、测试用例生成、测试报告生成和数据可视化四个主要用例。

表 4-1: 测试任务配置用例描述

ID	UC1
名称	测试任务配置
参与者	自动驾驶软件系统激光雷达测试人员
触发条件	自动驾驶软件系统激光雷达测试人员点击新建测试任务
前置条件	自动驾驶软件系统激光雷达开发人员已完成项目开发
后置条件	无
正常流程	1. 用户点击新建测试任务按钮 2. 用户上传或选择测试数据集 3. 用户选择要使用的变异算子 4. 用户上传或选择待测模型 5. 用户填写任务其余配置 6. 用户点击任务执行按钮
扩展流程	2a. 上传新的测试数据集 1. 用户点击上传测试数据集按钮 2. 上传点云数据文件和标签文件，填写数据集信息 3. 用户点击上传按钮 2b. 测试数据集校验失败 1. 系统提示测试数据集校验失败 4a. 上传新的测试模型 1. 用户点击上传测试模型按钮 2. 上传模型文件，选择模型类型 3. 用户点击上传按钮 4b. 测试模型校验失败 1. 系统提示测试模型校验失败

在测试任务配置用例中，用户首先新建任务，然后选择上传的新的测试数据集或者从先前上传的测试数据集中选择，紧接着，用户可以选择要使用的变

异算子以及待测模型，待测模型同样可以选择用户先去上传的待测模型或者上传新模型，最后填写其余相关配置，确认无误后就可执行任务，随后系统对的测试数据集、待测模型以及其他配置参数进行验证，确认无误后开始执行测试任务。在测试用例生成阶段，系统首先将测试数据集转换为指定格式，并对测试数据集进行备份，随后使用用户选择的变异算子对测试数据集进行变异生成，生成的点云数据为最终执行测试的测试用例。在测试报告生成阶段，系统将自动使用变异生成的测试用例对待测模型进行测试评估，并将得到的评估结果进行分析汇总后输出一份测试报告。对于数据可视化用例，用户首先上传要进行可视化的激光雷达点云数据，后端将对用户上传的点云数据进行格式检查，检查无误后返回。用户从上传的点云数据中选择要可视化的点云，执行可视化，即可对点云进行可视化处理。下面对四个用例进行详细的用例描述：

表 4-2: 测试用例生成用例描述

ID	UC2
名称	测试用例生成
参与者	自动驾驶软件系统激光雷达测试人员
触发条件	自动驾驶软件系统激光雷达测试人员点击执行测试任务
前置条件	自动驾驶软件系统激光雷达测试人员已完成测试任务配置
后置条件	无
正常流程	1. 用户点击执行测试任务按钮 2. 跳转至任务详情页面 3. 系统进行测试数据集预处理 4. 系统根据用户选择的变异算子对测试数据集进行变异
扩展流程	无

表 4-1给出了测试任务配置的用例描述。测试任务配置用例主要包括测试数据集管理、待测模型管理以及任务信息配置三个子用例。测试数据集管理用于让用户上传其要用来测试待测模型的测试数据集，系统将验证用户上传的测试数据集格式是否正确。对测试数据集进行校验的原因是本系统的点云处理库对点云数据文件的格式有一定要求，需要将点云数据转化为 PCD 格式进行处理，并且点云数据文件和标签文件也需要一一对应，用户需按照要求准备测试数据集。待测模型管理用于让用户上传其要测试的模型，系统将校验模型的类型是否正确。在本用例中，用户可以选择测试数据集、待测模型和要使用的

变异算子，然后提交测试任务，系统将根据用户填写的信息执行测试任务。另外，用户也可以上传新的测试数据集和测试模型。

表 4-2给出了测试用例生成的用例描述。测试用例生成主要包括测试数据处理和点云数据变异两个子用例。测试数据处理用于对用户上传的测试数据集进行格式转换，将点云数据格式转换为 PCD 格式，并对测试数据集进行备份。测试用例生成基于输入的测试数据集并使用用户选择的变异算子来生成点云数据变异体，生成的变异体即为最终执行模型测试的测试用例。

表 4-3: 测试报告生成用例描述

ID	UC3
名称	测试报告生成
参与者	自动驾驶软件系统激光雷达测试人员
触发条件	自动驾驶软件系统激光雷达测试人员点击执行测试任务
前置条件	系统完成测试用例生成步骤
后置条件	无
正常流程	1. 系统完成测试用例生成 2. 系统启动执行框架 3. 系统执行测试任务 4. 系统运行完毕，展示简要测试结果 5. 用户点击任务详情，查看详细的测试报告
扩展流程	无

表 4-3给出了测试报告生成的用例描述。测试报告生成用例主要包括测试用例运行和测试结果处理两个子用例。测试用例运行子用例中，在系统完成测试用例生成后，系统检测执行框架是否正常，然后启动执行框架并开始对测试模型进行测试。模糊测试系统将使用生成的测试用例对模型进行测试评估，评估完成获取测试结果。对于测试结果处理子用例，系统将对测试结果进行分析处理，每个变异生成的测试数据集对模型的评估结果都将被分析并保存在数据库中，等所有测试结果汇总完成后形成测试报告并输出。

表 4-4给出了数据可视化的用例描述。数据可视化用例主要包括点云数据处理和点云可视化两个子用例。对于点云数据处理子用例，在用户上传或选择已上传的要进行可视化的点云数据集后，后端将对点云数据进行校验，检测数据集是否存在且格式是否正确，然后返回数据集的点云数据文件列表。对于点

表 4-4: 数据可视化用例描述

ID	UC4
名称	数据可视化
参与者	自动驾驶软件系统激光雷达测试人员
触发条件	自动驾驶软件系统激光雷达测试人员点击进入可视化界面
前置条件	无
后置条件	无
正常流程	1. 用户点击进入可视化界面 2. 系统展示可视化界面 3. 用户上传或选择点云数据集 4. 系统数据校验后返回数据集文件列表 5. 用户选择点云数据 6. 用户点击可视化按钮 7. 系统展示点云演示图
扩展流程	无

云可视化子用例，用户在上传的点云数据集中选择要可视化的点云文件，然后提交请求到后端，后端读取用户请求的点云文件并对其进行可视化处理。

4.1.2 功能性需求

根据用例分析，本系统分为四大模块，分别为测试任务配置模块、测试用例生成模块、测试报告生成模块以及点云可视化模块。

测试任务配置模块。用户可以在数据集列表界面进行测试数据集管理，用户按照要求的格式准备并上传点云文件和标签文件。同时，用户可以在模型列表界面上传并管理测试模型。在新建任务界面中，系统将展示已提交的任务信息列表，用户可以新建测试任务，选择要使用的变异算子、测试数据集以及测试模型。所有配置填写完成后，用户可以点击任务执行按钮开始执行测试任务。

测试用例生成模块。在用户提交测试任务后，系统将按照用户的配置运行测试任务。模糊测试系统首先对测试数据集进行预处理，将点云数据格式转化为统一格式。然后进行测试用例生成，系统根据用户配置的激光雷达点云数据

变异算子，对测试数据集中的点云数据进行变异，得到的变异体作为模糊测试的测试用例。

测试报告生成模块。在完成测试用例的生成后，系统将开始执行模型评估操作。模糊测试系统首先将使用生成的测试用例对用户选择的自动驾驶激光雷达模型行评估。评估完成得到当前模糊测试变异算子生成的测试用例集的评估结果。系统将输出的评估结果分析处理并保存下来，等所有测试用例集均评估完成后，生成完整的测试报告。

点云可视化模块。用户在可视化界面中选择要进行可视化的点云数据集，若要上传新的数据集则点击跳转进行数据集上传。后端接收到用户选择的点云数据集后对点云数据集进行校验，若数据集存在且格式正确，则返回数据集点云文件列表，紧接着，用户从上传的点云数据中选择一个要进行可视化的点云数据提交后端进行可视化处理，后端在接收到用户请求后将读取指定的点云数据并进行可视化处理。

4.1.3 非功能性需求

本系统的目的是帮助自动驾驶软件激光雷达测试人员对自动驾驶软件系统的激光雷达模型进行模糊测试，以评估自动驾驶软件系统激光雷达模型的充分性，并找出可能存在的缺陷。为了便于使用，本系统需要满足以下非功能性需求。

性能。系统前端各个界面的响应时间不应高于 1 秒，系统执行单个测试用例生成的时间不应超过 30 秒，单个测试用例执行测试评估的时间不应超过 30 秒。

可靠性。由于模糊测试执行周期较长，因此系统需要有较高的可靠性，并且需要在用户输入异常数据或系统运行出现故障时有相应的提示信息或者操作，另外，在出现异常中断时，需要有对应的中断恢复机制。

易用性。由于本系统整体流程专业性较强，需要进行基本的任务配置，用户可能未接触过自动驾驶激光雷达模糊测试，本系统需要对测试数据集的具体格式进行详尽的说明以避免出现错误，还需要对模糊测试的变异算子和使用方法提供详细的介绍说明以降低使用难度。同时，由于模糊测试耗时可能长达数十分钟，为了提升用户体验，系统应在模糊测试任务进行时给出当前正在运行的变异算子信息以及整体运行进度。

可扩展性。由于自动驾驶软件系统的激光雷达模型在不断发展变化，并且

对模糊测试变异算子也可能有新增或者修改，因此，应该尽可能有较好的可扩展性。

4.2 概要设计

本节以上述需求分析总结出的功能性需求和非功能性需求为基础，对系统进行总体概要设计。首先介绍系统的整体架构，然后从 4+1 视图中的逻辑视图、进程视图以及开发视图三个角度详细描述系统，最后介绍系统的持久化设计。

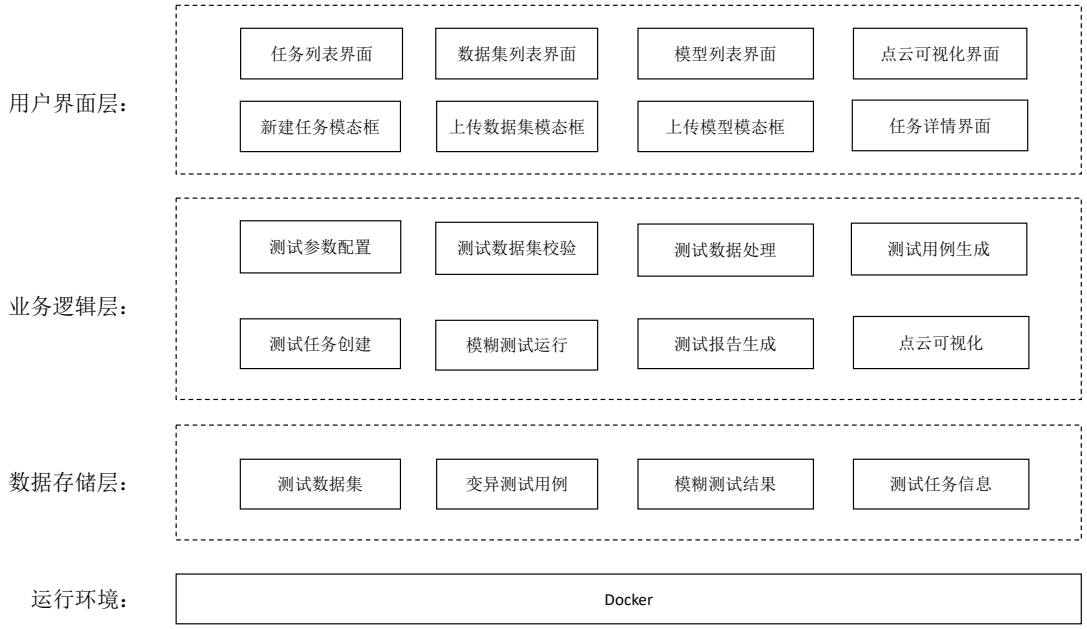

图 4-2: 系统架构图

4.2.1 系统架构设计

系统整体架构如图 4-2所示。基于对上述系统需求及可行的实现技术的分析研究，考虑将系统整体分为三层：用户界面层、业务逻辑层和数据存储层。本系统中，用户界面层主要负责测试任务的展示以及与用户的交互，主要包括任务列表界面、数据集列表界面、模型列表界面、点云可视化界面、任务详细界面、新建任务模态框、上传数据集模态框以及上传模型模态框。前端使用 Html5+JavaScript 实现静态 UI 界面，与后端接口的交互数据格式采用较为通用

的 JSON 数据格式。该层主要通过客户端界面的 UI 控件等实现与用户的交互，并采用 JavaScript 编写的事件处理函数来接受响应以及传送请求。业务逻辑层中，与前端的交互逻辑以及系统整体控制逻辑使用 Spring Boot 框架实现，选用 Spring Boot 框架的原因是它的可扩展性强且支持众多服务组件，而具体的测试用例生成等服务则采用 c++ 程序去完成。本系统中业务服务可分为测试参数配置、测试数据集校验、测试数据处理、测试用例生成、测试任务创建、模糊测试运行、测试报告生成以及点云可视化。所有的业务服务均采用传统的分层设计，可以较大程度减少业务耦合且具有良好的可拓展性。数据存储层使用 mySQL 数据库存储持久化数据，使用 JDBC 连接数据库，使用 Spring Data JPA 进行实体类映射。其中生成的测试用例保存在磁盘，其他的测试信息等数据存入数据库。为了实现实现系统的平台无关性，并实现快速部署，将利用 Docker 容器对系统进行打包，使其成为一个可跨平台运行的 Docker 镜像。每个 Docker 容器都相当于一个能够满足系统运行环境的服务器。

4.2.2 架构模型设计

本节采用 Philippe Kruchten 所提出的“4+1”视图 [49] 来展示系统的基本架构设计，对本系统的概要设计进行进一步描述。本节分别从“4+1 视图”中的逻辑视图、进程视图和开发视图三个方面对系统的架构进行描述。

逻辑视图设计。如图 4-3所示为系统的逻辑视图，图中展示了自动驾驶软件的激光雷达模糊测试系统的基本逻辑。逻辑视图从系统的功能性需求上描述系统，通过将系统之间的调用逻辑抽象成服务来对系统进行描述。逻辑视图能表达系统的核心逻辑，通常以面向对象的方法为基础，将系统抽象为对象或类进行描述。本系统的主要实体可以抽象为测试配置（Test Config）、测试数据（Test Data）、被测模型（Model Under Test）测试报告（Test Report）和点云数据（PointCloud Data）等，系统中的主要服务可以抽象为用户服务（UserServer）、测试任务服务（TestTaskServer）、测试配置服务（TestConfigServer）、数据生成服务（DataGenServer）、测试运行服务（TestRUNServer）、报告服务（ReportServer）、数据处理服务（DataProcessServer）和点云数据可视化服务（PCDVisualServer）。其中，用户服务主要提供用户管理功能，测试任务服务提供变异算子和待测系统选功能，测试配置服务提供配置信息处理功能，数据生成服务提供测试用例的生成功能，测试运行服务提供对被测模型执行所有测试用例的功能，报告服务提供测试结果的处理以及测试报

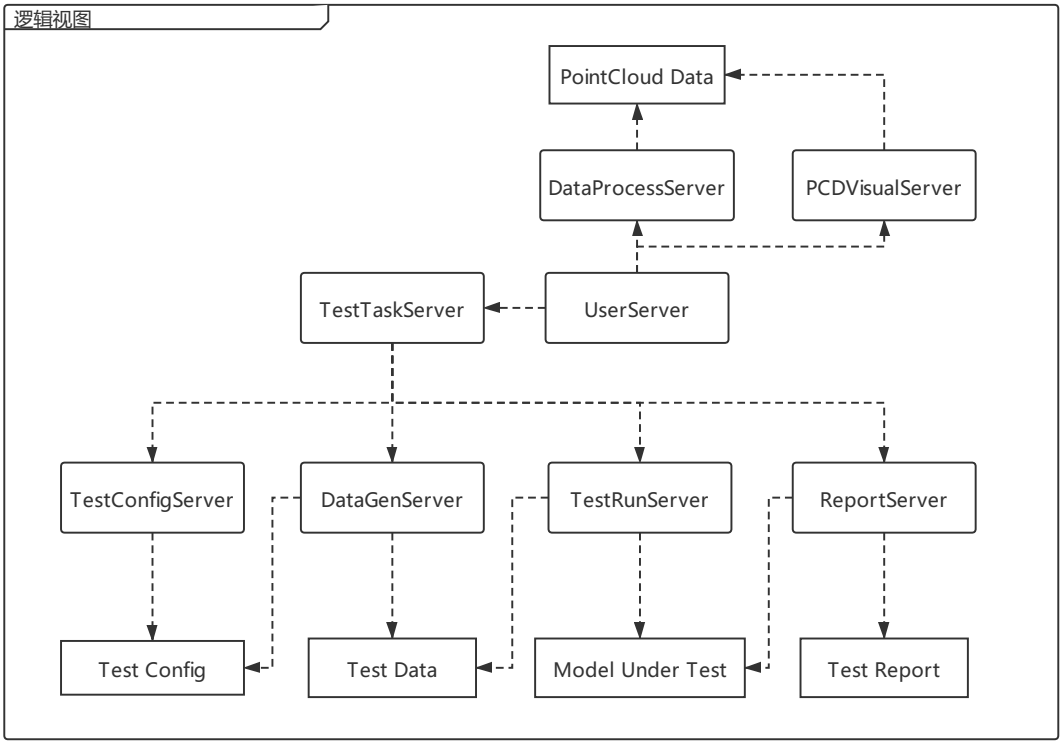

图 4-3: 系统逻辑视图

告生成功能，数据处理服务提供对点云数据的处理功能，点云数据可视化服务提供对点云数据的可视化功能。

进程视图设计。如图 4-4所示为系统的进程视图，进程视图描述了系统的进程、线程、对象等运行时概念以及他们相关的并发、同步、通信问题。本系统中，用户首先上传测试数据集到主程序，主程序调用数据处理进程对测试数据集进行处理，然后返回处理结果。同时，用户可以上传测试模型，主程序将调用数据处理进行对模型进行校验及处理。当用户配置好模糊测试任务的配置信息后，主程序将测试任务参数（如测试数据集、变异算子和待测模型等）传递给用例生成进程。用例生成进程根据用户选择的测试数据集以及变异算子等配置信息进行模糊测试用例的生成，每完成一个变异算子的用例生成任务，就将结果返回给主进程，主进程判断用例生成的结果。如果用例生成成功，主进程将测试任务参数传递给测试评估进程，由测试评估进程启动评估程序来使用生成的测试用例集对被测模型进行评估，得到被测模型的评估结果发送给报告生成进程。报告生成进程将收到的测试结果处理后保存起来，直到所有生成的测试用例集测试完毕，报告生成进程将所有测试结果汇总处理形成测试报告输出。

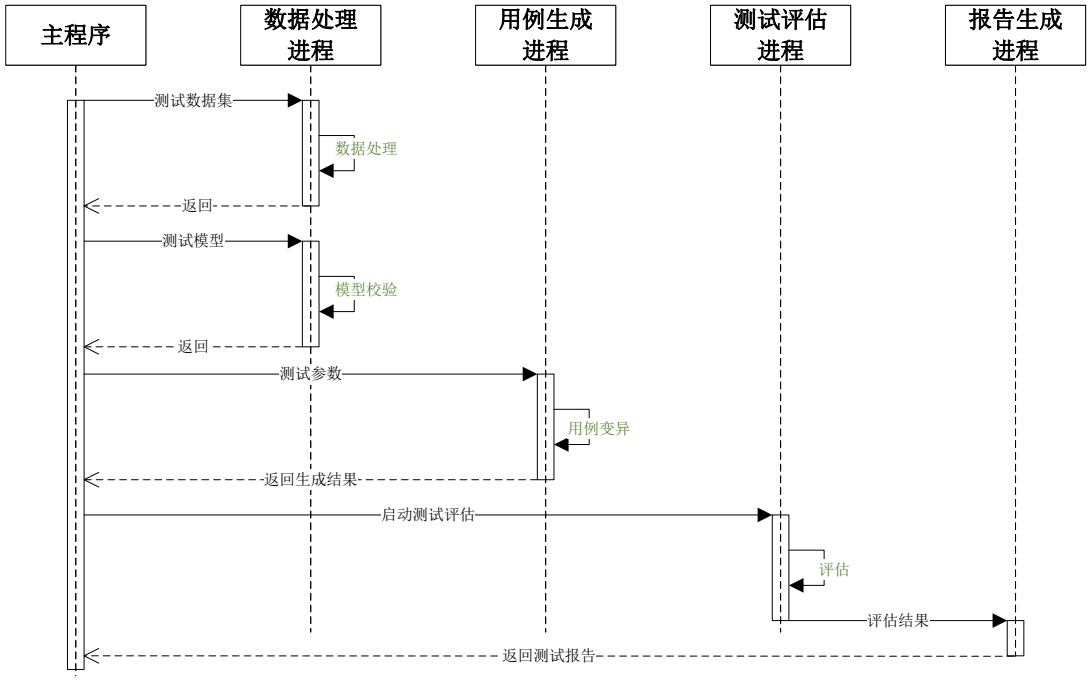

图 4-4: 系统进程视图

开发视图设计。如图 4-5所示为系统的开发视图，开发视图是从系统的模块构成和开发过程的角度来描述系统，并展示了软件包的层次结构。在用户界面层中，Static 包存放的是用户交互页面的静态资源，包括 css、js 以及图片等资源，Html 包存放的是与用户交互的网页的 Html 资源文件。业务逻辑层主要基于 Springboot 框架实现，主要包含 Controller、Service、FuzzingTest、DataGenerate、Util、DAO 以及 VO 七个包。Controller 包属于控制层，主要是处理前端传来的外部请求，并调用相应的服务进行处理。Service 包属于服务层，主要包含 DataGenService（数据生成服务）、TestTaskService（测试任务服务）、TestReportService（测试报告服务）以及 PointCloudVisualService（点云可视化服务）。FuzzingTest 包负责被测系统的启动以及测试结果的处理，其中，TestTaskStarter（测试任务启动器）用于根据配置开启测试评估阶段，TestResult Process（测试结果处理）主要负责将评估阶段获得的评估结果进行处理，保存汇总后返回 TestReportService（测试报告服务）进行测试报告的生成。DataGenerate 包负责模糊测试用例的生成，主要包含 DataGenerator（数据生成器）以及 PCDMutator（点云数据变异器）。Utils 包主要包含系统中公共的工具文件，包括 PCDConvertor（点云文件转换器）以及 TestTaskProcess（测试任务处理）。DAO 包用于保存与数据库的交互逻辑，VO 包则主要包含与

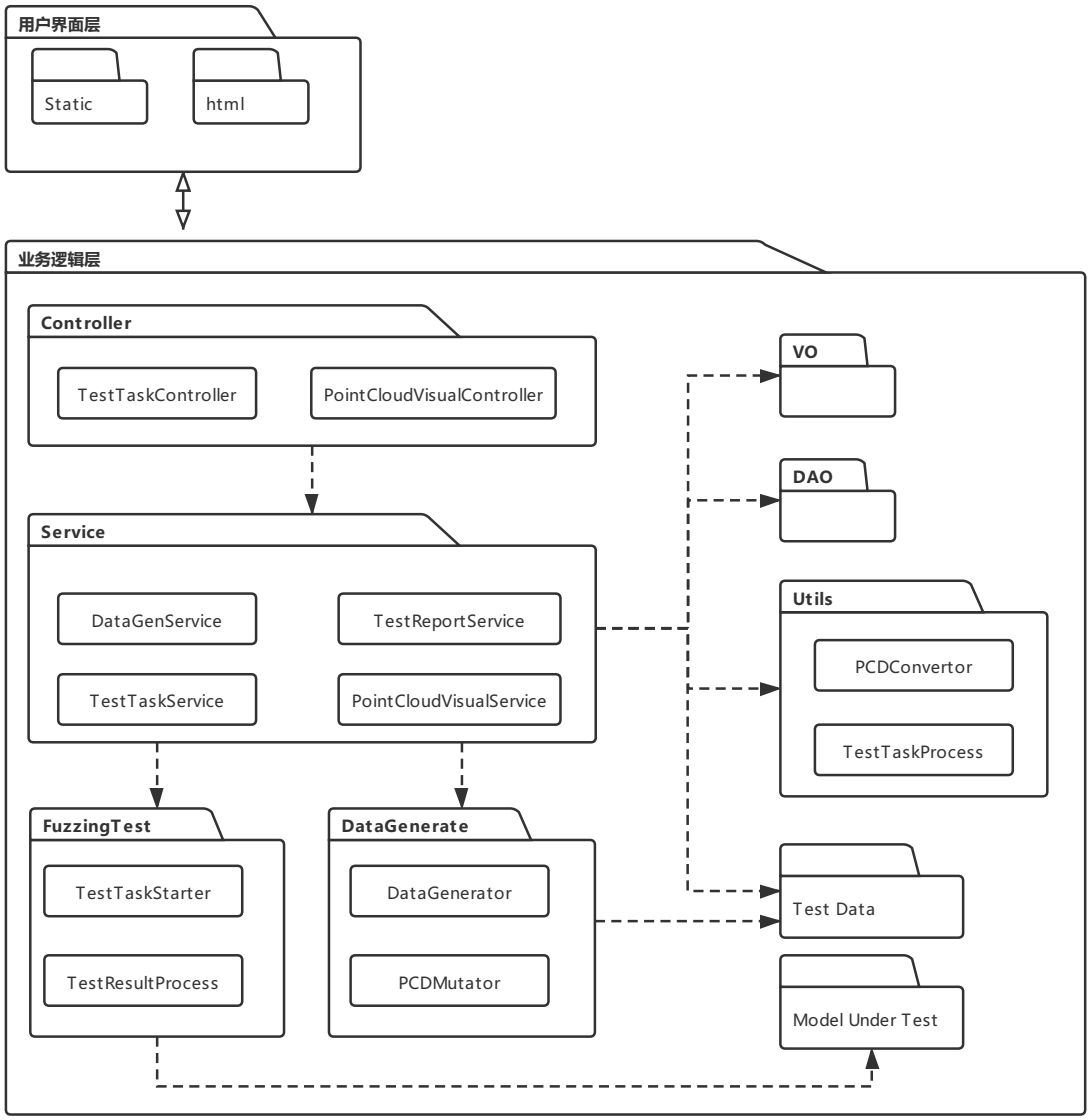

图 4-5: 系统开发视图

前端传递的数据对象。Test Data 以及 Model Under Test 是系统中重要的两个对象，分别对应模糊测试数据对象以及被测的激光雷达目标检测模型。

4.2.3 持久化设计

本文提出的自动驾驶软件的激光雷达模糊测试系统使用 MySQL 数据库来持久化对象，按照上述的需求分析，定义如下持久化对象：

如表 4-5所示为系统的 BigTestTask 数据库表字段，每一条记录对应一个总任务。该数据库表记录在用户新建测试任务时写入到数据库中，表中不仅详细

表 4-5: BigTestTask 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增 ID 字段
mutator	VARCHAR(200)	选择的变异算子
dataSetID	INT(11)	测试数据集 ID
modelID	INT(11)	被测模型 ID
beginTimes	Datetime	测试任务开始时间
finishTimes	Datetime	测试任务结束时间
process	INT(11)	任务执行进度
status	INT(11)	任务状态
testResult	TEXT	运行结果

记录了该次任务的主要配置信息，如选择的变异算子、测试数据集 ID 和被测模型 ID 等，也记录了任务的执行情况，如任务的开始时间、结束时间、任务执行进度、状态以及任务执行结果等。通过 BigTestTask 数据库表，用户可以方便的查看每个任务的执行情况，并在任务中断时为系统提供必要的任务恢复信息。

表 4-6: TestTask 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增 ID 字段
bigTaskID	INT(11)	所属任务 ID
mutator	INT(11)	使用的变异算子编号
beginTimes	Datetime	测试任务开始时间
finishTimes	Datetime	测试任务结束时间
testTimes	INT(11)	总执行次数
finishTimes	INT(11)	已执行次数
status	INT(11)	任务状态
testResults	TEXT	运行结果

如表 4-6所示为系统的 TestTask 数据库表字段，每一条记录对应一个总任

务中的一个变异算子的执行情况。用户在每次进行测试任务时可以选择多个变异算子，系统将依次使用用户选择的变异算子进行模糊测试，在测试任务中，每使用一个变异算子，就会在 TestTask 表中生成一条记录，TestTask 表详细记录了以该变异算子为用例生成的基础进行的模糊测试的详细信息，如使用的变异算子编号、所属的任务 ID、测试开始时间、结束时间以及测试的执行次数、状态和结果等。通过 TestTask 数据库表，系统可以保存每个子任务的执行结果，并随时查看和管理子任务。

表 4-7: DataSet 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增 ID 字段
dataSetName	VARCHAR(200)	数据集名称
dataSetPath	VARCHAR(200)	数据集路径
dataSetNotes	VARCHAR(200)	备注
upLoadTimes	Datetime	数据集上传时间
filesNum	INT(11)	点云文件个数

如表 4-7所示为系统的 DataSet 数据库表字段，每一条数据对应用户上传的一个数据集相关信息。用户填写数据集名称并上传数据集后，系统将数据集存放在指定路径下，同时在数据库中记录数据集的相关信息。每上传一个数据集 DataSet 表就会增加一条记录，记录的具体信息包括数据集名称、数据集路径、上传时间以及点云文件个数等。通过 DataSet 数据库表，系统可以方便的查询所有上传的数据集的具体信息，并且随时对数据集进行管理。

如表 4-8所示为系统的 TestModel 数据库表字段，每一条数据对应用户上传的一个被测模型相关信息。用户填写模型名称并上传模型文件后存放在指定路径下，同时在数据库中记录模型的相关信息，每上传一个模型就会增加一条记录，记录的具体信息包括模型名称、模型路径、上传时间以及模型类型等。通过 TestModel 数据库表，系统可以快速查询所有上传的模型的具体信息，并随时对模型进行管理。

表 4-8: TestModel 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增 ID 字段
modelName	VARCHAR(200)	模型名称
modelPath	VARCHAR(200)	模型路径
modelNotes	VARCHAR(200)	备注
modelType	INT(11)	模型类型
upLoadTimes	Datetime	上传时间

4.3 本章小结

本章节主要介绍了系统的需求分析以及概要设计。首先对本文的自动驾驶软件激光雷达模糊测试系统进行用例分析，根据用例分析的结果，将系统划分为测试任务配置、测试用例生成、测试报告生成以及点云可视化四个模块。其次从功能性需求和非功能性需求两个方面对系统进行需求分析。随后通过系统的概要设计，介绍了系统的整体架构，并从逻辑、进程和开发三个角度对系统的架构模型进行详细描述。最后给出系统的持久化设计。

第五章　详细设计与实现

本章主要介绍针对自动驾驶软件的激光雷达模糊测试系统的详细设计与具体实现。根据第三章的需求分析，将本系统划分为测试任务配置、测试用例生成、测试报告生成和点云可视化四大模块。

5.1　测试任务配置模块

5.1.1　详细设计

测试任务配置模块主要实现模糊测试执行前的准备工作以及与用户端的交互，包括测试数据集管理子模块、测试模型管理子模块以及测试参数配置子模块。

在测试数据集管理子模块中，用户上传的测试数据集将被放到指定目录下并将该数据集的信息存入数据库的 DataSet 表中，紧接着后端会对测试数据集的布局以及格式进行验证。用户首先在前端的数据集列表界面中点击“上传测试数据集”按钮，在弹出的模态框中，填写数据集名称和数据集备注，并上传点云数据文件以及对应的标签文件。紧接着，后端使用 @RequestParam 接收前端传来的数据，将点云数据以及标签文件存入指定的任务空间路径中，并检查点云数据和标签文件的格式和数量是否正确。若点云数据文件不是 PCD 格式，则需要调用 PCDConvertor 类中的方法将其转化为 PCD 格式。另外，该模块还提供测试数据集列表显示以及数据集的删除服务。用户在进入数据集列表界面时，前端将自行向后端提交数据集列表获取请求，后端在接收到请求后查询数据库 DataSet 表获得数据集列表信息返回前端。通过在数据集列表中点击选定数据集的“删除”按钮，可以向后端发送请求删除指定的数据集，后端接收到请求后将删除指定数据集并修改数据库记录。

在测试模型管理子模块中，用户上传的测试模型将被放到指定目录下并将该模型的信息存入数据库的 TestModel 表中。用户首先在前端的模型列表界面中点击“上传测试模型”按钮，在弹出的模态框中填写测试模型名称和备注，

然后选择模型类型文件并上传训练好的测试模型文件。紧接着，后端在接收前端传来的数据后，首先验证模型文件格式是否符合选定的模型类型，验证通过后将模型文件存入指定的任务空间路径中，并将模型信息存入数据库中。另外，该模块还提供模型列表显示以及模型删除服务。在进入模型列表界面时，前端将自行向后端提交模型列表获取请求，后端接收到请求后查询数据库 TestModel 表获得模型列表信息返回前端。通过在模型列表界面中点击选定模型的“删除”按钮，可以向后端请求删除指定的模型，后端接收到请求后将删除指定模型文件并修改数据库记录。

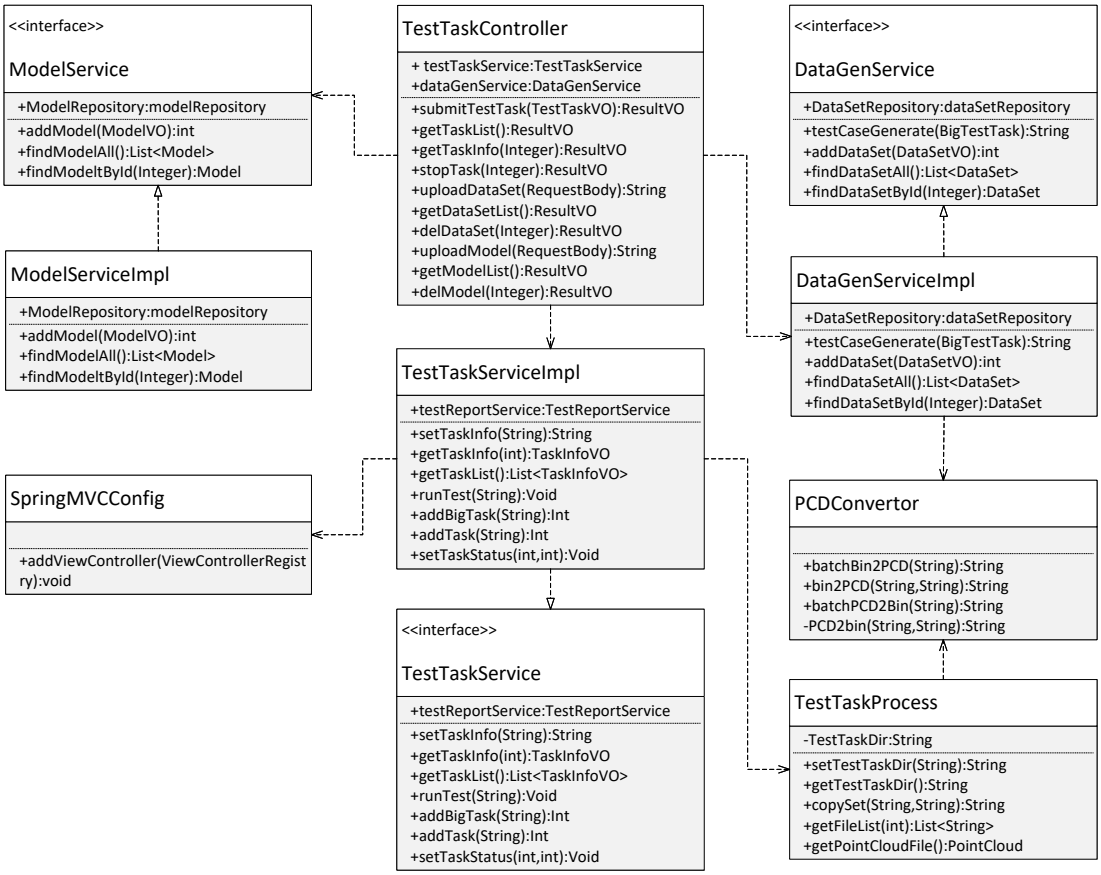

图 5-1: 测试任务配置模块核心类图

在测试参数配置子模块中，用户在前端界面填写任务参数并提交模糊测试任务。用户首先在任务列表界面点击“新增任务”按钮，在弹出的模态框中选择变异算子、测试数据集以及测试模型等任务配置信息，所有这些配置信息会在前端封装为 JSON 数据格式发送到后端进行处理。后端获取任务参数后新建任务并将任务信息存入数据库的 BigTestTask 表和 TestTask 表中。另外，该模块

还提供任务列表显示服务，用户在进入任务列表界面时，前端将自行向后端提交任务列表获取请求，后端在接收到请求后查询数据库 BigTestTask 表获得任务列表信息返回前端。同时，通过点击选定任务的“详情”按钮，跳转到指定任务的详情页面。至此测试任务配置模块完成。

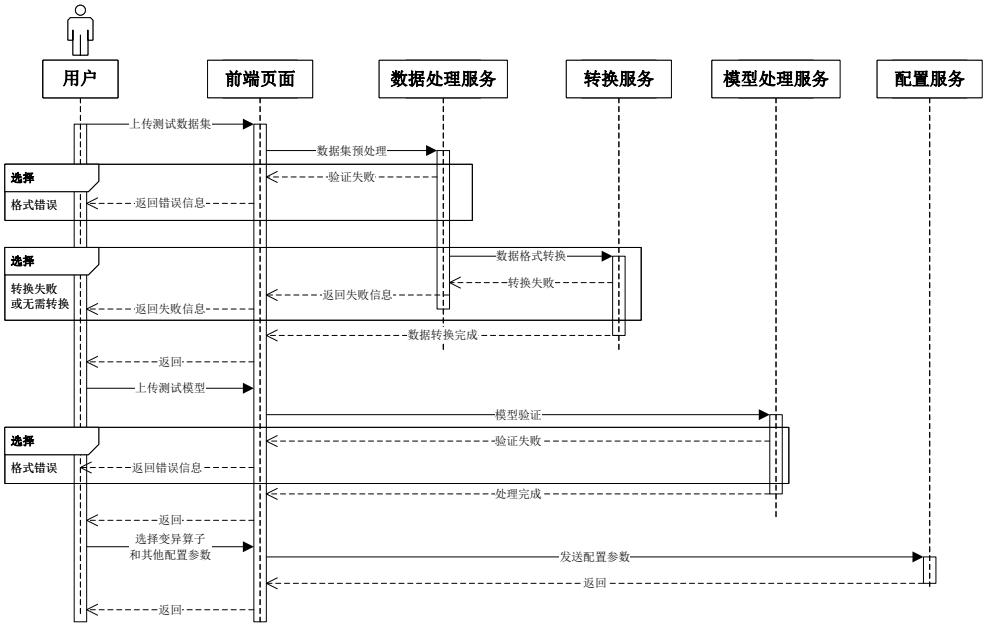

图 5-2: 测试任务配置模块顺序图

5.1.2 核心类设计

如图 5-1所示为系统的测试任务配置模块的核心类图，其中省略了视图对象类、数据库映射类以及持久化实体类等非核心类。该模块设计的类主要在 Controller、Service、Utils 以及用户界面层的 Static 和 html 包中。其中 TestTaskController 逻辑控制类主要是处理与前端用户的交互，包括接收用户上传的测试数据集和模型、管理测试数据集和模型、接收用户提交的任务参数以及处理用户提交的任务查询请求等。TestTaskService 主要为测试任务配置模块提供任务信息的管理服务。TestTaskProcess 类主要为测试任务配置模块提供基础任务配置功能，主要是进行测试数据集路径获取以及测试数据集格式的检验。DataGenService 类主要为测试任务配置模块提供测试数据集的信息管理服务。ModelService 类主要为测试任务配置模块提供测试模型的信息管理服务。SpringMVCConfig 类主要为系统提供前端页面路由服务。PCDConvertor 类是点云文件格式转换工具类，主要为系统提供 PCD 格式与其他点云文件格式的互相

转换，方便系统对点云数据进行处理，统一的格式也有利于系统的可扩展性。

5.1.3 流程设计

如图 5-2所示为系统的测试任务配置模块顺序图。用户可以选择要使用的测试数据集，也可以上传新的测试数据集，如果上传新测试数据集，系统会对数据集进行预处理，验证数据集的格式是否正确，若格式错误则返回错误信息，否则判断是否需要进行数据格式转换，将点云数据文件格式转换为 PCD 格式，若转换失败则返回失败信息，否则返回成功信息提示用户数据上传成功。接下来，用户可以预览并选择模糊测试变异算子，选择测试模型或上传新测试模型并填写其余配置信息，并提交到后端进行处理，后端将配置信息交给配置服务进行测试任务配置。

```
public ResultVO uploadDataSet(@RequestParam("pcd_upload")
    MultipartFile[] pcdFiles, //省略其余参数){
    if(dataGenService.checkDataSet(pcdFiles, labelFiles) !=
        0){// 数据集校验
        return ResultVOUtil.error("upload erro! ");
    }
    // 省略数据集存放路径创建代码
    for(int i=0;i<pcdFiles.length;i++){// 保存点云文件
        try{
            if (!pcdFiles[i].isEmpty()) {
                String uniqueName =
                    pcdFiles[i].getOriginalFilename();
                pcdFiles[i].transferTo(new File(realPath +
                    File.separator + "pcd" + File.separator +
                    uniqueName));
            }
        } catch (Exception e) { // 省略异常处理代码 }
    }
    for(int i=0;i<labelFiles.length;i++){
    // 保存label文件，与pcd文件保存逻辑类似，省略}
    DataSetVO dataSetVO = new DataSetVO();
// 省略dataSetVO的赋值代码
    int dataSetId = dataGenService.addDataSet(dataSetVO); //
        数据集信息存入数据库
    dataSetVO.setDataSetId(dataSetId);
    return ResultVOUtil.success(dataSetVO);
}
```

图 5-3: 测试数据集上传关键代码

5.1.4 关键代码

```
public ResultVO submitTestTask(@RequestBody TestTaskVO
    testTaskVO){
  BigTestTask bigTestTask = new BigTestTask();
  // 配置任务信息
  bigTestTask.setMutator(testTaskVO.getMutators().toString());
  bigTestTask.setMutatorNums(testTaskVO.getMutators().size());
  bigTestTask.setDataSetId(testTaskVO.getDataSetId());
  bigTestTask.setModelId(testTaskVO.getModelId());
  // 添加测试任务
  Integer taskID = testTaskService.addTestTask(bigTestTask);
  if( taskID == 0){
     return ResultVOUtil.error("Add Test Task Error! ");
  }
  bigTestTask.setId(taskID);
  // 添加子任务
  int mutatorSize = testTaskVO.getMutators().size();
  for(int i = 0; i <= mutatorSize; i++){
     TestTask testTask = new TestTask();
     // 省略任务信息配置代码
     testTaskService.addTestTask(testTask);
  }
  // 调用数据生成服务执行生成器
  dataGenService.testDataGenerate(bigTestTask);
  // 配置返回VO信息
  testTaskVO.setId(taskID);
  // 省略testTaskVO其余信息配置
  return ResultVOUtil.success(testTaskVO);
}
```

图 5-4: 新增测试任务关键代码

如图 5-3所示为测试任务配置模块的测试数据集上传关键代码，由于代码长度较长，只列出部分核心关键代码。本方法主要提供用户上传测试数据集的处理逻辑，通过 @RequestMapping 将前端的请求路由到该方法进行处理，紧接着通过 @RequestParam 获取前端传来的点云数据文件、标签文件、数据集名称以及备注等信息。在该方法的处理逻辑中，首先对点云文件以及标签文件格式进行校验和格式转换，若校验失败则返回错误信息，否则继续。紧接着，根据数据集名称在指定路径创建一个新的目录，然后将点云文件以及标签文件分别存入到对应的路径下。在完成以上操作后，新建 DataSetVO 对象并赋值，调用 DataGenService 服务的 addDataSet 方法去将数据集信息添加到数据库中，并获取生成的数据集 ID。最后，配置 DataSetVO 的其余信息并将信息封装后返回到

前端进行显示。到此，测试数据集上传完成。

如图 5-4所示为测试任务配置模块的新增测试任务关键代码，由于代码长度较长，只列出部分核心关键代码。本方法主要提供用户提交模糊测试任务的处理逻辑，通过 @RequestMapping 将前端的请求路由到该方法进行处理，紧接着通过 @RequestBody 获取前端传来的参数 TestTaskVO 对象，该对象主要包含测试数据集 ID、被测模型 ID 以及选择的变异算子等。在该方法的处理逻辑中，首先创建一个新的任务对象 BigTestTask，根据前端传来的参数对象 TestTaskVO 来配置 BigTestTask 任务对象的变异算子、算子数量、测试数据集 ID 以及被测模型 ID。在所有信息配置完成后，调用 TestTaskService 服务的 addTestTask 方法去将任务信息添加到数据库中，并获取生成的任务 ID。紧接着，调用数据生成服务 DataGenService 去启动数据生成器对数据进行变异生成。最后，配置 TestTaskVO 的其余信息并将信息封装后返回到前端进行显示。到此，测试任务新增完成。

5.2 测试用例生成模块

5.2.1 详细设计

测试用例生成模块主要实现自动驾驶软件激光雷达模糊测试系统的测试用例生成功能，根据用户选择的变异算子对原始数据集进行变异，并将变异生成的新测试用例集交付给测试报告生成模块进行测试评估。

在前一阶段的测试任务配置模块中，用户提交测试任务请求后，后端在接收到配置信息后进行一系列的验证和处理，完成后调用测试用例生成服务，即进入测试用例生成的阶段。在测试用例生成阶段中，主要服务为测试用例生成服务。测试用例生成服务调用数据处理工具类拷贝原始数据集，以避免在用例生成时覆盖了原始数据集。紧接着测试用例生成服务调用数据生成器并根据变异算子以及测试数据集路径等参数，对指定测试数据集进行批量变异处理。数据生成器主要逻辑是批量的变异业务逻辑，而真正的对单个点云数据文件进行变异处理的是 PCD 变异器，数据生成器不断调用 PCD 变异器对点云数据进行变异处理。

PCD 变异器是使用 C++ 语言进行编写，使用点云库 (PCL) 对点云数据进行处理。在 PCD 变异器中，首先获取要进行变异生成的 PCD 点云数据文件，根据预先设定好的变异规则对点云数据进行处理，并在变异完成后将点云数据流

重新保存为 PCD 点云数据文件格式。为了提高点云数据的处理速度，PCD 文件中的点云数据应该写入为二进制格式而不是 ASCII 码格式。

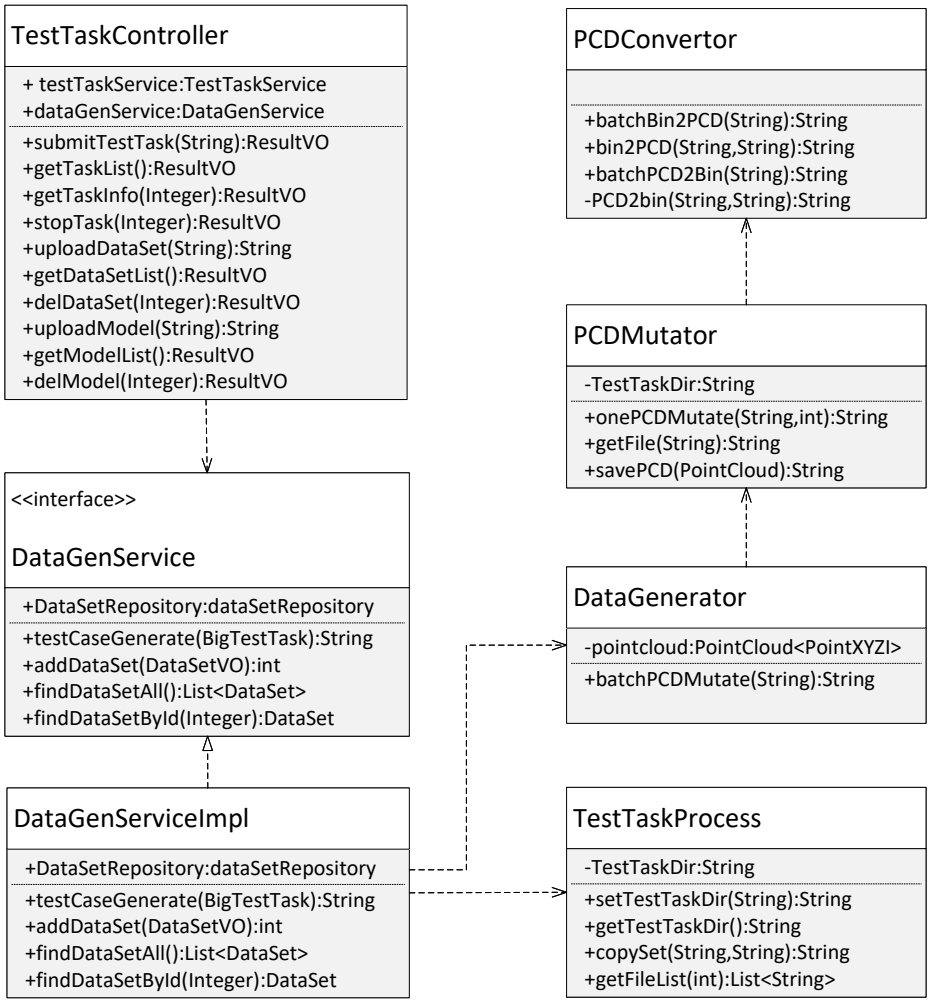

图 5-5: 测试用例生成模块核心类图

5.2.2 核心类设计

如图 5-5所示为系统的测试用例生成模块核心类图。该模块设计的类主要在 Controller、Service、Utils 以及 DataGenerate 文件夹中。其中 TestTaskController 主要是在接收到用户的测试任务执行请求后，调用相应的测试用例生成服务进行处理。DataGenService 服务是测试用例生成模块的主要入口服务，其他模块与服务通过调用 DataGenService 服务中的方法来使用测试用例生成功能。TestTaskPreparation 类是测试数据集的预处理类，在测试用例生成模块中主要提供测试用例集的备份服务。DataGenerator 类是测试用例生成的处理和调

度类，主要功能是根据选定的变异算子调用变异器相应功能对点云数据进行变异。PCDMutator 是测试用例生成模块的核心功能，它将根据传递来的参数（变异算子、数据集路径等）使用相应的变异功能对指定的点云数据文件进行处理，生成变异点云数据文件并保存到指定路径下。PCDConvertor 是一个工具类，提供点云数据文件格式 PCD 与其他格式的互相转换。

5.2.3 流程设计

如图 5-6所示为系统的测试用例生成模块顺序图。在测试任务配置阶段完成后，系统将在任务测试链中调用数据生成服务开始测试用例的生成。首先进行测试用例集的备份，因为测试用例的变异生成会对覆盖原本的数据集，因此要备份原始测试数据集一遍接下来其他变异算子的使用，若备份成功则开始进入变异生成阶段，否则返回备份错误信息。在变异生成阶段中，数据生成服务将变异参数传递给变异服务，变异服务对点云数据进行变异生成，然后调用存储服务将点云数据保存为 PCD 格式，若变异失败则返回失败信息，否则进入下一阶段。

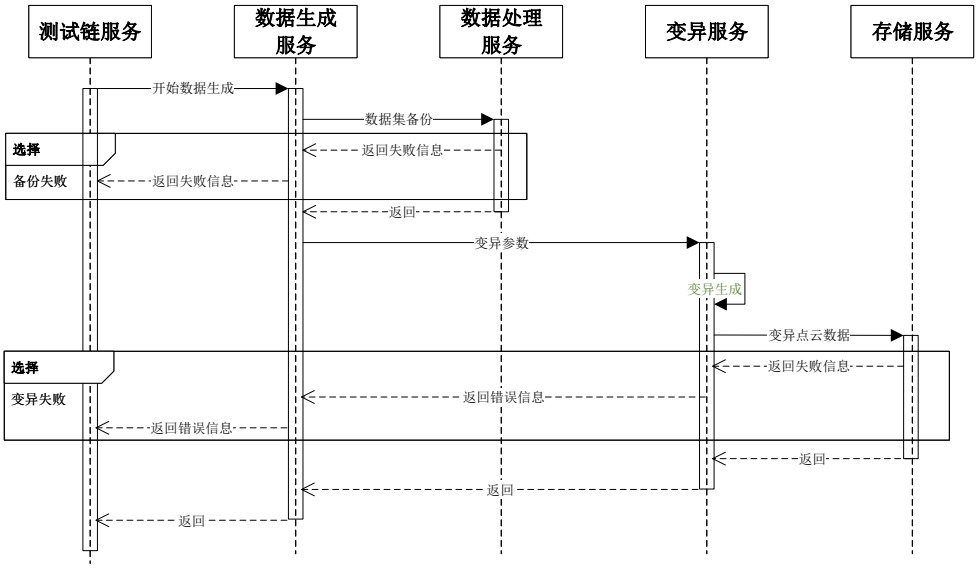

图 5-6: 测试用例生成模块顺序图

5.2.4 关键代码

如图 5-7所示为测试用例生成模块的关键代码。图中为根据测试数据集文件列表批量调用核心变异生成器进行变异生成的代码，其中，参数 *filenameList* 数组为测试数据集文件路径列表数组，*choice* 为选择的变异算子。首先，根据 *filenameList* 测试数据集文件列表数组进行循环，在循环体中，首先根据数组中的点云数据路径读取对应的点云 PCD 文件转化为 PointCloud<PointXYZI> 格式存入到 *original_pcd* 中，读取成功后，将 *original_pcd* 和 *choice* 作为参数去调用 one_pcd_mutat 方法对点云数据进行变异处理。最终调用 savePCDFileBinary 将返回的点云数据保存为 PCD 格式。

```
void batch_pcd_mutate(vector<string> filenameList, int
    choice) {
    cout << "batch_pcd_augment begin" << endl;
    int num = 1;
    // 循环读取PCD文件
    for (vector<string>::iterator iter =
        filenameList.begin(); iter != filenameList.end();
        ++iter) {
        cout << "index:" << num << endl;
        num++;
        clock_t start, finish;
        start = clock();
        const char* filename = iter->c_str();
        //根据pcd文件列表中的pcd文件名，打开一个对应的pcd文件
        pcl::PointCloud<pcl::PointXYZI>::Ptr
            original_pcd(new pcl::PointCloud<pcl::PointXYZI>);
        if (pcl::io::loadPCDFile<pcl::PointXYZI>(filename,
            *original_pcd) == -1) {
            PCL_ERROR("Couldn't read file pcd\n");
        }
        // 调用one_pcd_mutate变异一个pcd
        pcl::PointCloud<pcl::PointXYZI>::Ptr augment_pcd =
            one_pcd_mutate(original_pcd, choice);
        // 保存为PCD格式
        pcl::io::savePCDFileBinary(filename, *augment_pcd);
        finish = clock();
        printf("%f seconds\n", (double)(finish - start) /
            CLOCKS_PER_SEC);
    }
}
```

图 5-7: 测试用例生成模块关键代码

5.3 测试报告生成模块

5.3.1 详细设计

测试报告生成模块主要功能是启动指定框架并使用生成的测试用例对指定模型执行测试，最终得到测试结果汇总分析输出测试报告。测试模型需要对应的框架来运行，因此首先需要检测框架是否正常运行，确认框架正常运行后，发送指令使框架使用指定的测试数据集对测试的模型进行评估。

框架在对模型评估结束后，得到的评估结果将被记录下来。我们将保存每个变异算子生成的测试数据集得到的评估结果，并在最后所有变异算子运行完毕后，根据每个变异算子的运行情况生成一份测试报告，显示模糊测试总体结果以及每个变异算子对应的测试数据集的评估结果。

5.3.2 核心类设计

如图 5-8所示为系统测试报告生成模块的核心类图。该模块设计的类主要在 Controller、Service 以及 Utils 中。其中 TestTaskController 在该模块中的主要功能是测试链的功能，在上一阶段的测试用例生成完成后，调用测试任务服务去启动模型评估。TestTaskService 服务主要是去调用测试任务启动器启动框架使用指定测试数据集对目标模型进行评估。同时也能对测试任务的基本信息进行管理，获取任务信息和测试报告。TestTaskStarter 类主要用于发送命令到目标框架来对目标模型进行评估，同时利用该类对框架进行管理。TestReportService 服务通过调用测试结果处理类对评估的结果进行分析处理，得到每次评估的具体结果值，最后汇总后形成测试报告保存到数据库中。TestResultProcess 类主要用于获取模型评估的结果，并将结果解析得到具体结果值。

5.3.3 流程设计

如图 5-9所示为系统的测试报告生成模块顺序图。在测试用例生成阶段完成后，系统会在任务测试链中调用测试任务服务开始对目标模型进行评估。首先调用评估运行服务进行评估前验证，验证框架是否正常运行，若失败则尝试重启框架，重启失败则返回失败信息。若验证成功，则开始进行评估。评估运行服务使用命令“python ./pytorch/train.py evaluate –config_path=./configs/all.fhd.config –model_dir=./data/KITTI_DATASET_ROOT/model_dir/”

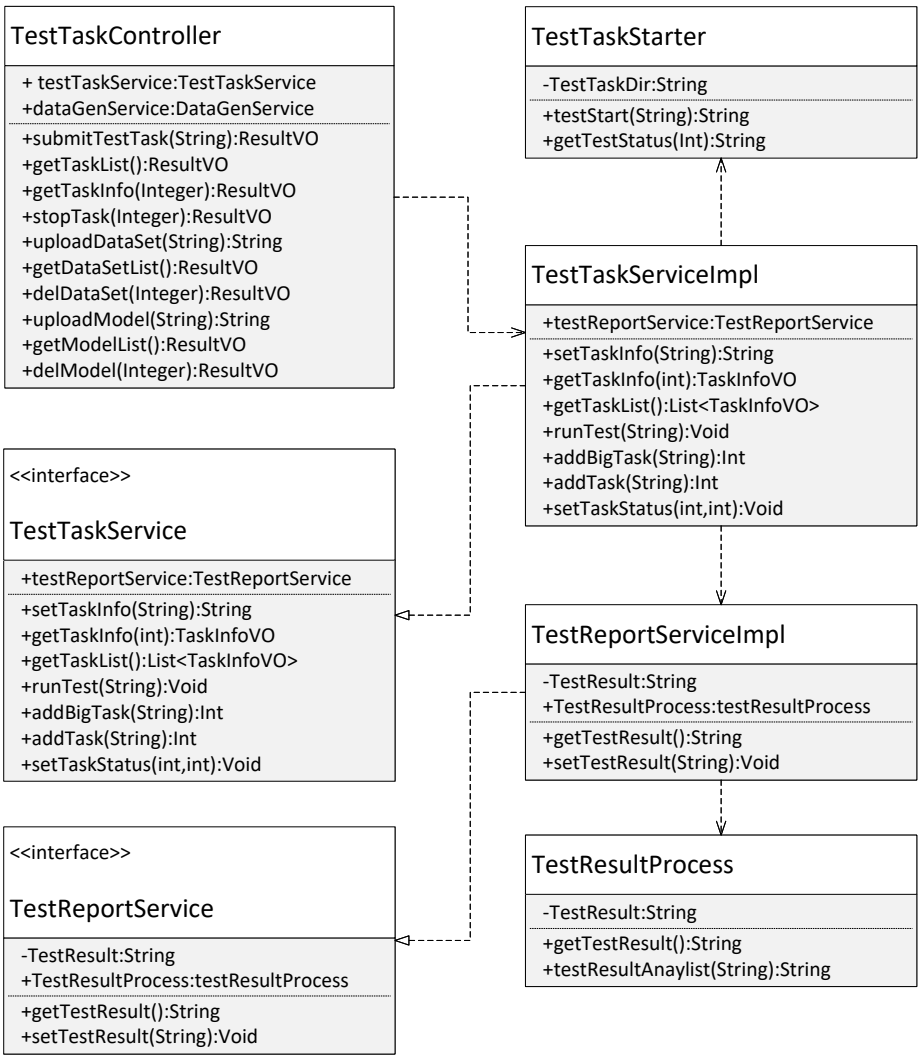

图 5-8: 测试报告生成模块核心类图

启动框架开始对目标模型进行评估。在所有模糊测试变异算子都执行完毕后，调用报告生成服务进行测试报告的生成，测试报告服务请求测试结果服务获得评估的结果，将评估结果分析处理形成测试报告后返回给测试任务服务。

5.3.4 关键代码

如图 5-10为测试报告生成模块的一部分关键代码。图中的方法为定时任务，设置为每分钟执行一次，目的是读取数据库查看是否有任务需要进行模型评估，由于测试用例生成所需时间比较长，同步调用模型评估会使得程序阻塞时长过久，因此采用定时扫描获取任务进行模型评估。首先通过调用 TestTaskService 服务获取需要进行评估的任务 BigTestTask，紧接着调用工具类

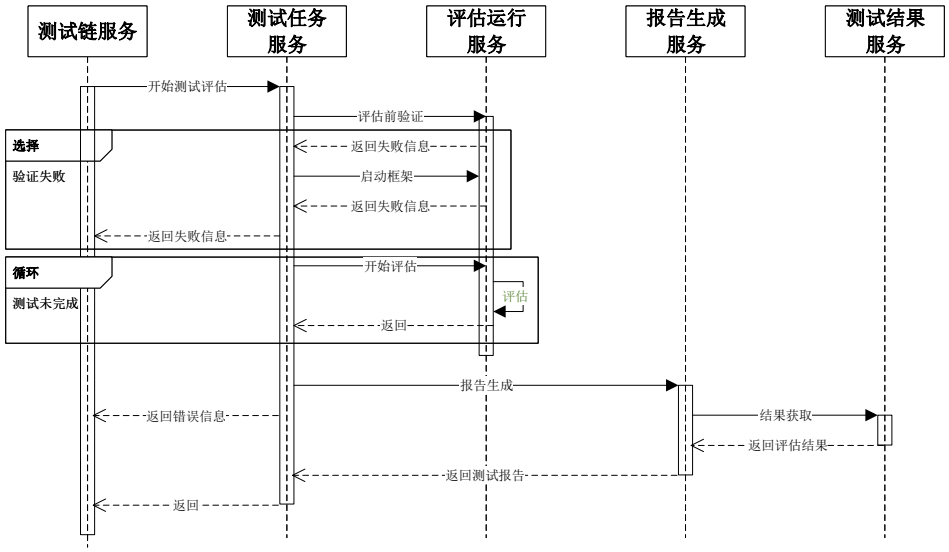

图 5-9: 测试报告生成模块顺序图

TestEvaluationStarter 去获取评估框架的状态，如果框架状态为 *FREE* 说明框架目前空闲，可以进行评估任务，则调用 TestTaskService 服务的 runTest 方法执行模型评估，并修改评估框架以及测试任务的状态。

```
    @Scheduled(fixedDelay = 1000 * 60) //
        单位是毫秒，设置时间为1分钟
    public void testTaskEvaluationTimer(){
        // 读取待评估任务
        BigTestTask bigTestTask =
            testTaskService.getEvaluationTask();
        if(bigTestTask.getId() != -1){
            // 检查评估框架是否空闲
            if(TestEvaluationStarter.getEvaluatorStatus() ==
                TestEvaluationStarter.FREE){
                // 执行评估
                testTaskService.runTest(bigTestTask);
                // 修改评估框架状态
                TestEvaluationStarter.setEvaluatorStatus(
                    TestEvaluationStarter.EVALUATING);
                // 修改任务状态
                testTaskService.setTaskStatus(bigTestTask.getId(),
                    2);
            }
        }
    }
```

图 5-10: 测试报告生成模块关键代码

5.4 点云可视化模块

5.4.1 详细设计

点云可视化模块主要实现激光雷达点云数据的可视化功能，通过对点云数据的三维可视化，可以使得用户直观的查看点云的可视化演示图，方便用户比较点云的变异效果。

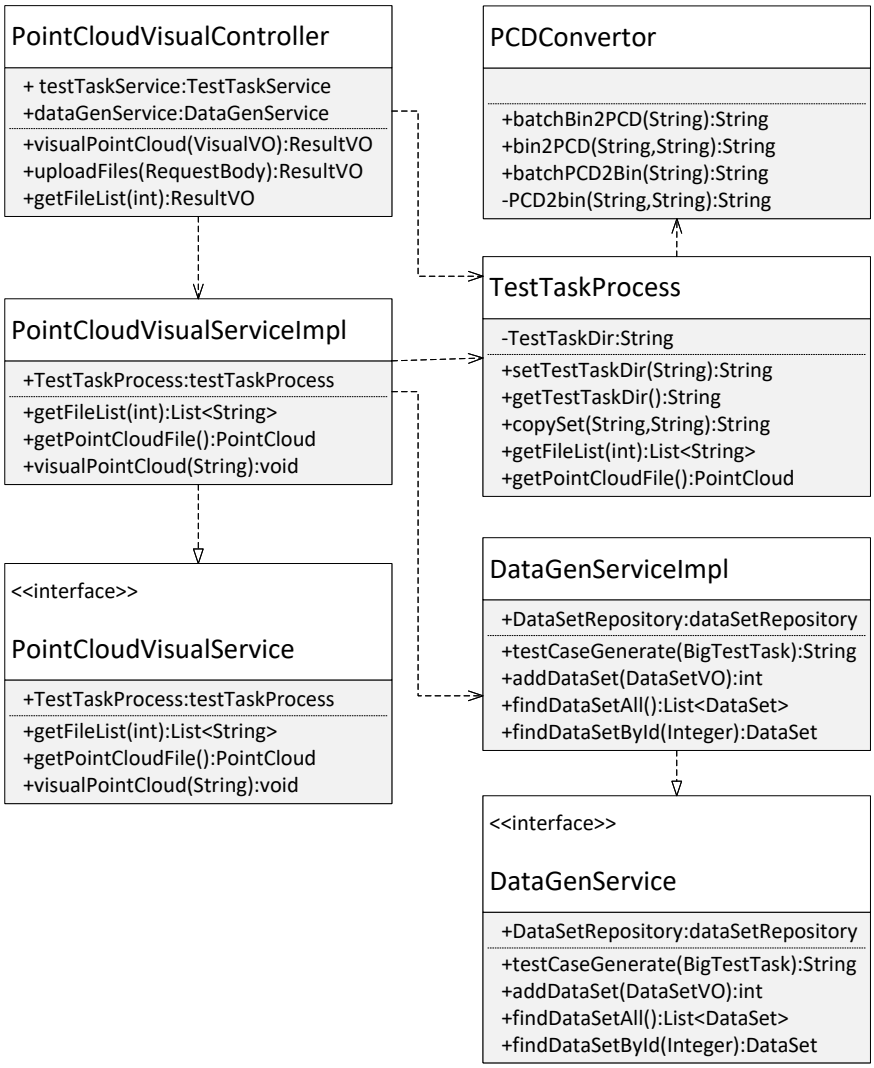

图 5-11: 点云可视化模块核心类图

点云可视化模块主要涉及点云数据处理和点云数据可视化两个子模块。在点云数据处理阶段，与测试任务配置模块中的数据处理方式类似，用户首先在前端的点云可视化界面中点击“上传点云数据”按钮，接着后端使用

@RequestParam 接收前端传来的点云数据，然后将数据集存入指定的路径文件夹中。紧接着，系统将对用户上传的点云数据集进行格式校验，如果用户上传的点云数据不是 PCD 格式，系统将调用 PCDConvertor 中的方法将点云数据转化为 PCD 格式，完成以上步骤，则点云数据处理完成，后端获取用户上传的点云数据文件列表返回前端，然后进入可视化阶段。在可视化阶段中，用户在前端从上传的点云数据中选择要可视化的点云数据文件，点击可视化按钮，后端将获取到要可视化的点云数据名称，读取对应的点云数据文件，配置可视化信息并执行可视化。至此，点云可视化模块完成。

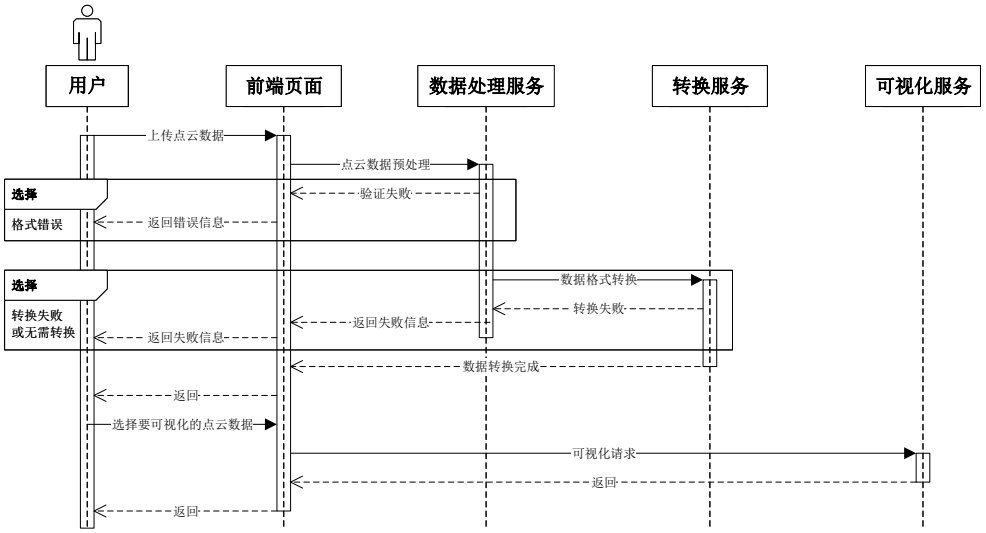

图 5-12: 点云可视化模块顺序图

5.4.2 核心类设计

如图 5-11所示为系统点云可视化模块的核心类图。该模块设计的类主要在 Controller、Service 以及 Utils 中。其中 PointCloudVisualController 在该模块中的主要功能是接收前端的请求并与前端进行交互，在接收到前端的可视化或数据上传请求后，调用相应的服务对用户的请求进行处理，处理完后将处理结果返回前端。PointCloudVisualServiceImpl 服务继承自 PointCloudVisualService 接口，主要是为 PointCloudVisualController 提供点云数据的可视化服务，通过去调用 TestTaskProcess 工具类获取点云数据，然后对点云数据进行可视化处理。TestTaskProcess 类主要提供点云数据列表以及点云数据的获取，并且对数据进行处理，例如调用 PCDConvertor 对点云数据进行格式转换。

5.4.3 流程设计

如图 5-12所示为系统的点云可视化模块顺序图。首先用户上传要可视化的点云数据，可以上传任意数量的点云数据文件，后端将使用数据处理服务对点云数据进行校验，若点云格式不是 PCD 格式，则将点云数据转换为 PCD 格式方便进行处理，完成后返回点云文件列表到前端页面展示。用户选择要可视化的点云数据，点击可视化按钮，后端获得要可视化的点云文件信息后，调用可视化服务读取对应的点云文件并对其进行可视化处理。

5.5 示例展示

本文实现的针对自动驾驶软件的激光雷达模糊测试系统主要包含 5 个主页面以及一些页面弹出模态框，包括任务列表、数据集列表、模型列表、任务详情页面以及点云可视化页面，弹出模态框有新增测试任务模态框、上传数据集模态框以及上传模型模态框。本节挑选较具代表的任务列表页面、任务详情页面以及新增任务模态框作为代表，以截图的形式分别展示，并作简要说明。

图 5-13: 任务列表页面展示图

如图 5-13所示为系统的模糊测试任务列表页面。该页面主要以列表的形式展示系统已提交的任务的基本信息，用户可以通过在导航栏点击任务列表进入该页面。在任务列表界面，用可以方便的查看所有任务的基本信息，包括任务 ID、任务状态、测试数据集、测试模型以及变异算子数量等任务关键信息。如果用户想查看某个任务的详细信息，可以通过点击表格最右列的详情按钮来跳转到该任务的详情页面。另外，用户可以通过点击右上角的新增任务按钮来提交新的测试任务。

如图 5-14所示为系统的任务详情页面。该页面主要负责展示具体任务的详细执行信息并提供任务管理功能，用户可以通过在任务列表页面点击任务详情

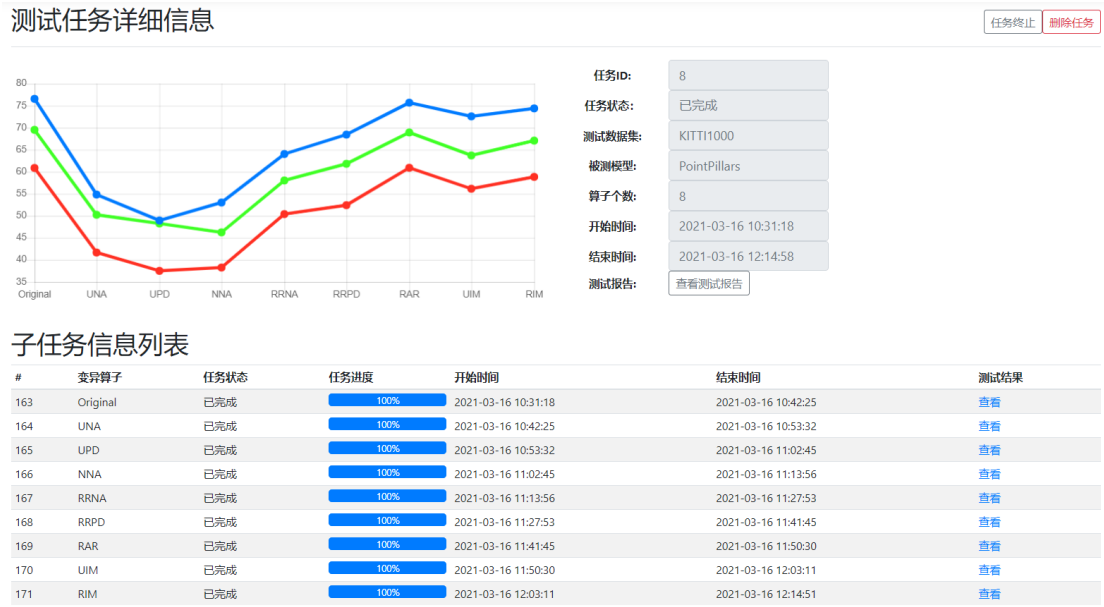

图 5-14: 任务详情页面展示图

可进入该任务的详情页。在任务详情页右上角，用户可以通过点击任务终止按钮来终止任务，也可以点击任务删除按钮来删除任务。在页面右上区域，我们可以看到任务执行结果的信息图表，其中，x 轴为使用的变异算子，y 轴为对应的评估结果 AP 值，蓝色的线为 $bbox\ AP$ 值，绿色的线为 $bev\ AP$ 值，红色的线为 $3d\ AP$ 值。在信息图表的右边为任务的详细信息，其中包括查看测试报告按钮，点击可以查看任务的测试报告。页面下方为子任务信息列表，每一行为一个变异算子执行的测试信息，包括使用的变异算子、任务状态、任务进度以及时间等，并可以点击查看按钮来查看子任务的执行结果。

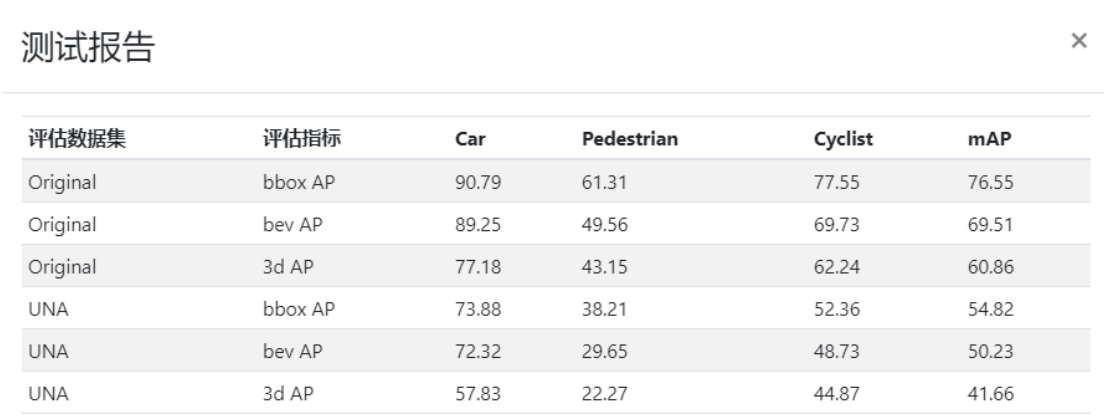

图 5-15: 测试报告展示图

如图 5-15所示为系统的测试报告展示模态框。该模态框主要负责展示测试

任务最终的测试报告，用户可以通过在任务详情页面点击测试报告按钮来弹出该模态框。测试报告主要由表格形式展示，包括变异算子、评估指标、目标类型以及最终的平均精度均值。其中，变异算子为 Original 的行代表原始对照数据集的结果。通过该测试报告，用户可以详细对比模型在不同变异算子以及不同指标下的表现，进一步发现模型存在的问题。

图 5-16: 新增测试任务模态框展示图

如图 5-16所示为系统的新增测试任务模态框。该模态框主要提供新增任务的配置信息填写并提交任务请求的功能，用户通过在任务信息列表页面点击新增任务按钮即可触发弹出新增测试任务模态框。在模态框左上区域为模糊测试变异算子的选择区域，用户可以通过点击勾选来选择对应的变异算子，勾选的变异算子将在任务中执行。在变异算子选择区域的右边，是变异算子的效果展示及说明，用户点击变异算子时，将在展示区域显示对应的变异算子效果图和说明。在模态框中间是测试数据集以及被测模型的选择框，用户可以选择想要使用的测试数据集以及要测试的模型，如果想上传新的测试数据集或新模型，可以通过点击右边的按钮跳转到数据集列表页面或模型列表页面进行上传。最

后，可以通过点击模态框右下角的提交任务按钮来提交执行新任务。

5.6 本章小结

本章主要介绍了针对自动驾驶软件的激光雷达模糊测试系统的详细设计与实现。根据系统需求分析与概要设计，分别对测试任务配置、测试用例生成、测试报告生成和点云可视化四大模块进行详细设计、核心类设计、流程设计以及关键代码介绍，并在给出了系统实现的示例展示。

第六章　系统测试与实验评估

本章将基于系统的需求分析以及详细设计，对系统进行功能性测试，并展示测试结果。最后通过组织和开展实验，对本系统的自动驾驶软件激光雷达模糊测试及其变异算子的有效性进行评估。

6.1　测试环境

表 6-1: 系统测试环境配置信息

字段	测试环境
服务器	Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz Tesla V100-SXM2-32GB * 2 物理内存 100G
操作系统	Ubuntu 16.04
数据库	Mysql 5.7.29
浏览器	Chrome 浏览器 84.0
Python 版本	Python3.6.5
其他软件	CUDA10.0、CUDNN7.4.2、PyTorch 1.1.0 Cmake3.13.2、Docker20.10.2、Spring Boot2.0 JDK1.8.0、PCL1.9.1

根据系统的需求以及具体的实现，如表 6-1所示为系统的测试环境配置信息。本系统部署在实验室环境下的服务器下，该服务器装有两个 32GB 显存的 Tesla 显卡为深度学习模型的训练与测试提供加速服务。后端请求处理服务使用 Spring Boot 进行构建，点云数据基于 PCL 库进行处理，模糊测试任务运行模块基于 CUDA、CUDNN 和 PyTorch 构建。软件和工具使用官方提供的稳定版本，其它选项则参照实际开发环境来进行配置。

表 6-2: 测试任务配置模块测试用例表

测试项	操作/输入	预期结果	测试结果
新建任务	1. 点击“新建任务”按钮	弹出新建测试任务模态框	通过
上传数据集	1. 点击数据集列表界面中的“上传数据集”按钮， 2. 填写数据集名称， 3. 选择本地数据文件上传， 4. 等待上传完成	数据集成功上传	通过
上传模型	1. 点击模型列表界面中的“上传模型”按钮， 2. 填写模型名称， 3. 选择本地模型文件上传， 4. 等待上传完成	模型文件成功上传	通过
选择变异算子	1. 点击新建任务界面中的“新建任务”按钮， 2. 勾选变异算子， 3. 查看每个变异算子的简介和预览是否正常显示， 4. 提交测试任务， 5. 等待任务完成， 6. 查看是否按给定变异算子进行模糊测试	按所选变异算子进行模糊测试	通过
查看任务	1. 点击“任务列表“选项	展示历史任务及其完成状态	通过
查看任务详情	1. 点击”任务详情“按钮	展示任务详细运行情况	通过
提交任务	1. 填写任务配置信息 2. 点击“提交任务”按钮	任务成功提交，等待执行	通过
任务终止	1. 在任务详情页面点击“任务终止”按钮， 2. 点击确认	任务终止	通过
查看测试报告	1. 在任务详情页面点击“查看测试报告”按钮，	弹出测试报告模态框，正确显示报告	通过

6.2 功能测试

本系统的功能测试主要是基于需求分析中用例分析中描述的内容来设计，功能测试的测试用例则由给定输入或操作以及期望的输出或展示结果组成，通过对比测试用例的期望结果与系统的实际结果，来验证系统功能的正确性，进而对系统中可能存在的缺陷进行发现和修复，以保证系统功能符合预期。本系统的功能测试主要从测试任务配置、测试用例生成、测试报告生成以及点云可视化四个方面来展开。其中，测试用例生成和测试报告生成是系统中的一个后端连续过程，用户在此过程中无法直接从外界感知，因此本文中测试用例生成和测试报告生成主要是通过系统调用相关函数来完成。

表 6-3: 测试用例生成模块测试用例表

测试项	操作/输入	预期结果	测试结果
启动数据生成服务	调用 testCaseGenerate 函数	成功执行测试用例生成	通过
数据集批量变异	调用 batchPCDMutate 函数	成功对测试数据集进行变异	通过
单个点云变异	调用 onePCDMutate 函数	成功对单个点云数据进行变异	通过
测试数据集备份	调用 copySet 函数	成功备份测试数据集	通过
PCD 文件转换	1. 调用 bin2PCD 函数 2. 调用 batchBin2PCD 函数 3. 调用 PCD2bin 函数 4. 调用 batchPCD2bin 函数	1. 成功将 bin 文件转为 PCD 2. 成功批量将 bin 转为 PCD 3. 成功将 PCD 转为 bin 文件 4. 成功批量将 PCD 转为 bin	通过
读取点云文件	调用 getFile 函数	成功读取点云文件	通过
保存 PCD 文件	调用 savePCD 函数	成功保存点云数据到 PCD 文件	通过

（1）测试任务配置模块。表 6-2所示为系统的测试任务配置模块测试用例，测试任务配置模块主要提供测试任务的配置以及与用户的交互操作。主要验证系统提供的新建任务、上传数据集、上传模型、选择变异算子、查看任务、查看任务详情、提交任务、任务终止以及查测试报告等功能的实际使用是否符合预期结果，测试结果显示，对应的测试用例全部通过。

（2）测试用例生成模块。表 6-3所示为系统的测试用例生成模块测试用例，测试用例生成模块主要为系统提供测试用例的生成以及点云数据的处理功能。主要验证模块是否能正常启动数据生成服务，是否能正常进行数据集的批量生成，是否能正常进行单点云变异，是否能正常备份测试数据集，是否能正常对 PCD 文件进行转换，是否能正常读取点云数据文件以及将点云数据保存为 PCD 文件。经过测试用例验证，以上功能的实际使用情况均符合预期结果，对应的测试用例全部通过。

表 6-4: 测试报告生成模块测试用例表

测试项	操作/输入	预期结果	测试结果
启动数据报告服务	调用 runTest 函数	成功执行评估和报告生成	通过
进行模型评估	调用 testStart 函数	成功使用测试数据集对目标模型进行测试	通过
获取测试状态	调用 getTestStatus 函数	成功获取测试任务状态	通过
获取测试报告	调用 TestReportService 服务中的 getTestResul 函数	成功输出测试报告	通过
获取单个变异算子测试结果	调用 TestResultProcess 服务中的 getTestResul 函数	成功获取测试结果	通过
测试结果分析处理	调用 testResultAnaylist 函数	成功取得测试结果并进行分析处理	通过

（3）测试报告生成模块。表 6-4所示为系统的测试报告生成模块测试用例，测试报告生成模块主要为系统提供模型评估以及报告生成的功能。本测试主要验证模块是否能正常启动数据报告服务，是否能正常进行模型评估，是否能正常获取测试的状态，是否能正常生成测试报告，是否能正常获取单个变异算子测试结果以及正常进行测试结果分析处理。经过测试用例验证，以上功能

的实际使用情况均符合预期结果，对应的测试用例全部通过。

表 6-5: 点云可视化模块测试用例表

测试项	操作/输入	预期结果	测试结果
选择数据集	1. 选择要可视化的数据集 2. 点击确认按钮 3. 等待读取数据集文件	成功读取数据集文件列表并显示	通过
点云可视化	1. 选择要可视化的数据集 2. 点击确认按钮 3. 选择数据集中要可视化的点云文件 4. 点击“可视化”按钮	成功将点云数据可视化显示	通过

（3）点云可视化模块。表 6-5所示为系统的点云可视化模块测试用例，点云可视化模块主要提供点云数据的可视化功能。本测试主要验证是否能正常读取要可视化的数据集以及是否能正常对选择的点云数据进行可视化显示。经过测试用例验证，以上功能的实际使用情况均符合预期结果，对应的测试用例全部通过。

6.3 实验评估

本节主要评估本文所提出的自动驾驶软件激光雷达模糊测试的有效性以及变异算子的有效性。首先说明研究的问题，然后介绍实验数据集、实验对象以及评估指标，并详细说明实验设置，最后给出实验结果。

6.3.1 研究问题

为了进一步验证本文设计的自动驾驶激光雷达变异算子有效性以及自动驾驶软件的激光雷达模糊测试技术的有效性，本文将对以下两个问题进行探索：

问题一：本文提出的变异算子是否有效？

问题二：本文提出的自动驾驶软件激光雷达模糊测试是否有效？

6.3.2 实验数据集

KITTI 数据集 [50] 是目前自动驾驶领域最重要的测试集之一，广泛应用于自动驾驶感知和预测方面。KITTI 数据集包含多场景点云数据，是目前自动驾驶激光雷达点云目标检测模型的最常用训练和测试数据集。因此，本文使用 KITTI 数据集作为实验数据集。

6.3.3 实验对象

本文以 3D 点云目标检测网络 PointPillars [51] 为实验对象。PointPillars 是在 VoxelNet 和 SECOND 的基础上进行改进得到的点云目标检测网络，是当前较为流行的自动驾驶激光雷达目标检测室网络。PointPillars 主要由三部分组成：特征编码网络 PillarFeatureNet、卷积中间网络 PillarFeatureScatter 以及区域生成网络 RPN。首先 PillarFeatureNet 将输入的点云数据编码成为稀疏伪图像，然后由 PillarFeatureScatter 从伪图像提取特征信息，最后由 RPN 对检测框进行分类与回归。选择 PointPillars 作为实验对象的原因在于，与 VoxelNex 和 SECOND 等网络相比，PointPillars 在实时计算速度以及准确度上都要更优，另外，PointPillars 在 GitHub 上开源，且附带了一组设计良好的测试套件，可以减少部署和测试所带来的工作量。

6.3.4 评估指标

表 6-6: 实验评估指标表

指标名称	指标含义
bbox AP	二维检测框准确率
bev AP	鸟瞰图检测框准确率
3*d AP*	三维检测框准确率

评估指标如表 6-6所示，平均精准度 (Average Precision，*AP*) 是目标检测模型常见的评价指标，*AP* 值的计算较为复杂，一般取 PR 曲线（precision-recall curve）下的面积作为 *AP* 值，简单来说就是目标检测结果和真值的 IoU（Intersection over Union）在阀值范围内的平均准确率，*AP* 值越高则说明目标检测模型的效果越好。本实验从三个不同的评估指标来评估实验结果，包括

bbox AP、*bev AP* 以及 3*d AP*，其中 *bbox AP* 为二维检测框的准确率，*bev AP* 为鸟瞰图检测框的准确率，3*d AP* 为三维检测框的准确率。同时，每个评价指标都分为汽车（Car）、行人（Pedestrian）以及自行车（Cyclist）三种检测目标。

6.3.5 实验设置

针对上述所提出的两个问题，本文通过设置以下两个实验进行研究：

实验一：由于模糊测试在进行变异生成的时候可能使得产生的大量点云数据是偏离原本语义的，这样的点云数据是没有意义的，对于模糊测试来说也是低效的。因此我们需要验证变异算子的有效性。首先我们从 KITTI 数据集中随机选择 100 个点云文件作为对照数据集 *D*，然后，分别使用变异算子对这 100 个点云数据文件进变异生成，得到对应的测试实验数据集，其中，由于 RAR、UIM 以及 RIM 变异算子生成的点云数据在视觉上并没有差别，因此本实验不考虑这 3 个变异算子。随后，我们邀请了 10 名同学，先对他们进行基本的激光雷达点云数据科普，让他们熟悉点云数据的基本特征，同时，隐瞒本文的变异算子效果。让 10 名同学分别仔细观察实验数据集和对照数据集中点云数据可视化后的区别，同时对照对应的摄像机图像数据，判断其是否在可理解的相同语义下。主要判断实验数据集的点云中的目标（如，汽车、行人和自行车等）和对照点云相比是否依旧能清晰辨别，然后填写调查问卷。调查问卷提供的选项主要有相同、相似以及不同，其中，相同代表与对照点云相比实验点云中目标均能清晰识别，相似代表对照点云相比实验点云中目标都能识别但有些目标较模糊，不同则代表对照点云相比实验点云中有些目标无法识别。每位同学观察点云数据后填写相应的调查问卷，最后统计调查结果。

实验二：证明了本文所述模糊测试变异算子的有效性后，我们进一步验证本文所述自动驾驶软件激光雷达模糊测试的有效性。为了证明本文所述模糊测试是有效的，能提高测试的充分性，我们需要对比模糊测试结果与仅使用原始数据测试结果。首先使用 KITTI 数据集中的 Velodyne point clouds (29 GB) 点云数据以及 Training labels of object data set(5 MB) 点云标签数据，将 KITTI 数据集中带标签的 7481 个点云文件分为训练集（5237 个）和测试集（2244 个），然后使用训练集对 PointPillars 进行训练（steps 为 99040，epochs 为 50），得到训练好的模型 *P*，再使用测试集作为原始测试数据集 *T*，数据集 *T* 的测试结果为对照结果。紧接着，上传数据集 *T* 和模型 *P*，并选择所有的变异算子进行模糊

测试，最终得到原始对照数据集以及每个变异算子生成的评估数据集对模型 P 的评估结果。

6.3.6 实验结果

表 6-7: 变异算子有效性评估统计表

评估数据集	相同 (个)	相似 (个)	不同 (个)
UNA 实验组 D_{UNA}	708	258	34
UPD 实验组 D_{UPD}	686	275	39
NNA 实验组 D_{NNA}	699	265	36
RRNA 实验组 D_{RRNA}	840	142	18
RRPD 实验组 D_{RRPD}	868	119	13

实验一：表 6-7所示为变异算子有效性评估统计表，该表总结了实验二中所有 10 名同学调查问卷最终统计分析结果。从表中可以看出，所有变异算子生成的测试数据集中，问卷调查结果为相同或相似的点云数据比例均超过 95%，只有少量点云数据与原始数据出现较大差别。由此可得，变异算子生成的测试数据集中绝大部分点云数据与原始数据集中的点云数据是语义相同或相似的，因此，本文所述变异算子是有效的。

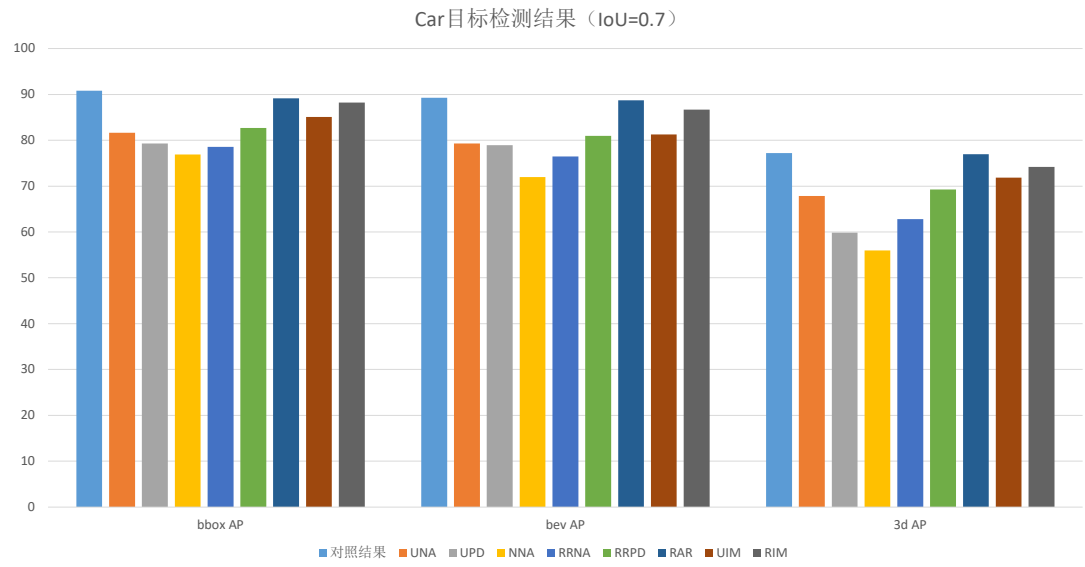

图 6-1: 实验二 Car 目标检测结果

实验二：图 6-1、图 6-2以及图 6-3分别为实验二模型测试的汽车、行人以及自行车目标检测结果柱状图，每个图分别总结了实验二中原始对照数据集以及每个变异算子生成的测试数据集对模型 *P* 的评估结果。为了直观表示实验结果，图 6-1的汽车（Car）选取 IoU 为 0.7 的评估结果 *AP* 值，图 6-3的行人（Pedestrain）和图 6-2的自行车（Cyclist）则选取 IoU 为 0.5 的评估结果 *AP* 值。从图 6-1可以得出，汽车类型测试结果影响相对较小，只有 UNA、UPD、NNA 和 RRNA 测试数据集导致模型 *AP* 值出现不同程度下降，主要是在 3*d AP* 中，中下降较为明显。而图 6-2结果显示，相比汽车类型，模型在自行车类型的测试结果 AP 值出现了更为明显的下降，但主要变化仍旧集中于 UNA、UPD、NNA 和 RRNA 测试数据集。另外，如图 6-3所示，模型在行人类型的测试结果受变异测试数据集影响最大，特别在 3*d AP* 中，UPD 测试结果相比对照结果甚至下降了超过 50%。以上测试结果表明，模型对于点云增删的扰动无法有效处理，其中，汽车类型的效果受影响最小，而行人类型的效果受影响最大。并且，小范围内的点云增删对模型准确率也有一定的影响。另外，模型对于坐标轴旋转以及点云强度的变化的可靠性更好。由此可得，本文提出的模糊测试技术能有效揭露模型存在的数据偏见问题，这些问题可能导致模型出现致命错误。因此本文提出的自动驾驶软件激光雷达模糊测试是有效的。

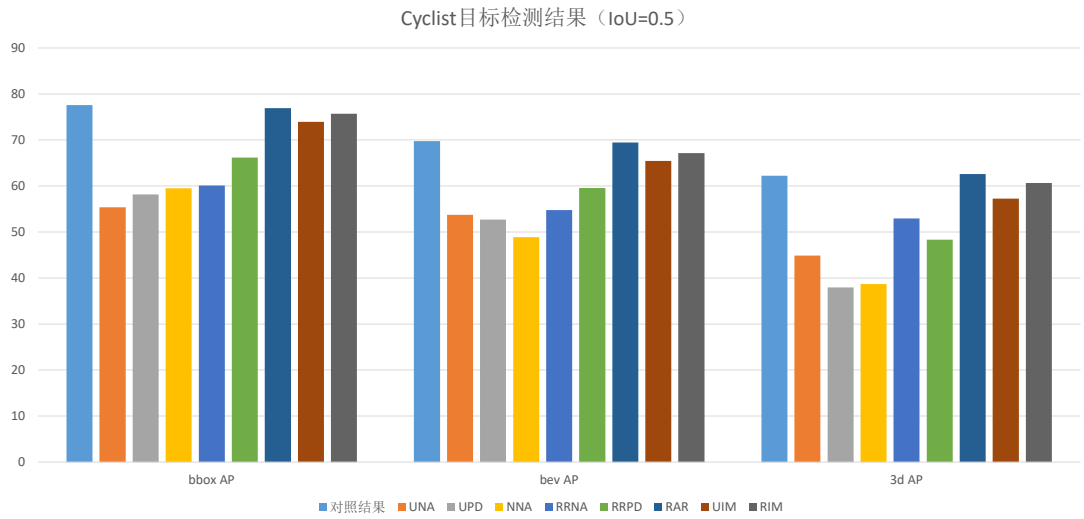

图 6-2: 实验二 Cyclist 目标检测结果

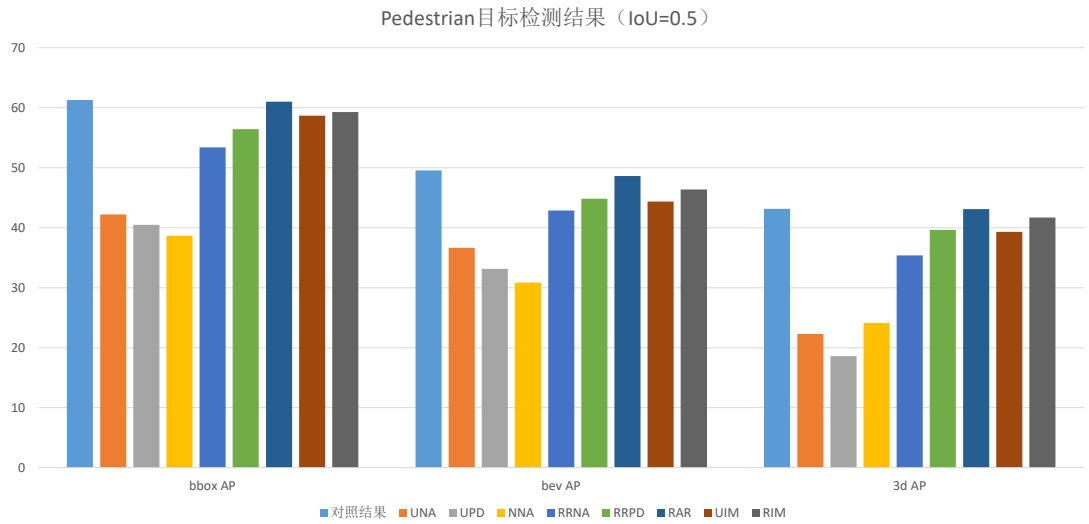

图 6-3: 实验二 Pedestrain 目标检测结果

6.4 本章小结

本章主要介绍自动驾驶软件激光雷达模糊测试系统的系统测试以及实验评估，首先基于系统的需求分析以及详细设计，进行功能性测试，并展示测试结果。紧接着，为了验证本文提出的模糊测试技术以及变异算子的有效性，提出两个研究问题：（1）本文提出的自动驾驶软件激光雷达模糊测试是否有效？（2）本文提出的变异算子是否有效？然后，通过设置并开展两个实验来对以上问题进行研究。实验一通过让同学对比可视化后的变异点云数据与原始点云数据，并填写调查问卷，最终统计结果表明绝大部分点云数据与原始数据集中的点云数据是语义相同或相似的，这表明，本文设计的变异算子是有效的。实验二以点云目标检测网络 PointPillars 为实验对象进行评估实验，设置评估指标，通过模糊测试结果可以推断，本文提出的自动驾驶软件激光雷达模糊测试是有效的。

第七章　总结与展望

7.1　总结

近年来，自动驾驶受到越来越多的关注。在自动驾驶中，激光雷达传感器在所有环境传感器中扮演着越来越重要的角色。随着越来越多激光雷达被安装到自动驾驶车辆上，激光雷达的安全性和稳定性也受到越来越多的关注，如何有效对自动驾驶激光雷达进行有效全面的测试成为研究的关键点。本文引入模糊测试的思想，通过分析自动驾驶激光雷达点云数据的特征并结合实际场景信息，提出针对自动驾驶场景的激光雷达点云数据变异算子，并基于此设计并实现了一个自动驾驶软件的激光雷达模糊测试系统。具体来说，本文从激光雷达点云的数量、坐标、强度以及存储格式等特性出发，设计了 8 个自动驾驶场景激光雷达点云特殊变异算子。随后，基于 Spring Boot 框架以及 PCL 点云代码库实现了一个自动驾驶软件的激光雷达模糊测试系统。该系统使用 PCL 库处理点云数据，基于设计的点云特殊变异算子进行测试用例生成，并对测试模型进行自动化部署及评估，最终生成测试报告，并且提供了点云可视化功能以及简洁可用的图形用户界面，具备较强的可用性。

本文基于系统的需求分析及设计对系统进行功能测试，以保证系统的可用性。紧接着基于点云目标检测网络 PointPillars 对本系统进行实证研究，进一步验证本文提出的自动驾驶软件的激光雷达模糊测试技术及其变异算子的有效性，并对以下两个问题进行探索：（1）本文提出的变异算子是否有效？（2）本文提出的自动驾驶软件激光雷达模糊测试是否有效？通过两个实验进行验证，实验一结果表明，变异算子生成的测试数据集大多都处于语义相同或模糊的范围，只有少量点云数据与原始数据出现较大差别，说明本文提供的模糊测试变异算子生成的测试用例与原始对照数据大部分是语义相同的，本文所述变异算子是有效的。在实验二中，大部分模糊测试算子生成的测试用例集评估激光雷达目标检测模型得到的 AP 值要低于原始数据集的评估结果 AP 值，说明本文提出的模糊测试能够揭露模型存在的问题，因此该模糊测试是有效的，能够提高自动驾驶软件激光雷达测试的充分性。

7.2 展望

本文设计开发的自动驾驶软件的激光雷达模糊测试系统主要职责是对自动驾驶软件的激光雷达目标检测模型进行测试，虽然起到了一定的作用，但从长远考虑，对此的研究还有很多改进和提高的空间。主要从一下方面开展后续工作：

第一，由于时间问题，本文在实验一只使用了 PointPillars 三维目标检测网络进行实验。为了更好的证明本文提出的测试技术的有效性，还可以使用 VoteNet、SECOND 以及 Part-A2 等网络进行验证，从而提高说服力。另外，实验二是通过邀请同学对数据真实性进行判断，后面我们将尝试邀请领域专家对我们的模糊测试算子有效性进行判断，提高实验的可靠性。

第二，本文提供的变异算子虽然能使得激光雷达目标检测模型评估结果的准确率出现下降，但缺少有效的评估结果回馈机制来引导变异算子的变异方向，从而进一步提高测试的效率以及命中率。在传统软件的模糊测试中，一般通过代码覆盖率作为测试的回馈机制，但代码覆盖率在深度学习模型中是没有太大的参考意义的。在查阅大量资料后，我们了解到可以利用神经元覆盖率来指导变异算子的变异方向，但由于这类实现属于白盒测试，需要对激光雷达神经网络进行深入学习，这是一个长期研究工作，我们将在后续实验中进行研究尝试。

第三，本文虽然提供了 8 个自动驾驶激光雷达点云数据变异算子，但相比于自动驾驶十分广大的输入空间，依旧是不够的。在后续实验中，我们将从感兴趣区域（ROI）内外的点云添加删除以及目标模版植入变换等方面持续加入新的变异算子，提高测试的充分性。

致　　谢

时光如梭，转眼间在硕士研究生生活就要落下帷幕，在论文完成之际，我想借此机会向我的老师、同学、家人以及朋友表达我由衷的感谢。

首先，我要感谢我的导师陈振宇老师以及指导老师冯洋老师，感谢他们为我的研究指明了方向，并在我的研究生期间不断给予我帮助。陈振宇导师经验丰富，在我的论文选题期间为我提供了很大的帮助。指导老师冯洋也对我的论文提出了许多建议，让我能够及时修改论文中的问题。感谢他们的悉心教导，我才能顺利完成我的毕业论文。同时也十分感谢其他老师在我研究生期间教会了我很多知识，他们对待教学认真负责的态度给我留下了深刻的影响。

其次，我要感谢我实验室的同学们，感谢他们在我遇到困难时对我热心的帮助和积极的鼓励。同时，十分感谢我的室友，感谢他们在学习和生活中给予我的帮助，也感谢他们为我的研究生生活增添了许多快乐。

最后，我要感谢我的家人，特别是我的父母，感谢他们对我的养育之恩，也感谢他们一直以来对我的支持和鼓励。同时，我也要感谢我的女朋友，感谢她一直以来对我的关心、理解和支持，在我遇到困难时，总是能在背后给予我支持，谢谢！

参考文献

[1] YUE X, WU B, SESHIA S A, et al. A LiDAR Point Cloud Generator: from a Virtual World to Autonomous Driving[J/OL]. CoRR, 2018, abs/1804.00103. http://arxiv.org/abs/1804.00103.

[2] LI J, ZHAO B, ZHANG C. Fuzzing: a survey[J/OL]. Cybersecur., 2018, 1(1): 6. https://doi.org/10.1186/s42400-018-0002-y.

[3] GEIGER A, LENZ P, STILLER C, et al. Vision meets robotics: The KITTI dataset[J/OL]. Int. J. Robotics Res., 2013, 32(11): 1231 – 1237. https://doi.org/10.1177/0278364913491297.

[4] VEIT M, CAPOBIANCO A. Go'Then'Tag: A 3-D point cloud annotation technique[C/OL] // LÉCUYER A, LINDEMAN R, STEINICKE F. IEEE Symposium on 3D User Interfaces, 3DUI 2014, Minneapolis, MN, USA, March 29-30, 2014. [S.l.]: IEEE Computer Society, 2014: 193 – 194. https://doi.org/10.1109/3DUI.2014.6798886.

[5] WONG Y, CHU H, MITRA N J. SmartAnnotator: An Interactive Tool for Annotating RGBD Indoor Images[J/OL]. CoRR, 2014, abs/1403.5718. http://arxiv.org/abs/1403.5718.

[6] BOYKO A, FUNKHOUSER T A. Cheaper by the dozen: group annotation of 3D data[C/OL] // BENKO H, DONTCHEVA M, WIGDOR D. The 27th Annual ACM Symposium on User Interface Software and Technology, UIST '14, Honolulu, HI, USA, October 5-8, 2014. [S.l.]: ACM, 2014: 33 – 42. https://doi.org/10.1145/2642918.2647418.

[7] KAPOOR A, GRAUMAN K, URTASUN R, et al. Active Learning with Gaussian Processes for Object Categorization[C/OL] // IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007.

[S.l.]: IEEE Computer Society, 2007: 1 – 8.
https://doi.org/10.1109/ICCV.2007.4408844.

[8] GROUEIX T, FISHER M, KIM V G, et al. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation[J/OL]. CoRR, 2018, abs/1802.05384.
http://arxiv.org/abs/1802.05384.

[9] QI C R, SU H, MO K, et al. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation[J/OL]. CoRR, 2016, abs/1612.00593.
http://arxiv.org/abs/1612.00593.

[10] MEGIE G. Laser Remote Sensing: Fundamentals and Applications[J/OL]. Eos, Transactions American Geophysical Union, 1985, 66(40): 686 – 686.
https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/EO066i040p00686-05.

[11] MANÈS V J M, HAN H, HAN C, et al. The art, science, and engineering of fuzzing: A survey[J]. IEEE Transactions on Software Engineering, 2019.

[12] LCAMTUF. cross_fuzz[EB/OL]. [2021/3/25].
https://lcamtuf.coredump.cx/cross_fuzz/.

[13] M.SECURITY. funfuzz[EB/OL]. [2021/3/25].
https://github.com/MozillaSecurity/funfuzz.

[14] GRIECO G, CERESA M, BUIRAS P. QuickFuzz: An automatic random fuzzer for common file formats[J]. ACM SIGPLAN Notices, 2016, 51(12): 13 – 20.

[15] BRUBAKER C, JANA S, RAY B, et al. Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations[C] // 2014 IEEE Symposium on Security and Privacy. 2014: 114 – 129.

[16] SOMOROVSKY J. Systematic fuzzing and testing of TLS libraries[C] // Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 1492 – 1504.

[17] KOSCIELNIAK B J. DETECTION OF TIMING SIDE-CHANNELS IN TLS[J], .

[18] SPENSKY C, HU H. Llfuzzer[EB/OL]. [2021/3/25]. https://github.com/mit-ll/LL-Fuzzer.

[19] DEWEY K, ROESCH J, HARDEKOPF B. Language fuzzing using constraint logic programming[C] // Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. 2014 : 725 – 730.

[20] HOLLER C, HERZIG K, ZELLER A. Fuzzing with code fragments[C] // 21st {USENIX} Security Symposium ({USENIX} Security 12). 2012 : 445 – 458.

[21] YANG D, ZHANG Y, LIU Q. Blendfuzz: A model-based framework for fuzz testing programs with grammatical inputs[C] // 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. 2012 : 1070 – 1076.

[22] ZALEWSKI M. American Fuzzy Lop[EB/OL]. [2021/3/26]. http://lcamtuf.coredump.cx/afl/.

[23] GOOGLE. honggfuzz[EB/OL]. [2021/3/26]. https://github.com/google/honggfuzz.

[24] A.HELIN. radamsa[EB/OL]. [2021/3/26]. https://github.com/aoh/radamsa.

[25] JHA S, BANERJEE S S, TSAI T, et al. ML-based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection[J/OL]. CoRR, 2019, abs/1907.01051. http://arxiv.org/abs/1907.01051.

[26] ZHU F, MA L, XU X, et al. Baidu Apollo Auto-Calibration System - An Industry-Level Data-Driven and Learning based Vehicle Longitude Dynamic Calibrating Algorithm[J/OL]. CoRR, 2018, abs/1808.10134. http://arxiv.org/abs/1808.10134.

[27] 冯洋, 夏志龙, 郭安, et al. 自动驾驶软件测试技术研究综述 [J]. 中国图象图形学报, 2021, 26(1) : 13 – 27.

[28] WANG J, ALIPOURI Y, HUANG B. Dual Neural Extended Kalman Filtering Approach for Multirate Sensor Data Fusion[J/OL]. IEEE Trans. Instrum. Meas., 2021, 70 : 1 – 9.
https://doi.org/10.1109/TIM.2020.3041825.

[29] MAGNIER V, GRUYER D, GODELLE J. Automotive LIDAR objects detection and classification algorithm using the belief theory[C/OL] // IEEE Intelligent Vehicles Symposium, IV 2017, Los Angeles, CA, USA, June 11-14, 2017. [S.l.] : IEEE, 2017 : 746 – 751.
https://doi.org/10.1109/IVS.2017.7995806.

[30] WANG H, WANG B, LIU B, et al. Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle[J/OL]. Robotics Auton. Syst., 2017, 88 : 71 – 78.
https://doi.org/10.1016/j.robot.2016.11.014.

[31] NAVARRO P J, FERNÁNDEZ-ISLA C, BORRAZ R, et al. A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data[J/OL]. Sensors, 2017, 17(1) : 18.
https://doi.org/10.3390/s17010018.

[32] FILGUEIRA A, GONZáLEZ-JORGE H, LAGüELA S, et al. Quantifying the influence of rain in LiDAR performance[J/OL]. Measurement, 2017, 95 : 143 – 148.
https://www.sciencedirect.com/science/article/pii/S0263224116305577.

[33] HADJ-BACHIR M, DE SOUZA P. LIDAR sensor simulation in adverse weather condition for driving assistance development[H/OL]. 2019.
https://hal.archives-ouvertes.fr/hal-01998668.

[34] GOODIN C, CARRUTH D, DOUDE M, et al. Predicting the Influence of Rain on LIDAR in ADAS[J/OL]. Electronics, 2019, 8(1).
https://www.mdpi.com/2079-9292/8/1/89.

[35] VAN SPRUNDEL I. Fuzzing: Breaking software in an automated fashion[J], 2005.

[36] RAWAT S, JAIN V, KUMAR A, et al. VUzzer: Application-aware Evolutionary Fuzzing[C/OL] // 24th Annual Network and Distributed System Security Sym-

posium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017. [S.l.] : The Internet Society, 2017.
https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/vuzzer-application-aware-evolutionary-fuzzing/.

[37] LUK C, COHN R S, MUTH R, et al. Pin: building customized program analysis tools with dynamic instrumentation[C/OL] // SARKAR V, HALL M W. Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation, Chicago, IL, USA, June 12-15, 2005. [S.l.] : ACM, 2005 : 190 – 200.
https://doi.org/10.1145/1065010.1065034.

[38] WANG J, CHEN B, WEI L, et al. Skyfire: Data-Driven Seed Generation for Fuzzing[C/OL] // 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017. [S.l.] : IEEE Computer Society, 2017 : 579 – 594.
https://doi.org/10.1109/SP.2017.23.

[39] GODEFROID P, PELEG H, SINGH R. Learn&Fuzz: Machine Learning for Input Fuzzing[J/OL]. CoRR, 2017, abs/1701.07232.
http://arxiv.org/abs/1701.07232.

[40] RAJPAL M, BLUM W, SINGH R. Not all bytes are equal: Neural byte sieve for fuzzing[J/OL]. CoRR, 2017, abs/1711.04596.
http://arxiv.org/abs/1711.04596.

[41] PEI K, CAO Y, YANG J, et al. Deepxplore: Automated whitebox testing of deep learning systems[C] // proceedings of the 26th Symposium on Operating Systems Principles. 2017 : 1 – 18.

[42] JOVANOVIC Z, JAGODIC D, VUJIčIć D, et al. Java Spring Boot Rest WEB Service Integration with Java Artificial Intellgence Weka Framework[C] // . 2017.

[43] GUTIERREZ F. Spring Boot, Simplifying Everything[M/OL] // . 2014 : 263 – 276.
http://dx.doi.org/10.1007/978-1-4302-6533-7_19.

[44] RAD B B, BHATTI H J, AHMADI M. An Introduction to Docker and Analysis of its Performance[J]. IJCSNS, 2017, 17(3): 228.

[45] RUSU R B, COUSINS S. 3D is here: Point Cloud Library (PCL)[C/OL] // IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011. [S.l.]: IEEE, 2011. https://doi.org/10.1109/ICRA.2011.5980567.

[46] 张健, 李新乐, 宋莹, et al. 基于噪声点云的三维场景重建方法 [J]. 计算机工程与设计, 2020, 4.

[47] ZHOU Z Q, SUN L. Metamorphic testing of driverless cars[J/OL]. Commun. ACM, 2019, 62(3): 61–67. https://doi.org/10.1145/3241979.

[48] FILGUEIRA A, GONZÁLEZ-JORGE H, LAGÜELA S, et al. Quantifying the influence of rain in LiDAR performance[J]. Measurement, 2017, 95: 143–148.

[49] KRUCHTEN P. The 4+1 View Model of Architecture[J/OL]. IEEE Softw., 1995, 12(6): 42–50. https://doi.org/10.1109/52.469759.

[50] GEIGER A, LENZ P, URTASUN R. Are we ready for autonomous driving? the kitti vision benchmark suite[C] // 2012 IEEE Conference on Computer Vision and Pattern Recognition. 2012: 3354–3361.

[51] LANG A H, VORA S, CAESAR H, et al. PointPillars: Fast Encoders for Object Detection from Point Clouds[J/OL]. CoRR, 2018, abs/1812.05784. http://arxiv.org/abs/1812.05784.

学位论文原创性声明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：_____

_____年_____月_____日

《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名：_____

_____年____月____日

论文题名	自动驾驶软件的激光雷达模糊测试系统设计与实现				
研究生学号	MF1932245	所在院系	软件学院	学位年度	2021
论文级别	☐ 硕士　☐ 硕士专业学位 ☐ 博士　☐ 博士专业学位 （请在方框内画勾）				
作者 Email	mf1932245@smail.nju.edu.cn				
导师姓名	陈振宇 教授，冯洋 助理研究员				

论文涉密情况：

☐ 不保密

☐ 保密，保密期 (_____年_____月_____日 至 _____年_____月_____日)

注：请将该授权书填写后装订在学位论文最后一页（南大封面）。