南 京 师 范 大 学

毕 业 设 计（论 文）

（2025 届）

题　　目：	VR APP 的自动化异常检测工具设计与实现
学　　院：	计算机与电子信息学院/人工智能学院
专　　业：	计算机科学与技术
姓　　名：	朱正阳
学　　号：	19210217
指导教师：	段博佳

南京师范大学教务处　　制

摘　要

虚拟现实（Virtual Reality，VR）应用的质量保障仍面临诸多挑战，主要由于其交互复杂性和开发生态系统的碎片化。现有的测试方法高度依赖人工操作，既耗时又低效。本文提出了 VRExplorer，一种面向 Unity 平台 VR 应用的基于模型的自动化测试方法。我们首先收集并分析了 GitHub 上 104 个高质量的开源 VR 项目，从中提取出常见的交互模式与对象特征，用以构建抽象模型。基于此模型，我们提出了一个具有可扩展性与适应性的三层测试框架——EAT，包括实体接口层（Entity Interface Layer）、动作抽象类层（Action Abstract Class Layer）和任务层（Task Layer）。该框架系统性地将 VR 交互逻辑解耦为可复用的组件。在 EAT 框架基础上，VRExplorer 集成了 Unity 的 NavMesh 系统与基于概率的有限状态机（FSM），实现对场景的探索及预定义交互任务的执行。其工作流程包括场景配置、接口实现和行为执行三个阶段。我们在 9 个多样化的 Unity VR 项目上进行了实验评估，结果显示 VRExplorer 在覆盖率和效率方面均显著优于基线方法 VRGuide。在实验问题 2 中，VRExplorer 的 ELOC 覆盖率最高提升了 96.5%，方法覆盖率提升了 61.5%。此外，消融实验结果也突出了各交互模块的关键作用，进一步验证了设计的合理性。这些发现表明 VRExplorer 在真实开发场景中具备良好的可扩展性与通用性，适用于自动化 VR 测试。

关键词：基于模型的软件测试，虚拟现实，场景探索

Abstract

Ensuring the quality of virtual reality (VR) applications remains a challenge due to the complexity of interactions and the fragmented development ecosystem. Existing testing approaches heavily rely on manual efforts, which are labor-intensive and inefficient. In this paper, we present VRExplorer, a model-based automated testing approach designed for Unity-based VR applications. We begin by collecting and analyzing 104 high-quality open-source VR projects from GitHub, from which we extract common interaction patterns and object features to construct abstraction models. Based on this abstraction, we propose a scalable and adaptable three-layer testing framework named EAT, consisting of the Entity Interface Layer, the Action Abstract Class Layer, and the Task Layer. The framework systematically decouples VR interaction logic into reusable components. Built upon EAT, VRExplorer integrates Unity's NavMesh system and a Probability-based Finite State Machine (FSM) to explore scenes and execute predefined interaction tasks. It operates through a sequence of scene configuration, interface implementation, and behavior execution. Experimental evaluation on 9 diverse Unity-based VR projects shows that VRExplorer consistently outperforms the baseline approach VRGuide, achieving significantly higher coverage and better efficiency. Specifically, VRExplorer yields up to 96.5% improvement in ELOC coverage and 61.5% in method coverage across a broad range of projects. Furthermore, ablation results highlight the essential contributions of each interaction module, confirming the rationality of our design. These findings demonstrate the scalability and versatility of VRExplorer for automated VR testing in real-world development scenarios.

Key words: Model-based Software Testing, Virtual Reality, Scene Exploration

目　　录

第 1 章 绪论

近年来，随着元宇宙概念的兴起和虚拟现实技术的快速发展，VR 应用在教育、医疗、娱乐等多个领域展现出广泛的应用前景。然而，VR 应用的复杂交互和大规模场景带来了前所未有的测试挑战，传统手动测试方式效率低、覆盖不足，难以满足高质量需求。特别是在以 Unity 为主的开发环境中，功能性异常常因交互逻辑复杂或插件不兼容而频繁出现。因此，亟需设计高效、可扩展的自动化测试方法，提升 VR 应用的测试覆盖率与开发质量，降低人工测试成本。

1.1 研究背景

1.1.1 VR APP 的应用价值

近年来，随着元宇宙（Metaverse）概念的兴起以及智能头戴设备（Head-Mounted Display，HMD）的快速发展，虚拟现实（Virtual Reality，VR）技术逐渐走进大众视野。虚拟现实（VR）是指基于计算机的技术，该技术模拟了人为创建的环境的视觉，听觉和触觉影响，目的是将用户浸入感知的现实中[1]。

根据《财富》的数据，全球 VR 市场规模在 2023 年的价值为 251 亿美元，预计将从 2024 年的 326.4 亿美元增长到 2032 年的 2444.4 亿美元。因此，确保 VR 应用质量变得非常重要[7]。

VR 应用（VR APP）主要指运行在 Oculus、Pico 等智能头戴设备上的应用程序。当前的 VR 应用程序，特别是部署在诸如 Meta Quest 2 之类的独立设备上的应用程序，主要集中在医疗，教育，视听娱乐和培训模拟上[2-6]，近年来一直在迅速发展并引起了人们的关注，VR 技术的应用也渗透到人们各行各业，在医疗、教育、航空、军事、工业生产领域和日常娱乐等方面有着广泛应用。

首先，VR 技术在教育领域的应用逐渐受到重视。通过虚拟环境，学生能够进行互动式学习，进行实验、模拟和角色扮演等，这对于传统的教学模式是一种极大的补充。

其次，在医疗领域，VR 的应用则主要集中在手术模拟、病患康复等方面，医生可以在虚拟环境中进行手术演练，减少实际手术的风险，而病患也可以通过虚拟场景进行康复训练，达到更好的治疗效果。VR 在娱乐行业中的应用也很广泛，尤其是在

游戏领域。通过 VR 设备，用户可以进入一个完全虚拟的世界，体验极致的游戏互动。这种高度沉浸感的体验，使得传统的 2D 或 3D 游戏无法比拟。

除此之外，VR 还在建筑设计、房地产展示、社交娱乐、心理治疗、远程工作等多个领域展现出巨大的潜力。随着元宇宙的构建和 5G 技术的普及，VR 技术的应用场景将进一步扩展，成为数字世界与现实世界之间的重要桥梁。

1.1.2 VR APP 测试存在的问题与挑战

VR 设备通常在基于 Android 和 Linux 的操作系统上运行，这可能会带来一系列从 Android 应用程序继承下来的常见错误。作为 VR 开发的主要引擎，Unity 被广泛采用；然而，开发人员在开发生命周期中经常会遇到各种功能性错误。这些错误通常是由于通过 Unity 和第三方 VR 交互包模拟物理交互和集成功能的复杂性造成的。目前，VR 应用测试和错误检测在很大程度上依赖于人工测试，这仍然是高度劳动密集型和低效的[8]。因此，迫切需要能够系统地探索大型环境并与虚拟对象进行交互的自动 VR 测试工具，从而模拟人类测试人员的操作。这种工具不仅能简化测试过程，还能提高 VR 应用开发的可靠性和效率。然而，VR 应用自动测试领域仍然存在挑战。

第一个挑战是复杂的场景探索和 VR 交互。与移动应用测试不同，VR 应用测试面临着独特的困难，因为场景空间和动作的复杂性大大增加。VR 测试的核心是探索虚拟环境和与虚拟对象交互，这就引入了一个巨大的状态空间。在测试过程中，任务通常由线性和非线性子任务组成，例如找到一把钥匙、用它打开一扇门、转动把手和按下按钮逃生。这些任务可能涉及各种复杂的交互，要求测试人员按照特定顺序触发各种动作和决策序列。因此，这些交互的多样性和复杂性，再加上大规模的虚拟环境空间，使得以系统和可重复的方式有效模拟类似人类的行为来探索场景变得十分困难。

第二个挑战是 VR 开发生态系统的分散性导致的可扩展性和多功能性。VR 开发生态系统高度分散，开发包和插件种类繁多。例如，XR Interaction Toolkit（XRIT）是 Unity 的官方软件包，无需使用第三方软件包即可开发 VR 应用程序[9-10]。除 XRIT 外，还有一些第三方插件，如 SteamVR、MRTK 和 VRTK，它们在交互实现的具体细节上存在差异[11-17]。此外，VR 应用针对的平台多种多样，包括基于 PC 的系统、独立 VR 头显、移动设备、基于网络的环境等。这种分散性使得很难使用单一的可扩展工具来满足所有 VR 应用程序的测试需求。

1.1.3 研究意义

本在本课题中，VR 应用中的异常主要来源于开发过程中的不当代码实践、不兼容的 Unity 及其插件包版本等因素。由于 VR 应用的项目文件数量庞大，场景和模型资源复杂多样，给 VR 应用中的异常检测带来了极大的挑战。目前，主流的 VR 应用公司和游戏公司通常依赖大量的黑盒和白盒测试来发现和报告程序缺陷，这不仅消耗了大量人力资源，而且测试覆盖面有限。虽然已有一些框架试图解决 VR 应用中的自动化异常检测问题，但它们在场景探索（Scene Exploration）中仅通过简单点击操作进行检测，未能模拟抓取、拖拽等更复杂交互的场景检测。

针对以上问题，本课题从软件工程和软件可靠性的角度出发，重点研究如何自动化进行 VR 场景探索，尽可能提高代码覆盖率、场景交互覆盖率，从而检测 Unity 引擎开发的 VR 应用中的常见异常。通过引入自动化场景探索技术，全面触发场景中的交互事件，并结合软件工程方法和静态分析手段，以提升 VR 应用开发的质量与效率，减少手动测试所带来的时间和成本。

1.2 国内外的研究现状

现有的 VR 异常实证研究涵盖了从性能优化到隐私安全的多个方面，为异常检测工具的开发提供了丰富的经验和方法；在场景探索和寻路方面，现有的工作提及到的交互序列问题和复杂的交互动作问题仍然比较难以解决；在 VR 智能体的研究方面，现有的工作已经在游戏检测领域初步尝试结合了 LLM 和强化学习方法。

1.2.1 自动化游戏测试

Glib 是一种基于代码的数据增强技术的工具，用于自动检测游戏 GUI 故障[18]。Macklon 等人提出了一种新方法，通过利用 <canvas> 上对象的内部表示，自动检测 <canvas> 游戏中的视觉错误[19]。Prasetya 等人研究了使用基于图的路径查找技术来解决自动化游戏导航和探索中的挑战[20]。ix4XR，基于 Java 的多代理编程框架，促进了游戏测试，通过一种名为 Environment 的接口使测试代理与待测游戏进行交互，开发者需要实现这一接口[21-22]。通过对游戏任务和 action 进行序列化建模，不关心底层实现细节，而是在高层上抽象任务结构（Goal Struction）和行为（Action），需要程序员手动去设定序列；通过接口和底层函数（如寻路）交互。该方法创新性的将交互动作进行序列化建模。然而这一工作需要程序员手动进行。因此如果能够通过 LLM

对 Unity 场景和代码文件进行学习，自动生成结构化、形式化模型，也许能够更好的解决智能体的决策问题。Aplib，基于信念-愿望-意图（BDI）模型的库工具，支持开发能够执行复杂测试任务的智能代理[23]。ix4XR 和 Aplib 都采用形式化方法来抽象动作和目标，将任务结构化为多个动作和目标的顺序组合。这种形式化改进了游戏测试框架的可扩展性和多功能性。

1.2.2 移动应用测试

Chen 等人提出了一种基于模型的 HarmonyOS 应用程序 GUI 测试方法。该方法引入了一个页面转换图（Page Transition Graph，PTG）模型，通过遍历抽象语法树（AST）和调用图（CG）构建[24]。它还提出了一种通用的 HarmonyOS 应用程序 GUI 测试工具，该工具利用自动化测试框架 arkxtest，并采用深度优先搜索（DFS）策略[25]。AutoConsis 利用专门定制的对比语言-图像预训练（CLIP）多模态模型自动分析 Android GUI 页面，借助大语言模型（LLM）从文本描述中提取相关数据[26]。Fastbot2 是另一种基于模型的自动化 GUI 测试工具，作者提出了一种能够记忆测试关键数据的概率模型，并提出了一种基于模型的引导测试策略，该策略通过强化学习算法进一步增强[27]。

1.2.3 基于强化学习的测试

先前的研究表明，强化学习（RL）提高了游戏测试的效率。Tufano 等人定义了一种方法，利用 RL 训练一个代理以类人方式进行游戏，同时识别导致帧率下降的区域[28]。RLbT 采用基于好奇心的 RL 方法通过最大化覆盖率来自动化游戏测试[29]。实证评估将 RLbT 应用于一款名为《Lab Recruits》的 3D 游戏，并将基于好奇心的方法与几种替代基准进行比较。Bergdahl 等人采用了一种模块化方法，其中 RL 补充了经典的测试脚本，而不是替代它[30]。该工作扩展到包括漏洞检测和连续动作模拟，如鼠标移动，重点关注导航而非战斗。Wuji 是一个即时游戏测试框架，利用进化算法、RL 和多目标优化进行自动化游戏测试[31]。它是网易雷火事业群伏羲 AI Lab 提出的一款基于演化强化学习的针对多人在线对战游戏的测试智能体。Wuji 模型忽略了 GUI，简化为考虑游戏技能释放和移动方向；首先对异常特征进行分析，对异常分成了 5 类；评估时通过手动注入异常作为“Groud Truth”，分析了两款网易游戏；衡量指标采用代码覆盖率、状态覆盖率。该论文创新性的将强化学习引入了游戏测试智能体。然而，

Wuji 无法解决 VR 应用中的场景探索问题，因为它只能学会释放技能和移动，并不具备处理交互物体的复杂逻辑。

1.3 研究内容

本论文围绕 VR 应用程序在自动化测试中存在的覆盖率不足、探索能力有限等问题，提出了一种基于交互建模的自动化测试方法——VRExplorer，并基于所设计的面向 VR 场景的 EAT 测试框架进行了系统实现与评估。

首先，为确保测试方法具备良好的通用性与代表性，本文选取了多个具有代表性的 Unity VR 项目，开展了项目调研与特征分析。通过系统提取并分析 VR 应用中常见的交互行为与对象类型，为测试建模阶段提供了可靠的依据。

随后，本文设计并实现了 EAT 测试框架，构建了基于“Entity-Action-Task”层次结构的交互建模体系，全面覆盖了 Triggerable、Grabbable、Transformable 等多种典型交互方式，从而提升了测试系统的通用性与扩展能力。EAT 框架不仅为交互行为建模提供了结构化支持，也为后续测试过程中的动作规划与执行奠定了基础。

在测试流程设计方面，本文提出了一套完整的测试执行流程，包括场景配置、接口注入与行为驱动探索等关键环节，实现了对 VR 场景中关键交互对象的自动识别与有效操作，提升了测试的自动化程度与执行效率。

最后，本文通过实验评估验证了所提出方法的有效性。对比实验，分析 VRExplorer 在测试上的性能表现。

1.4 论文结构

本文正文部分一共分为 5 个章节，具体内容与结构如下：

第 1 章为绪论部分，首先介绍了本研究的背景与动机，包括 VR 应用程序的应用价值、当前测试中存在的问题以及本研究的意义。随后回顾了国内外在 VR 漏洞实证研究、寻路与场景探索、VR 测试与智能体方面的研究现状。最后明确了本研究的主要内容与结构安排。

第 2 章介绍了本研究所依赖的相关技术，包括虚拟现实技术基础、Unity 引擎的使用、自动化软件测试的基本概念与流程，以及 VR APP 测试中涉及的场景探索与交互机制。

第 3 章重点阐述了 VR APP 自动化测试方法的设计过程。包括项目的收集与分析、交互行为与对象特征的模型抽象、提出的 EAT 测试框架的各层设计（Mono 层、Entity 层、Action 层、Task 层），以及整体测试流程的构建和执行方式。

第 4 章前半部分为实验设计部分，首先提出了实验所关注的研究问题，然后介绍了 Baseline 方法、各项实验参数、评估指标的选取依据，并对速度参数进行了预实验。此外，对实验项目的选择、实验环境与配置、接口实现及消融实验的配置进行了详细说明。后半部分，展示并分析了实验结果，分别从三个问题出发，比较了所提出的 VRExplorer 与 Baseline 方法在不同实验设置下的表现。通过代码覆盖率、时间成本等指标进行评估，进一步验证了所提方法的有效性和适应性。

第 5 章对全文进行了总结，归纳了本研究的主要工作和成果，并探讨了当前研究的局限性以及未来可能的研究方向，为后续工作提供了思路与参考。

第 2 章 相关技术

本章围绕虚拟现实应用的关键技术基础展开，旨在为后续研究内容提供必要的技术背景支撑。通过梳理当前虚拟现实系统的构成与发展，介绍主流开发平台的核心能力，并结合软件工程中的自动化测试方法和相关应用技术，为理解虚拟现实应用中的异常检测与测试策略奠定理论基础。

2.1 虚拟现实技术

虚拟现实技术，是一种利用计算机生成的三维环境，使用户能够通过特定的设备沉浸其中，并与虚拟世界进行交互的技术。它通过视觉、听觉、触觉甚至嗅觉等多感官模拟，使用户产生一种身临其境的感觉。近年来，随着计算机图形学、传感器技术和人工智能的快速发展，虚拟现实技术得到了广泛的应用，并在多个领域展现出巨大潜力。其技术核心在于创建高度逼真的虚拟环境，并通过硬件设备使用户能够感知和交互。

VR 系统依赖于计算机图形学和渲染引擎来构建虚拟环境，并利用物理引擎模拟真实世界的物理特性。此外，人工智能和机器学习技术的结合也提高了虚拟现实的智能化水平，例如 AI 驱动的虚拟角色、智能导航等。

2.2 Unity 引擎

Unity 是目前最流行的 VR 开发引擎之一，它提供了强大的工具和插件，使开发者能够轻松创建虚拟现实应用。下面介绍 Unity 在 VR 领域的核心技术支持。

Unity 提供了对多个 VR 平台（如 Oculus、HTC Vive、PlayStation VR、Windows Mixed Reality 等）的原生支持，并集成了 XR 插件管理器，使开发者能够快速部署 VR 应用。通过 Unity 的 XR 交互工具包（XR Interaction Toolkit），开发者可以轻松实现 VR 交互功能，如抓取、移动物体等。

VR 应用对渲染性能要求极高，Unity 提供了多种优化方案，如实时光照、LOD（细节层次）技术、Occlusion Culling（遮挡剔除）等，以提高渲染效率。此外，Unity 的 URP（通用渲染管线）和 HDRP（高清渲染管线）可用于优化 VR 场景的视觉效果，使其更加真实。

Unity 对 VR 应用提供了丰富的支持。Unity 支持各种输入设备，包括 VR 手柄、眼动追踪、手部追踪等。开发者可以使用 Unity 的 Input System 或 OpenXR 标准来管理 VR 输入，并结合物理引擎模拟真实世界的物理交互。另外，Unity 支持多人 VR 应用的开发，开发者可以使用 Photon、Mirror 等网络框架来构建多人在线 VR 体验。例如，在多人 VR 会议中，用户可以在同一虚拟空间中互动，实现远程协作。此外，Unity 拥有丰富的 VR 开发资源，包括 Asset Store（资源商店）提供的各种 VR 插件，如 VR Interaction Framework、VR Builder 等，使开发者能够更快地构建 VR 应用。此外，Unity 还支持 WebXR，使 VR 应用可以在浏览器中运行，提高了可访问性。

2.3 自动化软件测试

自动化软件测试是一种利用脚本或专用工具执行测试用例，以减少人工干预、提高测试效率和准确性的技术。它广泛应用于功能测试、回归测试、性能测试等场景，尤其适用于需要频繁重复执行的测试任务。

自动化测试通常包括测试脚本编写、测试环境配置、执行测试以及结果分析。常见工具包括 Selenium（Web 自动化）、Appium（移动应用测试）、JMeter（性能测试）等。相比手动测试，自动化测试能更快地发现缺陷，提高软件质量，同时降低长周期项目的测试成本。

在持续集成（CI）与持续部署（CD）流程中，自动化测试也是关键环节之一，它能够确保代码变更不会引入新问题，从而提升软件开发的稳定性和可靠性。

本节探讨的自动化软件测试技术，重点放在测试脚本编写、测试用例生成部分。本节通过分析不同的测试方法，简述本文使用基于模型的方法的底层原因。

2.4 VR 异常测试相关技术

2.4.1 VR 应用异常及 VR 漏洞实证研究

VR 应用作为一种高度复杂的软件系统，其在开发过程中容易产生多种异常，主要包括性能问题和功能性问题。性能问题和功能性问题是影响 VR 应用稳定性和用户体验的两大类重要问题，它们各自源于不同的开发环节，具有不同的特点和挑战。

性能问题是指由于不当的代码实践或不合理的资源管理，导致 VR 应用在运行过程中出现响应迟缓、帧率下降等问题。VR 应用尤其需要保持较高的帧率（通常要求至少 30-60 帧/秒）以确保用户体验的流畅性，若帧率过低，容易导致 VR 头显中的运

动伪影和延迟，严重时可能导致晕动症等不适症状。常见的性能问题通常来源于以下几个方面：

垃圾回收压力过大：在 Unity 引擎中，垃圾回收（Garbage Collection，GC）是一种常见的性能瓶颈。频繁的堆内存分配和不当的内存管理可能导致垃圾回收压力过大，进而影响 CPU 的性能。尤其是在 VR 应用中，长时间运行可能导致 GC 频繁触发，进而导致帧率下降。

过于复杂的图形模型：VR 应用的图形表现往往要求较高的渲染质量，而复杂的三维模型和大量的多边形会增加 GPU 负担，影响渲染效率。尤其是一些具有大量多边形的模型，可能会导致图形渲染的瓶颈，造成帧率波动或画面卡顿。

资源模型错误：VR 应用中的资源管理至关重要，如果模型资源没有得到有效优化，可能会导致不必要的资源浪费，甚至影响 UI 的渲染效果。例如，当使用错误的纹理格式或未优化的贴图时，可能导致 UI 闪烁或画面出现紫色错误等问题。

功能性问题主要是指软件由于设计缺陷或代码实现错误，导致应用的功能无法正常工作，或产生不符合预期的行为。常见的功能性问题包括：

软件崩溃（Crash）：当程序在运行过程中遇到未处理的异常或致命错误时，可能导致应用崩溃。在 VR 应用中，崩溃不仅会影响用户体验，还可能导致用户的设备卡死或重启，造成很大的不便。

空指针引用（Null Reference）：空指针错误是开发中最常见的一类错误，尤其是在 Unity 中，由于对象生命周期管理不当，可能会发生空指针引用，从而导致应用崩溃或功能异常。

内存泄漏：内存泄漏是指程序分配了内存，但没有及时释放，导致系统内存占用逐渐增加。长时间的内存泄漏会导致应用性能下降，甚至系统崩溃。

这些问题虽然看似与性能无关，但它们直接影响 VR 应用的稳定性，进而影响用户的整体体验。因此，在 VR 应用的开发过程中，必须时刻关注这些潜在的功能性问题。

VR 应用程序漏洞的实证研究，主要是指通过对开源 VR 项目的程序漏洞的统计性分析，提取其关键特征，并进行分类，以及进一步的分析每一种漏洞的产生原因。这是漏洞检测工作的关键前置问题，只有明确漏洞及其产生类型才能更好的去检测漏洞、评估工具的性能。目前国内外的研究中，软件工程领域有一些针对游戏漏洞和 VR 应用漏洞的实证分析，其中的分析方法很值得借鉴。

文献[32]是德克萨斯大学圣安东尼分校联合上海交通大学针对 VR 应用开发者如何优化性能问题的实证研究（Empirical Study）。作者通过对 45 个开源 VR 项目数据集的 git commits messages 手动分析（两位研究者独立分析，合并、讨论确定最终结果，通过 Cohen's-Kappa 值达成一致）；然后通过静态分析（Static Analysis），使用 srcML 解析 C#代码，GumTree 解析抽象语法树，通过 MetaID 检测 unity 依赖文件，将性能问题的优化方法分为图形简化、渲染优化、API 代码优化、堆分配避免和值捕获。最终说明了优化方法在官方 VR 项目和独立开发者之间的区别等若干研究性问题。该论文对 VR 性能问题做了深刻的特征分析，并且从代码角度说明了性能问题产生的原因和解决方法。

文献[33]是中山大学软件学院和香港浸会大学合作的提出了一款名为 VR-SP Detector 的工具，用于检测主要面向 Unity Oculus VR 应用的安全性和隐私性漏洞的工具。首先通过对 apk 包合政策隐私协议的提取，通过 PolicyLint 进行分析比对隐私方面的漏洞；然后对 manifest 文件进行分析，并通过污点追踪，最终得出结论，隐私和安全性问题在 VR 应用中很常见。该论文主要关注隐私和安全性漏洞，重点在于对打包后的 apk 文件以及隐私政策文本的分析，以及逆向工程后对源代码的分析。本课题的 VRExplorer 是基于源代码的分析，可以借鉴文章中提到的对 C#代码的静态分析方法、以及对 git commit messages 的分析方法。

2.4.2 VR 寻路算法

寻路问题是一个非常经典且被充分研究的问题，在计算几何和图论领域中，经典的最短路算法比如 Dijkstra 和 Flody；游戏中的寻路算法比如 A*算法等；全局优化问题中的 Prim 和 Kruskal 算法用于解决最小生成树等。在复杂的 VR 和 3D 游戏中，寻路问题需要考虑更多。例如每帧更新的动态的障碍物，以及代价问题（最短路可能会有其他游戏代价）和智能体的随机性问题（比如偶尔走入“危险区域”更符合拟真玩家的行为）。而场景探索，则是 VR 测试中需要解决的问题。在寻路的基础之上，VR 测试智能体需要尽可能的去探索场景，而除了障碍物以外，还有可能出现交互之后才能继续探索的区域、事件等。这给场景探索带来了复杂性和多样性，简单的 DFS 等暴力方法无法解决。

文献[34]探讨 3D 游戏中的寻路算法问题，包括动态避障和精细控制（比如有选择性的走入危险区域），也提到了决策序列问题。该论文发现 Unity 中的插件 NavMesh

可以很好的解决寻路问题。这是一个底层基于启发式的 A*算法和图形学方法实现的寻路插件。通过对场景模型的 Mesh 构成的图进行 Dijkstra 等最短路算法的启发式综合进行最短路求解，同时能够很好的解决动态避障问题。操作上只需要通过烘焙静态网格以及设置 NavMeshAgent 即可进行寻路。

2.4.3 VR APP 场景探索和测试技术

软件测试是软件生命周期中的重要组成部分。VR 应用作为一种软件，需要严格和规范化的自动测试，更需要自动化的测试工作流。目前国内外提出了基于代码异味的检测、基于强化学习算法的智能体检测等。其中，智能体检测是指通过一个模拟玩家的智能体在场景中漫游，去运行游戏中的所有分支情况，尽可能的去触发函数执行链条，从而达到检测功能性问题的目标。

Harms[35]提出了一种完全自动化的 VR 应用程序测试方法，该方法不要求用户在固定的测试环境中执行预定义任务。相反，它通过分析实际使用会话的录制内容来生成任务树。然后，这些任务树被用来识别可用性问题，指的是用户行为模式中可能暗示的潜在可用性问题。具体而言，通过扫描场景中的所有可交互 VR 物体，找到其中具有 event handling 脚本的，并扩展使其能够在 event log 的时候，自动化用户行为记录（保存到 log file）：物体抓取、释放、物体使用、物体不使用、头部移动；将这些记录保存到中央服务器；提出了任务树生成算法，任务树能够代表记录的用户行为。该论文关注 VR 应用的可用性测试，从用户的角度出发，需要用户自己进行场景探索的同时，框架在后台记录 Task Tree 和行为日志，通过对 Task Tree 的规律分析，寻找 VR 应用的可用性异味（Smells）。然而该框架仍然需要用户去自己探索场景，并不能达到自动化场景探索和异常检测的目标。

文献[36]这篇工作针对 VR 应用开发测试做了实证性研究（测试代码比重、测试有效性），主要阐明了 VR 应用在开发过程中的自动化测试方法和评估测试样例的性能，并对测试样例进行了分类。该论文考虑到了传统安卓应用和 VR 应用之间的差异性问题，比如考虑到了 VR 应用的交互测试、物理碰撞测试等。在方法构成上有借鉴意义。

Rzig 等人对 314 个开源 VR 应用程序进行了实证研究[8]。分析结果显示，79%的评估 VR 项目没有包含任何自动化测试。在有自动化测试的项目中，功能方法与测试方法的中位数比例低于其他类型的软件项目。

VRTest 是 VR 场景探索的一篇相关工作，为德克萨斯大学圣安东尼奥分校 Xiaoyin Wang 老师提出的场景自动化探索工具[37]。它通过提取 VR 场景中的信息并控制用户摄像头，通过特定的测试策略探索场景并与虚拟物体交互。它包括两种内置策略：VR-Monkey，采用纯随机探索，以及 VRGreed，使用贪婪算法探索 VR 场景中的可交互物体。通过自动控制相机移动、追踪物体交互事件（点击等），探索可交互物体。该方法首先计算拥有 renderer 组件，即可见物体的 Bounding Boxes 包围盒；然后通过状态变化报告器，获得物体的 EventTrigger 组件，获得组件的 Entry 条目，对回调增加监听，将状态变化报告器添加到所有可交互物体，在对应 event 触发时向上报告。并且通过 Evaluation 实验，对比了 Moneky 随机算法和 Greedy 贪心寻路算法，通过若干指标：可交互物体的探索程度、时间效率等，对 5 个开源 VR 项目进行实验，并检测异常。

VRGuide 是 VRTest 的改进工作，主要通过计算几何方法，对 VRTest 的相机寻路进行了优化[38]。它应用了一种名为 Cut Extension 的计算几何技术，优化相机路线以覆盖所有可交互物体。结果表明，VRGuide 在两个项目中在测试超时之前实现了更高的测试覆盖率，并且在剩下的六个项目中，与 VRTest 相比，它平均减少了 31%的测试时间，并达到了饱和覆盖率。文章提到了 Art Gallery Problem 美术馆问题和 Watchman Route Problem 观察者路由问题 。这是一个关于在多边形美术馆里，选择保安的最短路径，让他的沿途可以看到每一个角落的问题。传统的方法是 The Cut Theory。而 VRGuide 提出了动态割寻路，计算距离最近的割的距离；相比 VRTest 的 Greedy 寻路，得到提升。在评估时，该工作将 VRTest 作为 Baseline，将测试案例扩展到 8 个 VR 应用。

这两篇工作的核心都是场景探索。Unity 中有复杂的游戏场所，让开发人员手动探索是非常耗时耗力的。这两篇工作初步将模拟交互的思想放到 VR 测试中。然而，目前的交互仅限于模拟“鼠标点击”这一简单操作，无论是 VRTest 还是 VRGuide 都没有考虑到复杂的 VR 交互，比如说抓取物体、开门、拉弓箭等操作；同时也无法处理需要一定固定交互序列的操作，比如先找到钥匙，再开门然后再拉下电闸。

文献研究了生成性人工智能（GenAI）在 VR 探索测试中的视场（FOV）分析的潜力[39]。该研究特别探讨了 GenAI 如何协助测试实体选择和行动建议，结果显示，GPT-4o 在任意视场内的物体识别准确率达到 63%。然而，该模型在物体组织和定位等任务上表现不佳。此外，研究还识别出了一些关键情境，可以提高建议动作在不同

视场中的准确性。Rafi 等人提出了 PredART，一种预测虚拟物体放置的人工评分的方法，可以作为自动化增强现实（AR）测试中的测试预言机[40]。

第 3 章 VR APP 自动化测试方法设计

本章介绍了我们提出基于模型的自动化测试方法的设计概述、项目分析与模型抽象过程，以及 EAT 框架的设计和 VRExplorer 的算法实现。

3.1 概述

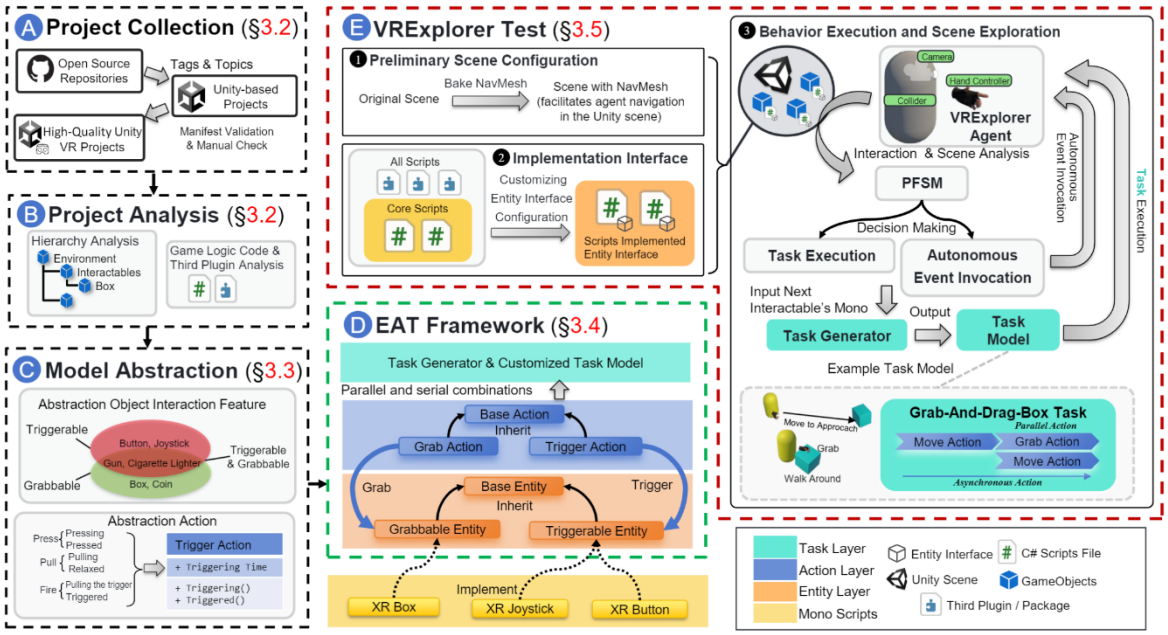


Figure 1: Overview of *VRExplorer*

图 3-1 VR APP 自动化场景探索与测试方法概述图

本节概述了本文采用的方法。

在“项目收集与初步分析”部分，我们描述了开源 VR 项目的收集过程。

在“模型抽象”部分，我们介绍了如何通过初步的实证分析将 VR 交互抽象为模型。

在“EAT 框架”部分，我们详细阐述了 EAT 框架，该框架旨在解决 VR 开发中的可扩展性和通用性问题。

在“VRExplorer 测试流程”部分，我们介绍了 VRExplorer，这是一种基于模型的自动化 VR 应用测试方法，依赖于 EAT 框架、NavMesh 和基于概率的有限状态机。VRExplorer 测试和场景探索包括准备工作（场景配置和接口定制化实现）与场景探索与测试。

图 3-1 展示了方法的完整流程图。

3.2 项目收集与分析

3.2.1 项目收集

我们首先利用 GitHub API 通过基于 Python 的网络爬虫脚本发送查询请求。在 GitHub 上筛选与 Unity 相关的开源项目，使用标签和主题进行过滤，例如“vr”“unity”“xr”。初步筛选后，共识别出 971 个项目。

随后，我们利用 manifest.json 文件作为补充标准，该文件提供了有关每个仓库中使用的包和库的关键信息。经过人工审核和编译测试，我们剔除了因编译错误、仓库质量不佳以及 Unity 或第三方插件版本冲突等问题导致不可用的项目。最终，筛选出 104 个高质量的 VR 相关项目，构建了一个涵盖各种 VR 交互任务和复杂空间环境的数据集。

3.2.2 项目分析

在数据集的基础上，我们进行初步分析并提取 VR 交互的抽象模型。主要通过手动分析场景对象的层次结构和核心游戏逻辑代码。层次结构分析可以剔除与核心可交互对象无关的内容，如静态墙壁、灯光、游戏管理器等。

此外，我们深入研究了继承自 MonoBehaviour 的交互脚本，这些脚本被挂载在 Inspector 窗口中的交互对象上，同时也分析了第三方 VR 交互库提供的脚本。这一分析帮助我们理解 VR 可交互对象的特征，并为下一步的模型抽象奠定基础，这是 EAT 框架的核心组成部分。

3.3 模型抽象

基于前述的模型抽象，我们提出了 EAT 框架，这是一个三层结构，包括实体接口层、行为抽象类层和任务层，旨在提升 VR 应用测试的可扩展性和通用性。

3.3.1 交互行为抽象

我们将具体的 VR 交互行为抽象为通用的交互动作。例如，按下按钮、开关门、打开或关闭灯、拉动操纵杆等操作，都可以抽象为一个持续过程和一个事件触发。以拉动操纵杆为例，这个过程可以被抽象为一个连续的拉动过程，随后触发一个事件。

我们将具备此类特征的交互归类为“触发动作”，该动作由一个异步方法 Triggering() 和一个同步方法 Triggered() 组成。

3.3.2 对象交互特征抽象

我们将具体的 VR 可交互对象抽象为不同的交互特征。例如，箱子、金币等可以归类为“可抓取”，意味着它们可以被拾取或移动；按钮、操纵杆等可以归类为“可触发”，表示它们具有多个状态，如“触发中”和“已触发”。

某些对象可能同时具有多个特征，例如枪支和打火机，它们既可抓取也可触发，意味着用户不仅可以拿起它们，还可以进行功能性操作，例如开火或点燃。

3.4 EAT 框架

在模型抽象的基础上，为了更好的解决通用性和泛化性问题，我们提出了 EAT 框架，这是一个的三层框架，包括实体接口层（Entity interface layer）、行为抽象类层（Action abstract class layer）和任务层（Task layer）。

3.4.1 Mono 层

该层由继承自 MonoBehaviour 类的核心 VR 交互逻辑类组成，通常在 Inspector 窗口中挂载到 VR 可交互对象上。这些类负责处理 VR 环境中的交互逻辑，如物体的抓取、按钮的点击、射线检测等。例如，一把能够发射子弹的枪可以挂载 XRGun 类，该类继承自 MonoBehaviour，并包含诸如对子弹预制体的引用指针变量、Shoot()函数等。此外，该类还可以包含射击间隔、弹药数量、音效播放等属性，以提供更丰富的交互体验，并通过与物理引擎或动画系统的结合，实现更逼真的 VR 射击效果。

3.4.2 Entity 层

基于抽象对象交互特性，我们使用特定的接口来表示 VR 可交互对象的相应特性和属性。我们定义了一个基础接口 BaseEntity，其中包含 Transform 属性（用于表示对象的位置、旋转和缩放），以及 Name 属性。所有其他实体接口都继承自该基础接口。

实体层提供了针对动作层所需特性的定制接口。通过组合多个接口，可以通过多重继承高效地表示复杂特性，从而在定义 VR 交互时实现更大的灵活性和模块化。

图 3-2 给出了一个例子，一个具有 Grabbable 和 Triggerable 特性的枪类 XRGun，可以通过继承 MonoBehaviour 并同时实现多个接口来实现：

```
class XRGun : MonoBehaviour, IGrabbable, ITriggerable {
    // Implementation of grabbable and triggerable
        features
}
```

图 3-2 一个具有可抓取和可触发特性的枪类脚本 XRGun.cs

这种方法使 XRGun 能够无缝集成多种交互能力，使其既可抓取又可触发。通过利用接口组合，我们可以灵活地定义和扩展对象行为，而无需修改基础类层次结构，从而促进代码的可重用性和模块化设计，提高 VR 交互系统的灵活性。

3.4.3 Action 层

该层基于前文提到的行为抽象，将具有类似特征的动作归入同一动作类别，以提高代码的复用性和可扩展性。我们定义了一个抽象基类 BaseAction，其中包含一个虚拟异步方法（Asynchronous Method）Execute()，用于执行具体的交互行为。所有其他行为类均继承自该基类，并根据不同的交互需求，实现各自的执行逻辑。

行为层不仅能够与实体层交互，还承担着连接高层逻辑和底层交互的桥梁作用，为 VRExplorer 提供功能性 API 接口，模拟真实玩家的交互动作。例如，它可以用于触发按钮点击、操控虚拟角色移动、拾取物体等操作。此外，行为层还支持扩展，以适应更多复杂的 VR 交互场景，例如手势识别、物理碰撞反馈以及多模式交互，使系统更具灵活性和适应性。

3.4.4 Task 层

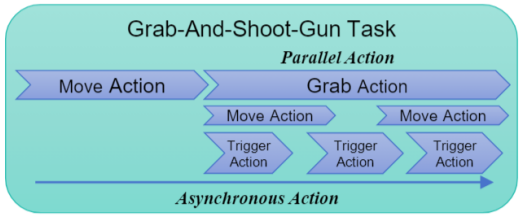
任务是多个行为层的行为的串行和并行组合。并行的多个行为之间在同一时刻同时执行各自的异步方法，串行的多个行为之间有先后顺序，必须等待前一个行为执行完成才能继续后续行为。任务层包含任务生成器（Task Generator）和多个预定义的任务，任务生成器通过接受 MonoBehaviour 实例或者实体数组 BaseEntity[] 生成对应的任务。预定义的任务是我们根据前期对项目的分析，抽象了一些常用的任务模型。

如图 3-3 所示，我们定义了 Grab-And-Shoot-Gun 任务。如图(b)所示，这个任务分为三个步骤，靠近桌子，拿起枪，边走边射击。其对应的模型如图(c)所示，横轴代表着异步任务的时间轴，首先是一个 Move Action，接着是一个 Parallel Action，其内部的三个行为之间彼此并行。对应的代码如图(a)所示。

```
1. private List<BaseAction> GrabAndShootGunTask(
2.     IGrabbableEntity grabbableEntity,
3.     ITriggerableEntity triggerableEntity)
4. {
5.     List<BaseAction> task = new List<BaseAction>()
6.     {
7.         new MoveAction(_navMeshAgent, moveSpeed,
8.             grabbableEntity.transform.position),
9.         new GrabAction(leftHandController, grabbableEntity,
10.            new List<BaseAction>()
11.            {
12.                new ParallelAction(new List<BaseAction>()
13.                {
14.                    new MoveAction(_navMeshAgent, moveSpeed,
15.                        GetRandomTwitchTarget(transform.position)),
16.                    new TriggerAction(2.5f, triggerableEntity)
17.                })
18.            })
19.    };
20.    return task;
21.}
```

(a) Code Snippet

(b) Process Diagram



(c) Action Model

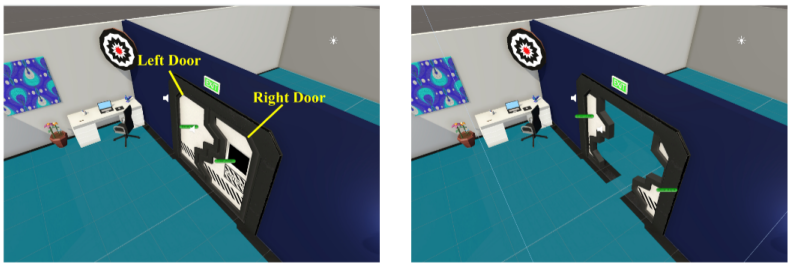
图 3-3 Grab-And-Shoot-Gun 任务实例示意图

3.5 VRExplorer 测试流程

我们在 EAT 框架的基础上开发了名为 VRExplorer 的 VR APP 自动化测试工具。对于一个待测试的开源项目，主要的步骤包括准备工作（场景配置和接口定制化实现）与场景探索与测试。

3.5.1 准备工作：场景配置

由于导航部分使用 Unity 提供的 NavMeshAgent 系统，在场景配置时，需要将地板、墙壁等地形物体设置为静态，并进行导航网格烘焙。场景中可能会开启的门等物体不需要设置静态，但需要挂载上 NavMesh Obstacle 组件，并开启 Carve 选项，使其能够动态的切割导航网，如图 3-4 所示。



(a) Doors Closed that separate the two rooms' NavMesh　　(b) Doors Opened

图 3-4 启用 Carve 选项并附加 NavMesh Obstacle 组件的动态门

3.5.2 准备工作：定制化接口和接口实现

定制化接口和接口实现是基于 EAT 框架的 VRExplorer 的核心，旨在解决复杂场景探索和 VR 交互的挑战，以及由于 VR 开发生态系统碎片化导致的可扩展性和通用性问题。通过使用接口封装实现细节，使其能够对各种 VR 应用进行测试。

为了确保覆盖率并验证交互功能的可靠性，测试人员需要为目标项目中与核心 VR 交互逻辑相关的 Mono 层脚本选择并自定义适当的实体（Entity）层接口，并实现相应的接口功能。如图 3-5 所示，橙色部分表示在接口实现过程中引入的额外代码。

```
using BNG;
using UnityEngine;
using VRExplorer;
/// <summary>
/// Apply forward force to instantiated prefab
/// </summary>
public class LaunchProjectile : MonoBehaviour, ITriggerableEntity, IGrabbableEntity
{
    #region Entity
    public float TriggeringTime => 1.5f;       // Time to hold this entity
    public string Name => Str.Gun;       // Tag for Task Generator to recognize this entity
    public Grabbable Grabbable
    {
        get
        {
            var g = GetComponent<Grabbable>();
            if(g) return g;
            return gameObject.AddComponent<Grabbable>();
        }
    }
    public Transform Destination => null;     // Grabbable.Destination, not used in this case
    public void Triggerring() { }
    public void Triggerred() { Fire();}
    public void OnGrabbed() { }
    #endregion

    public GameObject projectilePrefab = null;
    public Transform startPoint = null;
    public float launchSpeed = 1.0f;
    public void Fire()
    {
        GameObject newObject = Instantiate(projectilePrefab, startPoint.position, startPoint.rotation);
        if(newObject.TryGetComponent(out Rigidbody rigidBody)) ApplyForce(rigidBody);
    }
    private void ApplyForce(Rigidbody rigidBody)
    {
        Vector3 force = startPoint.forward * launchSpeed;
        rigidBody.AddForce(force);
    }
}
```

图 3-5 实现接口后的 Launch Projectile.cs 脚本

考虑到实现接口给开发者带来的额外工作量，我们提供了一套预定义的常用 Mono 脚本，这些脚本继承自 MonoBehaviour，适用于基于 XRIT 框架开发的项目，并已实现相应的实体层接口。如图 3-6 所示，测试人员只需将我们提供的预定义 Mono 脚本添加到目标对象上，并通过在检视器中添加 UnityEvent 来配置待测试功能。

图 3-6 预定义的 Mono 脚本组件：XRTriggerable

3.5.3 行为执行和场景探索

在完成场景配置和接口实现后，VRExplorer 将在场景运行时通过状态机决策，执行相应的行为，从而进行场景探索和 VR 交互。

我们为 VRExplorer 设计了两类行为：任务执行（task execution）和自主事件触发（autonomous event invocation）。

在执行任务时，VRExplorer 会感知当前场景，并将 MonoBehaviour 实例或实体数组（BaseEntity[]）输入到任务生成器中，从而获取并执行相应的任务。相比之下，自主事件的设计来源于我们对项目的分析，分析发现 VR 环境中的玩家不仅会与对象进行交互，还会主动触发一些事件，例如释放技能以获得加速效果等。

为了全面覆盖这类测试场景，VRExplorer 提供了一个可配置的 UnityEvents 列表，测试人员可以在测试前通过检查器轻松配置需要覆盖的功能。这两类行为共同构成了一个行为空间，每个行为可以看作是一个节点，整个行为空间则形成一个由这些节点组成的有向图。

在行为决策过程中，VRExplorer 使用一个基于概率的有限状态机（Probability-based FSM）来决定节点之间的转换，从而定义了这个有向图的拓扑结构。我们用$\mathcal{T} = T_1, T_2, \ldots, T_N$表示所有任务状态的集合，其中 T_i是执行序列中的第 i 个任务状态，i 取

值范围是 1 到 N。类似地，$\mathcal{E} = E_1, E_2, \ldots, E_M$表示所有探索状态的集合，其中$E_j$是执行序列中的第 j 个探索状态，j 的取值范围是 1 到 M。

在每个 FSM 的决策点，会有一个变量决定转移的方向。转移到某个探索状态 E_j 的概率记为 p，因此转移到任务状态T_i的概率为$1 - p$。状态转移的具体情况如表 3-1 所示，一个具体的状态转移示例如图 3-7 所示。

表 3-1 基于概率的 FSM 状态转移表

Current State	Next State	Condition	Probability
T_i	T_{i+1}	Default	$1-p$
T_i	E_j	Default	p
E_j	E_{j+1}	Default	p
E_j	T_{i+1}	Default	$1-p$
T_i	T_{i+1}	$\mathcal{E}$ exhausted	1.0
E_j	E_{j+1}	$\mathcal{T}$ exhausted	1.0

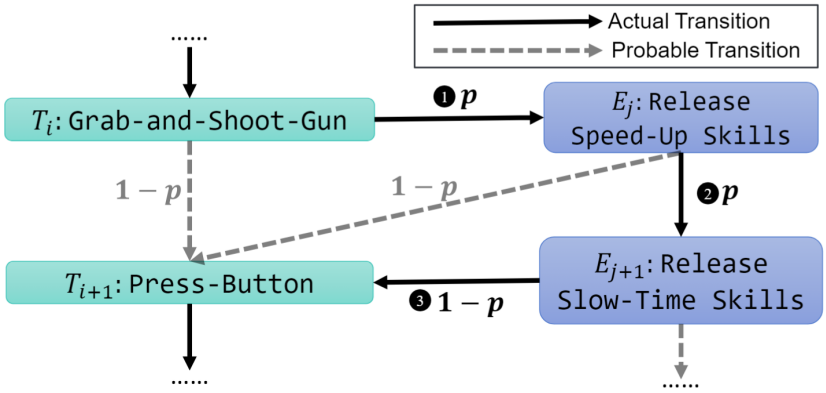


图 3-7 状态转移示例

在导航的路径规划过程中，我们提供了两种路径规划算法：贪心算法（Greedy algorithm）：基于最短路径原则的局部优化方法；回溯剪枝算法（Backtracking with Pruning）：用于全局最优解搜索的方法。在后续实验中，我们选择贪心算法作为基准。

综上，VRExplorer 持续获取场景信息，决策当前要执行的行为，执行行为并接收反馈，重复此过程，直至覆盖所有交互对象和函数。

第 4 章 实验设计与结果分析

本章系统地评估了 VRExplorer 在自动化测试虚拟现实应用中的有效性与优势。我们设计了一系列实验，旨在从多个维度验证该工具的性能，包括与现有基线方法的对比、在不同类型 VR 项目中的适应性，以及其内部模块对整体覆盖性能的影响。首先，我们提出了三个核心研究问题，并围绕这些问题制定实验方案和参数配置。随后，我们选取具有代表性的 VR 项目，采用标准化评估指标，全面衡量 VRExplorer 在代码覆盖率、可交互对象探索能力以及效率方面的表现。通过深入的对比与分析，我们希望从实证层面验证 VRExplorer 的优势，并为未来的 VR 测试方法设计提供参考依据。

4.1 实验研究问题

为了评估 VRExplorer 的性能，我们进行了实验，研究以下三个问题：

问题 1： VRExplorer 在效率和性能方面与基线方法相比如何，特别是在方法覆盖率、ELOC 覆盖率以及 VR 场景中的可交互对象覆盖率方面？

问题 2： VRExplorer 在可扩展性和通用性方面的表现如何，特别是在应用于使用不同 VR 生态系统开发包和插件的 VR 项目时？

问题 3：EAT 框架中的不同模块如何影响 VRExplorer 的代码覆盖效果？

4.2 Baseline 方法

由于 VR 应用具有复杂的 VR 交互和碎片化的开发生态系统等特点，当前的自动化游戏测试工具和 Android 应用测试框架并不适用于 VR 应用测试。基于已有研究，我们选择 VRGuide 作为基线方法，这是一种最先进的 VR 应用自动化测试方法。研究表明，VRGuide 在方法覆盖率和可交互对象覆盖率方面优于 VRTest 的 VRMonkey 和 VRGreed 方法。

4.3 参数

由于 VRExplorer 和 VRGuide 都是模拟玩家探索场景的，因此测试过程中的移动速度、转弯速度和初始位置都会在一定程度上影响交互对象的覆盖效率。虽然较快的速度通常可以提高性能，但过快的速度可能会带来物理和交互问题（例如，测试工

具可能会由于惯性而无意中脱离平台）。因此，我们的目标是将测试速度保持在一个合理的范围内，以平衡效率和稳定性。另外，转换到下一状态的概率由参数p决定。

4.4 评估指标

参考已有研究，我们选择可执行代码行覆盖率（ELOC）、方法覆盖率、可交互对象覆盖率以及收敛时间成本作为评估指标。ELOC 覆盖率与方法覆盖率：我们使用 Unity 官方提供的 Code Coverage 工具自动记录 C# 脚本的代码覆盖情况。测试工具运行于 Unity 编辑器模式，覆盖数据会以 .html 格式的历史报告以及 .xml 格式的汇总数据导出。可交互对象覆盖率：与 VRGuide 略有不同，VRGuide 主要定义可交互对象为那些能够接收鼠标点击事件的对象，而在实际 VR 场景中，我们认为更合理的定义是所有能够触发、抓取、移动并生成 VR 交互事件的对象。例如，可以被抓取和开火的枪、用于开门的钥匙或控制玩具车的遥控器。这些对象通过人工分析场景中的 MonoBehaviour 脚本进行确认。收敛时间成本：当测试工具不再尝试探索场景的更多部分时，即认为进入收敛状态。收敛时间成本指的是测试工具达到该状态所需的时间。

表 4-1 不同速度参数的实验结果

Approach	Parameters		Metrics			
	Move Speed (m/s)	Turn Speed (deg/s)	ELOC Coverage(%)	Method Coverage(%)	Interactable Object Coverage(%)	Convergence Time Cost(s)
VRGuide	3	30	66.53	70.59	91.43	175.0
	6	30	66.53	70.59	94.29	180.0
	6	60	66.53	70.59	94.29	145.0
VRExplorer	3	30	81.67	82.35	100.00	163.9
	6	30	81.67	82.35	100.00	81.5
	6	60	81.67	82.35	100.00	81.4

4.5 速度参数的预实验

为了探索合理的速度参数，我们在 unity-vr-maze 项目上测试了三组参数：移动速度 3 m/s，转向速度 30 度/s；移动速度 6 m/s，转向速度 30 度/s；移动速度 6 m/s，转向速度 60 度/s。

所有实验的初始位置均设为 (0,4.5,0)，确保测试工具合法地站在地面上。如图 4-1 和表 4-1 不同速度参数的实验结果所示，实验结果表明，转向速度对效率影响较小，而移动速度影响较为明显。

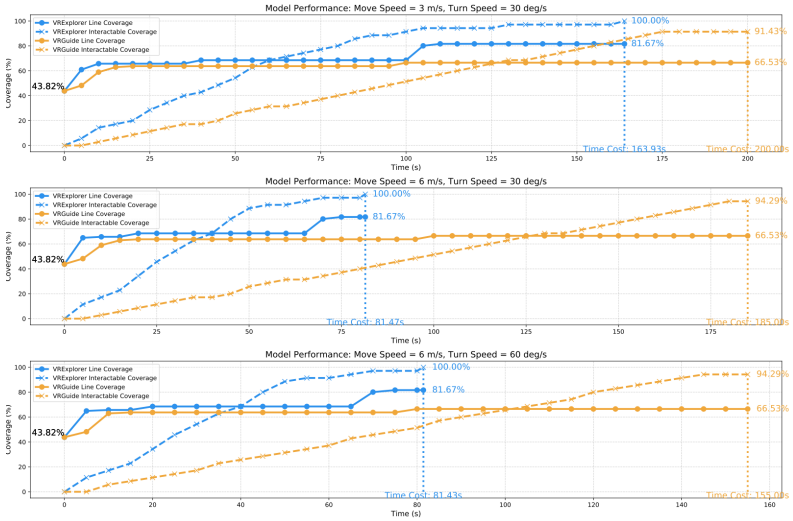


图 4-1 unity-vr-maze 项目在不同速度参数上的覆盖性能

4.6 实验项目选择说明

对于问题 1，我们从文献[44]中的研究对象中选择了三个项目，因为它们包含点击事件，VRGuide 和 VRExplorer 都可以对其进行测试，从而可以直接比较效率和性能。

对于问题 2，我们从收集到的 VR 数据集中选择了六个具有代表性的项目。这些项目包含各种复杂的交互和多样的场景，涵盖各种类型和 VR 生态系统开发包和插件，包括 XRIT、MRTK、SteamVR、Oculus XR Plugin、Open XR Plugin 等。这一选择确保了对 VRExplorer 适应性和多功能性的全面评估。

对于问题 3，我们对这六个项目中的两个进行了深入的消减研究，因为它们具有多种交互类型。通过系统地删除 EAT 框架的特定模块，我们评估了对性能的影响。这一分析有助于量化 EAT 框架各模块的影响，为我们的方法设计提供了实证依据。

4.7 实验环境与配置

4.7.1 硬件实验环境

我们在 Windows 11 计算机上运行实验，具体配置如下：Unity 版本：2021.3.13f1c1 LTS；开发工具：Visual Studio 2022；处理器：AMD Ryzen 7 5800H with Radeon Graphics；内存：32GB RAM；显卡：NVIDIA GeForce RTX 3060 Laptop GPU（6GB）。

4.7.2 实验参数配置

参数配置方面，为了兼顾效率和实用性，我们选择（移动速度 ＝6 米/秒，转弯速度 ＝60 度/秒）作为 VRGuide 和 VRExplorer 在所有后续实验中的标准参数。我们将初始位置设置为场景中心，这是最恰当的。由于所有测试工具的这两个参数设置相同，因此在后续实验结果中不再详细赘述。当测试工具达到收敛状态时，实验将终止。参数 p 设为 0.5，表示两种状态转换的倾向相同。

4.7.3 接口实现

对于每个待测试项目，我们都会人工分析场景中的可交互对象，并挑选出对象引用的所有与 VR 交互逻辑相关的代码（称为核心代码，Core Code），然后人工选择并实现 VRExplorer 的实体层接口，并附加我们预先定义的 Mono 脚本。减少内部有效性威胁的具体措施将总结中讨论。

4.7.4 消融研究配置

在我们的方法中，模块（Module）代表了 EAT 框架三层和 Mono 层中特定行为的相应部分。在 问题 1 和 问题 2 中，我们使用了完整的模块 VRExplorer，而在问题 3 中，为了探索 EAT 框架的不同模块对其覆盖性能的影响，我们删除了待测项目中的特定部分模块，例如，删除可触发模块会删除 Entity 层的接口 Triggerable、Action 层的类 TriggerAction、Task 层、所有涉及触发的任务以及 Mono 层 `XRTriggerable.cs` 中预定义的 Mono 脚本。

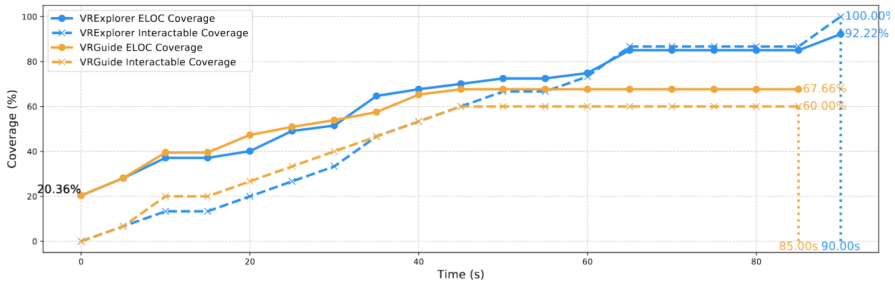
4.8 问题 1 结果分析

表 4-2 问题 1 结果

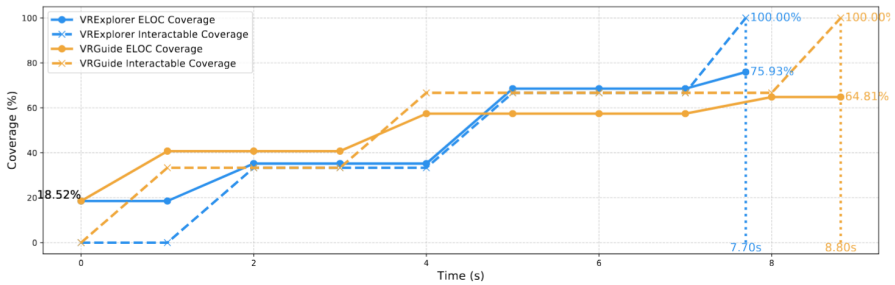
Project Name	Approach	Metrics				Other Information		
		ELOC Coverage (%)	Method Coverage (%)	Interactable Object Coverage(%)	Convergence Time Cost(s)	Core LOC/ Total LOC	Interactable Count	Bugs Reported
unity-vr-maze	VRGuide	66.53	70.59	94.29	145.0	496/23,229	35	0
	VRExplorer	81.67	82.35	100.00	81.4			0
UnityCityView	VRGuide	67.66	78.38	60.00	45.0	247/28,080	15	1
	VRExplorer	92.22	100.00	100.00	89.3			1
UnityVR	VRGuide	64.81	84.62	100.00	8.8	134/22,878	3	2
	VRExplorer	75.93	92.31	100.00	7.7			2

如前文所言，表 4-1 展示了 unity-vr-maze 在不同速度参数下的覆盖性能。三个参数的结论一致：VRExplorer 达到收敛状态所需时间比 baseline 方法 VRGuide 更短，并且在 ELOC 覆盖率和可交互对象覆盖率方面表现更好。图 4-2 展示了

UnityCityView 和 UnityVR 的覆盖性能。在图 4-2(a)中，尽管 VRGuide 更早收敛，但其 ELOC 覆盖率和可交互对象覆盖率明显低于 VRExplorer。在图 4-2(b) 中，VRExplorer 同样在收敛时间上优于 VRGuide，并且其 ELOC 覆盖率和可交互对象覆盖率也更高。



(a) Coverage Performance of UnityCityView



(b) Coverage Performance of UnityVR

图 4-2 UnityCityView 和 UnityVR 的覆盖率测试结果

具体来看，表 4-2 展示了 VRExplorer 在多个指标上优于 VRGuide 的表现。对于 unity-vr-maze，ELOC 覆盖率提升了 22.8%（计算方式为：(81.67 - 66.53) / 66.53 × 100%），方法覆盖率提升了 16.7%（(82.35 - 70.59) / 70.59 × 100%），可交互对象覆盖率提升了 6.1%（(100.00 - 94.29) / 94.29 × 100%）。与此同时，收敛时间减少了 43.9%（(81.4 - 145.0) / 145.0 × 100%）。

对于 UnityCityView，ELOC 覆盖率提升了 36.3%（(92.22 - 67.66) / 67.66 × 100%），方法覆盖率提升了 27.6%（(100.00 - 78.38) / 78.38 × 100%），可交互对象覆盖率提升了 66.7%（(100.00 - 60.00) / 60.00 × 100%）。不过，由于 VRGuide 的可交互对象覆盖率仅为 60%，使其更早收敛，因此 VRExplorer 的收敛时间几乎增加了一倍。

对于 UnityVR，ELOC 覆盖率提升了 17.1%（(75.93 - 64.81) / 64.81 × 100%），方法覆盖率提升了 9.1%（(92.31 - 84.62) / 84.62 × 100%），收敛时间减少了 12.5%（(8.8 - 7.7) / 8.8 × 100%）。两个方法均实现了 100% 的可交互对象覆盖率。

实验结果表明，VRExplorer 在效率和覆盖性能方面持续优于 VRGuide。它在所有评估的 VR 项目中都实现了更高的 ELOC 覆盖率、方法覆盖率和可交互对象覆盖率。在 UnityCityView 中，尽管收敛时间较长，VRExplorer 仍显著提高了覆盖率。而在 unity-vr-maze 和 UnityVR 中，VRExplorer 不仅提高了覆盖率，还减少了收敛时间，展示出其在 VR 测试中的高效性。

回答问题 1：VRExplorer 在覆盖率方面优于 VRGuide，实现了更高的 ELOC、方法和可交互物体覆盖率，同时大部分情况下能提高场景探索效率。这些结果证实了它在 VR 自动测试方面相比 baseline 方法的有效性。

4.9 问题 2 结果分析

表 4-3 问题 2 结果

Project Name	Approach	Metrics		Other Information	
		ELOC Coverage(%)	Method Coverage(%)	Core LOC / Total LOC	Interactable Count
VR-Basics	VRGuide	41.38	53.22	707/36,738	8
	VRExplorer	80.17	91.93		
VR-Room	VRGuide	40.97	50.63	822/37,772	11
	VRExplorer	77.61	83.54		
VGuns	VRGuide	28.68	38.89	730/46,363	7
	VRExplorer	77.57	77.78		
VR-Adventure	VRGuide	54.12	65.00	225/10,685	3
	VRExplorer	91.76	95.00		
Edutainment-Escape-Room	VRGuide	38.08	58.06	903/15,410	24
	VRExplorer	70.61	88.17		
EscapeGameVR	VRGuide	41.77	55.26	656/7,674	14
	VRExplorer	71.08	73.68		

表 4-3 展示了我们从收集到的 VR 数据集中挑选出的六个具有代表性的项目中，VRExplorer 与 baseline 方法 VRGuide 在性能方面的对比结果。

在实验过程中我们观察到，相比于问题一中使用的项目，这些更具通用性的项目中，VRGuide 由于仅依赖点击操作，其可交互对象覆盖率极为有限，因此讨论可交互对象覆盖率变得没有意义。因此，在分析中我们排除了可交互对象覆盖率和收敛时间成本两个指标，重点关注核心逻辑的代码覆盖情况。

对于 VR-Basics，ELOC 覆盖率提升了 93.8%（(80.17 - 41.38) / 41.38 × 100%），方法覆盖率提升了 72.8%（(91.93 - 53.22) / 53.22 × 100%）。

对于 VR-Room，ELOC 覆盖率提升了 89.4%（(77.61 - 40.97) / 40.97 × 100%），方法覆盖率提升了 65.0%（(83.54 - 50.63) / 50.63 × 100%）。

对于 VGuns，ELOC 覆盖率提升了 170.7%（(77.57 - 28.68) / 28.68 × 100%），方法覆盖率提升了 100.0%（(77.78 - 38.89) / 38.89 × 100%）。

对于 VR-Adventure，ELOC 覆盖率提升了 69.6%（(91.76 - 54.12) / 54.12 × 100%），方法覆盖率提升了 46.2%（(95.00 - 65.00) / 65.00 × 100%）。

对于 Edutainment-Escape-Room，ELOC 覆盖率提升了 85.5%（(70.61 - 38.08) / 38.08 × 100%），方法覆盖率提升了 51.8%（(88.17 - 58.06) / 58.06 × 100%）。

对于 EscapeGameVR，ELOC 覆盖率提升了 70.2%（(71.08 - 41.77) / 41.77 × 100%），方法覆盖率提升了 33.3%（(73.68 - 55.26) / 55.26 × 100%）。

平均来看，ELOC 覆盖率提升了 **96.5%**（(93.8 + 89.4 + 170.7 + 69.6 + 85.5 + 70.2) / 6），方法覆盖率提升了 **61.5%**（(72.8 + 65.0 + 100.0 + 46.2 + 51.8 + 33.3) / 6）。

实验结果表明，VRExplorer 在多个使用不同生态包和插件开发的 VR 项目中都表现出一致的优异性能。相比 VRGuide，VRExplorer 在 ELOC 和方法覆盖率两个代码覆盖指标上均有显著提升。特别是在 VGuns 项目中，ELOC 覆盖率提升高达 170.7%，突显了该工具在复杂多样的 VR 生态系统中处理能力的强大。

对于如 VR-Basics 和 VR-Room 这类主题更为通用的项目，VRExplorer 依然优于 VRGuide，在 ELOC 覆盖率和方法覆盖率方面实现了大幅提升。即使在由于 VRGuide 限制而导致可交互对象覆盖率不具参考价值的情况下，VRExplorer 仍然在代码覆盖方面保持出色表现。总体趋势表明，VRExplorer 能够有效适应各种 VR 项目的测试需求，无论其使用的开发包或插件如何不同。

回答问题 2：VRExplorer 在应用于使用不同 VR 生态系统软件包和插件开发的各种 VR 项目时，表现出卓越的扩展能力和多功能性。在所有评估项目中，VRExplorer 的 ELOC 和方法覆盖率始终优于 VRGuide。这些结果表明，VRExplorer 能够在不同开发环境中测试 VR 应用程序，适用于各种 VR 测试场景。

4.10 问题 3 结果分析

为了理解 EAT 框架中各交互模块的贡献，我们进行了消融实验，通过有选择地禁用 Triggerable、Grabbable 和 Transformable 模块，并将其产生的代码覆盖率与完整的 VRExplorer 系统进行比较。

表 4-4 展示了在两个项目（VR-Basics 和 VR-Room）中，VRExplorer 在不同模块组合下的 ELOC 和方法覆盖率。与完整启用 EAT 的 VRExplorer 相比，移除任何一个模块都会导致覆盖率下降，但不同模块在不同项目中的影响程度不同。

在 VR-Basics 中，完整的 VRExplorer 模块实现了 80.17% 的 ELOC 覆盖率和 91.93% 的方法覆盖率。当移除 Triggerable 模块时，ELOC 下降至 68.10%（下降了 12.07%），方法覆盖率下降至 77.42%（下降了 14.51%）。Transformable 模块的影响相对较小，其移除导致 ELOC 降至 59.28%（下降了 20.89%），方法覆盖率为 77.10%（下降了 14.83%）。这些结果表明，在该项目中，Triggerable 和 Grabbable 类型的交互更为关键。

表 4-4 问题 3 结果

Project Name	Approach	Metrics	
		ELOC Coverage (%)	Method Coverage (%)
VR-Basics	VRGuide	41.38	53.22
	VRExplorer	80.17	91.93
	VRExplorer (remove Triggerable)	68.10	77.42
	VRExplorer (remove Transformable)	59.24	70.00
VR-Room	VRGuide	40.97	50.63
	VRExplorer	77.61	83.54
	VRExplorer (remove Grabbable)	58.52	69.62
	VRExplorer (remove Triggerable)	64.12	67.00

相反地，在 VR-Room 中，移除 Grabbable 模块造成了最显著的影响。完整的 VRExplorer 实现了 67.34% 的 ELOC 和 83.61% 的方法覆盖率。移除 Grabbable 模块后，ELOC 降至 50.80%（下降了 16.54%），方法覆盖率降至 69.91%（下降了 13.70%）。移除 Triggerable 模块也对性能产生了显著影响：ELOC 降至 55.66%（下降了 11.68%），方法覆盖率降至 67.12%（下降了 16.49%）。

这些结果表明，不同类型的交互对覆盖率的贡献因项目交互设计的差异而不同。然而，在两个项目中，移除任意模块都会导致明显的覆盖率下降，进一步验证了整合三种交互感知模块的重要性。完整的 EAT 框架始终实现了最高的覆盖率。

回答问题 3：从 EAT 框架中移除任何模块都会导致覆盖率下降，但不同项目和交互类型的影响程度不同。可触发模块和可抓取模块对 ELOC 和方法覆盖率的影响最为显著。完整的 EAT 框架在所有情况下都达到了最高的覆盖率，验证了结合所有交互模块的必要性，也验证了我们方法设计的有效性和合理性。

第 5 章 总结与展望

本章对我们的工作进行总结，并分析内在和外在潜在的局限性因素，以及未来工作的后续展望。

5.1 工作总结

本论文针对当前虚拟现实（VR）应用程序在自动化测试领域存在的覆盖率不足、探索能力有限等问题，基于我们设计的面向 VR 场景的 EAT 测试框，提出了一种基于交互建模的自动化测试方法 VRExplorer。主要工作总结如下：

首先，我们进行了项目收集与特征分析：选取了多个具有代表性的 Unity VR 项目，提取并分析了其中常见的交互行为与对象类型，为测试建模提供依据。

接着，我们阐述了 EAT 框架设计与实现：构建了包含 Entity、Action 和 Task 等层次的交互行为建模体系，全面覆盖 Triggerable、Grabbable、Transformable 等多种交互方式，提升了测试系统的通用性与扩展性。

然后，我们介绍了实验测试流程设计与实现：设计了从场景配置、接口注入到行为驱动探索的一体化流程，使测试系统能够自动识别关键对象并完成有效交互操作。

在实验分析时，我们进行对比实验与评估：通过与 Baseline 方法 VRGuide 的对比实验，验证了 VRExplorer 在 ELOC 覆盖率、方法覆盖率和时间效率等多个维度的显著优势，尤其在处理复杂交互与通用 VR 生态上表现出更强的适应性与鲁棒性。我们还进行了消融实验分析，进一步通过模块消融研究，分析了各类交互模块在实际项目中的贡献，验证了 EAT 框架中各子模块的必要性与有效性。

5.2 局限性分析与未来展望

5.2.1 外部有效性威胁

尽管我们已经覆盖了尽可能多的 VR 开发插件包和工具，如 XRIT、MRTK、SteamVR、Oculus XR 插件、Open XR 插件等，但仍然存在一些未涵盖的其他开发工具。此外，我们的框架目前仅适用于 Unity 下的 VR 应用，无法推广到其他引擎，例如 Unreal Engine 下的 VR 应用。

5.2.2 内部有效性威胁

内部效度的主要威胁来自于核心代码的选择和接口的实现。为了尽量减少错误并确保有效性，在核心代码选择阶段，我们安排了 4 名测试人员单独选择代码，然后讨论最终结果。在接口实现阶段，我们让 4 名测试人员一起讨论，确保获得一致的定制实现。

5.2.3 局限性与未来工作

目前，VRExplorer 仍需要手动分析场景并实现定制化接口，这仍然会带来一定的工作量和一定的学习门槛。

未来的工作一方面，我们希望将强化学习、多模态大模型植入 VRExplorer，从而通过场景的视觉分析、测试人员的自然语言分析等方式，使其能够更加智能化地实现测试工具，并基于强化学习进行自主学习；另一方面，我们将继续在更多第三方 VR 开发库和其他引擎中进行实验测试。

参考文献

[1] Radoeva R, Petkov E, Kalushkov T, et al. Overview on hardware characteristics of virtual reality systems[C].2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). IEEE, 2022: 01-05.

[2] Basu A. A brief chronology of Virtual Reality[J]. arXiv preprint arXiv:1911.09605, 2019.

[3] Hamilton D, McKechnie J, Edgerton E, et al. Immersive virtual reality as a pedagogical tool in education: a systematic literature review of quantitative learning outcomes and experimental design[J]. Journal of Computers in Education, 2021, 8(1): 1-32.

[4] Liu Y, Sun Q, Tang Y, et al. Virtual reality system for industrial training[C].2020 International conference on virtual reality and visualization (ICVRV). IEEE, 2020: 338-339.

[5] Muhanna M A. Virtual reality and the CAVE: Taxonomy, interaction challenges and research directions[J]. Journal of King Saud University-Computer and Information Sciences, 2015, 27(3): 344-361.

[6] Rajeswaran P, Varghese J, Kumar P, et al. Airwayvr: Virtual reality trainer for endotracheal intubation[C].2019 IEEE conference on virtual reality and 3d user interfaces (VR). IEEE, 2019: 1345-1346.

[7] Fortune Business Insights[EB/OL]. Virtual Reality (VR) Market Size, Share & Industry Analysis, By Component (Hardware, Software, and Content), By Device Type (Head Mounted Display (HMD), VR Simulator, VR Glasses, Treadmills & Haptic Gloves, and Others), By Industry (Gaming, Entertainment, Automotive, Retail, Healthcare, Education, Aerospace & Defense, Manufacturing, and Others), and Regional Forecast, 2024.[2024-09-04]. https:.www.fortunebusinessinsights.com/industry-reports/virtual-reality-market-101378.

[8] Rzig D E, Iqbal N, Attisano I, et al. Virtual reality (vr) automated testing in the wild: A case study on unity-based vr applications[C].Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2023: 1269-1281.

[9] XR Interaction Toolkit[EB/OL]. https:.docs.unity3d.com/Packages/com.unity.xr.inter-

action.toolkit@3.0/manual/index.html

[10] Juránek V[EB/OL]. Virtual reality toolkit for the Unity game engine[OL]. 2021 [2025-01-24]. https:.theses.cz/id/6zl8up/.

[11] SteamVR[EB/OL]. 2024. https:.github.com/ValveSoftware/steamvr_unity_plugin.

[12] Bovet S, Kehoe A, Crowley K, et al. Using traditional keyboards in vr: Steamvr developer kit and pilot game user study[C].2018 IEEE Games, Entertainment, Media Conference (GEM). IEEE, 2018: 1-9.

[13] Valve Corporation[EB/OL]. 2024. SteamVR Plugin. https:.assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647.

[14] Mixed Reality Toolkit[EB/OL].2022. https:.github.com/microsoft/MixedRealityToolkit-Unity.

[15] Ong S, Siddaraju V K, Ong S, et al. Introduction to the mixed reality toolkit[J]. Beginning Windows Mixed Reality Programming: For HoloLens and Mixed Reality Headsets, 2021: 85-110.

[16] VR Toolkit[EB/OL]. 2024. https:.www.vrtk.io.

[17] Baruah R. Virtual Reality with VRTK4: Create Immersive VR Experiences Leveraging Unity3D and Virtual Reality Toolkit[M]. Apress, 2019.

[18] Chen K, Li Y, Chen Y, et al. Glib: towards automated test oracle for graphically-rich applications[C].Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2021: 1093-1104.

[19] Macklon F, Taesiri M R, Viggiato M, et al. Automatically detecting visual bugs in HTML5 Canvas Games[C].Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. 2022: 1-11.

[20] Prasetya I, Voshol M, Tanis T, et al. Navigation and exploration in 3D-game automated play testing[C].Proceedings of the 11th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation. 2020: 3-9.

[21] Prasetya I, Dastani M, Prada R, et al. Aplib: Tactical agents for testing computer games[C].International Workshop on Engineering Multi-Agent Systems. Cham: Springer International Publishing, 2020: 21-41.

[22] Prasetya I, Pastor Ricós F, Kifetew F M, et al. An agent-based approach to automated game testing: an experience report[C].Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation. 2022: 1-8.

[23] Shirzadehhajimahmood S, Prasetya I, Dignum F, et al. Using an agent-based approach for robust automated testing of computer games[C].Proceedings of the 12th International Workshop on Automating TEST Case Design, Selection, and Evaluation. 2021: 1-8.

[24] Chen Y, Wang S, Tao Y, et al. Model-Based GUI Testing for HarmonyOS Apps[C].Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. 2024: 2411-2414.

[25] Arkxtest[EB/OL]. 2022. https:.gitee.com/openharmony/testfwk_arkxtest.

[26] Hu Y, Jin H, Wang X, et al. Autoconsis: Automatic gui-driven data inconsistency detection of mobile apps[C].Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice. 2024: 137-146.

[27] Lv Z, Peng C, Zhang Z, et al. Fastbot2: Reusable automated model-based gui testing for android enhanced by reinforcement learning[C].Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. 2022: 1-5.

[28] Tufano R, Scalabrino S, Pascarella L, et al. Using reinforcement learning for load testing of video games[C].Proceedings of the 44th international conference on software engineering. 2022: 2303-2314.

[29] Ferdous R, Kifetew F, Prandi D, et al. Towards agent-based testing of 3D games using reinforcement learning[C].Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. 2022: 1-8.

[30] Bergdahl J, Gordillo C, Tollmar K, et al. Augmenting automated game testing with deep reinforcement learning[C].2020 IEEE Conference on Games (CoG). IEEE, 2020: 600-603.

[31] Zheng Y, Xie X, Su T, et al. Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning[C].2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2019: 772-784.

[32] Nusrat F, Hassan F, Zhong H, et al. How developers optimize virtual reality applications: A study of optimization commits in open source unity projects[C].2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021: 473-485.

[33] Guo H, Dai H N, Luo X, et al. An empirical study on oculus virtual reality applications: Security and privacy perspectives[C].Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. 2024: 1-13.

[34] Prasetya I, Voshol M, Tanis T, et al. Navigation and exploration in 3D-game automated play testing[C].Proceedings of the 11th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation. 2020: 3-9.

[35] Harms P. Automated usability evaluation of virtual reality applications[J]. ACM Transactions on Computer-Human Interaction (TOCHI), 2019, 26(3): 1-36.

[36] Rzig D E, Iqbal N, Attisano I, et al. Virtual reality (vr) automated testing in the wild: A case study on unity-based vr applications[C].Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2023: 1269-1281.

[37] Wang X. VRTest: an extensible framework for automatic testing of virtual reality scenes[C].Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings. 2022: 232-236.

[38] Wang X, Rafi T, Meng N. Vrguide: Efficient testing of virtual reality scenes via dynamic cut coverage[C].2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2023: 951-962.

[39] Qin X, Weaver G. Utilizing generative ai for vr exploration testing: A case study[C].Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering Workshops. 2024: 228-232.

[40] Rafi T, Zhang X, Wang X. Predart: Towards automatic oracle prediction of object placements in augmented reality testing[C].Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. 2022: 1-13

致　　谢

写完这篇论文，我的本科生涯也即将画上句点。回首四年，有太多人值得感谢。

首先感谢我的父母，你们无私的爱与支持是我一路走来的最大动力。感谢我的女友，你的陪伴让这段旅程充满温暖。

特别感谢南京师范大学的各位老师：段博佳老师在大一大二时为我指明了生涯规划方向；王琼老师在算法竞赛上给予我专业指导；刘日晨老师带我走进科研的世界。感谢吉根林、李峻、宋歌、谢捷等老师在专业课程上的悉心教导，感谢辅导员陆思冰老师的关心帮助。还要感谢夏泽禹学长和彭章荣学长，你们在实习和算法学习上给予的指导让我受益匪浅。

感谢季润民同学、杨大宇同学、杨欣彤同学、马朝阳同学、师铭骏同学、张惠捷同学、从圣元同学在生活和学习上帮助我颇多。感谢徐嘉龙同学、王雅琳学姐、冯暄越学长、张宇辰学长、董文杰学弟、陈子润学弟、陆昊宇学弟、徐耀学弟、朱禹丞学弟在竞赛和项目中的合作。

衷心感谢中山大学郑子彬老师、香港浸会大学戴弘宁老师的指导，感谢郭瀚阳、廖泽钦、吴婧雯、苏健钟等师兄师姐在科研上的帮助。也感谢厦门大学林俊聪老师、东南大学徐翔宇老师在保研过程中给予的支持。

最后，感谢所有未能一一提及的老师、同学和朋友。你们的善意与帮助，我将永远铭记于心。