DOI:10.7544/issn1000-1239.20211139
卷(期):起止页,年

自动驾驶系统中视觉感知模块的安全测试

所属类别：工业互联网安全
稿件编号：2021-11139

吴 昊[1] 王 浩[1] 苏 醒[1] 李明昊[1] 许封元[1] 仲 盛[1]

（计算机软件新技术国家重点实验室（南京大学） 南京 210023）

（hako.wu@smail.nju.edu.cn）

Security Testing of Visual Perception Module in Autonomous Driving System

Wu Hao[1], Wang Hao[1], Su Xing[1], Li Minghao[1], Xu Fengyuan[1], and Zhong Sheng[1]

（*State Key Laboratory for Novel Software Technology（Nanjing University）*, *Nanjing* 210023）

Abstract In recent years, developments of visual perception techniques based on deep learning have significantly promoted the prosperity of autonomous driving in Internet of vehicles scenarios. However, frequent security issues of autonomous driving systems have raised concerns about the future of autonomous driving. Since the behaviors of deep learning systems lack interpretability, testing the robustness of autonomous driving systems based on deep learning is challenging. The existing efforts on security testing for autonomous driving have limitations in scene description, security defect detection, and defect interpretation. Aiming at testing the security of the visual perception module of autonomous driving, we design and implement a scene-driven security testing system. Our work proposes a flexible scene description method that balances authenticity and richness. We utilize the real-time rendering engine to generate scenes for autonomous driving security testing. We design an efficient scene search algorithm for nonlinear systems that dynamically schedules search plans based on the testing feedback. We also design a failure analyzer to profile the cause of security issues automatically. We reproduce the latest dynamic automatic driving testing system, which is based on the real-time rendering engine, and test the CILRS system and CIL system with our system and the SOTA system. The experimental results show that our work's failure discovery rate is 1.4 times that of the SOTA scene-driven dynamic testing system in the same amount of time. Further experiments show that our system can find 1,939 and 1,671 scenes through 3,000 dynamically-searched scenes, respectively, which cause security issues in the CIL and CILRS system's visual perception module. The searched scenes are in three environments: the fields, the country, and the city, the average search time for each failure-causing scene is 16.86s. Our analyzer determined from a statistical perspective that the areas where the CILRS system is prone to cause failures are on both sides of the road, and rainy weather and red or yellow objects are more likely to lead to failures.

Key words Internet of vehicles; visual perception module; deep learning security; black-box testing; scene search

摘要 近年来，基于深度学习的视觉感知技术的发展极大地促进了车联网领域中自动驾驶的繁荣，然而自动驾驶系统的安全问题频出引发了人们对自动驾驶未来的担忧.由于深度学习系统的行为缺乏可解释性，测试基于深度学习的自动驾驶系统的安全性极具挑战性.目前已有针对自动驾驶场景的安全性测试工作被提出，但这些方法在测试场景生成、安全问题检测和安全问题解释等方面仍存在不足之处.针对基于视觉感知的自动驾驶系统，设计开发了一种场景驱动的、可解释强的、运行高效的安全性测试系统.提出了一种能够平衡真实性与丰富度的场景描述方法，并利用实时渲染引擎生成可以用于驾驶系统安全性测试的场景；设计了一种高效的针对

收稿日期：2021-11-16；**修回日期**：2021-12-22
基金项目：国家重点研发计划项目 2021YFB3100300；国家自然科学基金项目 61872180；江苏省“双创计划”；江苏省“六大人才高峰”高层次人才项目
This work was supported by National Key Research and Development Program of China 2021YFB3100300, National Natural Science Foundation of China 61872180, Jiangsu “Shuang-Chuang” Program , and Jiangsu “Six-Talent-Peaks” Program.
通信作者：许封元（fengyuan.xu@nju.edu.cn）

非线性系统的场景搜索算法，其可以针对不同的待测试系统动态调整搜索方案；同时，还设计了一个故障分析器，自动化分析定位待测试系统的安全性缺陷成因.复现了现有基于实时渲染引擎的动态自动驾驶测试系统，并同时使用本系统和复现系统对 CILRS 系统和 CIL 系统进行安全测试，实验结果表明相同时间下本工作的安全问题发现率是复现的场景驱动的动态测试方法的 1.4 倍.进一步的实验表明，可以分别为具有代表性的深度学习自动驾驶系统 CIL 和 CILRS，从旷野、乡村与城市的 3 类环境中动态生成的共 3 000 个场景中，搜索到 1 939 和 1 671 个造成故障的场景，并且每个故障场景的搜索时间平均为 16.86 s.分析器从统计的角度判断出 CILRS 系统容易造成故障的区域在道路两侧，雨天和红色或黄色物体更易造成该自动驾驶系统发生故障.

关键词 车联网；视觉感知模块；深度学习安全性；黑盒测试；场景搜索

中图法分类号 TP391

车联网领域随着物联网与交通运输领域的深度融合蓬勃发展.随着深度学习的进步，车联网领域的自动驾驶技术得到了突破性的发展，并有演化成一场新的汽车工业革命的趋势[1,2,3,4].无论是特斯拉、蔚来等新型车企，还是福特、宝马等传统车企都陆续拿到了自动驾驶路测牌照，着眼于研发深度自动驾驶技术.迅猛发展的深度自动驾驶技术正逐渐成为车联网领域的主要支撑技术之一，正在改变未来的交通和出行方式.

视觉感知模块是自动驾驶进行环境感知的重要组件，也是车辆进行智能决策的重要基础[5].自动驾驶领域的重要企业特斯拉更是将视觉感知模块作为其驾驶系统的唯一环境感知模块.因此，自动驾驶系统视觉感知模块的安全性是自动驾驶系统正常工作的关键.虽然视觉感知模块的表现随着深度视觉技术的发展稳步提升，但是其从驾驶环境中感知到的特征语义难被理解、决策过程无法解释.如何对自动驾驶系统视觉感知模块的安全性进行充分测试，已经成为了一个迫在眉睫、亟待解决的问题[6].

诚然，围绕深度学习可解释性方面的工作有了一定的突破，但是距离分析清楚自动驾驶视觉感知模块的错误传导机理还有较远的距离.近年来，神经网络的黑盒攻击方法[7,8]的进步，启发大家提出了一些基于场景搜索的自动驾驶视觉感知模块安全性测试技术.这些场景驱动的测试方法利用黑盒测试的思路，为驾驶系统提供尽可能多的驾驶场景数据，观察自动驾驶系统的输出与测试预言（Test Oracle）之间的差异，进而分析自动驾驶系统视觉感知模块的安全性.

我们认为场景驱动的黑盒安全测试是在弄清深度学习可解释性之前，对于视觉感知模块安全性最为重要的测试手段.但目前对于将生成场景应用于视觉感知模块的测试，我们仍面临着 3 个挑战：

1）**平衡场景描述的真实性与丰富度**. 场景生成规则是场景驱动的测试系统的重要基础.保守的规则设计会造成场景覆盖能力不足；而过于灵活的规则设计又会破坏物体相对关系，伤害场景真实性.探究一个能同时兼顾场景真实性和丰富度[9,10]的场景生成规则极具挑战性.

2）**保证搜索算法的高效性和稳定性**.单个物体的属性（如颜色、形状）与物体间的相互关系（如位置、方向）组合十分复杂.为了能够高效稳定地测试系统安全性缺陷，需要：①动态地针对不同自动驾驶系统视觉感知模块产生个性化的场景搜索方案，以保证搜索过程的步骤较少；②同时尽可能缩短单步搜索的时间.

3）**解释测试结果的精确性与自动化**.以往测试系统需要介入人工分析缺陷原因.若想做到自动化地分析测试结果，系统需要能够精细地操纵场景中每个元素来定位系统安全性缺陷的成因.

围绕场景驱动的视觉感知模块的黑盒安全测试，学术界已经有了初步探索.在场景测试方法当中，基于实时渲染引擎的一系列测试方法由于其场景生成的灵活性受到了广泛关注.最初，基于实时渲染引擎的场景驱动的安全测试采用的是基于预定场景的测试思路.其中的代表性工作是 CARLA 0.8.X [11]，该工作使用 Unreal Engine 创建用以测试系统的固定驾驶线路.接着，Scenic[12]等人提出了一种场景生成的编程接口，以使此类测试程序更加系统化，奠定了静态的基于场景测试的基础.然而，其模拟环境较为固定，缺乏动态行为，同时对于非实体对象（如天气）的描述缺少自由度.基于实时渲染引擎的场景驱动的安全测试的最新工作，Paracosm[13]在 Scenic 的基础上，提出了基于随机搜索的动态场景生成方法来进行视觉感知模块的安全测试.由于该动态场景搜索方法相对简单，对待测试视觉感知模块的适应性不足，因此，安全问题的搜索过程不够高效.本文在前述工作的基础上，提出了一种基于结果反馈的动态场景搜索算法，从而提高了自动驾驶系统中视觉感知模块的安全

问题测试效率.更多细节将在第 1.5 节进行详细介绍.

本工作中，类比深度学习黑盒攻击策略，借助实时渲染引擎的开放度，设计并提出了一套可靠且具有可解释性的自动驾驶视觉感知模块安全性测试系统.主要的贡献有 3 方面：

1）本工作针对车联网场景下自动驾驶系统感知模块的安全问题，提出并设计了一套场景驱动的黑盒安全性测试系统.与现有工作相比，该系统引入基于结果反馈的动态化测试策略，通过自适应机制在循环中不断调整输入数据的生成，实现对感知模块高效、稳定的非入侵式安全测试.

2）面向自动驾驶视觉感知模块的动态测试新需求，本工作设计了细粒度场景描述方法、适应性动态搜索算法和自动化系统缺陷分析的动态测试技术，从测试粒度、反馈自适和可解释性 3 个方面对提出的黑盒安全性测试系统进行了优化.

3）本工作以两个代表性的开源自动驾驶系统为对象开展了测试系统的验证，分别动态生成了 3,000 个场景，并各自找到了 1,939 和 1,671 个故障场景，平均每 16.86s 可发现一个故障场景.实验表明，得益于动态自适的场景搜索方法，本工作的安全问题发现率是现有基于实时渲染引擎的场景驱动的动态测试方向中最好工作的 1.4 倍.

1 相关工作

1.1 基于神经元覆盖的自动驾驶测试

神经元覆盖是类比传统程序的分支覆盖而设计的.此类工作定义当测试输入经过神经元使神经元输出满足某种状态时，该神经元则称被测试样例覆盖（激活）了.此类工作，以最大化神经元覆盖为优化目标，来寻找输入样例.自 DeepXplore[14]引入神经元覆盖的概念并成功应用于基于视觉的深度自动驾驶领域后，已有大量关于神经元覆盖的工作出现，提出了各式各样的覆盖标准，并成功使传统软件测试方法如蜕变测试、模糊测试、符号测试等迁移到深度学习测试任务中[15-22].然而，这种类比属于一种机械类比，神经元的输出值的状态与传统软件测试中的分支是完全不同的概念，因而这种类比方法的有效性始终受到人们质疑[23].并且，基于神经元覆盖的设计也难以给出基于语义的测试样例故障原因，不利于自动驾驶系统安全性的进一步提升.

1.2 基于错误注入的自动驾驶测试

AVFI[24]使用软件故障注入模拟自动驾驶系统的硬件故障，以测试系统的容错性.随后 DriveFI[25]使用 Apollo 和 DriveAV 在 CARLA 模拟器和 DriveSim 上进行实验，利用贝叶斯网络模拟自动驾驶系统加速错误注入后的验证过程，能最大限度地挖掘出影响自动驾驶系统的故障.Kayotee[26]在上述工作的基础上，添加了描述误差传播的能力，并且能直接注入 CPU 和 GPU 的位翻转.这些工作是从容错性的角度考察自动驾驶系统的特性的，而且实际考察的是自动驾驶系统的硬件故障.这类工作与本工作讨论的场景和问题是正交的，本工作研究的是自动驾驶系统软件，特别是其视觉感知模块的安全性.

1.3 基于搜索的自动驾驶测试

基于搜索的自动驾驶安全性测试的核心思路如下：给定待测自动驾驶系统的输入空间，定义待测系统的特殊输入状态，通过在输入空间搜索，确定哪些输入会造成系统输出特殊状态，从而实现对输入空间的划分.Abdessalem 利用搜索算法在输入空间给输入打上标签，同时利用标签数据训练分类器，对输入空间进行了决策边界的划分[27,28].随后，他把对少体问题的状态空间的搜索过程扩展到多体问题状态空间中的搜索，提出了 FITEST 搜索测试方法[29].这样的方法的局限性在于，分类器的引入隐含假定了输入空间是局部连续线性的，例如使用决策树进行划分时，潜在认为了两个正例状态之间也是正例状态[21].对于 AEB 系统这样问题定义在线性域下的系统，分类器的设计是合理的，但是对于具有高度非线性系统的深度学习系统，这样的方法显然是不适用的.除此之外，Wicker 利用两者博弈的想法[30]，通过操纵图片上的像素点，利用蒙特卡洛树搜索博弈的渐进最优策略，来寻找造成模型出错的反例.这样的搜索策略的搜索空间是像素级的，搜索结果不具备在现实中的真实性.

1.4 基于真实数据的测试方法

基于真实数据的测试方法主要分为两种，一种是通过收集大量的用户驾驶数据来改善其自动驾驶系统的质量，如 Tesla[31]；另一种是实景测试[32]，在真实的公路环境下使用原型车辆进行测试，考虑到安全因素，该方法的测试条件比较严格[33].这些方法除了数据收集的成本较高外，收集到的数据分布也十分有限，这使得测试系统无法检测出自动驾驶系统在新环境中的安全性；同时过度收集的驾驶数据也存在侵犯用户隐私的问题.

1.5 基于生成数据的测试方法

基于生成数据的测试方法主要分为两种，一种是基于生成式对抗网络 [34-39]（generative adversarial networks，GAN），在 DeepRoad[34]中，考虑将正常天气的道路场景变换到雨雪天气下，从而测试系统在雨雪天气下的安全性，但其生成方法的场景丰富度存

在不足，也无法实现场景内容的灵活控制；另外一种是基于实时渲染引擎创建测试场景.Richer 利用游戏 GTA-V 来创建自动驾驶数据集[40,41],Sythia[42]使用 Unity 引擎合成数据集，CARLA 0.8.X [11]使用 Unreal Engine 创建用于测试系统的驾驶线路.这些工作的问题在于，测试的场景是预先定义好的，不能或只能很有限地对场景进行调整，这样的工作没有充分利用实时渲染可以动态调整场景的特点.特别是，使用固定的驾驶场景进行测试，很难对所有可能的场景进行覆盖.

Scenic[12]设计了一种场景描述语言，可以根据预定义规则生成一些场景用于自动驾驶视觉感知模块的安全性测试.这种描述语言尽管有很高的自由度，但对场景中非实体的对象刻画，如天气和太阳，比较困难，并且难以应用到场景变换的情况.最新的工作是 Paracosm[13]提出了一种可编程的自动驾驶测试场景生成方法.该工作通过将场景中的物体和环境参数化并提供了一套编程接口用于测试场景生成，该方法基于随机搜索的方式进行场景生成并对自动驾驶中视觉感知模块进行测试.但是考虑到可搜索的参数空间规模十分巨大，因此，很难通过随机搜索高效地找到感知模块的安全问题.对此我们在 4.1 节进行了详细的比较和评估.

本工作延续了基于生成数据的测试方法的研究思路，基于实时渲染引擎来生成真实度高的场景来进行系统的安全性测试.相比于以往的工作，我们有 2 个显著改进，首先是采用了更加灵活丰富的场景描述方式，相比于 Scenic 方法，本方法增添了对非实体对象的描述，使得自动驾驶视觉感知模块在不同天气下的安全性也得到充分测试.第 2 个改进是提出了一套适应性场景搜索算法，相比于 Paracosm 方法，本文可以实现适应性的动态故障场景搜索，使得自动驾驶视觉感知模块的安全问题发现效率有了显著提升.

2 系统设计

本节将介绍针对自动驾驶视觉感知模块的安全测试系统的具体设计，其中包括安全性测试系统的形式化描述、工作流程、场景描述方法、动态场景生成器以及缺陷分析方法.在此之前，我们在表 1 中汇总了本文所用的一些关键变量.

Table 1 Summary of Notations in Our Approach

表 1 本文方法用到的变量说明

术语	符号	说明
转向角	s	控制指令的转向角输出
油门	t	控制指令的油门输出
场景	w	搜索系统搜索的具体场景
自动驾驶模型	m	自动驾驶系统模型
示性函数	δ	判断动作是否出错的函数
环境	E	环境的参数化表示
天气	Q	天气的具名元组表示
实体	D	实体对象的参数化表示
静态物品	G	实体对象
场景参数	$W_E(Q,D)$	场景参数化表示
误差允许上限	ε	蜕变测试松弛度

2.1 安全性测试系统的形式化描述

自动驾驶系统本质上是一个策略π，在给定环境信息序列 O 的情况下映射到控制指令 A 上，即 $\pi: O \to A$.其中输入的环境信息包含相机图像 I 和当前车速 V；输出控制指令包含刹车、油门和转向角，其中刹车可以被看成反向油门，这样输出的控制指令可表示为(s,t).

基于场景搜索的测试系统首先会生成一个具体的场景 $w \in W$，该场景所表示的环境信息为 $o = \phi(w) \in O$.给定自动驾驶模型$m = M$ 可以得到车辆下一步的执行动作 $a = m(o) \in A$.假定一个具体的环境信息存在正确的控制指令 $\hat{a}$，通过定义示性函数：

$$\delta(a,\hat{a}) = \begin{cases} 1, & \text{自动驾驶车辆动作出错} \\ 0, & \text{自动驾驶车辆动作正常} \end{cases} \tag{1}$$

来确定模型当前输出的动作 a 是否会造成出错.

定义场景搜索算法$L \in \mathrm{L}$，其能够基于一个预定的场景 w，以令$\delta = 1$为目标，生成一个场景 w'.对于一个确定的待测试模型 m，把$L(m,\cdot)$简记成$L_{\mathrm{m}}(\cdot)$.可以使用算法 L 度量自动驾驶系统 m 的故障率：

$$\mathrm{Err}(m) = \sum_{\mathrm{w} \in W} \frac{\delta(m(\phi(L_m(w))),\hat{a})}{|W|} \tag{2}$$

也可称为，模型 m 在算法 L 的δ测试表现，$\mathrm{Err}(m;L;\delta)$.

2.2 安全性测试系统的工作流程

测试系统的架构设计如图 1 所示.为了能够精确控制场景生成，本文设计了一组属性配置方案，分别用于控制场景中单个对象的属性和相互关系（2.3

节）.1）动态场景生成器（2.4 节）首先读取配置文件，得到场景中对象的分布函数，并随机采样出一个初始场景描述.2）实时渲染引擎（如 Unreal Engine）会根据场景描述渲染出一个可供系统测试的驾驶场景.3）待测试驾驶模型读取场景.4）输出决策至缺陷分析器进行分析.5）缺陷分析器（2.5 节）会根据模型输出生成约束，适应性地指导动态场景生成器产生新的场景描述，进行下一轮测试.6）若缺陷分析器发现待测试系统的安全问题，会自动生成缺陷报告，以供后续的自动驾驶系统改进.下面将对系统中的核心技术和组件进行介绍.

2.3 丰富且真实的场景描述

为了实现丰富且真实的场景描述，我们设计了对象属性描述文件和环境配置方案，以用于描述实时渲染引擎所需要的资产（Assets）属性、属性分布和添加新地图；同时我们对场景中的全部对象进行了参数化描述，以便进行缺陷场景的自动化搜索.

2.3.1 对象描述与环境配置

本系统将场景中出现的对象分为了 5 类：

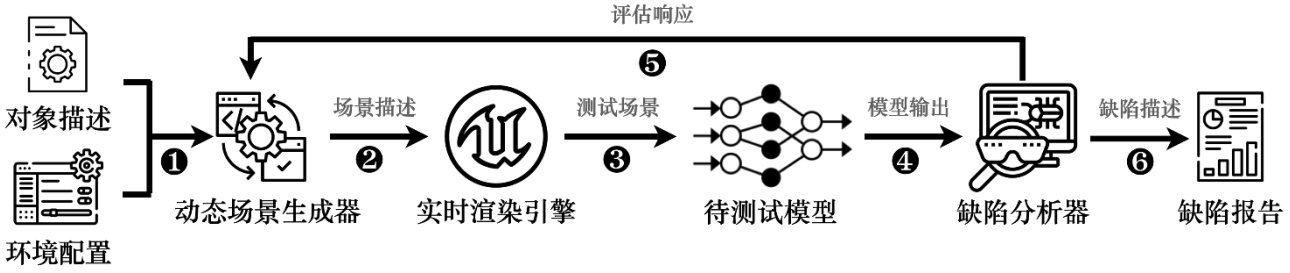


Fig.1 The workflow of the testing system

图 1 测试系统的工作流程

1）**环境 *E*** 是用于场景生成的预设的基础道路环境.一个道路环境应该至少包含道路与路边建筑.为了保证所描述场景的真实性如限定对象出现的合理位置，我们对环境进行了区域划分.一个典型的环境包含的区域有路外非驾驶区域、行人道、左右车道、十字路口.

2）**天气 *W*** 包括太阳高度角、降雨量、雾浓度等，其值是连续变化的.一个基本天气表示为在某个范围上的概率分布密度函数.天气之间可能会有相互作用，带来联合概率分布.因此，我们需要为单个场景设置多个天气分布，并使用联合概率分布函数进行抽样.

天气分布与环境配置相关性很低，若考虑天气分布与环境的相关性，则直接改变天气分布更合理，例如干旱地区大雨的概率比潮湿地区小很多；而车辆、行人和静态物体的分布是依赖环境的.

3）**车辆 *V*** 是环境中进行碰撞模拟和重力模拟的载具，包括汽车、自行车、摩托车，特别地，从现实角度考虑，自行车、摩托车会额外在载具上加入一个驾驶员.车辆在环境中不同区域出现的概率是不同的；我们也为车辆设置了正常和异常两种状态，并约束在不同的模式下车辆在各个区域的概率分布.例如，正常情况下车辆无论如何不会出现在人行道或是逆行车道上.

4）**行人 *P*** 的描述与车辆相似，设计上的差异在于：行人之间不存在类别差别，只有衣服、身材、相貌差别.

5）**静态物品 *G*** 是没有使用碰撞模拟和物理模拟的实体对象.如果不考虑相互作用直接采样得到初始分布，很有可能产生穿模的问题，因此，我们使用了基于几何体计算的 OBB（oriented bounding box，即定向包围盒）碰撞检测算法，其思路是将空间中的每个实体对象用 OBB 包围起来，通过计算不同实体对象之间的 OBB 是否重叠来判断是否发生了碰撞[43].

具体来说，先将物品投影到地面上，使用 OBB 代替三维物体进行相交性检查.同时为了弥补三维物品在垂直地面的维度上的层次丢失，引入图层的概念（图 2），每个图层上均存在物体的 OBB 投影，碰撞检测时需针对物品的多个图层同时进行.在场景生成时，会随机选择静态物品，并依次添加到环境当中，若新物体与老物体未发生碰撞，则物品生成有效.

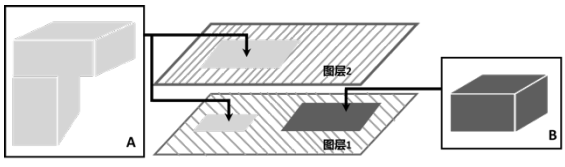


Fig.2 Multi-layer projection OBB.

图 2 多图层投影 OBB.

2.3.2 参数化描述

为了能够便捷地进行场景搜索以及保持系统多个模块之间概念的一致性，对场景设计了参数化描述方案.$W_E(Q,D)$ 用来表示场景的参数：E 为环境的参

数表示，其在一次测试过程中是一个定值；Q 为天气的参数表示；D 为实体对象的参数表示.

天气 Q 可以被表示为一个长度为 n 的具名元组，元组中的每一项为从 W 选择的天气与其对应的强度.$Q=\{q_1:x_1,q_2:x_2,\ldots,q_n:x_n\}$，其中 q_i 是天气的名字，x_i 是天气的具体取值，且处于定义的天气要求的范围内.

实体对象 D 包括车辆 V、行人 P 和静态物品 G，这些对象无须区分，可被统一化处理.对于一个场景，本系统挑选共计 m 个实体对象进行创建，使用一个大小为 $m\times 8$ 的矩阵（公式 3）表示实体状态，n_i 表示挑选的实体对象的名字，a_i 表示这个实体对象所取属性离散值在属性集合中的索引.坐标*(x,y,z)*是物体在实时渲染引擎创建的空间中的坐标；方向角*(r,p,y)*是物体语义正面法向量在渲染引擎坐标系下的欧拉角.

$$D_{m\times 8}=\begin{bmatrix} n_1 & a_1 & x_1 & y_1 & z_1 & r_1 & p_1 & y_1 \\ n_2 & a_2 & x_2 & y_2 & z_2 & r_2 & p_2 & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n_m & a_m & x_m & y_m & z_m & r_m & p_m & y_m \end{bmatrix} \tag{3}$$

对于相同的实体对象，由于其在矩阵 ***D*** 所处行不同，故而自然就区分开了.换言之，多个重复对象在我们的表示中用多行表示.

2.4 动态场景生成器

为了能够适应性地对测试系统进行安全测试，我们设计了一个动态场景生成器.生成器的使用分为两个阶段，首先是场景初始化阶段，在每轮测试开始前，动态场景生成器会根据环境配置和对象描述，按照天气、车辆、行人、静态物体的顺序进行一次抽样，合成对象的分布函数，生成场景描述.在使用阶段，动态场景生成器可以根据缺陷分析器的输出动态地生成下一个用于测试的场景.动态场景生成器的核心是一个适应性的场景搜索算法，该算法可以针对不同的待测系统生成不同的搜索方法，使得测试系统能够快速稳定地找到待测系统缺陷.

2.4.1 基于蜕变测试的评估方法

测试预言用于判断测试中待测系统输出是否正确.对于自动驾驶系统而言，在某个具体场景下，正确的输出是难以定义的.这是因为，对于自动驾驶系统，在某个范围内的输出都不会造成驾驶错误.另外，由于自动驾驶系统控制的车辆本身是物理连续的，单次系统错误输出可能不会造成严重的安全后果.因此，采取某个具体值作为测试预言是不合理的.沿用先前的工作[11]，我们采取蜕变测试并且予以松弛的方式作为测试预言.由于不同的测试场景需要的测试预言是不同的，使用我们的测试系统时，可以根据经验设计不同的预言规则.这里我们给出了两种测试预言：通常深度自动驾驶测试中关注的是输出转向角的正确性，因为转向角往往决定了系统是否会造成危险后果，**预言一**使用了这样的设计；但当测试车辆前有车辆，却因为变换场景造成没有刹车，此时也应判定自动驾驶系统出了错，**预言二**使用了该设计.

预言一：场景在变换前后造成的转向角应小于某一阈值，如果 $\delta(a,\hat{a})=|s-\hat{s}|>\varepsilon$，则代表发生故障.

预言二：测试车辆前有车辆等实体障碍，待测试车辆都应刹车，如果 $\delta(a,\hat{a})=(|s-\hat{s}|>\varepsilon_1)\vee(|t-\hat{t}|>\varepsilon_2)$，则代表发生故障.

$\hat{s}$ 和 $\hat{t}$ 是场景变换前自动驾驶系统的转向角和油门的输出；ε 和 $\varepsilon_1,\varepsilon_2$ 均为误差允许限，也称蜕变测试的松弛限.

2.4.2 适应性场景搜索算法

本搜索算法有 3 个设计需求：1）搜索算法应该是场景**合理**的，不能超出环境配置和对象描述所定义的合理状态.2）搜索到的相邻两个场景应该是驾驶**语义不变**的.3）搜索算法应该是**高效**的，单次搜索耗时不能太长.

我们通过拒绝采样的方式保证搜索到的场景的**合理性**.即，每次变换场景后，都要验证场景的合理性，如果不符合原本对象属性文件中定义的分布，则重新变换.驾驶**语义不变性**是通过控制相机所处车辆前一定距离的对象的位置不变保证的.例如，在自动驾驶车辆前有一辆车将造成车辆的刹车行为，那么在搜索过程中就不应变换这辆车的空间位置，仅可改变方向角和车辆颜色.最后，算法的**高效性**是通过变步长的随机搜索设计保证的，在搜索过程中，步长预算是根据上次搜索过程是否被接纳选择进行调整的.算法的细节描述见算法 1.

算法 1．适应性场景搜索算法.

输入：初始场景描述 w，搜索步长ε，变化维数 N_d，迭代更新预算 N_t,权衡天气参数 Q 和实体对象参数 D,示性函数 δ，接口封装好的自动驾驶系统 m, 在实际渲染引擎中创建的传感器 ϕ；

输出：搜索到的新场景 w_f.

① $W_{init}\leftarrow E,Q,D;\hat{a}\leftarrow m(\phi(W_{init}));i=1;$ /*

初始化场景、控制以及搜索轮数*/

② for each iteration $i \le N_t$

③ $n \leftarrow \text{Randint}(1, \min(\text{len}(Q), N_d))$；

④ $RV_O \leftarrow \text{randomWeather}(n)$；

⑤ $RV_D \leftarrow \text{randomEntity}(N_d - n)$； /*生成一个符合分布的 N_d 维随机向量*/

⑥ $scale \leftarrow \varepsilon / \sqrt{|RV_Q|_2^2 + |RV_D|_2^2}$； /*控制搜索步长为 ε */

⑦ $RV_O, RV_D \leftarrow RV_O * scale, RV_D * scale$；

⑧ $Q', D' \leftarrow$ 将 RV_O, RV_D 附加到 Q, D 上；

⑨ $w' \leftarrow (E, Q', D')$； /*生成新的场景，如果天气等参数超出阈值或者未通过碰撞检测，重新生成*/

⑩ $a \leftarrow m(\phi(w'))$；

⑪ if $\phi(a, \hat{a}) == 1$ /*搜索到新场景，结束搜索，返回有问题的场景 w' */

⑫ $w_f \leftarrow w'$；

⑬ break；

⑭ else if $d\delta(a, \hat{a}) / da > 0$

⑮ $Q, D \leftarrow Q', D'$；

⑯ 缩短步长 ε；

⑰ end if

⑱ $i \leftarrow i + 1$；

⑲ end for

在算法 1 描述的搜索过程中，每次迭代，若 $d\delta(a, \hat{a}) / da > 0$，就记录此次搜索结果作为下次搜索的起点.本工作中，用测试预言一和测试预言二代替这个条件.需要说明的是，上述搜索算法在迭代次数预算中未必能找到一个造成驾驶错误的世界 w_f，根据公式 2，据此计算出自动驾驶系统的表现度量.

图 3 是测试系统进行的一次场景搜索过程示意图.图中，红线框框出的是动态生成的车辆，蓝线框是动态生成的行人，绿线框是动态生成的静态网格体对象，天气影响了全局的渲染效果，如图中的建筑物阴影和树木的阴影、树叶的摆动角度等.测试系统通过动态调整这些物体空间位置及其内部属性，改变渲染画面，寻找造成故障的场景.

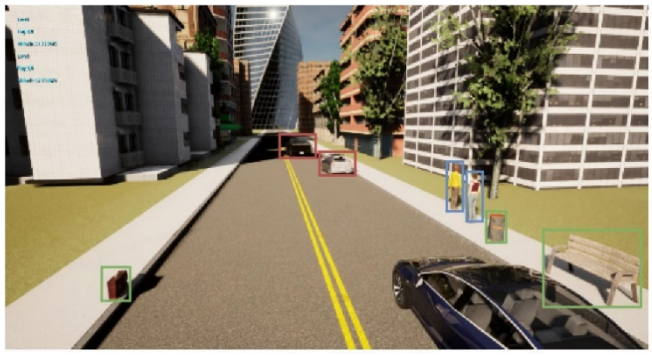


Fig.3 Schematic diagram of the scene search

图 3 一次场景搜索示意图

2.5 精确且自动的缺陷分析

对于造成示性函数取值为 1 的场景，缺陷分析器会分析、解释哪些物体或属性引发了系统异常.需要说明的是，造成模型输出异常的缘故是搜索过程的一整条路径，而非某次具体的迭代过程；对于带有高度非线性深度学习模块的系统而言，单纯分析路径很难判断究竟是什么原因造成了系统异常.

我们通过依次将天气置零和移除场景中的实体对象来寻找造成自动驾驶系统出现问题的缘由.造成自动驾驶出错的原因可能是相互耦合的，例如，对面行驶来的车辆由于雾天被识别错误造成车辆停止.为了找到造成出错的一组物体，采取迭代贪心搜索的策略，以 $\delta = 0$ 为停止标志进行场景搜索.算法细节见算法 2.

算法 2．自动化缺陷分析算法.

输入：当前场景下应执行的输出动作 $\hat{a}$，致使驾驶动作出错的世界 w_f，示性函数 δ，接口封装后的自动驾驶系统 m，在实时渲染引擎中创建的传感器 ϕ；

输出：造成故障的天气或实体对象 R.

① $w \leftarrow w_f; a \leftarrow m(\phi(w))$;/*初始化场景以及控制*/

② while $\delta(a, \hat{a}) == 1$

③ $weights \leftarrow []$；

④ for *object* in w

⑤ if 移除 *object* 会影响驾驶语义

⑥ continue；

⑦ end if

⑧ $w' \leftarrow w - object$； /*将天气置 0 或移除实体对象*/

⑨ $a' \leftarrow m(\phi(w'))$；

⑩ $weights \leftarrow$ 将 $d\delta(a', \hat{a}) / da'$ 加入列表；

⑪ end for

⑫ $object \leftarrow$ 取出 *weights* 中第一个值最大的元组的 *object*；

⑬ $R \leftarrow$ 将 *object* 加入实体对象；

⑭ $w \leftarrow w - object$；

⑮ $a \leftarrow m(\phi(w))$；

⑯ end while

⑰ return *R*.

算法 2 的 $\hat{a}$ 是当前场景下，自动驾驶系统应输出的动作，通过前文所述方式保持语义不变，同时将置零天气或移除其他实体对象情况下的自动驾驶系统输出作为 $\hat{a}$.可以预见到，算法 2 中最差情况是将处于驾驶语义保持区域外的所有实体对象清除，天气置 0，此时自动驾驶系统的输出就是 $\hat{a}$ ，故而该算法一定能够正确终止.搜索结果 *R* 是一个组合，在这个组合中，天气和实体对象的地位是相同的.

3 系统实现

3.1 待测系统

选取目前最佳的条件自动驾驶系统 CILRS[44]作为测试对象，其使用 ResNet-34 作为图像特征提取的卷积神经网络，权重参数为使用 CARLA 的 NoCrash 数据集训练得到的预训练模型.同时，为了显示出测试系统在不同自动驾驶系统上的能力，还选择了基础的条件自动驾驶系统 CIL[45]作为对比的基线.最后，对 CIL 和 CILRS 分别进行了封装，部署到测试系统中.

3.2 测试平台

从预制资产丰富程度和 API 使用的灵活度的角度出发，本文选择了 CARLA 0.9.11 作为测试系统开发平台.出于便捷导入静态资产和地图的目的，我们从源码编译 Unreal Engine 4.24 引擎和 CARLA 0.9.11，部署于 Windows 平台.运行时，从 Unreal Engine 编辑器中启动 CARLA 以快速迭代构建环境，验证场景生成算法的正确性.

3.3 场景搭建

受限于深度学习模型推导效率，通常部署于深度自动驾驶中的 CNN 网络的输入图像分辨率不会特别高，需要对相机捕获的数据进行预处理，裁剪出感兴趣区域（region of interest，ROI）.在这种情况下，场景中相对路面比较远的其他部分不会被捕获到相机画面中来.但是，比较高或者比较低的道路两侧的建筑的高度实际会影响到道路的光照效果，一定会对自动驾驶系统的预测产生影响.为了测试这种影响，我们设计了 3 种环境，分别为旷野、乡村、城市，具有不同的环境物体高度，通过 Unreal Engine 编辑器创建，如图 4 所示.

(a) 旷野环境 (b) 乡村环境 (c) 城市环境

Fig.4 Three environments

图 4 3 种环境

选取 CARLA 0.9.11 默认提供的 10 种天气参数（太阳方位角、太阳高度角、云量、降雨量、积雨量、风强度、空气湿度、雾浓度、雾距、雾密度）作为测试系统可以调节的天气参数.其中在不同场景中，太阳方位角和太阳高度角是必须存在的天气参数.雾浓度、雾距离、雾密度三者之间具有相关性，必须同时存在.

我们对 CARLA 提供的 89 种可有效生成的静态物体、28 种车辆、26 种行人进行了对象属性测算.28 种车辆的差异体现在型号带来的大小、形状和颜色的差异.26 种行人则包含男性、女性两种性别，年龄分成幼年、青年、老年 3 个年龄段.在 89 种静态物体中，有一些重复的内容，例如盒子包含 6 种，但有明显差异的只有 2 种，而且有一些物体在我们设计的场景下不应作为可以动态生成在道路和人行道上的内容，例如秋千.最终，选择全部的 28 种车辆和 26 种行人以及 15 种具有代表性的静态物体，将测算的数据按照格式要求写在对象属性文件中.

实时渲染引擎中需要一个用以控制自动驾驶的车辆和一个捕获当前场景图像的相机传感器，由于 CARLA 中车辆的物理模拟使用了相同的蓝图作为实现，所以在该平台上选择哪种车辆作为控制车辆是无

关紧要的.选取 28 种车辆中的 Tesla Model 3 作为控制车辆，传感器选择普通 RGB 单眼传感器，位于相对车辆中心前方 1.6m 处，相对地面高 1.4m，FOV 取为 100，画面分辨率为800×600，帧率为 25Hz.相机捕获的场景画面如图 5 所示.

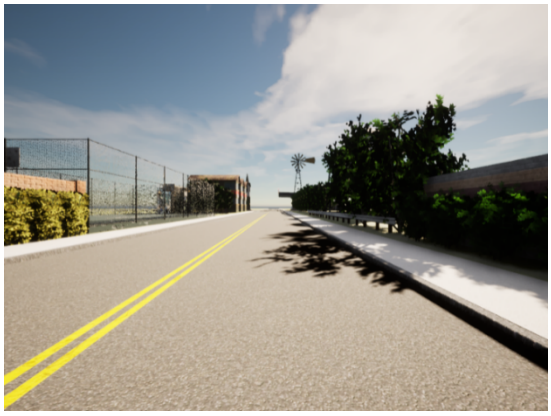


Fig.5 The image captured by the on-board camera in the country environment

图 5 乡村环境中车载相机捕获的画面

4 实验评估

4.1 安全问题检测能力对比分析

我们对最新的自动驾驶测试系统 Paracosm 进行了复现，并使用复现的 Paracosm 与我们的工作同时对自动驾驶系统 CILRS 和 CIL 进行安全性测试.我们选取了 CARLA 0.9.11 作为 Paracosm 和本系统的场景生成平台.考虑到，Paracosm 未指定如何设置具体的安全性问题检出标准，其在未来工作部分讨论了与本文类似的测试预言生成思路，但是并未指定具体的方法和参数选择.故在本章节的实验中，为了实验的公平性，复现的 Paracosm 系统和本文一样，选取了蜕变测试作为测试预言机制，将场景变换前自动驾驶系统输出的结果视为正确输出，详细的设计见 3.4 节.

对于参数选择部分，若搜索场景中仅发生了天气变换，系统会选取**预言一**作为故障判断的依据，同时取$\varepsilon=0.17$，即当车辆相对原始输出偏转角超过$15°$时认为出现了故障；对包含实体对象的变换情况，系统选取**预言二**作为故障判断的依据，在有实体对象的情况下，是否妥善刹车应是故障判断的标准，同时取$\varepsilon_1=0.17$，取$\varepsilon_2=0.2$.对于场景存在实体对象的情况，采用正态分布对实体数量进行抽样：$\mathrm{n}=\lfloor x\rfloor, x\sim \mathrm{N}(\mu=2,\sigma=2)$，当 n<0 时进行拒绝采样.

我们选取了 3 种类型的环境，即旷野、乡村和城市，对 CILRS 系统和 CIL 系统分别进行了安全性测试.请注意，3 种环境的主要区别在于道路两侧的建筑物高度不同，从而影响自动驾驶车辆上相机中的 ROI 的光照.将车辆生成于场景中的直道部分，设定车辆的驾驶分支为沿路行驶开始测试.让这两个测试系统分别在不同类型的环境下动态搜索 300 个场景.我们在表 2 中展示了本工作与 Paracosm 系统的安全问题检出率，检出率的计算方法如公式 2 所示.

Table 2 The discovery rate of security issues between this work and the Paracosm system

表 2 本工作与 Paracosm 系统的安全问题发现率

工作	测试系统	旷野/%	乡村/%	城市/%	平均/%
本工作	CILRS	45	43	30	39.3
Paracosm	CILRS	34	32	21	29
本工作	CIL	47	46	36	43
Paracosm	CIL	29	43	18	30

上述实验结果表明，本工作在旷野、乡村和城市 3 个代表性环境的 300 次场景搜索中，安全问题发现能力均要优于自动驾驶测试系统 Paracosm.整体来看，本工作在两个系统上的安全问题检出率皆是 Paracosm 的 1.4 倍.实验表明，本系统面向测试对象的适应性搜索算法设计相比于非适应性算法更加高效.

4.2 安全问题检测能力具体分析

在本节中，会对本文设计的系统进行详细的分析.为排除其他因素造成的影响，选择设计的 3 个环境中的旷野环境作为测试环境.其实验参数与 4.1 节相同.基于该设定分别对 CIL 和 CILRS 进行实验，并针对每个系统进行了 1000 轮场景搜索.使用公式 2 计算故障率，并将结果列于表 3 中.

从表中可以看出，存在实体对象时，场景变换造成的故障率较只有天气的情况更高.并且在考虑天气和所有实体对象情况下，我们对 CILRS 的故障发现率达到了 58.4%.

Table 3 Failure rate of Autonomous Driving System

表 3 自动驾驶系统的故障率

系统	仅天气 /%	仅车辆和行人 /%	仅静态物体 /%	全部对象 /%
CIL	31.5	62.4	61.2	65.7
CILRS	31.8	50.9	53.7	58.4

比较 CILRS 的实验结果与 CIL 的实验结果，可以发现两者在只包含天气的情况下表现相近，而在拥有表示场景拥挤程度的实体对象的情况下，CIL 故障发生率高于 CILRS.CILRS 是在 CARLA100 数据集下训练的，着重解决拥挤场景情况下驾驶系统的正确性

预测问题.实验结果印证了 CILRS 确实缓解了拥挤场景下自动驾驶系统故障率高的问题.换言之，CILRS 的安全性优于 CIL.

4.3 蜕变测试的松弛限

在章节 3.4 中的测试预言的定义中，使用松弛的蜕变测试来避免采取固定真值测试时，自动驾驶系统测试可能会出现大量假阳性的问题.在章节 5.2 中，使用 $\varepsilon = 0.17$ 对不包含实体对象的天气场景进行实验，使用 $\varepsilon_1 = 0.17$，$\varepsilon_2 = 0.2$ 对包含实体对象的场景进行实验.这两个值的选取是经验选取的.本小节通过调整 ε 进行实验，从而分析松弛限的取值对实验结果的影响.

选取 CILRS 系统的沿路行驶模式，在直道上进行测试，测试系统拥有生成全部实体对象的能力.分别固定 $\varepsilon_1 = 0.17$ 和固定 $\varepsilon_2 = 0.2$，每次固定进行 100 次实验，将故障发现率绘在图 6 中.

图 6 中，折线 ε_1 是固定 $\varepsilon_2 = 0.2$ 时，ε_1 取值[0,0.5]，故障发现率与松弛限的关系.折线 ε_2 是固定 $\varepsilon_1 = 0.17$ 时，ε_2 取值[0,0.5]，故障发现率与松弛限的关系.注意当 $\varepsilon = 0$ 时，故障发现率并不为 100%，这是由于 ε_1 与 ε_2 并非同时取 0 带来的.

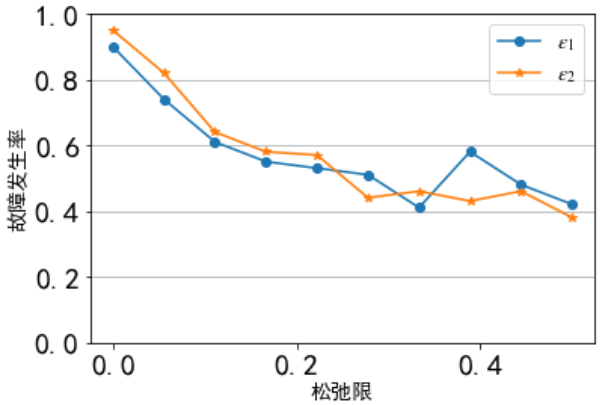


Fig.6 The relationship between failure discovery rate and slack limit using metamorphosis test

图 6 使用蜕变测试故障发现率与松弛限的关系

从图 6 中，可以观察到随着松弛限逐渐变大，故障发现率不断下降.如果松弛率比较小，那么系统就会报出大量的假阳性.如果松弛率比较大，那么系统又会忽略可能的危险错误.正如前面解释设计自动驾驶测试预言的困难性，如何折中松弛率是一个很复杂的问题，在 DeepTest[15]中使用统计标准差作为松弛限，DeepRoad[34]直接使用经验取值.根据此图，ε 在 0~0.1 区间内，随自身增加，故障发现率下降速度比较快，推测排除了大量的假阳性，而松弛限在 0.1~0.22 比较平滑，可以看作一个比较合理的取值范围.另外，对自动驾驶任务而言，略高的假阳性能够避免故障场景的错误排除，对实际任务并非不可接受.这是自动驾驶任务故障的危害性程度带来的.

4.4 场景覆盖分析

本节从不同的环境、不同的驾驶模式对深度自动驾驶系统进行测试，验证测试系统的覆盖能力.

4.4.1 环境测试

我们提供的 3 种测试场景，旷野、乡村和城市的主要区别在于道路两侧的建筑物高度不同，从而影响自动驾驶车辆上相机中的 ROI 的光照不同.在地图中的直道区域，分别对 CIL 和 CILRS 进行测试，结果列于表 4 中.

Table 4 Testing results of autonomous driving systems in different environments

表 4 自动驾驶系统在不同环境下的测试结果

自动驾驶系统	旷野/%	乡村/%	城市/%
CIL	65.7	64.4	63.8
CILRS	58.4	56.7	52.0

从表 4 可以得到 2 个结论：

1）在旷野、乡村、城市 3 个不同的环境下进行测试，故障发现率是比较接近的，这在一方面说明了环境对故障发现率的影响比较小，另一方面印证了我们设计不同环境的依据——自动驾驶中的 ROI 设计使得自动驾驶系统更关注路面而非道路两侧的环境.

2）有趣的是，在我们的设想中，低光照环境下的城市，故障发现率应该高于正常光照或者高光照情况下的旷野和乡村条件，因为按照直觉，夜间驾驶相对白天驾驶更容易出现问题.然而表格 4 的数据与我们预料的结果相反，低光照情况下的故障搜索效率反而低了.通过观察实验结果，我们推测在正常光照条件下，自动驾驶车辆的转向角一方面依赖实体对象，另一方面依赖道路中心的双黄线.一旦双黄线部分被遮蔽后，就很有可能造成驾驶输出故障.在低光照情况下，双黄线始终被遮蔽，这就使得驾驶输出主要依赖场景中的其他实体对象.由前文可知，我们设计的驾驶语义保持的方式使得决定驾驶语义变化区域的实体对象没有改变，这样反而使得故障的搜索能力减弱了.

4.4.2 驾驶模式测试

在条件自动驾驶系统中，除车载相机拍到的图像

外，高层的控制指令决定了当前驾驶动作应采取的分支动作.而自动驾驶系统有 4 个驾驶模式，即沿路行驶、向左、向右和直行.我们在不同的场景中分别选择了直路来测试沿路行驶模式，选择十字路口分别测试向左、向右和直行 3 种指令.注意，在地图的不同位置进行测试时，场景初始化器需要根据区域配置重新合成对象分布进行抽样以适应车载相机实际的 FOV.测试结果见表 5.可以看到，在不同分支的情况下，CILRS 的安全性均优于 CIL.

Table 5 Testing results of the autonomous driving system in different branches

表 5 自动驾驶系统在不同分支下的测试结果

自动驾驶系统	沿路 /%	直行 /%	向左 /%	向右 /%
CIL	65.7	63.1	68.6	59.5
CILRS	58.4	51.8	51.1	47.2

4.5 CILRS 驾驶系统脆弱性分析

在章节 2.5 中，介绍了如何对测试系统发现的安全性问题进行进一步解释.本章节中，将以 CILRS 系统为例，通过回答如下问题，来展示测试系统的自动化测试能力.

问题一：哪些天气更容易造成 CILRS 系统出现故障？

使用缺陷分析器将拥有所有对象控制能力的实验中的天气逐个置零，确定哪些天气更容易造成自动驾驶系统发生故障.由于不同天气是通过采样得到的，采样得到的数量不同，故而使用造成故障的天气相对该天气出现次数的比值作为对比值，将结果画在图 7 中，注意，天气可能只是造成故障的原因之一，不一定是决定因素.

缺陷分析器给出的造成故障的原因可能是多个对象，故而该图中总和大于 100%.从图 7 中可以看出，造成路面信息模糊（特别是道路标志模糊）的积雨最容易造成自动驾驶系统的不稳定.随之是干扰了相机传感器画面的降雨.风的强度主要影响到了降雨的雨水飘落的倾斜程度和道路两侧树叶的吹动，后者属于环境内容，根据在表 4 给出的结果，其对自动驾驶系统的影响并不突出.可见 CILRS 系统在这些天气条件中，最易受到降雨环境的干扰.

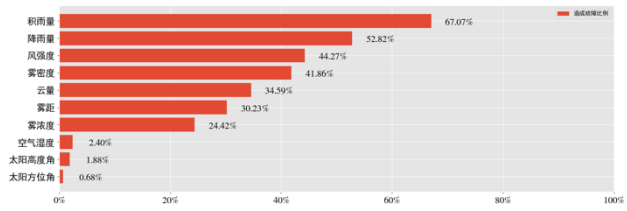


Fig.7 Proportion of CILRS system failure caused by the weather

图 7 天气造成 CILRS 系统故障的比例

问题二：哪些区域是 CILRS 驾驶系统的关键区域？

将深度自动驾驶的关键区域定义为，当这个区域出现实体对象时，自动驾驶的输出更容易出现不稳定性.对先前在直道上进行测试，存在实体对象的实验结果进行分析.**首先**，搜索到的造成故障的场景和原始场景会一并交给缺陷分析器进行处理，分析出造成故障的物体.**接着**，把故障物体的区域绘在图 8 中，横坐标是在沿道路方向的 x 轴，纵坐标是道路切向方向的 y 轴.在我们的设置中，自动驾驶车辆的坐标为(0,-2.27).图 8 的统计结果显示 CILRS 系统的敏感区域在道路两侧的人行道上.

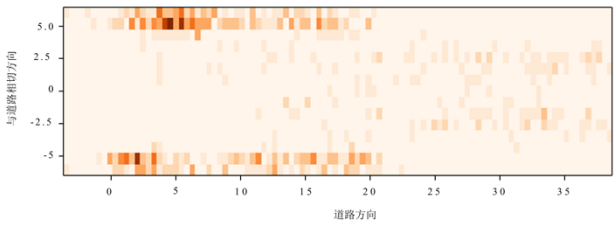


Fig.8 The key area of the CILRS driving system.

图 8 CILRS 驾驶系统的关键区域.

问题三：哪些物体更容易造成 CILRS 系统出现故障？

将造成系统故障的物体相对其采样次数的比值用作比较物体造成 CILRS 系统故障的概率的标准.去除重复物体，取错误率最高的 5 个实体对象，将对象及其故障率列在表 6 中.

通过图 8 和表 6，可以发现 2 点：

1）位于道路上的车辆可能不是造成不稳定性的主要原因，相反，处于道路两侧人行道上的物体可能更容易造成自动驾驶系统的不稳定.观察 CARLA100 数据集，我们发现自动驾驶系统针对复杂道路条件进行了训练，忽略了人行道上物体的复杂性.

2）黄色与红色是 CILRS 系统的敏感颜色.这是很自然的，因为交通信号灯的颜色恰好是黄色和红色，当人行道上出现黄色或者红色物体时，CILRS 很有可能将其误判成信号灯.

测试系统发现的 CILRS 的脆弱性正是 CILRS 优化的方向，可以使用本文测试系统在网络训练后校验系统的安全性.网络优化方案可以是数据增强、结构优化等，优化后再次使用测试系统进行校验可以确定深度网络是否满足了安全性需求.例如对 CILRS 系统，建议增加雨天天气、更丰富的道路两侧场景的训练数据以，提高系统的稳定性，设计双保险机制缓解对黄色、红色物体的敏感性.

Table 6 The entity objects and their failure rate

表 6 实体对象及其故障率

对象	故障率/%
位于人行道上的无靠背长椅	33.8
身穿红色上衣、蓝色裤子的行人	32.3
黄色方形纸箱	31.2
身穿红色上、，橘黄色裤子的行人	30.8
红色 Toyota Prius 汽车	28.6

4.6 特征分析与启发

为了进一步分析造成 CILRS 系统故障的原因，我们打开了 CILRS 系统，观察 ResNet 的特征提取层.我们发现，大多数情况下搜索到的故障场景和原场景的特征提取结果虽然有所差异，但差异不是特别明显，在一系列分析过程中，我们发现了一个有趣的样例，即，**身穿红色衣服的行人在人行道上行走**.

如图 9 所示，设置环境在旷野上，车辆位于直行道路区域上，使用沿路行驶模式进行测试，此时车速为 4（归一化数值），一个身穿红色衣服的行人在右侧人行道的中央位置沿着人行道向前走.自动驾驶车辆停留在初始位置，根据行人所处的不同位置，可以得到 CILRS 系统预测的输出的变化，将其画在图 10 中.

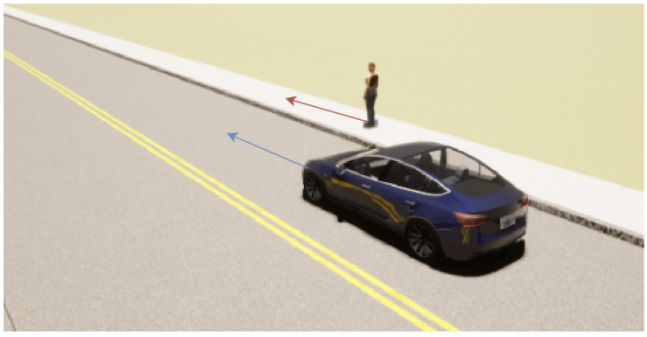


Fig.9 Failure analysis example: Pedestrian in red walking on the side of the road

图 9 故障分析样例：身穿红色衣服的行人在道路旁行走

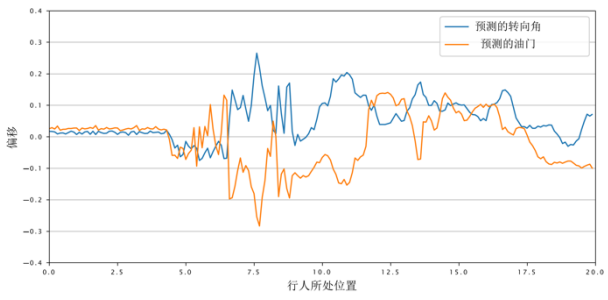


Fig.10 CILRS system output with pedestrian position changes

图 10 CILRS 系统输出随行人位置的变化

根据图 10 可以看到，行人出现前，CILRS 系统输出比较稳定.而在行人开始出现时，CILRS 系统受到了影响和波动，然而始终处于故障范围之外.大约在车辆中心位置前 7.5m 处，预测输出变为猛踩刹车（刹车效应优先于转向效应）.随后自动驾驶系统的输出仍处于波动状态，直到行人远离后车辆输出才趋向稳定（bias<0.1）.这样的表现显然是不正常的，当行人距离车比较近的时候，如果 CILRS 判定此时应刹车，则应立即输出刹车，不应该在行人继续向前走之后才输出刹车.将没有行人和行人在相对车辆中心位置前面 7.5m 处时的相机输入及 CILRS 卷积层前 3 层输出的绘在图 11 中.作为对比，将一张长椅放在位于人行道上造成自动驾驶系统输出变化最大的位置，此时自动驾驶输出相对不存在任何物体时的偏移 *steer_bias*=0.030<0.17，*throttle_bias*=0.05<0.2，不会被判定为系统故障.

图 11 中，有长椅的场景第 3 层卷积输出的结果与不存在任何物体的场景的卷积输出的结果比较相似，而行人所出现的场景卷积输出和不存在任何物体的场景的卷积输出有明显差异.

图 11 中，前 2 列在原始画面和特征提取前 2 层画面有明显差异，然而在第 3 层却比较相近.第 1 列和第 3 列原始画面和前 3 层都有很大的差异.这说明尽管画面中相同位置都出现了物体，但特征提取出来的内容是不同的.这启发我们或许可以在某一或某些特征层设置监控器，通过监控器的变化分析提前预警系统是否会发生故障.

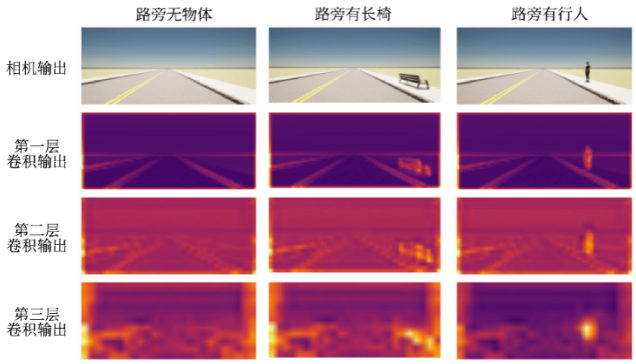


Fig.11 Input and convolutional layer output of CILRS

图 11 CILRS 输入及卷积层输出

4.7 系统运行效率分析

本小节对测试系统的运行效率进行分析.部署测试系统的硬件平台为 AMD R5 3600X+RTX 3070，使用的软件平台为 Windows10 + Unreal Engine 4.24.3 + CARLA 0.9.11 + Python 3.9.1.

将动态场景生成器根据初始场景搜索到一个造成自动驾驶系统出错的场景或是超出迭代预算的用时定义为一次测试用时.在考虑所有实体对象，对象

数量根据正态分布采样得到，包含渲染过程的情况下，一次实验的平均用时为 16.86s.排除渲染过程，采用分模块测试取平均值的方式，对每个模块及其内部细节进行效率的测试，结果见表格 7.

与十几秒的单次实验耗时相比，各个模块内部耗时是很低的，系统主要的性能瓶颈在渲染效率上.

Table 7 Average time consumption of each module of the testing system

表 7 测试系统各模块平均时间消耗 ms

初始化			搜索		
天气	行人与车辆	静态物体	天气	行人与车辆	静态物体
0.181	0.852	2.705	1.087	0.260	<0.1

5 结论

为了保障车联网场景下的自动驾驶系统安全性，本文设计并实现了一套场景驱动的针对自动驾驶视觉感知模块的安全测试系统.该系统构建了一套真实且丰富的场景描述方法，极大地拓展了测试系统的数据分布；本系统可以动态地为不同自动驾驶模型产生安全测试方案，可以实现高效、稳定地发现安全缺陷；最后本系统设计了一套精细的自动化安全问题分析工具，该工具可以帮助自动驾驶开发人员快速定位系统的安全性问题.我们相信，本工作将启发更多的自动驾驶感知模块的测试方案，会为车联网场景下的自动驾驶领域提供重要的安全基础.

作者贡献声明：吴昊负责论文撰写和解决方案设计；王浩负责系统设计和实现；苏醒负责系统实现和实验；李明昊负责实验和结果分析；许封元定义问题及修改论文；仲盛负责把控研究方向和修改论文.

参 考 文 献

[1] Silberg G, Wallace R. Self-driving cars: The next revolution [OL]. [2021-10-15] https://www.cargroup.org/publication/self-driving-cars-the-next-revolution/

[2] Zhang Rui, Li Jintao. A survey on algorithm research of Scene Parsing based on Deep Learning[J]. Journal of Computer Research and Development, 2020, 57(4): 859-875(in Chinese)
（张蕊, 李锦涛. 基于深度学习的场景分割算法研究综述[J]. 计算机研究与发展, 2020, 57(4): 859-875 ）

[3] Mozaffari S, Al-Jarrah O Y, Dianati M, et al. Deep learning-based vehicle behavior prediction for autonomous driving applications: A review[J]. IEEE Transactions on Intelligent Transportation Systems, 2020, PP(99): 1-15

[4] Huang Yu, Chen Yue. Autonomous driving with deep learning: A survey of state-of-art technologies[J]. arXiv preprint arXiv: 2006.06091, 2020

[5] Zhu Xianglei, Wang Haichi, You Hanmo, et al. 2021. Survey on testing systems in autonomous vehicles[J]. Journal of Software, 2021, 32(7):2056-2077 (in Chinese)
（朱向雷,王海弛,尤翰墨, 等. 自动驾驶智能系统测试研究综述[J].软件学报,2021,32(7):2056-2077）

[6] Woehrle M, Gladisch C, Heinzemann C. Open questions in testing of learned computer vision functions for automated driving[C]//International Conference on Computer Safety, Reliability, and Security. Berlin, German: Springer, Cham, 2019: 333-345

[7] Andriushchenko M, Croce F, Flammarion N, et al. Square attack: A query-efficient black-box adversarial attack via random search[C]//European Conference on Computer Vision. Berlin, German: Springer, Cham, 2020: 484-501

[8] Zhang Xinhai, Tao Jianbo, Tan Kaige, et al. Finding critical scenarios for automated driving systems: A systematic literature review[J]. arXiv preprint arXiv:2110.08664, 2021

[9] Li Jiangkun, Deng Weiwen, Ren Bingtao, et al. An Evaluation Method of Test Scenario Complexity for Intelligent Vehicles[J]. China Society of Automotive Engineers Congress and Exhibition, 2020: 110-117 (in Chinese)
(李江坤,邓伟文,任秉韬, 等. 一种智能汽车测试场景复杂度的评估方法[J]. 2020 中国汽车工程学会年会论文集(1) ,2020: 110-117)

[10] Wang Runmin, Zhu Yu, Zhao Xiangmo, et al. Research progress on test scenario of autonomous driving[J]. Journal of Traffic and Transportation Engineering, 2021, 21(2)：21-37.(in Chinese)
(王润民, 朱宇, 赵祥模, 等. 自动驾驶测试场景研究进展[J]. 交通运输工程学报, 2021, 21(02): 21-37)

[11] Dosovitskiy A, Ros G, Codevilla F, et al. CARLA: An open urban driving simulator[C]//Conference on robot learning. Long Beach, California: PMLR, 2017: 1-16.

[12] Fremont D J, Dreossi T, Ghosh S, et al. Scenic: a language for scenario specification and scene generation[C]//Proc of the 40th ACM SIGPLAN Conf on Programming Language Design and Implementation. New York, NY：ACM， 2019: 63-78

[13] Majumdar R, Mathur A, Pirron M, et al. Paracosm: A test framework for autonomous driving simulations[J]. Fundamental Approaches to Software Engineering, 2021, 12649: 172-195

[14] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated Whitebox Testing of Deep Learning Systems[C]//Proc of the 26th Symposium on Operating Systems Principles. New York, NY: ACM, 2017:1–18.

[15] Tian Yuchi, Pei Kexin, Jana S, et al. Deeptest: Automated testing of deep-neural-network-driven autonomous cars[C]//Proc of the 40th Int Conf on Software Engineering. New York, NY: ACM, 2018: 303-314

[16] Ma Lei, Juefei-Xu F, Zhang Fuyuan, et al. Deepgauge: Multi-granularity

testing criteria for deep learning systems[C]//Proc of the 33rd ACM/IEEE Int Conf on Automated Software Engineering. New York, NY: ACM, 2018: 120-131

[17] Odena A, Olsson C, Andersen D, et al. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing[C]// Proc of Int Conf on Machine Learning. Long Beach, California: PMLR, 2019: 4901-4911

[18] Guo Jianmin, Jiang Yu, Zhao Yue, et al. Dlfuzz: Differential fuzzing testing of deep learning systems[C]//Proc of the 2018 26th ACM Joint Meeting on European Software Engineering Conf and Symp on the Foundations of Software Engineering. New York, NY: ACM, 2018: 739-743

[19] Xie Xiaofei, Ma Lei, Juefei-Xu F, et al. Deephunter: a coverage-guided fuzz testing framework for deep neural networks[C]//Proc of the 28th ACM SIGSOFT Int Symp on Software Testing and Analysis. New York, NY: ACM, 2019: 146-157

[20] Sun Youcheng, Huang Xiaowei, Kroening D, et al. Deepconcolic: testing and debugging deep neural networks[C]//Proc of 2019 IEEE/ACM 41st Int Conf on Software Engineering: Companion (ICSE-Companion). Piscataway, NJ: IEEE, 2019: 111-114

[21] Sun Youcheng, Huang Xiaowei, Kroening D, et al. Testing deep neural networks[J]. arXiv preprint arXiv:1803.04792, 2018

[22] Ma Lei, Zhang Fuyuan, Xue Minhui, et al. Combinatorial testing for deep learning systems[J]. arXiv preprint arXiv:1806.07723, 2018

[23] Li Zenan, Ma Xiaoxing, Xu Chang, et al. Structural coverage criteria for neural networks could be misleading[C]//Proc of 2019 IEEE/ACM 41st Int Conf on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). Piscataway, NJ: IEEE, 2019: 89-92

[24] Jha S, Banerjee S S, Cyriac J, et al. Avfi: Fault injection for autonomous vehicles[C]//Proc of 2018 48th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks Workshops (DSN-W). Piscataway, NJ: IEEE, 2018: 55-56

[25] Jha S, Banerjee S, Tsai T, et al. Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection[C]//Proc of 2019 49th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks (DSN). Piscataway, NJ: IEEE, 2019: 112-124

[26] Jha S, Tsai T, Hari S, et al. Kayotee: A fault injection-based system to assess the safety and reliability of autonomous vehicles to faults and errors[J]. arXiv preprint arXiv:1907.01024, 2019

[27] Abdessalem R B, Nejati S, Briand L C, et al. Testing vision-based control systems using learnable evolutionary algorithms[C]//Proc of 2018 IEEE/ACM 40th Int Conf on Software Engineering (ICSE). Piscataway, NJ: IEEE, 2018: 1016-1026

[28] Ben Abdessalem R, Nejati S, Briand L C, et al. Testing advanced driver assistance systems using multi-objective search and neural networks[C]//Proc of the 31st IEEE/ACM Int Conf on Automated Software Engineering. Piscataway, NJ: IEEE, 2016: 63-74

[29] Abdessalem R B, Panichella A, Nejati S, et al. Testing autonomous cars for feature interaction failures using many-objective search[C]// Proc of 2018 33rd IEEE/ACM Int Conf on Automated Software Engineering (ASE). Piscataway, NJ: IEEE, 2018: 143-154

[30] Wicker M, Huang X, Kwiatkowska M. Feature-guided black-box safety testing of deep neural networks[C]// Proc of Int Conf on Tools and Algorithms for the Construction and Analysis of Systems. Berlin, German: Springer, Cham, 2018: 408-426

[31] TESLA. Obtain a copy of the data associated with your tesla account [EB/OL]. [2021-10-18].https://www.tesla.com/support/privacy

[32] Broggi A, Cerri P, Debattisti S, et al. Proud-public road urban driverless test: Architecture and results[C]//Proc of 2014 IEEE Intelligent Vehicles Symp. Piscataway, NJ: IEEE, 2014: 648-654

[33] Feng Yang, Xia Zhilong, Guo An, et al. 2021. Survey of testing techniques of autonomous driving software[J]. Journal of Image and Graphics, 2021, 26(01): 13-27 (in Chinese)
(冯洋, 夏志龙, 郭安, 陈振宇. 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(01): 13-27)

[34] Zhang Mengshi, Zhang Yuqun, Zhang Lingming, et al. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems[C]// Proc of 2018 33rd IEEE/ACM Int Conf on Automated Software Engineering (ASE). Piscataway, NJ: IEEE, 2018: 132-142

[35] Liang Xiaodan, Hu Zhiting, Zhang Hao, et al. Recurrent topic-transition gan for visual paragraph generation[C]//Proc of the IEEE Int Conf on Computer Vision. Piscataway, NJ: IEEE, 2017: 3362-3371

[36] Marchesi M. Megapixel size image creation using generative adversarial networks[J]. arXiv preprint arXiv:1706.00082, 2017

[37] Bansal M, Krizhevsky A, Ogale A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst[J]. arXiv preprint arXiv:1812.03079, 2018

[38] Bewley A, Rigley J, Liu Y, et al. Learning to drive from simulation without real world labels[C]// Proc of 2019 Int Confe on Robotics and Automation (ICRA). Piscataway, NJ: IEEE, 2019: 4818-4824

[39] Ding Shaohua, Tian Yulong, Xu Fengyuan, et al. Trojan attack on deep generative models in autonomous driving[C]//Proc of Int Conf on Security and Privacy in Communication Systems. Berlin, German: Springer, Cham, 2019: 299-318.

[40] Richter S R, Vineet V, Roth S, et al. Playing for data: Ground truth from computer games[C]//Proc of European Conf on Computer Vsion. Berlin, German: Springer, Cham, 2016: 102-118

[41] Richter S R, Hayder Z, Koltun V. Playing for benchmarks[C]//Proc of the IEEE Int Conf on Computer Vision. Los Alamitos, CA: IEEE Computer Society.,2017: 2213-2222

[42] Ros G, Sellart L, Materzynska J, et al. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes[C]//Proc of the

IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2016: 3234-3243

[43] Zhang Rong, Liu Xiaogang, Wei Jianjun. Collision detection based on OBB simplified modeling[C]//Journal of Physics: Conference Series. IOP Publishing, 2019, 1213(4): 042079

[44] Codevilla F, Santana E, López A M, et al. Exploring the limitations of behavior cloning for autonomous driving[C]//Proc of the IEEE/CVF Int Conf on Computer Vision. Los Alamitos, CA: IEEE Computer Society, 2019: 9329-9338

[45] Codevilla F, Müller M, López A, et al. End-to-end driving via conditional imitation learning[C]//Proc of 2018 IEEE Int Conf on Robotics and Automation (ICRA). Piscataway, NJ: IEEE, 2018: 4693-4700

Wu Hao, born in 1994. PhD. His main research interests include intelligent mobile computing and privacy and security.

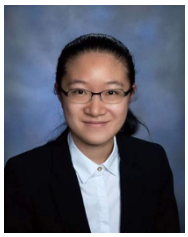
吴昊，1994 年生.博士.主要研究方向为智能移动计算及其隐私安全.

Wang Hao, born in 1996. Master. His main research interests include security in games and binary analysis.

王浩，1996 年生.硕士.主要研究方向为游戏安全和二进制分析.

Su Xing, born in 1997, Bachelor. His main research interests include software analysis and blockchain.

苏醒，1997 年生.学士.主要研究方向为软件分析和区块链.



Li Minghao, born in 1999. PhD candidate. Her main research interests include computer networks, cloud computing, and edge computing.

李明昊，1999 年生，博士研究生.主要研究方向为计算机网络、云计算和边缘计算.

Xu Fengyuan, born in 1983. PhD, professor, PhD supervisor, Senior member of CCF. His main research interests include Mobile System and Security, the Real-Time Cyber-Physical Metaverse, Autonomous Trust and Privacy Mechanisms, and Intelligent Edge Computing Systems.

许封元，1983 年生.博士，教授，博士生导师，CCF 会员.主要研究方向为移动端系统安全、实时虚实融合元宇宙、 自主信任与隐私机制以及智能边缘计算系统.

Zhong Shen, born in 1974. PhD, professor, PhD supervisor, Senior member of CCF. His main research interests include cryptography, game theory and its application in Computer Networks and Distributed System.

仲盛，1974 年生.博士，教授，博士生导师，CCF 会员.主要研究方向为密码学、博弈论及其在计算机网络、分布式系统中的应用.