将复杂移动应用用户界面手势的视频录制内容转换为可重放场景

Carlos Bernal-C´ardenas，*IEEE* 会员，Nathan Cooper，*IEEE* 学生会员，Madeleine Havranek，*IEEE* 学生会员，Kevin Moran，*IEEE* 会员，Oscar Chaparro，*IEEE* 会员，Denys Poshyvanyk，*IEEE* 会员，以及 Andrian Marcus，*IEEE* 会员

摘要——移动应用的屏幕录制内容很容易获取，并且包含大量与软件开发人员相关的信息（例如，缺陷或功能请求），因此成为众包应用反馈的一种常见方式。因此，这些视频正逐渐成为开发人员必须管理的常见软件制品。鉴于移动开发的独特约束，包括快速的发布周期和迅速演进的平台，自动分析各种丰富软件制品的技术有助于移动开发人员。遗憾的是，与其他类型的（文本）制品相比，由于屏幕录制内容具有图形特性，自动分析它们面临着严峻挑战。为应对这些挑战，本文提出 V2S+，一种将 Android 应用使用情况的视频录制内容转换为可重放场景的自动化方法。V2S+ 主要基于计算机视觉技术，并改进了近期的目标检测和图像分类解决方案，以检测和分类视频中捕获的用户手势，并将其转换为可重放测试场景。由于 V2S+ 采用基于计算机视觉的方法，因此适用于混合型和原生安卓应用程序。我们对 V2S+ 进行了广泛评估，使用了 243 个视频，其中描绘了从用户操作的功能及复现的缺陷中收集的 4,028 个基于图形用户界面的操作；这些视频来自 90 多款热门原生和混合型安卓应用。结果表明，V2S+ 能够根据屏幕录制内容准确回放场景，并能够以较低开销，在实体设备上的原生应用程序场景中复现 ≈ 90.2%的连续操作，以及在模拟器上的混合型应用程序场景中复现 ≈ 83%的连续操作。与三家工业合作伙伴开展的案例研究展示了从开发人员角度看 V2S+ 的潜在效用。

索引词—缺陷报告、屏幕录制、目标检测。

arXiv:2301.01191v1 [cs.SE] 3 Jan 2023

1 引言

MOBILE 应用程序开发人员依靠各种软件制品，在整个开发过程中做出明智的决策。这些信息来源包括用户评论、崩溃报告、缺陷报告和电子邮件等。截图或屏幕录制等图形信息正日益成为这些软件制品的常见组成部分 [1]。这主要是因为这类信息相对容易收集，而且由于移动应用以图形用户界面为驱动的特性，其中包含丰富的信息，能够轻松展示复杂的概念，例如缺陷或特征请求。事实上，许多众包测试和缺陷报告框架都内置了屏幕录制功能，帮助开发人员收集移动应用使用数据并记录故障 [2], [3], [4], [5]。屏幕录制可用于呈现应用程序的使用情况，开发人员可借助这些制品：（i）了解用户与应用的交互方式 [6], [7]；（ii）处理终端用户提交的缺陷报告和功能请求 [8]；以及（iii）辅助理解与测试相关任务中的错误 [9]。然而，尽管可视化移动开发制品日益普及，开发人员仍必须手动检查并解读屏幕截图和视频，以获取相关信息，这既耗时，也容易产生歧义。理解过程所需的人工工作，使开发工作流程更加复杂，而该流程本已受到语言二分性 [10] 以及移动软件特有的若干挑战制约，包括：（i）频繁发布带来的压力 [11], [12], ；（ii）平台和 API 的快速演进 [13], [14], ；（iii）来自用户的持续嘈杂反馈 [15], [16], [17], [18], [19],；以及（iv）移动设备生态系统的碎片化 [20],[21], [22] 等 [23]。自动处理图形软件制品十分必要，这将有助于开发人员将精力转向核心开发任务。

为了改进并自动化对视频相关移动开发产物的分析，我们推出了视频转场景+(V2S+)，这是一种易于使用且模块化的自动化方法，可将 Android 应用使用情况的视频屏幕录制转换为可重放场景。我们设计 V2S+，使其仅依赖于从安卓设备录制的视频文件，因此主要采用计算机视觉技术。该方法基于我们此前发表的工作 [24]。V2S+ 将最新的深度学习（DL）目标检测和图像分类模型加以调整，以准确检测并分类在屏幕上执行的不同类型的用户操作

• *C. Bernal-C´ardenas* 任职于微软雷德蒙德的 *Liquid* 团队，地址：美国华盛顿州雷德蒙德，邮编 *98052*。电子邮件：*carlosbe@microsoft.com*

• *N. Cooper*、*M. Havranek*、*O. Chaparro* 和 *D. Poshyvanyk* 任职于威廉与玛丽学院计算机科学系，地址为弗吉尼亚州威廉斯堡市，邮编 *23185*。电子邮箱： *[nacooper01*、*mrhavranek]@email.wm.edu*、*oscarch@wm.edu*、*denys@cs.wm.edu*

• *K. Moran* 任职于乔治梅森大学计算机科学系，地址为弗吉尼亚州费尔法克斯市，邮编 *22030*。电子邮箱：*kpmoran@gmu.edu*

• *A. Marcus* 任职于德克萨斯大学达拉斯分校计算机科学系，地址为德克萨斯州理查森市，邮编 *75080*。电子邮箱：*amarcus@utdallas.edu*

稿件收到日期：*2021*年*10*月；

随后，这些分类动作会被转换为可重放场景，从而能够在目标设备上自动重现用户交互（来自视频），使 V2S+ 成为首个纯图形化的安卓录制与回放技术。

除了帮助自动处理移动开发产物中已有的图形数据外，V2S+ 还可用于改进或增强目前尚未充分利用屏幕录制的其他开发任务，例如创建和维护自动化图形用户界面测试套件，以及众包功能测试和可用性测试。

我们对 V2S+ 进行了全面评估，以考察其在根据用户与原生应用和混合应用交互时录制的视频生成可重放场景方面的准确性、鲁棒性、效率和实用性。用户使用多种手势与这些应用进行交互，包括单指和多指手势。我们在实体设备上的一组热门原生安卓应用中评估了 V2S+ 。用户使用单指手势，在 Google Play 市场中 32 个类别的热门应用上重现缺陷并体验了多项特征。我们还使用用户在 2 个类别的热门应用中执行多指动作时录制的视频（例如，双指旋转、双指缩放等。）评估了 V2S+ 。我们选择分别开展研究，以评估 V2S+在原生应用和混合应用上的表现，从而考察其对这些应用的普适性，并凸显 V2S+对混合应用的支持，因为这类应用可用的图形用户界面测试工具相对较少。混合应用采用 Web 标准（例如，HTML 和 CSS）与原生软件开发工具包相结合的方式构建，面向多个平台（例如，iOS 和安卓），而原生应用仅使用原生软件开发工具包构建，且面向单一平台 [25]。与其他测试工具（例如，CRAFTDROID [26] 和 MONKEY [27]）不同，V2S+ 无需任何图形用户界面元数据，而是利用检测到的屏幕上每个触摸事件的空间位置；无论应用框架如何（Web 和/或原生），这一做法都可用于应用测试。对于这些混合应用，我们分析了 V2S+在模拟设备上 7 个类别的 14 款热门应用中录制的用户场景上的表现。我们使用为所有这些场景收集的视频录制内容评估了 V2S+ 。我们还评估了该技术的开销，并与三家工业合作伙伴开展案例研究，以了解 V2S+的实际适用性。

我们的评估结果表明，V2S+ 具有准确性，能够在收集的视频中正确重现 ≈ 83% - 90% 的事件。该方法也具有鲁棒性，适用于目前 Google Play 上众多流行的原生应用和混合应用。就效率而言，我们发现 V2S+ 带来的开销在可接受范围内，且开发人员认为它可能很有用。

总而言之，我们工作的主要贡献如下：

- V2S+，这是首个完全基于应用使用屏幕录制内容运行的安卓录制与回放方法。V2S+ 采用计算机视觉领域的目标检测和图像分类解决方案，以有效识别并分类单指和多指用户操作
  n the video f框架 o f 屏幕录制 ding;
- 用于从屏幕录制内容中识别用户交互的数据集生成和模型训练自动化流水线。该流水线已在我们的在线附录中公开 [28]，并且完全自动化且附有文档，供他人使用和改进。
- 对 V2S+ 进行广泛实证评估的结果，评估了其在来自 90 多个应用程序的 243 个视频中的准确性、鲁棒性和效率；
- 与三家开发商业应用的工业合作伙伴开展的案例研究结果，突出了 V2S+ 的潜在实用价值，以及有待改进和扩展的领域；
- 在线附录 [28]，包含由 V2S+ 重放的视频示例、实验数据、源代码、训练好的模型以及我们的评估基础设施，以促进该方法和评估结果的可复现性。

本文是在第 42 届国际软件工程大会（ICSE'20） [24]上发表的前期研究成果的基础上进行的大幅扩展。扩展内容包括以下新贡献：(i) V2S 的新版本，称为 V2S+，该版本完全使用 Python 实现，能够识别并重放单指和多指手势；(ii) 扩展的视频数据集，其中包含一组新的混合应用，以及包含多指手势的新移动应用使用场景；以及 (iii) 两个新的研究问题，重点评估 V2S+ 重放使用单指和多指手势在原生应用和/或混合应用上录制的视频的能力。

2 背景

我们简要讨论用于图像分类和目标检测的深度学习技术，并将其改编用于 V2S+ 中的触摸/手势识别。

2.1 图像分类

近年来，利用由专用层构成的神经网络的深度学习技术，在将各种图像分类到指定类别方面展现出巨大潜力。借助卷积神经网络（CNN）实现高精度图像识别的先进方法 [29], [30], [31], [32], [33] ，在图像分类任务上的准确率已达到人类水平。

通常，卷积神经网络中的每层 都会对输入模型的数据执行某种形式的计算变换。初始层通常接收一幅输入图像。接下来通常是一个卷积层，它通过对图像像素应用预定义尺寸的滤波器 （也称为 卷积核）来提取图像特征；滤波器中的内容通过成对的矩阵乘法（即 卷积运算）进行变换。每个滤波器都以固定步幅 作为滑动窗口遍历整幅图像，从而提取特征图。卷积层通常与最大池化 层结合使用，以进一步降低网络中传递的数据维度。卷积运算本质上是线性的。由于图像通常是非线性数据源，因此一般会使用修正线性单元（ ReLU）等激活函数

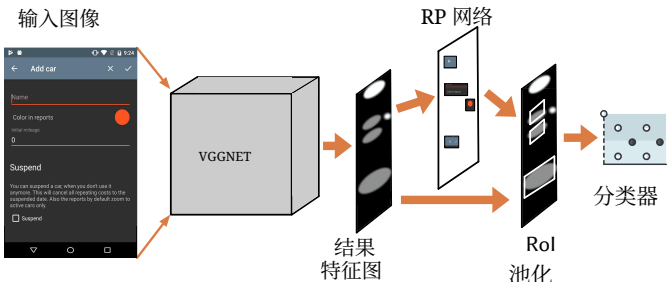


图 1：FASTER R-CNN 架构示意图

来引入一定程度的非线性。最后，全连接层（或一系列此类层）会与*Softmax* 分类器 配合使用，以预测图像类别。卷积神经网络的训练过程通常通过梯度下降更新网络各层之间的权重，并通过反向传播误差梯度来完成。

近年来，一些改进卷积神经网络模型或采用其他架构的新模型，在图像分类方面展现出良好前景。EfficientNet [34] 使用一个独立的神经网络来学习如何构建高效且性能出色的卷积神经网络架构。Transformer [35] 架构传统上仅用于自然语言处理（NLP）任务，如今也通过一种新颖的分词方案应用于图像分类；该方案可将图像转换为一系列视觉词元，而非子词词元 [36]。此外，多层感知机（MLP）在图像分类方面也展现出良好潜力，其性能可与最先进的模型相媲美 [37]。

V2S+ 针对图像片段不透明度分类这一特定任务，实现了一个定制的卷积神经网络，以帮助识别由触摸指示器表示的图形用户界面交互（见第 3.2 节）。

2.2 目标检测

在图像分类任务中，通常会为整张图像分配一个较为宽泛的标签（例如，鸟或人）。然而，图像通常包含多个组成部分，其中包含需要识别的不同物体。

与图像分类类似，近年来，目标检测的深度学习模型取得了长足进步，使目标跟踪和计数，以及人脸和姿态检测等应用成为可能。推动这些进展的最具影响力的神经网络架构之一，是 Girshick 等人 [38]提出的基于区域的卷积神经网络（R-CNN）。R-CNN 架构将图像区域提议（RPs）算法与卷积神经网络的分类能力相结合。图像区域提议算法旨在识别可能包含感兴趣内容的图像区域。R-CNN 使用选择性搜索算法 [39]生成一组区域提议的边界框。随后，将所有识别出的图像区域输入预训练的 ALEXNET [29]（即特征提取器），将图像特征提取为向量。然后将这些向量输入支持向量机（即分类器），以判定每个图像区域是否包含感兴趣的类别。最后，使用贪心非极大值抑制算法（即回归器）选出重叠区域中分类为目标物体且可能性最高的区域。最近，DETR [40] 被提出，作为一种使用卷积神经网络和编码器-解码器 Transformer 架构进行目标检测的端到端方法。DETR 无需 R-CNN 架构所需的许多手工启发式规则，使架构更加简单；作者发现，其准确率和速度均与超优化的 FASTER R-CNN [41]架构相当。

我们采用 FASTER R-CNN 架构（图 1），它通过引入一个独立的神经网络来预测图像区域提议，从而改进了 R-CNN 架构。通过将区域提议网络的训练整合到网络的端到端训练中，模型的速度和准确率均得到提升。对于 V2S+，我们调整 FASTER R-CNN 模型，以检测视频帧中代表用户操作的触摸指示器（参见第 3.2.2 节）。

3 V2S 方法

本节概述 V2S+ 方法，该方法可将安卓屏幕录制自动转换为可重放的场景。图 2 展示了 V2S+ 的执行流程，该流程分为三个主要阶段：(i) 触摸检测阶段，用于识别输入视频每一帧中的用户触摸；(ii) 动作分类阶段，将检测到的触摸分组并分类为离散的用户操作（即，轻触、长按、手势和多指手势）；以及 (iii) 场景生成阶段，将这些操作导出并格式化为可重放的脚本。

V2S [24] 最初使用 Java、Bash 和 Python 实现。此后，我们重新设计了 V2S+ 结构，采用纯 Python 实现，并兼顾可扩展性和模块化。现在，V2S+的每个阶段和组件都可以通过抽象类轻松替换和/或扩展，从而使整个流水线能够灵活适应各种研究和开发情境。在详细讨论各个阶段之前，我们先介绍方法中的一些初步内容、输入规范和要求。

3.1 输入视频规范

为了让 V2S+ 能够处理视频，视频必须满足一些要求，以确保我们的计算机视觉（CV）模型能够正常运行。首先，视频帧尺寸必须与目标安卓设备的全分辨率屏幕尺寸一致，才能兼容指定的预训练目标检测网络。过去几年推出的几乎所有现代安卓设备都满足这一要求。这些视频既可以通过内置的 Android 屏幕录制工具录制，也可以借助第三方应用程序 [42]录制。第二，输入视频的录制帧率必须至少达到 30“帧每秒”（每秒帧数），而大多数现代安卓设备同样都能满足或超过这一要求。这是因为帧率直接影响在视频帧中解析“快速”手势（例如，快速滚动）实际动作的准确度。第三，视频应从应用程序打开后的初始屏幕/状态，或从设备的主屏幕开始录制。这样可以确保重现过程中不会

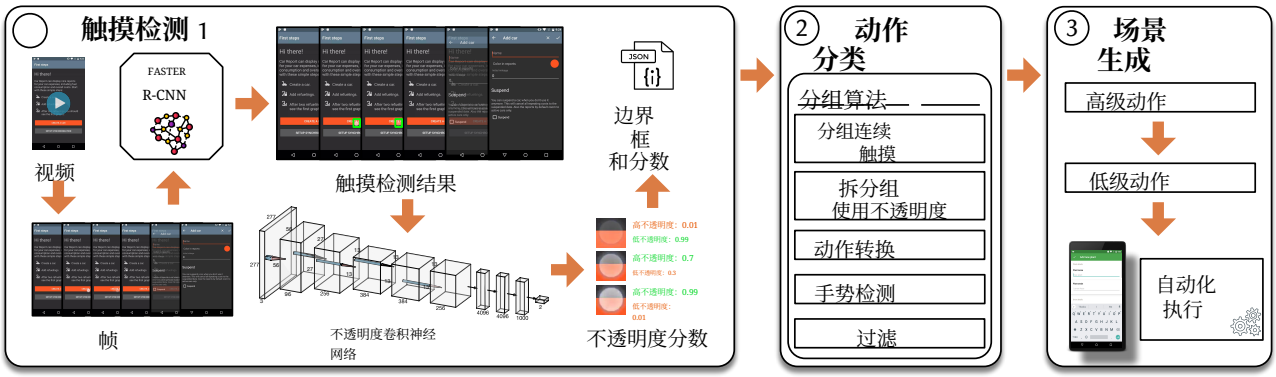


图2：V2S+ 方法及各阶段（编号）

遗漏重现视频所需的任何动作。最后，录制视频时必须在设备上启用“显示触摸”选项，该选项可通过高级设置菜单访问 [43]，并且至少从安卓4.1起，几乎所有安卓设备默认都提供此选项。启用该选项后，屏幕上会显示触摸指示器，这是一个半透明的小圆圈，可在用户用手指触摸设备屏幕时提供视觉反馈。用户首次触摸屏幕时，指示器的不透明度为完全不透明；手指离开屏幕后，指示器则会逐渐淡去（参见图 3）。

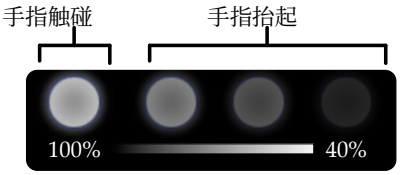


图 3：触摸指示器不透明度级别示意图

3.2 阶段1：触摸检测

此阶段的目标是准确识别用户在视频录制期间触摸设备屏幕的位置。为此，V2S+ 利用第 2 节中概述的深度学习技术，既准确找出视频帧中出现的触摸指示器的位置，又识别其不透明度，以确定用户的手指是在按下屏幕还是从屏幕上抬起。使用深度学习技术而非传统图像处理技术（例如，Canny边缘检测 [?] 和颜色分析）的主要原因是，后者在对比度水平不同的视频帧上表现不一致（例如，触摸指示器与背景之间的对比度），因为它们需要根据图像使用不同的参数值。例如，Canny边缘检测需要指定两个阈值或参数（即，梯度的最小值和最大值），以帮助识别图像中的边。可是，由于每个应用在运行期间可能显示各种视觉内容，所有视频中的图像可能具有不同的对比度水平，因此，为所有视频中的图像寻找这些参数的最优值是一项挑战。在研究项目开始时，我们尝试了一组参数值；然而，我们很快意识到，寻找最优参数将需要付出大量努力（即，使用不同值进行大量实验），同时也无法保证所选参数能够泛化到其他图像和应用。通过使用深度学习技术，我们考虑了移动应用视频屏幕录制中图像之间对比度的变化，同时控制实验效应-

固化并提升泛化能力。

更具体地说，我们改编了 FASTERR-CNN 的一个实现 [41]，[44]，，该实现利用 VG-

GNET [30] 提取参考点（RPs）的特征，以执行触摸指示器检测（见图 1）。为了区分检测到的低不透明度和高不透明度触摸指示器，我们构建了一个不透明度卷积神经网络（OPACITY CNN），它是 ALEXNET 的一个改进版本 [29]。鉴于我们采用了广为人知的深度学习（DL）架构，本文重点介绍所做的改进，并在附录中提供模型规格 [28]。

触摸检测阶段首先接收一个符合第 3.1 节所述规格的视频。首先，对视频进行解析，并将其分解为组成它的各个帧。然后，利用 FASTER R-CNN 网络检测每一帧中是否存在触摸指示器。最后，OPACITY CNN 将每个检测到的触摸指示器分类为低不透明度或高不透明度。此阶段的输出是一个结构化 JSON，其中包含各个视频帧中的一组触摸指示器边界框，并根据不透明度对每个检测到的触摸指示器进行分类。

3.2.1 解析视频

在 V2S+ 解析视频并提取单帧之前，必须先对帧率可能变化的视频进行标准化，以确保帧率恒定。某些 Android 设备可能会为了提高效率而录制可变帧率视频 [45]。这可能导致帧与帧之间的时间间隔不一致，而 V2S+ 会在动作分类和场景重放阶段利用这些间隔来合成触摸动作的时序。因此，为避免这一问题，我们将帧率标准化为 30帧/秒，并使用 FFmpeg 工具提取单帧 [46]。

3.2.2 Faster R-CNN

从输入视频中解析出各个帧后，V2S+ 会应用其目标检测网络，以定位触摸指示器的边界框。不过，在使用目标检测之前，必须先对其进行训练。如第 2 节所述，我们使用的深度学习模型通常需要大量人工标注的数据集才能发挥作用。然而，

为了避免人工整理数据，并使 V2S+方法切实可行，我们设计了一个完全自动化的数据集生成和训练流程。为了启动 V2S+的目标检测训练数据集的生成，我们利用了现有的大规模 REDRAW 数据集，其中包含安卓屏幕截图 [47]。该数据集包含从 Google Play 上最受欢迎的安卓应用程序中，使用完全自动化的执行技术提取的 1.4 万多个屏幕截图。

为了创建用于训练这些独立模型的数据集，我们从不同应用的屏幕截图中随机抽取 5k 张互不重复的截图，并以编程方式在每张截图的随机位置叠加一张触摸指示器 图像。在此过程中，我们采取了两个步骤，以确保合成数据集能够反映触摸指示器的实际使用情况：(i) 将指示器图标的不透明度设为 40% 至 100% 之间的不同值，确保模型经过训练后能够检测手指离开屏幕的情况；(ii) 将指示器图标放置在屏幕边缘，以捕捉指示器可能被遮挡的情况。每张屏幕截图重复这一过程三次，生成 15k 张互不重复的图像。随后，我们以 70%-30% 的比例随机划分该数据集，分别创建训练集和测试集。划分时，我们确保除一张屏幕截图外，其余截图只出现在测试集中；那张重复出现的截图中，触摸指示器的位置和不透明度值有所不同。测试时，我们发现，15k 张屏幕组成的训练集足以将模型训练至极高的准确率（即 $>97\%$）。

为了训练这些模型，我们使用 *TensorFlow* 目标检测 *API* [48]，该 API 提供了知名深度学习架构的函数和配置。我们在第5.1节中详细介绍 V2S+ 的目标检测网络的训练过程。请注意，尽管训练过程完全自动化，但针对给定设备屏幕尺寸只需运行一次，之后即可重复用于推理。模型训练完成后，会对每一帧运行推理，从而为每个屏幕生成一组 边界框 预测结果，并附带置信度分数。

3.2.3 不透明度卷积神经网络

V2S+ 定位出每个视频帧中的屏幕触摸后，还必须确定每个检测到的触摸指示器的不透明度，以便辅助

动作分类阶段。这有助于 V2S+ 准确识别连续帧中位置非常相近的多个动作，或滑动结束的情况。由于安卓需要计算滑动的动量，这对于快速轻扫尤为重要。例如，如果用户正在键盘上打字，并快速多次轻触同一个按键，或双击返回按钮，就会出现多个位置相近的触摸指示器。因此，利用触摸指示器的不透明度识别一个动作何时结束、另一个动作何时开始，有助于动作分离，并准确识别涉及单指和多指的手势。

为了区分低不透明度和高不透明度的触摸指示器，V2S + 采用改进版的 ALEXNET[29] 架构作为不透明度卷积神经网络，用于预测触摸指示器的裁剪图像是否完全不透明（即

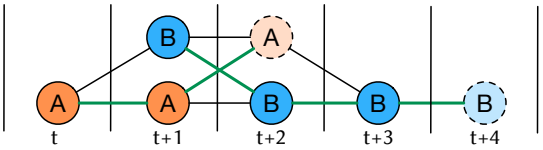


图 4：用于拆分离散动作的图遍历问题示意图。带虚线的淡化节点表示手指正从屏幕上抬起时的触摸。

手指触碰屏幕）或不透明度较低（即 表示手指正从屏幕上抬起）。

与目标检测网络类似，为了提高实用性，我们完全自动化了训练数据集的生成和训练过程。我们再次使用 REDRAW 数据集，随机选取 1 万张不同的屏幕截图，并从截图中随机裁剪出与触摸指示器大小相同的区域，然后生成数量相同的全不透明度和部分不透明度样本。对于低不透明度样本，我们将透明度级别设为 20% 至 80% 之间的不同值，以增加训练集样本的多样性。在初步实验中，我们发现不透明度卷积神经网络达到较高准确率所需的训练样本少于目标检测网络（即 $>97\%$）。与 FASTER R-CNN 模型类似，该模型只需训练一次；不过，它可以在不同屏幕尺寸下重复使用。最后，V2S+ 对上一步检测到的所有触摸指示器进行分类。随后，V2S+ 生成一个 JSON 文件，其中包含所有检测到的边界框、置信度和不透明度分类。

3.3 阶段2：动作分类

触摸检测阶段生成的 JSON 文件包含详细数据，包括边界框、不透明度信息，以及与每个检测到的触摸指示器对应的视频帧[1]。该 JSON 文件作为动作分类阶段的输入，用于将单次触摸分组并分类为高级动作（例如，轻触）。

这些动作的分类包含三个主要部分：(i) 分组算法，用于将连续帧中的触摸关联并分组成代表单个离散动作的组；(ii) 动作转换，用于识别分组后触摸的动作类型；(iii) 单指/多指动作识别，用于识别检测到的动作属于单指还是多指。动作分类阶段的结果是一个结构化动作列表（即轻触、长按、手势或多指手势），其中每个动作都是一系列触摸；对于多指手势，则是一系列动作，并与视频帧和屏幕位置相关联。

3.3.1 动作分组

V2S+的动作分组 步骤分为两步：第一步滤除模型置信度低于 0.7 的检测到的触摸。第二步根据定制的启发式规则和图连接算法，将属于同一原子动作的触摸分组。之所以需要这一过程，是因为在屏幕上执行的离散动作会持续跨越

1. 接下来，我们将“触摸指示器”和“触摸”交替使用。

多帧，因此需要进行相应的分组和分段。

连续触摸分组。 第一种启发式方法会将连续帧中出现的触摸指示器归为同一组。为避免将触摸指示器误检（这种情况很少发生），会丢弃仅存在于两帧或更少帧中的触摸。原因是，我们在实践中观察到，即使是最快的触摸屏轻触，持续时间也至少跨越五帧。

离散动作分割。有时，连续发生的单指触摸速度极快，以至于两次触摸之间没有空帧。在其他情况下，某个动作的触摸指示器可能尚未完全消失，用户就开始了新动作，导致同一帧中出现两个或更多触摸指示器。当用户快速滑动列表并轻触其中一个选项，或在键盘上快速打字时，这两种情况都很常见。或者，在多指动作中，同一手势可能包含同一帧中出现的两个或更多触摸，但必须分别对每根手指的动作进行分类。V2S+ 能够按如下方式确定一个动作的结束位置和另一个动作的开始位置。

V2S+ 使用基于启发式的方法分析连续或重叠的触摸组，并按手指将其分割为离散动作。我们将触摸指示器的分组建模为图连通性问题（见图 4）。在此模型中，触摸指示器表示为节点，竖线用于分隔连续帧，边表示构成离散动作的连续触摸之间可能存在的连接。目标是推导出遍历该图所需的适当边，使动作 A 的所有触摸位于同一组中，而动作B 的所有触摸位于另一组中（如图 4 中的绿色边所示）。我们的算法将归为一组的连续触摸列表分解为一个图。从第一个节点开始，算法访问后续每个节点，并尝试将其与前一个节点连接。如果后续帧中只有一个节点，则连接这两个相邻节点。如果后续帧中有多个节点，我们的算法会计算前一个节点与两个后续节点之间的空间距离，并将前一个节点与距离更近的邻居分为一组（如帧 t和 $t+1$之间所示）。但是，如果某一帧中的多个节点与前一个节点的距离相近（如图 4 中帧 $t+1$ 和 $t+2$ 之间所示），则使用节点的不透明度来完成连接。例如，很明显，帧 $t+2$ 中的节点 A 表示一根手指正从屏幕上抬起。因此，它必须与之前发生的动作 A （即在 $t+1$中）连接，而不是与刚刚开始的动作 B 连接。

最后，在此过程中，会分析所有相连节点的不透明度，以确定是否需要进一步拆分。不同的动作之间可能没有空帧。因此，如果在一串连续相连的节点中检测到低不透明度节点，就会将这些节点拆分为代表不同动作的组。

在此阶段结束时，将识别出代表原子操作（例如，轻触、长按或手势）的不同触摸组。

算法1： 单指/多指动作识别 别

输入： 检测到的动作（DA）
输出： 单指和多指动作（SFAs，MFAs）

```
DAc = countMultiTouchFrames(DA)
pMFAs = select(DA, DAc > 0.5) // 潜在的多指动作
SFAs = select(DA, DAc ≤ 0.5) // 初始的单指动作
SDA = sortActions(pMFAs) // 按起始帧
S = createNewStack()// 条目为动作组
initializeStack(S, SDA_1) // 推入第 1 个动作
// 对每个动作
对于每个 Action A ∈ SDA执行
    AG = popActionGroup(S)
    // 对组中的每个操作
    对于每个 Action at ∈ AG 执行
        如果 checkOverlap(A, at)则
            addToActionGroup(AG, A)
            跳出
        end
    end
    如果 notInActionGroup(A,AG)那么
        PushNewActionGroup(S, A)
    end
结束
MFAs = [] // 多指动作列表
对于 Action Group AG ∈ S执行
    c = countActions(AG)
    // 该组是否只有一个操作?
    if c == 1 then
        addSFA(SFAs, AG)
    else
        addMFA(MFAs, AG)
    end
结束
```

3.3.2 动作转换

此过程会分析得出的触摸组，并根据每组的持续时间和触摸位置对其进行分类。如果某个动作中的所有触摸指示器都保持在屏幕上指定的小半径范围内，则该动作会被归类为轻触或长按。为准确确定该半径的阈值，我们参考 Android 开源项目的定义：触摸容差（即触摸在不再被解释为轻触之前可以“游移”的像素距离）为 8px [49], [50], [51]。如果某个动作持续 20 帧或更短，则归类为轻触；否则归类为长按。其他所有动作均归类为手势。

过滤。 V2S+ 会移除触摸指示器平均不透明度较低的动作（例如， $<$ 0.1%）。这些动作横跨一个帧组，可能代表 FASTER R-CNN 误分类的触摸指示器罕见的序列。V2S+ 还会移除大小小于或等于两帧阈值的帧组，因为它们也可能表示罕见的误分类。

经过此阶段后，动作已被检测并分类，每个动作都表示为一个包含触摸位置和类型（即，轻触、长按或手势）的帧组。

3.3.3 单指/多指动作识别 别

V2S+ 能够检测并分类单指动作和多指动作至关重要。许多应用程序依赖多指动作来实现应用中的核心功能（例如，Google 地图利用双指旋转让用户旋转屏幕上的地图）。如果 V2S+ 不支持这些多指动作，那么它的实用性

就会被限制在某些应用程序和手势中，这是不理想的。

V2S+ 首先识别包含单指动作（SFA）和多指动作（MFA）的帧组。

多指动作是由发生在同一帧中的单指动作组成的集合，其中每个单指动作对应一根手指。在通过上述步骤（即第3.3.1节和第3.3.2节所述步骤）识别出每个动作后，V2S+ 根据这些动作重叠的帧对其重新分组，从而对单指动作和多指动作进行适当分类，并将其正确转换为将在设备上执行的低级事件（参见第3.4节）。V2S+ 通过执行算法1来完成这一过程。

最初，将超过50%的帧包含多个检测到的触摸的帧组视为潜在多指动作（算法1第1—2行）。其他帧组则视为单指动作（第3行）。根据我们的初步实验和观察，真正的多指动作中包含多个检测到的触摸的帧所占比例远高于50%，而单指动作的这一比例则远低于50%。因此，50%是此类阈值的合理值。此外，当用户在键盘上快速轻触时，触摸指示器可能会同时出现在屏幕上，但仍必须将每根手指的动作识别为单指动作而不是多指动作。引入这一50%的阈值后，我们降低了将这些手势关联起来的可能性，因为它们不太可能在超过其持续时间50%的时间内发生重叠。

V2S+ 随后在*sortActions*（第 4 行）中按照潜在的多指动作 的起始帧对其进行排序，并通过调用*createNew-Stack*（第 5 行）创建一个栈（S），用于存放动作组（即多指动作）。栈中的每个组都是一个动作列表，这些动作会在动作组的持续时间内（在设备上）同时执行。栈顶的组是当前正在构建的组。V2S+ 通过*initializeStack*初始化栈，将检测到的按时间顺序排列的第一个动作压入栈中，以开始构建第一个组（第 6 行）。

接下来，V2S+ 会考察每个经过排序的检测到的动作（第 7 行：$A \in SDA$），以及栈顶正在构建的组中的每个动作（第 8–9 行：$at \in AG$）。V2S+ 使用*checkOverlap* 函数来判断两个动作 A 和 at 是否发生在共同的帧中：如果 A 的第一帧出现在 at的最后一帧之前，则说明两个动作重叠，此时 V2S+ 会将 A 追加到当前动作组 AG 中（第 10–13 行）。例如，考虑动作 A=手势：[3,.....，其中 [3,.....，20] 是该手势发生时所在帧的列表。对于一个组AG=[手势$_a$:[1,.....，20],手势$_b$:[2,.....，17]], 算法 1 会将 A 追加到 AG，因为A 的第一帧（即第 3 帧）出现在手势$_a$的最后一帧（即第 20 帧）之前，这一点可由帧索引（第 $3 <$ 帧、第 20 帧）看出。

如果 V2S+ 已将每个动作 $at \in AG$ 与 A进行比较，且这些动作之间均不重叠（第 15 行），V2S+ 便完成构建 AG，并将 A 压入栈 S，开始一个新的动作组。继续上面的示例，如果此时 AG为 [手势$_a$:[1,...、20], 手势$_b$:[2,...、17],手势$_c$:[3,..., 20]] 而 A 为手势:[70,..., 92],V2S+ 不会将 A 追加到 AG，因为 A的起始帧出现在任何动作 $at \in AG$ 的结束帧之后（即帧 #70 $>$ 帧 #20 ）。因此，V2S+ 会将 A 压入 S，而 A 会成为正在构建的全新动作组 S中的第一个动作。

由于 V2S+ 会先按照检测到的动作的起始帧对其排序，因此，当某个 A 不属于 AG 时，安全关闭 AG，因为后续动作也不会与其重叠。

所有动作处理完毕后，V2S+ 会遍历创建的动作组（第 20 行）。如果组长度为 1（第 22 行），则该组只包含一个动作，并被视为单指动作（第 23 行）。否则，V2S+ 会将该组视为多指动作（第 25 行）。

多指动作分类。V2S+ 成功识别单指动作和多指动作后，会根据动作涉及的手指数进一步对多指动作进行分类。具体做法是，统计一个多指动作的所有帧中，每帧触摸指示器数量出现频率最高的值。例如，如果某个多指动作的大多数帧中包含三个触摸指示器，则该多指动作会被分类为三指手势。

我们采用这种方法，是因为我们观察到，在少数情况下，V2S+ 会错误地将类似触摸指示器的应用程序图像识别为屏幕上的额外轻触。执行多指动作时，这个轻触通常不会妨碍场景成功执行，因此我们不会将其从多指动作中移除。然而，如果 V2S+ 仅依赖分类阶段在多指动作中检测到的动作数量，流水线就会错误地判断多指动作中的手指数。例如，由于这个额外的轻触，V2S+ 会将上文提到的三指手势分类为四指手势，从而降低 V2S+的分类准确率。相反，由于构成多指动作的大多数动作都在相同的帧中开始和结束，而额外的轻触通常持续时间较短，因此，通过统计多指动作所有帧中每帧触摸指示器数量出现频率最高的值，可以更准确地判断多指动作涉及的手指数。

V2S+ 支持对一指至十指手势进行分类，以支持涉及用户双手所有手指的动作。我们认识到，在实践中，多指手势并不常见，因此在 V2S+ 的评估中（见第 5 节），我们主要关注双指多指动作（例如捏合缩放或旋转），以考察 V2S+ 对多点触控的支持。

3.4 阶段3：场景生成

在动作分类阶段推导出所有动作后，V2S+ 便使用 Android 调试桥（adb）生成命令，以便在设备上重放分类动作。为此，V2S+将分类后的高级动作转换为 sendevent 命令格式的低级指令。sendevent 是 Android Linux 内核中包含的一种实用工具。随后，V2S+ 使用经过修改的 RERAN [52]二进制文件，在设备上重放这些事件。

生成场景脚本。sendevent 命令使用有限的指令集来控制安卓设备的用户界面。主要关注的指令包括 start_event、end_event、用户手指触碰屏幕的坐标（x, y），以及旧版 Android API 级别的设备所需的某些特殊指令。要创建脚本，需生成低级动作，

按时间顺序排列，但单指动作和多指动作采用不同的技术进行转换，因为它们的转换流程略有不同。

单指动作。要生成单指事件，每个动作都从 start_event 命令开始导出（图5中的第1行）。对于归类为轻触的动作，V2S+ 会提供一个 $(x,\ y)$ 坐标对，该坐标对取自检测到的轻触边界框的中心。对于手势，V2S+ 会遍历构成该手势动作的每次触摸，并将每个触摸指示器的 $(x,\ y)$ 坐标对追加到指令列表中（图5中的第3–4行、第7–8行等。）。对于长按，V2S+ 采用与手势类似的处理方式，但只使用来自初始检测到的触摸指示器边界框的一个 $(x,\ y)$ 坐标对。然后，V2S+ 通过适当的 end_event 命令结束一个动作的指令集（例如，图5中的第275行和第276行）。

```
/* 时间 |   输入位置 |动作代码|  值 */1 [0.000] /dev/input/event1: 0003 0039
00000001 // 设置动作 ID2 [0.000] /dev/input/event1: 0003 003a 00000081 // 设定压
力3 [0.000] /dev/input/event1: 0003 0035 00000184 // x 坐标4 [0.000] /
dev/input/event1: 0003 0036 000000c3 // y 坐标5 [0.000] /dev/input/event1: 0003
0030 00000005 // 触摸半径6 [0.000] /dev/input/event1: 0000 0000 00000007 [0.
033] /dev/input/event1: 0003 0035 0000018c // x 坐标8 [0.033] /
dev/input/event1: 0003 0036 000000c2 // y 坐标9 [0.033] /dev/input/event1: 0000
0000 00000010 [0.066] /dev/input/event1: 0003 0035 00000190 // x 坐标11 [0.
066] /dev/input/event1: 0003 0036 000000c4 // y 坐标12 [0.066] /
dev/input/event1: 0000 0000 00000000 ...272 [3.000] /dev/input/event1: 0003
0035 00000409 // x 坐标273 [3.000] /dev/input/event1: 0003 0036 000000c3 // y 坐
标274 [3.000] /dev/input/event1: 0000 0000 00000000275 [3.000] /
dev/input/event1: 0003 003a 00000000 // 压力 = 0276 [3.000] /dev/input/event1:
0003 0039 ffffffff // 标记动作结束277 [3.000] /dev/input/event1: 0000 0000
00000000
```

图5：单指手势 sendevent 示例

多指动作。V2S+ 对多指动作的转换方式与单指动作不同。这是因为，为了实现多指动作预期的回放行为，必须将同一帧内发生的所有触摸同时发送到设备并在设备上执行。这意味着，不同于像单指动作那样按顺序直接转换单个手指的动作（见图5），必须将屏幕上每个手指的动作分解为每帧中的组成触摸操作，并与同一帧内发生的其他触摸操作一并转换。这需要加入一条命令，指示设备当前正在转换哪个手指，方法是标明其槽值（例如，图6中的第7行和第14行） [53]。

每个手指的动作导出时都以 start_event 命令开头（例如，图6中的第1行和第8行）。当一个手指的触摸结束时，V2S+ 会在脚本中附加 end_event 命令，以终止该槽的行为（例如，图6中的第105–107行和第109–111行）。由于每个手指的动作都是与特定槽关联后转换的，因此，即使另一个手指的动作结束，其他手指的动作仍可继续。

对于所有单指动作和多指动作，每条指令的速度和持续时间对于准确重放用户动作至关重要。为了得出这些动作的速度和持续时间，V2S+ 会根据视频帧之间的间隔为每个触摸位置添加时间戳（参见图5和图6中的时间值）

```
/* 时间 |   输入位置 |动作代码|  值 */1 [0.000] /dev/input/event1: 0003 0039
00000000 // 手指 1 动作 ID2 [0.000] /dev/input/event1: 0003 003a 00000032 // 手指
1 压力3 [0.000] /dev/input/event1: 0003 0035 00000188 // 手指 1 x 坐标4 [0.000] /
dev/input/event1: 0003 0036 00000429 // 手指 1 y 坐标5 [0.000] /
dev/input/event1: 0003 0030 00000005 // 手指 1 点击半径6 [0.000] /
dev/input/event1: 0000 0000 00000007 [0.000] /dev/input/event1: 0003 002f
00000001 // 切换到手指 28 [0.000] /dev/input/event1: 0003 0039 00000001 // 手指
2 动作 ID9 [0.000] /dev/input/event1: 0003 003a 00000032 // 手指 2 压力10 [0.
000] /dev/input/event1: 0003 0035 00000337 // 手指 2 x 坐标11 [0.000] /
dev/input/event1: 0003 0036 00000323 // 手指 2 y 坐标12 [0.000] /
dev/input/event1: 0003 0030 00000005 // 手指 2 点击半径13 [0.000] /
dev/input/event1: 0000 0000 00000000 14 [0.033] /dev/input/event1: 0003 002f
00000000 // 切换到手指 115 [0.033] /dev/input/event1: 0003 0035 0000018816 [0.
033] /dev/input/event1: 0003 0036 0000042917 [0.033] /dev/input/event1: 0000
0000 0000000018 [10.641] /dev/input/event1: 0003 002f 00000001 // 切换到手指 2 ...
104 [1.485] /dev/input/event1: 0000 0000 00000000105 [1.485] /
dev/input/event1: 0003 002f 00000000 // 切换到手指 1106 [1.485] /
dev/input/event1: 0003 003a 00000000 // 手指 1 压力 = 0107 [1.485] /
dev/input/event1: 0003 0039 ffffffff // 结束手指 1108 [1.485] /dev/input/event1:
0000 0000 00000000109 [1.485] /dev/input/event1: 0003 002f 00000001 // 切换到手
指 2110 [1.485] /dev/input/event1: 0003 003a 00000000 // 手指 2 压力 = 0111 [1.
485] /dev/input/event1: 0003 0039 ffffffff // 结束手指 2112 [1.485] /
dev/input/event1: 0000 0000 00000000
```

图6：双指手势 sendevent 示例

（即，在30帧/秒的情况下，每帧之间相隔33毫秒），从而在时间上分隔发送到设备的每条触摸命令。

此外，为了确定两个单指动作和多指动作之间的延迟，我们再次利用视频帧之间的时间间隔。我们所需的 30fps 帧率使 V2S+ 能够以毫秒级精度解析事件时间。更高的帧率可以提高重放时序的保真度。

场景重放。将所有动作转换为低级 sendevent 指令后，会将它们写入日志文件。随后，将该日志文件输入翻译器，由其将文件转换为可运行格式，从而能够直接在设备上重放。我们将转换后的文件和修改版 RERAN 引擎 [52]推送到目标设备。我们优化了原始 RERAN 二进制文件，以便更高效地重放事件轨迹。根据目标设备的架构，V2S+ 提供多种 RERAN 二进制文件，以便在更广泛的实体设备和模拟设备上进行重放[2]。最后，运行该二进制文件并使用转换后的文件，即可忠实重放最初输入视频中记录的用户操作。在线附录 [28] 中提供了更多 V2S+生成的 sendevent 脚本示例，以及更新版 RERAN 二进制文件。

4 V2S+实现

最初的 V2S 由 Java、Python 和 Bash 混合实现，因此难以使用和操作，无法适应不同的研究与开发场景。考虑到这一点，V2S+ 完全使用 Python 实现，并以模块化为设计理念，鼓励未来的开发人员和研究人员复用和扩展。每个阶段

2. 在 COVID-19 疫情期间，实体访问受到限制，这一点尤其有用。

组件可以通过抽象类轻松替换和/或扩展，从而使整个流水线能够根据预期用例灵活调整（例如，移动应用程序调试、演示等）。要执行 V2S+，用户需要在可修改的 JSON配置文件中指定所需输入视频、FASTER R-CNN、不透明度卷积神经网络、adb二进制文件和目标设备型号的路径；该文件会在整个流水线中使用。

4.1 阶段 1：触摸检测

V2S+ 从配置文件中读取输入视频的路径，该视频展示了一个自然应用使用场景。V2S+ 定义并使用一个独立的帧提取器（*FE*）实体，为触摸检测准备视频。此类别使用 FFmpeg [46] 工具将输入视频的帧率标准化为 30帧/秒，并提取各个帧。通过实现*AbstractVideoManipulator*类，可以替换 V2S+ 此部分的具体实现。

然后，V2S+ 让其触摸检测器（*TD*）类使用配置文件中指定的预训练 FASTER R-CNN，在*FE*提取的每一帧中定位各个触摸。通过实现抽象*AbstractTouchDetector*超类，可以重新配置和自定义此*TD* 类。

最后，V2S+ 使用不透明度检测器（*OD*）类，该类调用配置文件中指定的不透明度卷积神经网络，对*TD* 类定位的每次触摸进行分类，判断其不透明度值是高还是低。同样，可以通过*AbstractOpacityDetector*超类修改或自定义该类的行为。

阶段1的输出是一个结构化JSON文件，其中包含所有检测到的触摸的详细信息，包括对应的帧编号、屏幕上的空间位置以及检测到的不透明度。

4.2 阶段2：动作分类

此阶段接收阶段1输出的结构化 JSON 文件作为输入。随后，V2S+ 的*GUI*动作分类器（*GAC*）负责根据阶段1的输入，检测并分类输入视频中呈现的单指动作和多指动作。通过实现*AbstractGUIActionClassifier*类，可以对此进行调整。阶段2的输出是另一个结构化 JSON 文件，其中包含每个已分类动作的组成触摸操作（对于多指动作，则包含相应的动作）。

4.3 阶段3：场景生成

此阶段以阶段2的 JSON 文件作为输入。V2S+ 的 *Ac-tion2EventConverter（A2EC)*随后使用 sendevent 命令，将*GAC*检测到的高级动作转换为低级可重放输出脚本。开发人员可以根据所需的具体规范，实现 *AbstractAction2EventConverter* 类别，以改变该类别的行为。

V2S+ 随后使用 Android 调试桥（adb）连接到指定的目标设备，并将脚本以及相应的预编译 RERAN [52] 二进制文件推送到设备上。接着，V2S+ 执行回放并保存录制结果作为输出，以完成流水线。

5 实证评估

在本节中，我们将介绍用于评估 V2S+的方法。实证研究的目标是评估 V2S+的准确性、稳健性、性能和工业实用性 。本次评估的背景 包括：(i) 两组图像，分别包含 15,000 张和 10,000 张，分别用于评估 V2S+的 FASTER R-CNN 和不透明度卷积神经网络；(ii) 一组 90 多个安卓应用程序，包括 Google Play 上评分最高的 64 个应用、5 个发生过真实崩溃的开源应用、5 个存在已知缺陷的开源应用、5 个可控崩溃的开源应用、14 个混合应用，以及 4 个具备多点触控功能的应用；(iii) 2 台常用的实体目标 Android 设备（Nexus 5 和 Nexus 6P）以及 1 台模拟的目标 Android 设备（Nexus 5）。本研究的主要质量关注点 是 V2S+ 生成的可重放场景在多大程度上能够仿照原始用户应用使用行为。

为实现评估目标，我们提出了以下六个研究问题：

- 研究问题$_1$：V2S+ 识别屏幕上触摸指示器位置的准确度如何？
- 研究问题$_2$：V2S+ 识别触摸指示器不透明度的准确度如何？
- 研究问题$_3$：V2S+ 在根据不同应用程序的屏幕录制内容生成事件序列、准确模仿单指用户操作方面的效果如何？
- 研究问题$_4$：V2S+ 在根据不同应用程序的屏幕录制内容生成事件序列、准确模仿多指用户操作方面的效果如何？
- 研究问题$_5$：V2S+在场景生成方面的开销是多少？
- 研究问题$_6$：从业者是否认为V2S+ 有用？

5.1 研究问题$_1$：FASTER R-CNN 的准确性

为回答研究问题$_1$，，我们首先评估了 V2S+的 FASTER R-CNN 准确识别并定位屏幕录制帧中触摸指示器的能力，并使用边界框标示其位置。为此，我们按照第 3.2 节所述流程生成训练数据，并将训练集和测试集分别按 70%–30% 划分。V2S+ 使用的 FASTER R-CNN 目标检测实现与训练和推理所用图像的尺寸相关。因此，为确保 V2S+的模型能在不同设备上运行，我们训练了多个 FASTER R-CNN 模型。

对于 V2S+ 的原始实现，我们训练了两个不同的 FASTER R-CNN 模型（即 一个用于 Nexus 5，另一个用于 Nexus 6P）。对于所有这些模型，我们都将 REDRAW 数据集中的图像调整为目标设备的图像尺寸。正如我们在回答其他研究问题时所展示的，我们发现，与为每台设备重新采集原生尺寸的图像相比，调整已有的大型 REDRAW 数据集中的图像尺寸 [54], 在实际应用中能获得非常准确的检测结果。

我们使用*TensorFlow* 对象检测 *API* [48] 来训练模型。此外，在进行了初步实验后，我们对训练过程中的若干超参数进行了调整。这些调整影响了类别数量（即 1）以及每个类别或图像的最大检测结果数（即 10）。我们还调整了学习率



(a) 触摸指示器 (b) 误报检测

图 7：触摸指示器和检测失败

在 50k（即 3×10^{-5}）和 100k（即 3×10^{-6}）次迭代后，V2S+ 的学习率也进行了调整。每个训练过程运行了 150k 步，批量大小为 1；我们实现的 FASTER R-CNN 使用了一个在 MSCOCO 数据集 [55]上预训练的 VGGNET [30] 实例。我们在在线附录 [28]中提供了完整的模型参数。

为了验证 V2S+'S FASTER R-CNN 模型的准确性，我们采用平均精度均值 （mAP）进行评估；这项指标常用于评估目标检测任务中的技术。该指标通常通过对数据中所有类别的精确率 取平均值来计算；不过，由于我们只有一个类别（触摸指示器图标），因此仅报告该类别的结果。因此，我们的 mAP 按 $mAP = TP/(TP + FP)$ 计算，其中 TP对应于已识别且标签正确的图像区域， FP对应于已识别但标签错误的图像区域；在我们的案例中，这指的是被错误识别为触摸指示器的图像区域。我们使用预测区域与真实区域之间不同的交并比（IoU） [41]（交并比阈值从 0.5 到 0.95，每次递增 0.05），以判断预测结果是 TP 还是 FP。此外，我们还评估模型的平均召回率，以确定模型是否漏检任何触摸指示器实例。其计算方式为 $AR = TP/k$，其中TP 的定义与上述相同， k 则对应于可能的 TP 预测总数。

在使用默认触摸指示器 进行 FASTER R-CNN 初步实验时（见图 7a），我们发现，由于默认触摸指示器与应用中的其他图标和图像相似，因此偶尔容易出现误报检测（图 7b）。因此，我们分析了默认触摸指示器失效的特定情况，并将其替换为更醒目、对比度更高的触摸指示器（见图 7a）。我们发现，这种自定义触摸指示器略微提高了模型的准确率。需要注意的是，要替换设备上的触摸指示器，设备必须已获取 root 权限。虽然这对大多数开发人员来说是可接受的选择，但对终端用户而言可能较为困难。不过，即便使用默认触摸指示器，V2S+ 的 FASTER R-CNN 模型仍能达到极高的准确率。基于这些原因，我们仅在 RQ_1 和 RQ_2 中评估自定义触摸指示器。

5.2 RQ_2：不透明度卷积神经网络的准确性

为回答 RQ_2，，我们评估了 V2S+ 的不透明度卷积神经网络预测触摸指示器的不透明度是实心还是半透明的能力。为此，我们遵循第 3.2 节所述的数据集生成流程，其中生成数量相等的完全不透明和部分不透明样本。因此，生成的数据集共包含 10k 个完全不透明和部分不透明样本，数量各半，并按 70%–30% 的比例均分为训练集和测试集。我们结合使用 TensorFlow 框架和 Keras 实现了不透明度卷积神经网络。与先前使用的 FASTER R-CNN 模型不同，我们无需为每台设备单独创建一个模型。这是因为触摸指示器输入不透明度卷积神经网络时会调整大小。与 FASTER R-CNN 类似，我们使用两个类别（实心和半透明）的平均精度均值来评估不透明度卷积神经网络。

5.3 RQ_3：单指动作场景的回放准确性

为回答 RQ_3，，我们开展了三项不同的研究，以评估 V2S+ 在不同类型应用程序中回放单指动作的准确性：两项针对原生应用程序，一项针对混合应用。

5.3.1 原生应用程序上的回放准确性

我们开展了两项研究，旨在评估 V2S+重现原生应用程序屏幕录制中所呈现用户事件的能力的深度和广度。第一项是受控研究，通过一项用户研究衡量 V2S+的能力深度；在这项研究中，我们收集了终端用户录制的真实视频，内容包括：缺陷、真实崩溃、人工注入的崩溃，以及 20 款应用的正常使用场景（例如，Car Report、Another Monitor、等等。）。接下来，在热门应用研究中，我们通过录制 Google Play 上最热门的 64 款、种类更多样的应用的场景，衡量 V2S+能力的广度（例如，Tasty、Twitter、Airbnb、等等。）。这些应用的完整详情见在线附录 [28]。

受控研究。 在本研究中，我们考察了四类录制的使用场景，分别呈现：（i）正常使用；（ii）缺陷；（iii）真实崩溃；以及（iv）受控崩溃。正常使用场景是指展示热门应用不同特征的视频录制。缺陷场景是指展示开源应用中缺陷的视频录制。最后，受控崩溃是指在开源应用中注入崩溃。这样我们就能控制崩溃触发前的步骤数。

本研究从威廉与玛丽学院招募了八名参与者，包括1名本科生、3名硕士生和4名博士生（经威廉与玛丽学院人体受试者保护委员会（PHSC）批准，项目编号为 PHSC-2019-01-22-13374），由他们录制视频。每位参与者分别录制八段视频，即上述每类场景各录制两段。四名参与者使用实体 Nexus 5 录制视频，另外四名使用实体 Nexus 6P。总计录制了64段视频，涉及20款应用，均匀分布于所有场景。在录制每款应用之前，参与者要么先使用该应用以熟悉它，要么阅读缺陷/崩溃报告，再复现故障。所有真实缺陷和崩溃均取自以往关于移动测试和缺陷报告的研究 [56], [57], [58]。这些视频共呈现了1,141个单指动作，平均每段视频包含 ≈ 18 单指动作 。

热门应用研究。 在本研究中，我们考察了从 Google Play 获取的规模更大、类型更多样的应用。具体而言，我们从每个非游戏类别（即 32 个类别）中下载了评分最高的两款应用，共计 64 个应用程序（见表 4）。

随后，两位作者使用实体 Nexus 5，为每个应用录制两个场景，共涉及 32 个应用。

另一位作者则使用实体 Nexus 6P。作者尽可能自然地使用这些应用，这一评估流程与以往对安卓录制与回放技术的评估一致 [52], [59]。录制的场景代表了应用的特定用例，至少涵盖一项主要特征，且彼此独立。在实验过程中，我们发现某些录制的场景无法复现，原因可能是非确定性或动态内容（例如，随机弹出窗口）。因此，我们舍弃了这些场景，最终保留了来自 60 个应用的 111 个应用使用场景。值得注意的是，由于动态内容具有非确定性，现有技术（如 RERAN [52] 或 Barista [60]）几乎不可能复现这些场景，因此我们决定将其排除。这些使用场景总计包含 2,150 个 单指动作，平均每段使用视频包含 ≈ 19 个单指动作。

5.3.2 混合应用的回放准确率

我们开展了一项研究，以评估 V2S+ 在混合应用中重现场景的能力，即，这类应用使用安卓 API 原生实现某些特征，并通过基于 Web 的技术实现其他特征。开展这项研究的动机是，混合应用在 Google Play商店 [25] 中的日益普及，以及目前针对这类应用的图形用户界面测试工具相对匮乏。我们选择将 V2S+ 在混合应用上的表现与原生应用分开评估，以证明 V2S+ 是少数同时支持混合应用和原生应用的工具之一。

热门应用研究。本研究选取了来自 7 个热门应用程序类别、具有混合特征的 14 款应用，每个类别 2 款（见表5）。针对这 14 款应用，我们各设计了 2 个与应用核心混合功能交互的场景。我们招募了 7 名参与者（6 名本科生和 1 名研究生）。每个参与者在组内各录制了 28 个场景中的两次录制，因此每位参与者都能在 Nexus 5 模拟器上录制 8 个不同的场景。由于 COVID-19 相关限制，我们使用模拟器而非实体设备。为了让参与者尽可能自然地与每款应用交互，每位参与者都可以在录制前熟悉应用和场景。本研究共收集了 56 段视频，但由于内容非确定性或动态内容，或 APK问题，我们舍弃了 3 段视频。这些视频共呈现了 622 个单指动作，平均每段视频 ≈ 11 个单指动作。

5.3.3 场景评估

除了制作视频录制外，我们还要求参与者观看自己的场景，并记录视频中呈现的（真实标注）动作序列。参与者需要将每个场景中呈现的每个动作归为以下三种类型之一：(i) 轻触，即触摸指示器大致停留在同一位置，持续时间较短的动作；(ii) 长按，即触摸指示器在整个动作持续期间大致停留在同一位置，且持续时间比轻触更长的动作（即 通常约为 1 秒或更长）；或 (iii) 手势，即触摸指示器在屏幕上移动的动作其持续时间可长可短。因此，动作序列就是由动作类型构成的序列（依据上述定义的三种类型）。收集视频和动作序列后，两位作者对以下内容的正确性进行了核验：(1) 录制内容；以及 (2) 动作序列（与录制内容进行对照）。只有三种情况下，我们修订了动作序列，使其与视频正确对应，和/或剪掉了视频开头或结尾与应用使用场景无关的部分，其中包括参与者在录制内容中加入了打开或关闭应用程序，或与其他应用程序交互的情况。

为了衡量 V2S+ 重放展示单指动作的视频的准确度，我们使用了 3 种不同的指标。为了计算这些指标，我们手动为每段录制的视频推导出动作类型的真实序列。首先，我们使用莱文斯坦距离（常用于计算单词在字符层面的距离）来比较原始动作类型列表与 V2S+生成的分类动作列表。因此，我们将每种动作类型视为一个字符，将场景视为表示一系列动作的字符序列。莱文斯坦距离值越低，表示将 V2S+的输出转换为真实标注动作集所需的修改越少。此外，我们还计算最长公共子序列（LCS），以找出 V2S+输出的每个场景中与视频录制中的真实场景相吻合的最长序列。对于这一 LCS 指标，LCS 与真实序列之间的百分比越高，V2S+的轨迹就越接近与原始轨迹完全匹配。此外，我们还计算了 V2S+ 在所有场景中预测每种动作类型时相对于真实标注的精确率和召回率。最后，为了验证 V2S+ 生成的重放场景相对于原始视频录制的保真度，我们手动比较了每段原始视频与 V2S+重现的场景，并确定每段视频中被忠实重放的动作数量。

5.4 RQ_4：多指动作场景的回放准确率

为评估 RQ_4，，我们对热门应用程序进行了一项研究，其中一位作者参与其中，以展示 V2S+ 能否正确识别并重放现实场景中常见的多指动作。我们研究了 Google Play商店中两个不同类别的 4 款原生应用程序，如表1所示。之所以选择这些类别，是因为众所周知，其中有许多应用在使用核心应用功能时会依赖多指动作。对于这 4 款应用中的每一款，我们都列举了其支持且会触发特定行为的多指动作（同样列于表1），并设计了同时运用这些动作的人工场景和自然场景。人工场景是指用户仅按顺序执行多个多指动作的场景；记录这类场景是为了确保 V2S+ 能够复现各应用所支持的各种多指动作。自然场景则会运用应用的基本功能，并融入一个或多个多指动作；研究中纳入这类场景，是为了确保 V2S+ 能够在多指动作与其他支持的动作依次发生时准确重放它们

应用类别	Apps	演示操作
艺术与设计	Infinite Painter	双指捏合放大 双指拉开缩小 双指旋转
	Sketchbook	双指捏合放大 双指拉开缩小 双指旋转
导航	Google 地图	双指捏合放大 双指拉开缩小 双指旋转 双指轻点 双指“推移”
	Waze	双指捏合放大 双指拉开缩小 双指旋转 双指轻点

，例如轻触。对于艺术与设计类应用，我们设计了 1 个自然场景和 1 个人工场景。对于导航类应用，我们设计了 2 个自然场景和 2 个人工场景。与艺术与设计类应用相比，我们选择为导航类应用录制更多场景，因为如表1所示，导航类应用中有更多相关的多指动作。本研究共包含 12 段视频，展示了 115 个动作：44 个多指动作和 71 个单指动作。这些视频除了多指动作之外，还必须包含单指动作，因为自然场景需要在多指动作之间穿插单指动作，才能与应用形成有意义的交互。平均而言，每段视频包含 ≈ 4 多指动作和 ≈ 6 单指动作。

本文的一位作者在一台实体 Nexus 5 安卓设备上录制了这 12 个场景。录制前，这位作者熟悉了这些应用程序和场景，以确保场景的进展自然流畅。随后，在拍摄视频后，这位作者重新观看了每段录制内容，并将每个动作分类为 (i) 轻触、(ii) 长按或 (iii) 手势。随后使用这些分类，根据第 5.3.3 节所列的指标，对 V2S+ 复现手势序列的性能进行实证验证。

5.5 RQ_5：性能

为回答 RQ_5，，我们通过测量视频通过 V2S+ 方法三个阶段中每个阶段所需的平均时间，在普通硬件（即 单块 NVIDIA GTX 1080Ti）上评估了 V2S+ 。我们将此视为 V2S+ 性能的最坏情况，因为我们的方法在专用硬件上运行时速度可能会快得多。请注意，我们修改了 RERAN 引擎，使其运行速度足以支持更精确的滑动。因此，我们预计 V2S+ 的开销低于 RERAN 相应论文 [52] 中报告的开销。

5.6 RQ_6：感知有用性

我们的最终目标是将 V2S+ 集成到实际开发环境中。因此，作为评估的一部分，我们调查了三位分别为各自公司开发 Android 应用的开发人员对 V2S+的感知有用性。

开发人员（也称为 参与者）是通过作者直接联系到的。参与者 #1（P1）是 Google 搜索应用的图像搜索团队的一名前端开发人员 [61]，参与者 #2（P2）是 7-Eleven Android 应用的开发人员 [62]，参与者 #3（P3）是 Proximus 购物篮应用的后端开发人员 [63]。我们采用一套分为两部分的问题对参与者进行了访谈。第一部分旨在收集参与者的背景信息，包括他们在公司的角色、完成任务时使用的信息、这些信息的质量、收集信息所面临的挑战，以及他们在日常工作中如何使用视频。第二部分旨在评估 V2S+的潜在用途，以及其生成可重放场景的准确性。本部分还征求了参与者对改进 V2S+ 的反馈，其中包括李克特量表问题。我们在在线附录 [28]，中提供了所用访谈问题的完整列表，并在第 6.5 节讨论了其中一些问题。

表 2：触摸指示器检测准确率 - 原始

模型	设备	mAP	平均精度均值@.75	AR
FASTER R-CNN-原始版	Nexus 5	97.36%	99.01%	98.57%
FASTER R-CNN-原始版	Nexus 6P	96.94%	99.01%	98.19%
修改版 FASTER R-CNN	Nexus 5	97.98%	99.01%	99.33%
修改版 FASTER R-CNN	Nexus 6P	97.49%	99.01%	99.07%

参与者通过比较两段视频回答第二部分的问题，这两段视频展示了各自应用中相同的使用场景：一段视频展示在应用上手动执行该场景的过程，另一段则展示通过 V2S+’S 生成的脚本自动执行该场景的过程。具体来说，我们在每个应用上定义、录制并手动执行一个使用场景。随后，我们对生成的视频录制内容运行 V2S+ 。为了定义这些场景，我们在每个应用中找出一项涉及任意动作类型（即轻触、长按和手势）的功能。接着，我们生成一段视频来展示原始场景（即视频录制内容），并在旁边展示执行 V2S+’S 脚本时生成的重放场景。这两段录制内容都突显了在应用上执行的动作。我们向参与者展示了视频，以及 V2S+’S 的脚本，其中包含从原始视频中自动识别出的高级动作。

6 评估结果

我们展示并讨论 V2S+ 的评估结果。

6.1 研究问题$_1$：准确性 FASTER R-CNN

表 2 展示了 V2S+ 的 FASTER R-CNN 网络在不同设备和数据集上进行触摸指示器检测的精确率和召回率。第一列标明使用的是默认触摸指示器还是修改后的版本。第二列说明各个训练模型对应的目标设备。第三列给出平均精度均值，该值依据 COCO 检测指标[3] 计算，计算时采用预测区域与真实区域之间不同的交并比（IoU） [41]值（交并比阈值范围为 0.5 至 0.95，间隔为 0.05）。第四列显示预测区域与真实区域之间的交并比高于 75% 时的平均精度均值。所有模型的平均精度均值至少达到 ≈97%，这表明所作出的触摸指示器预测中有 ≈97%

3. https://cocodataset.org/#detection-eval

表 3：不透明度卷积神经网络的混淆矩阵。低不透明度原始图像（LO-Orig.）、高不透明度原始图像（HO-Orig.）、低不透明度自定义图像（LO-Cust.）、高不透明度自定义图像（HO-Cust.）

	总计	低不透明度原始图像	高不透明度原始图像	低不透明度自定义图像	高不透明度自定义图像
低不透明度	5000	97.8%	2.2%	99.7%	0.3%
高不透明度	5000	1.4%	98.6%	0.8%	99.2%

V2S+ 的目标检测网络所检测的结果均正确。只有当边界框交并比与真实边界框的匹配度至少达到 75% 时，平均精度均值才会提高。这说明，当模型能够预测出相当准确的边界框时，它几乎总能正确检测到触摸指示器（在 ≈99% 的情况下）。如表 2 最后一列所示，该模型的召回率也极高，至少能检测到 ≈98% 的触摸指示器。

> **RQ 的回答$_1$**：V2S+ 得益于其目标检测技术的出色性能，能够检测触摸指示器。所有 FASTER R-CNN 模型在不同设备上的精确率均至少达到 ≈ 97%，召回率均至少达到 ≈ 98%。

6.2 RQ$_2$：不透明度卷积神经网络的精确率

为展示不透明度卷积神经网络对触摸指示器两种不透明度级别进行分类时的精确率，我们在表 3 中给出了混淆矩阵。结果分别针对默认触摸指示器和修改后的（即自定义的）触摸指示器。原始触摸指示器的总体精确率为 98.2%，而自定义触摸指示器的总体精确率为 99.4%。这些百分比的计算方法是：将正确预测的标签（即低/高不透明度）数量除以原始触摸指示器和自定义触摸指示器中属于该标签的成员数量。因此，显然，V2S+ 的不透明度卷积神经网络能够非常准确地区分不同的不透明度级别。

> **回答 RQ$_2$**：V2S+ 得益于卷积神经网络在不透明度等级分类方面的精确率。OPACITY CNN 对两个触摸指示器的平均精确率均超过 98%。

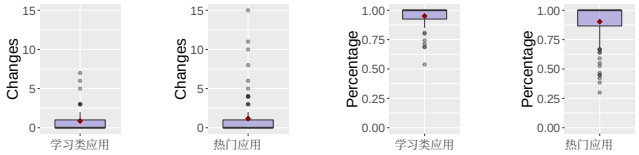
6.3 RQ$_3$：单指动作场景的回放准确率

我们展示并讨论 V2S+在原生应用程序和混合应用程序中针对单指动作场景的回放准确率结果。

6.3.1 原生应用程序的回放准确率

莱文斯坦距离。图 8a 和 8b 分别展示了将输出事件轨迹转换为真实标注所需的改动数量，涉及受控研究和热门应用研究中使用的应用。对于受控

学习类应用平均每条用户轨迹需要进行 0.85 次变更（即 操作变更的数量），才能将 V2S+ 的输出转换为真实事件轨迹；而对于热门应用，所需变更略多，为 1.17 次。总体而言，V2S+ 每条事件轨迹所需的变更极少，与真实标注非常相似。



(a)LD-学习类 (b)LD-热门 (c)LCS-学习类 (d) LCS-热门

图 8：有效性指标 - 原生应用

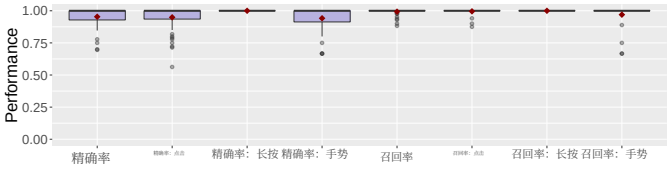


图 9：精确率和召回率 - 受控研究（原生应用）

最长公共子序列。图 8c 和 8d 分别展示了受控研究 和热门应用研究中每条轨迹与原始录制轨迹相匹配的事件百分比。平均而言，V2S+ 能够正确匹配受控研究中学习类应用真实标注中的 95.1% 的顺序事件，以及热门应用中的 90.2%。这些结果表明，V2S+ 能够生成在操作类型方面与原始轨迹高度匹配的动作序列。

精确率和召回率。图 9 和图 10 分别展示了受控研究和热门应用研究的精确率和召回率结果。为了绘制这些图，我们针对数据集中每个场景的每种预测动作类型，创建了一个不考虑顺序的“动作集合”。随后，将这些动作与真实标注的“动作集合”进行比较，并统计真正例、假正例和假负例，从而计算精确率和召回率。最后，计算所有动作类型的平均精确率和召回率。结果表明，事件轨迹的平均精确率在受控研究应用中为 95.3%，在热门应用中为 95%。各事件类型的结果也支持这一结论：不同类型的精确率普遍较高，但热门应用中的长按精确率除外。这主要是因为在所有热门应用的轨迹中，此类事件的数量较少（即9 次长按）。此外，图 9 和图 10 显示，各动作类型的召回率都很高：受控研究应用中所有事件类型的平均召回率为 99.3%，热门应用中则为 97.8%。总体而言，我们认为 V2S+ 能够准确预测各条轨迹中事件类型的正确数量。

成功率。我们还评估了两项 RQ$_3$ 研究中所有场景里每个重放操作的成功率。我们对 175 个视频进行了人工分析；如果可回放场景按照原始视频忠实地执行了应用功能，就将相应操作标记为成功。这意味着在某些情况下，视频不会与原始视频录制完全匹配（例如，一次键盘按键错误仍然产生了相同的功能结果）。

验证受控研究中的全部 64 个视频后，V2S+ 完整复现了 93.75% 的场景和 94.48% 的连续操作。V2S+ 完整复现了 96.67% 的

表 4：RQ_3 热门应用研究的详细结果。绿色单元格表示完整复现的视频，橙色单元格 >50% 表示部分复现的视频，红色单元格<50% 表示未复现的视频。蓝色单元格表示因非确定性/动态内容而未复现的视频。

应用名称	复现操作		应用名称	复现操作		应用名称	复现操作		应用名称	复现操作		应用名称	复现操作	
Ibis Paint X	11/11	36/36	Firefox	22/22	N/A	Tasty	20/20	14/36	SoundCloud	12/12	13/13	LetGo	17/17	15/15
像素艺术专业版	20/20	9/9	MarcoPolo	12/12	30/30	Postmates	26/26	12/12	Shazam	12/15	20/20	TikTok	14/14	11/11
Car-Part.com	20/20	16/16	Dig	13/13	N/A	Calm	9/9	11/16	Twitter	14/14	19/19	LinkedIn	18/18	13/13
CDL练习	8/8	13/13	Clover	15/15	19/19	Lose It!	36/36	N/A	News Break	1/18	9/9	CBSSports	25/25	16/16
丝芙兰	4/9	13/13	PlantSnap	39/39	18/24	U Remote	14/14	18/18	FamAlbum	19/19	8/26	MLBatBat	11/13	N/A
SceneLook	14/14	16/16	翻译器	20/20	28/28	LEGO	52/52	24/24	Baby-Track	14/14	12/12	G-Translate	14/17	15/15
欧定版圣经	16/16	19/19	Tubi	2/19	30/30	开发库	35/35	22/22	Walli	8/8	N/A	G-Podcast	9/9	15/15
圣经应用	12/12	15/15	Scan Radio	24/27	N/A	星座运势	24/24	19/19	ZEDGE	9/9	18/18	Airbnb	9/9	14/14
Indeed Jobs	15/15	19/19	Tktmaster	30/30	14/14	Waze	17/17	19/19	G-Photo	18/18	18/18	G-Earth	8/13	1/30
UPS Mobile	16/16	19/24	问候卡片	23/23	N/A	交通	26/26	18/18	PicsArt	18/18	39/39	DU Record	15/15	9/9
网络漫画	17/17	15/21	QuickBooks	47/47	28/28	WebMD	7/34	7/26	Google 文档	3/26	N/A	AccuWeath	13/13	21/21
MangaToon	16/16	28/28	Yahoo Fin	23/23	N/A	K-Health	10/10	15/24	M. Outlook	27/27	21/26	W. Radar	14/14	13/13

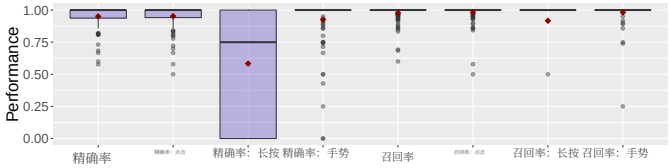


图 10：精确率和召回率——热门原生应用

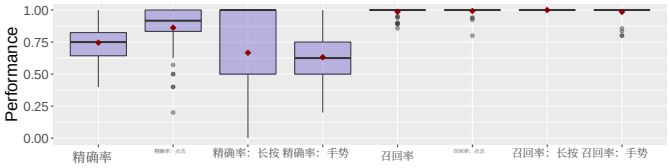
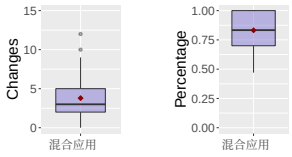


图 12：混合应用的精确率和召回率



(a) LD-Hybrid (b) LCS-Hybrid

图 11：有效性指标——混合应用

缺陷和崩溃场景，以及 91.18% 的应用使用情况。热门应用研究的详细结果如表 4 所示，其中列出了每个应用、场景（含动作总数）以及成功回放的动作。绿色单元格表示视频已被完全复现，

橙色单元格表示复现的事件超过 50%，红色单元格表示复现的事件不足 50%。蓝色单元格表示因非确定性/动态内容而未能复现的视频。在为热门应用录制的 111 个场景中，V2S+ 完整复现了 81.98% 的场景，以及 89% 的连续操作。总体而言，这表明该方法在种类极其多样的应用中具有出色的回放性能。V2S+ 未能复现场景的情况

表5：RQ_3 混合场景研究的详细结果。绿色单元格表示视频已完全重现，橙色单元格表示重现了 >50%，红色单元格表示重现了<50%。蓝色单元格表示因非确定性、动态内容或 APK 问题而未能重现的视频。

应用名称	场景1 复现操作		场景2 复现操作	
WikiHow	4/4	1/7	3/9	10/10
维基百科	3/14	13/13	23/23	17/17
Platt	2/10	14/14	3/3	3/3
USPS	16/16	16/16	24/24	18/23
CNBC	10/10	10/10	11/11	14/14
Seeking Alpha	7/7	5/5	0/5	1/7
Old Navy	7/9	1/11	0/6	N/A
Pinterest	22/22	0/21	24/24	24/24
Epocrates	10/13	12/17	10/10	6/6
Leafly	18/18	11/11	N/A	9/9
Newsbreak	5/5	5/5	9/9	5/5
卫报	5/9	1/8	9/9	9/9
Rainbow	12/12	2/19	12/12	11/11
VS	12/12	5/11	2/10	N/A

这主要是由于视频分辨率为 30帧/秒，导致手势事件存在轻微误差。我们将在第8节讨论解决这一局限的潜在方案。

> **RQ 的回答$_3$（原生应用）**：对于在实体设备上使用的热门原生应用，V2S+ 能够生成事件轨迹，平均需要进行 ≈1 次更改才能匹配原始用户场景。此外，按事件类型序列进行比较时，至少 90.2% 的事件与真实标注相符。总体而言，V2S+生成的事件类型的精确率和召回率分别为 ≈95% 和 ≈98%。最后，V2S+ 分别在 96.67% 和 91.18% 的情况下成功复现了与缺陷/崩溃和应用使用相关的视频。

6.3.2 混合应用的回放准确率

莱文斯坦距离。 图 11a 展示了将 V2S+ 的事件轨迹转换为混合应用研究中所用应用的真实轨迹所需的更改次数。在这项研究中，将 V2S+ 的输出转换为真实事件轨迹时，每条用户轨迹平均需要进行 3.79 次更改。

最长公共子序列。图 11b 展示了混合应用研究中每条轨迹与原始录制轨迹相匹配的事件比例。平均而言，V2S+ 能够正确匹配这些混合应用程序中真实数据集的 83.19% 顺序事件。

精确率和召回率。图 12 详细展示了混合应用研究的精确率和召回率结果。结果表明，混合应用的事件轨迹平均精确率为 74.54%。图 12 还显示，各动作类型的召回率都很高，所有事件类型的平均召回率为 98.63%。

成功率。 最后，我们还在混合应用研究中评估了每个回放动作的成功率。我们对这56个视频进行了人工分析；如果可回放场景能够根据原始视频忠实地运行应用功能，则将相应动作标记为成功。

在验证本研究的全部56个视频后，有3个因非确定性或APK文件问题而被剔除。V2S+ 完整复现了53个场景中的39个（73.58%），以及622个连续操作中的486个（78.15%）。详细结果见表5，其中展示了每个应用、场景（包括动作总数）以及成功回放的动作。

讨论。 结果表明，与原生应用相比，V2S+在混合应用上的表现较差。由于 COVID-19 相关限制，我们选择以线上方式开展混合应用研究，并要求参与者在个人电脑上的模拟设备上录制场景。对结果的分析表明，用这些 Android模拟器替代实体设备主要通过两种方式影响了结果质量：（i）在模拟器上使用鼠标或触控板完成动作并不自然，因为这无法像在实体设备上用手指操作那样轻松自如，导致动作更加“锯齿化”，也更容易被 V2S+错误分类和回放；（ii）模拟器的运行速度与电脑的处理能力直接相关。在我们的研究中，这表现为部分参与者录制的视频中出现了“卡顿”的动作，难以被 V2S+准确检测或回放。我们将在第8节讨论这些局限的潜在解决方案。

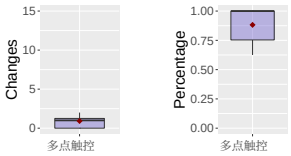
我们没有发现任何会影响混合应用回放结果的特定应用类型相关因素。因此，我们观察到的结果差异（混合应用与原生应用之间）可能并非源于应用类型的差异或 V2S+支持这些应用的能力，而更可能是视频收集流程的局限所致。

> **对 RQ 的回答$_3$（混合应用）**：对于在模拟设备上使用的混合应用，V2S+ 能够生成事件轨迹，平均需要进行 ≈3.5 次修改才能匹配原始用户场景。此外，若按事件类型序列进行比较，至少 83% 的事件与真实标注相符。总体而言，V2S+生成的事件类型的精确率和召回率分别为 ≈ 75% 和 ≈ 99%。

6.4 RQ$_4$：包含多指动作的场景的回放准确率

我们讨论 V2S+ 在包含多指动作的场景中的回放准确率结果。

莱文斯坦距离。 图13a展示了将输出事件轨迹转换为所需的更改次数用于多点触控研究的应用的真实标注。在这项研究中，平均每条用户轨迹需要进行 ≈ 0.92处更改，才能将 V2S+’S 的输出转换为真实事件轨迹。如果将视频分为人工和自然类别，人工场景平均每条轨迹仅需进行 ≈ 0.33处更改，而自然视频平均需要进行 ≈ 1.5处更改。相反，如果将这些视频中描绘的动作分为单指动作或多指动作序列，那么单指动作平均每个序列需要进行一次更改，而多指动作平均每个序列需要进行 0.083 处更改。



(a) LD-多指 (b) LCS-多指

图 13：有效性指标 - 多指场景

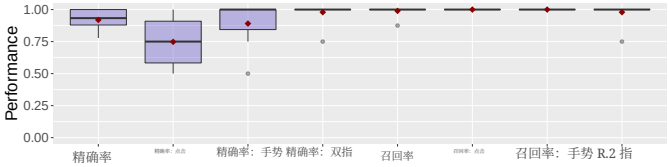


图 14：精确率和召回率 - 多指动作

最长公共子序列。 图 13b 展示了多点触控研究中每条轨迹的顺序事件与原始录制轨迹中相应事件匹配的百分比。平均而言，V2S+ 能够正确匹配这些多点触控场景真实标注中的 ≈ 88.2%的顺序事件。如果将视频分为人工和自然类别，V2S+ 能够正确匹配人工场景真实标注中 ≈ 96.7%的顺序事件，以及自然场景中 ≈ 79.7%的顺序事件。

如果将这些视频中的单指动作 与多指动作 分开，V2S+ 能够正确匹配真实标注中单指动作 的顺序事件的 ≈ 99.2%，以及多指动作的顺序事件的 ≈ 97.9%。根据我们的分析，这些结果高于将这些动作合并统计时得到的结果（见上文），主要有两个原因。首先，单独考虑多指动作 或单指动作 会缩短真实标注中的动作序列，因此更容易获得较高的最长公共子序列指标。其次，本研究所包含的视频只能呈现双指手势（因为涉及更多手指的手势无法产生有意义的应用程序行为），这也使多指动作 序列获得了较高的最长公共子序列指标。

精确率和召回率。 图 14 详细展示了多点触控研究的精确率和召回率结果。结果表明，在多点触控场景中，事件轨迹的平均精确率为 ≈ 91.9%。图 14 还显示，各动作类型的召回率都很高，多点触控场景中所有事件类型的平均召回率为 ≈ 99%。平均而言，V2S+ 在人工场景中的精确率为 ≈ 93.9%，召回率为 100 %；在自然场景中的精确率为 ≈ 89.9%，召回率为 ≈ 97.9%。如果将这些视频中呈现的动作按单指动作 和多指动作区分，我们发现，对于单指动作，精确率为 ≈ 83%，召回率为 100%；对于多指动作，精确率为 ≈ 97.9%，召回率为 ≈ 97.9%。

成功率。 最后，我们还评估了多点触控研究中每个回放动作的成功率。手动验证这项研究的 12 个视频后，我们发现，平均而言，V2S+ 能够完整复现 83.3% 的场景和 94.8% 的连续操作。对于人工场景，V2S+能够复现 83.3% 的场景和 95.7% 的连续操作。对于自然场景，V2S+ 能够复现 83.3% 的场景和 94.5% 的连续操作。详细结果如表6所示，其中列出了每个应用、场景（及其动作总数）以及成功

表6：RQ_4 多点触控场景研究的详细结果。绿色单元格表示视频已完整复现，橙色单元格表示复现了 >50%，红色单元格表示复现了 <50%。蓝色单元格表示由于非确定性／动态内容或 APK问题而未能复现的视频。

应用名称	人工场景	自然场景
Infinite Painter	5/5	19/19
Sketchbook	5/5	19/19
Google 地图	4/5	15/15
	4/4	3/8
Waze	3/3	13/13
	2/2	17/17

显示回放动作。由于该指标基于所描绘场景整体的演进过程，我们无法将单指动作 与多指动作 区分开来以计算这些指标。

> **对 RQ_4 的回答**：V2S+ 能够生成涉及多指动作的事件轨迹，这些动作平均需要 < 1 的变更才能匹配原始用户场景。此外，考虑事件类型的顺序时，至少有 88% 的事件与真实标注相匹配。总体而言，对于 V2S+重现的所有事件类型，精确率和召回率分别为 $\approx$ 92% 和 $\approx$ 99%。V2S+ 成功重现了≈ 83.3% 的多指场景。

6.5 RQ_5：方法性能

为了衡量 V2S+的性能，我们测量了所有录制视频中单个视频帧在三个组件上的平均处理时间，单位为秒/帧（s/f）：（i）帧提取过程（0.045 s/f）、（ii）触摸指示器检测过程（1.09 s/f）以及（iii）不透明度分类过程（0.032 s/f）。与其他处理时间相比，脚本生成时间可以忽略不计，并且每个视频只需执行一次。这意味着，一段长度约为 3 分钟的视频[4] 将需要 V2S+ ≈105 分钟来完成处理并生成脚本。不过，该过程是完全自动化的，可以在背景中运行，并且能够通过更先进的硬件加速。我们预计，回放脚本的开销将与 RERAN 相近或更低，因为 V2S+ 回放引擎本质上是 RERAN 的改进版本。

> **对 RQ 的回答$_5$**：V2S+ 能够完整处理一段平均时长为 3 分钟的屏幕录制，耗时 ≈105 分钟。

6.6 RQ_6：感知有用性

我们讨论了为评估 V2S+的感知准确性和有用性而开展的用户研究结果。

首先，三位行业参与者均（强烈）认同，V2S+ 生成的场景（见生成的脚本）与手动执行的场景相比是准确的。

参与者还一致认为，需要提供更多工具支持，帮助质量保证人员及其他利益相关者

4. 这一持续时间长于我们评估中录制的大多数场景。

生成视频录制内容。例如，P3 提到，虽然视频“在某些情况下比图像更有用”，但由于“时间限制”，“可能很难录制”。P1 表示，质量保证团队可以使用 V2S+ ，以更高效地制作视频。P2 赞同这一观点，并提到 V2S+ 可以帮助“质量保证团队编写/提供命令或步骤，然后由工具读取并执行，同时录制该场景和问题的视频。这种解决方案可以集成到持续集成流水线中”。此外，P3 提到，V2S+ 也可用于应用演示：V2S+ 可以“自动执行一个展示应用功能的脚本并录制视频。这样一来，演示就可以着重于业务讲解，而不是手动向应用输入内容或执行用户场景”。

P2 还表示，V2S+ 可用于分析应用中的用户行为，从而帮助改进某些应用界面和导航。他提到，V2S+“可以收集交互类型、轻触次数等信息，以便检测，例如，某些界面是否经常被使用，或者用户导航到某个界面后是否经常返回”。他提到，这些数据“可能对营销用途有帮助”。P3 认为，V2S+ 可能有助于复现难以复现的缺陷。

参与者为改进 V2S+ 提供了宝贵而具体的反馈。他们建议，在执行 V2S+ 的脚本时，为生成的视频添加边界框，标出每一步交互涉及的图形用户界面组件或屏幕区域。他们还提到，视频可以显示解释应用中发生情况的（弹出式）评论（例如，“10 秒后，按钮 X 报错”之类的评论），这有助于复现缺陷。他们希望脚本中包含更多信息，例如图形用户界面元数据，以便更深入、直观地了解每个步骤。例如，脚本可以使用交互对象的图形用户界面组件名称或 ID，并生成“用户轻触了发送按钮”之类的步骤，而不是“用户轻触了坐标 (10.111,34.56)”。P3 提到：“如果能使用图形用户界面组件的元数据而非坐标，以编程方式修改脚本就好了，这样脚本也更容易维护。”他们建议加入一个命令/步骤解释器，使用简单易写的语言编写，并将其转换为低级命令。

> **对 RQ 的回答$_6$**：开发人员认为，V2S+ 能够准确地根据输入视频复现场景中的应用使用，并且有望用于支持多种开发任务，包括自动复现缺陷、分析应用行为、帮助用户和质量保证人员生成高质量视频，以及自动执行场景。

7 相关工作

我们简要讨论与 V2S+ 相关的先前研究工作。

7.1 视频与屏幕捕获分析 s

Lin 等人 [64] 提出了一种名为 Screenmilker 的方法，通过 Android 调试桥自动提取敏感信息的屏幕截图（例如，用户输入密码）。该技术专注于从“实时”屏幕截图中提取键盘输入。Screenmilker 主要专注于提取敏感信息，而 V2S+ 会分析视频的每一帧，根据一系列视频帧生成高保真度的回放脚本。

Krieter 等人 [65] 使用视频分析，从 Android 应用的用户视频录制内容中提取事件的高层描述。他们的方法会生成日志文件，描述应用层正在发生的事件。与我们的工作相比，这项技术无法生成能够在设备上自动回放动作的脚本，而只能描述高层应用事件（例如，“*WhatsApp* 聊天列表已关闭”）。此外，我们的工作侧重于利用视频分析帮助错误复现和生成测试场景，而非高层次地描述使用场景。

Bao 等人 [66] 和 Frisson 等人 [67] 专注于利用计算机视觉技术提取用户交互，以便在编程任务期间对开发人员进行行为分析。我们的工作并不关注记录开发人员的交互，而是着重理解并提取移动应用中的通用用户操作，以生成高保真度的回放脚本。

其他研究者提出了一些方法，着重于根据屏幕截图或模型图生成安卓应用的源代码。这些方法所依赖的技术范围从基于计算机视觉 [68] 到基于深度学习[47], [69], [70]。

与 V2S+ 最相关的工作是 Hu等人 [71]提出的 AppFlow 方法。AppFlow 利用机器学习技术分析 Android 屏幕，并对可在这些屏幕上执行的测试用例进行分类（即，登录屏幕对应的测试用例可以是用户尝试登录）。不过，这项技术侧重于结合自动化动态分析生成具有语义意义的测试用例。相比之下，V2S+ 侧重于自动复现安卓设备上的任意类型用户交互，无论其描述的是使用场景还是错误。因此，V2S+ 可用于自动重现众包移动应用视频，而 AppFlow 主要关注生成测试，而非复现已有场景。

其他研究致力于检测重复的视频缺陷报告。例如，Cooper 等人 [72]提出了一种方法，用于处理每一份新提交的缺陷报告，并自动生成最相似的前 5 条记录列表，同时在问题跟踪系统中将该报告标记为可能的重复项。与 V2S+类似，TANGO 利用流行的计算机视觉技术分析视频输入。不过，TANGO 还会依据所提交缺陷报告中的文本内容得出结论，而 V2S+ 不会这样做；它使用文本检索和字符识别工具来分析这些额外信息。Kordopatis-Zilos 等人 [73] 提出了一种利用卷积神经网络提取近乎相同视频的方法，这些视频可能会被纳入缺陷报告。该框架使用三个不同的深度网络和两种特征聚合技术，提取并理解视觉特征，同时收集元数据，以便后续比较视频。与 V2S+非常相似，该方法利用卷积神经网络框架将视觉信息转换为数据，从而能够结合具体问题进行处理和组合。不过，上述两种方法都没有实现 V2S+所采用的同类动作分类或重放技术。

7.2 记录与回放

有许多工具可用于录制和回放移动平台上的测试 [52], [74], [75], [76], [77]。然而，这些工具中有许多需要使用 adb 录制低级事件，这通常要求对设备进行 root，或加载定制操作系统（OS），以捕获无法通过 adb 等标准工具获取的用户动作/事件。我们的方法使用 RERAN [52]回放系统级事件，同时依靠视频帧将触控叠加层转换为低级事件。这样一来，用户只需使用少量专用程序来录制和回放用户场景，便能更轻松地报告缺陷。

Hu 等人 [78] 开发了 VALERA，用于回放设备操作、传感器和网络输入（例如全球定位系统、加速度计等）、事件计划以及应用间通信。这种方法要求目标设备已获取 root 权限，并安装经过修改的 Android 运行时环境。这些要求可能带来实际限制，例如设备配置开销，以及已获取 root 权限的设备可能存在的安全隐患。开发人员 [79]通常不希望满足这些要求。相比之下，我们的方法无需对设备进行 root，也无需修改 Android 版本，只需进行屏幕录制即可运行。

Nurmuradov 等人 [80]为安卓应用引入了一种记录与回放工具，该工具通过在网页浏览器中显示设备屏幕来捕获用户交互。该技术使用记录过程中捕获的事件数据生成热力图，帮助开发人员了解用户如何与应用程序进行交互。这种方法的局限在于，用户必须通过 Web 应用与虚拟 Android 设备进行交互，这可能导致不自然的使用模式。该技术更侧重于基于会话的可用性测试，而 V2S+ 则侧重于回放从真实设备通过屏幕录制收集的用户或众包测试人员在“实际环境中”的应用使用情况。

Tuovenen 等人 [81] 将记录和重放技术应用于 Android 游戏应用测试。他们介绍了 MAuto，这是一款能够记录屏幕截图以总结用户在实际环境中的应用交互、为用户交互过程中发生的物体和事件创建测试脚本，并随后重放测试的工具。这使移动应用开发人员无需手动定义或运行这些测试用例，便能改进测试套件。MAuto 与 V2S+有许多相似之处，包括各自流水线的一般执行流程。不过，V2S+ 是一款端到端工具，而 MAuto 依赖 AKAZE 特征对输入屏幕截图进行基于图像的对象检测，再依赖 Appium 框架重放事件。此外，V2S+ 所需的用户配合更少

用户只需提交一段在设备上启用了触摸指示器的屏幕录像，以显示用户的点击。

其他研究则着重捕获高层次交互，以便回放事件 [74], [82], [83], [84]。例如，Mosaic [82],使用用户交互的中间表示，使回放不受设备限制。其他工具包括 HiroMacro [85] 和 Barista [60]，它们是允许用户记录与回放交互的安卓应用程序。这些工具需要安装或集成底层框架，例如 Replaykit [75], 、AirTest[74], 或 Troyd [76]。Android Bot Maker [86] 是一款安卓应用程序，可自动执行单个设备操作，但不支持记录高层次交互；用户必须手动输入动作类型和原始坐标（x，y）。与这些技术不同，V2S+的主要目标之一是创建一种易于上手的安卓记录与回放解决方案。例如，无需安装任何框架或添加任何插桩，唯一需要的输入就是可收集的屏幕录制。这使 V2S+ 适用于众包测试或 Beta 测试场景；相较于开发人员对其他工具的认知，V2S 易于使用，因此更有可能被开发人员用于自动化测试 [87]。

最后，随着通过多种测试服务 [88], [89], [90], 从移动应用用户处收集众包信息日益普遍，研究人员开始利用众包测试人员记录的使用数据，改进与移动应用自动化测试用例生成相关的技术。Linares-V´asquez 等人率先提出记录众包数据的概念，并通过推出 MONKEYLAB 框架 [91]，利用这些数据改进移动测试的输入生成策略。该技术根据记录的用户数据训练 N 元语言模型，以预测用于自动化移动测试用例生成的事件序列。Mao 等人开发了 POLARIZ 方法，该方法能够从众包数据中推断出适用于不同应用的“模式”事件序列 [9]。作者发现，结合基于众包和基于搜索的技术，可以提高 SAPIENZ 自动化测试工具的基于活动的覆盖率。上述两种方法都需要对设备进行插桩或使用已获取 root 权限的设备（MONKEYLAB），或者通过网页浏览器与应用交互（POLARIZ）。因此，V2S+ 与这些技术互为补充，因为它为开发人员提供了一种便捷的机制，可通过视频形式收集用户数据。

7.3 混合应用程序测试

应用程序测试是移动开发过程中的关键步骤。尽管混合应用在 Google Play商店中日益普及 [25], ，但由于混合应用带来的挑战，测试工具和技术往往会忽视它们。Wan 等人 [92] 指出，由于混合应用采用 Web 标准而非传统的 Android UI 元素构建，应用爬虫无法有效测试混合应用的全部功能。因此，依赖视觉信息的其他测试方法有助于补充应用爬取技术。V2S+提供了一种有效生成测试场景的替代方法，仅使用视频中包含的视觉信息来生成混合应用的测试场景。未来，使用 V2S+ 捕获的场景可用于为新型混合应用界面爬虫构建混合应用模型。

在探索其他测试挑战（如测试迁移）时，也曾强调与混合应用相关的问题。Talebipour 等人 [93] 探索了跨移动平台（例如，安卓或 iOS）的用户界面测试迁移。本文指出，由于界面设计语言和图标用法不同，在不同平台间迁移测试比在不同的 Android 设备或版本间迁移更具挑战性。混合应用面临的挑战与此类似，因为跨平台应用通常采用依赖网页技术的框架构建，例如 React Native。Lin 等人 [26]开发了 CRAFTDROID，这是一种通过依据图形用户界面元素的语义功能将其关联起来，从而在应用程序之间迁移测试预言机的技术。由于该工具无法解析混合应用用来实现功能的网页元素，因此无法用于混合应用。

过去支持混合应用测试的研究有限。最值得一提的是，Yazdani-BanafsheDaragh 等人 [94]提出了*DeepGUI*，这是一种黑盒测试框架，旨在适用于采用多种实现方式构建的应用程序（例如，原生应用和混合应用）。作者希望在以往测试工具的基础上进一步发展，通过学习基于应用程序屏幕像素的交互来实现这一目标。V2S+ 与*DeepGUI* 类似，都是跨平台方案，但其目标不同：它支持记录特定的用户提供的测试场景，而非基于学习到的模型生成输入序列。

8 局限性与有效性威胁

局限性。 我们的方法存在多种局限性，也为未来的工作提供了研究动机。目前，FASTER R-CNN 实现的一项局限性在于，它与单一设备的屏幕尺寸绑定，因此，对于 V2S+ 所应用的每种屏幕尺寸，都必须单独训练一个模型。然而，如第 3.2 节所述，训练数据的生成过程完全自动化，模型只需训练一次，即可用于任何具有相同屏幕尺寸的设备。扩大数据集规模，将所有类型的屏幕尺寸纳入其中，可以缓解这一局限性，但会增加训练时间。为方便研究社区使用我们的模型，我们已在在线附录 [28]中发布了适用于两种常见屏幕尺寸（Nexus 5 和 Nexus 6P）的训练好的模型。

另一项我们将在未来工作中解决的局限性是，目前我们的可重放轨迹与特定屏幕的尺寸绑定，而且不易于人类阅读。不过，将 V2S+ 与现有的设备无关型测试用例转换技术 [60], 以及用于生成自然语言报告的图形用户界面分析技术 [56]相结合，或可缓解这些局限性。

如第 6.3 节所述，V2S+ 忠实重放滑动手势的能力也受到视频帧率的限制。在评估过程中，我们的设备仅支持 30帧/秒，因此难以完整解析一小部分执行得非常迅速的手势操作。不过，可以通过改进安卓

硬件或软件来解决这一限制，使其能够以 60帧/秒或更高的帧率录制视频。根据我们的经验，这应该足以解析几乎所有快速的用户手势。

最后，V2S+ 也有局限性：与在模拟设备上使用相比，在实体设备上使用时效果明显更好。如第 6.3.1 节所述，用户必须以不自然的方式与模拟设备交互，加上计算机处理速度各异，导致生成的输入视频质量较差，V2S+无法很好地重现这些视频。我们预计，以更高的帧率（>30 每秒帧数）进行模拟器屏幕录制，或使用处理器或显卡速度更快、性能更强的机器，可能会取得更好的结果。

内部效度。在评估 V2S+的实验中，内部效度方面的威胁可能源于我们对回放视频正确性的人工验证。为尽可能减少主观性或错误，我们安排两位作者人工核验重放场景的正确性。此外，我们已发布所有实验数据和代码 [28], ，以便他人复现实验。

构念效度。对构念效度的主要威胁，源于我们为回答 RQ_3 和 RQ_4 而开展的热门应用研究中，手动创建视频可能产生的偏差。作者对 V2S+ 的了解可能影响了视频的录制方式。为减轻这一威胁，我们尽可能自然地录制视频（例如，以正常速度录制，并包含自然、快速的手势）。此外，在回答 RQ_3，的过程中，我们开展了一项正交研究：不熟悉 V2S+ 的用户在实体设备和模拟设备上自然地录制视频，从而获得了一组无偏的视频。

另一个潜在的混杂因素涉及用于训练、测试和评估 V2S+ 的 FASTER R-CNN 和不透明度卷积神经网络的屏幕数据集质量。为减轻这一威胁，我们使用了 REDRAW 数据集 [47] 中的屏幕；这些屏幕经过了多轮过滤和质量控制，以确保其中包含多种真实的图形用户界面。另一个潜在威胁涉及我们评估 V2S+ 实用性的方法。我们的开发者访谈仅评估了这项技术感知到的实用性；开发人员是否确实能从 V2S+ 中获益，则有待未来研究。

外部效度。 我们的结论能否推广，主要面临以下威胁：(i) 评估中使用的应用数量和多样性；(ii) 实验视频所呈现的使用场景是否具有代表性；以及 (iii) 受访开发人员所给反馈的普遍适用性。为缓解第一项威胁，我们开展了 2 项大规模研究：一项研究使用了从 32 个类别中选出的 Google Play 上 64 款热门原生应用，另一项研究则使用了从 7 个类别中选出的 Google Play商店中 14 款最受欢迎的混合应用。虽然理想情况下还应使用更多应用开展额外实验，但我们的实验应用集涵盖了数量相当可观、功能各异的应用，展现了 V2S+ 的相对适用性。为缓解第二项威胁，我们从 8 位参与者那里收集了展示缺陷、自然应用使用、真实崩溃和受控崩溃的场景。在热门应用研究中，我们还收集了多种场景，其中包含按顺序组合使用单指和多指动作，旨在提升对自然应用使用的概括性。

此外，我们并未声称从开发人员那里获得的反馈能够广泛适用于各个工业团队。不过，访谈中收到的积极反馈和对未来工作的建议，展现了 V2S+ 在实践中的潜在价值。

9 结论与未来工作

V2S+ 是一种可将安卓应用使用情况的视频录制自动转换为可重放场景的方法。全面评估结果表明，V2S+：（i）能够准确识别触摸指示器，并区分不同的不透明度级别；（ii）能够重现很大比例的完整场景，这些场景涉及在混合和原生应用中使用单指和多指动作，并在实体和模拟设备上执行；（iii）有望在各种任务中为实际开发人员提供帮助。

屏幕录制众所周知难以分析，而借助屏幕录制，V2S+ 成为同类首个自动转换工具。我们的所有发现表明，V2S+ 或可成为一项宝贵工具，帮助改进移动开发中固有的快速调试流程。我们预计，开发人员也可能非常愿意用它取代其他技术，这得益于V2S+较低的使用门槛：V2S+ 只需将屏幕录制作为输入，而且其模块化结构易于调整，能够适用于各种研究和开发任务。

我们未来的工作将着重于对 V2S+ 进行多项改进，包括：(i) 生成不依赖坐标的动作脚本，(ii) 生成自然语言用户场景，(iii) 通过行为分析改善用户体验，以及 (iv) 利用基于图形用户界面的信息，简化其他维护任务。

致谢

本研究部分由美国国家科学基金会的 CCF-1927679、CCF-1815186、CNS-1815336、CCF-1955837、CCF-1910976 和 CCF-1955853 项目资助。本文所述的任何观点、研究结果和结论均为作者所有，不一定反映资助方的意见。

参考文献

[1] M. Nayebi, 《心眼：社交编程的图像处理》，载于《第*42*届*IEEE/ACM*国际软件工程会议（*ICSE'20*）论文集》，2020年，第49–52页。[2] “Watchsend https://new.watchsend.com”，2019年。[3] “Testfairy https://testfairy.com”，2019年。[4] “Instabug https://instabug.com/screen-recording”，2021年。[5] “Bird eats bugs https://birdeatsbug.com”，2021年。[6] R. Schusteritsch、C. Y. Wei和M. LaRosa，《面向移动设备可用性测试的理想基础设施》，载于《*CHI*计算机系统人因扩展摘要集论文集》，系列：CHI'07，美国纽约州纽约市，2007年，第1839–1844页。[7] “Mr. tappy移动可用性测试设备 https://www.mrtappy. com”，2019年。[8] N. Bettenburg、S. Just、A. Schr¨oter、C. Weiss、R. Premraj和T. Zimmermann，《什么样的缺陷报告才算好？》，载于《第*16*届*ACM SIGSOFT*软件工程基础国际研讨会论文集》，系列：FSE'08，美国纽约州纽约市，2008年，第308–318页。

[9] K. Mao、M. Harman 和 Y. Jia，“Sapienz：多目标自动化移动测试》，“载于第 32 届 IEEE/ACM 自动化软件工程国际会议论文集，系列：ASE’17，美国新泽西州皮斯卡塔韦，2017年，第 16–26 页。[10] K. Moran、C. Bernal-C´ardenas、M. Linares-V´asquez 和 D. Poshy- vanyk，《克服语言二分性：迈向有效的移动应用开发程序理解》，载于第 26 届程序理解国际会议论文集，系列：ICPC’18，2018年，第 7–18 页。[11] G. Hu、X. Yuan、Y. Tang 和 J. Yang，《借助 appdoctor 高效、有效地检测移动应用缺陷》，载于第 9 届欧洲计算机系统会议论文集，系列：EuroSys’14，美国纽约州纽约市，2014年，第 18:1–18:15 页。[12] N. Jones，《优化移动测试工作的七项最佳实践》，Gartner，技术报告 G00248240，2013年。[13] M. Linares-V´asquez、G. Bavota、C. Bernal-C´ardenas、M. Di Penta、R. Oliveto 和 D. Poshyvanyk，《API变更与故障倾向性：对安卓应用成功的威胁》，载于第 9 届软件工程基础联合会议论文集，系列：ES-EC/FSE’13，美国纽约州纽约市，2013年，第 477–487 页。[14] G. Bavota、M. Linares-V´asquez、C. Bernal-C´ardenas、M. Di Penta、R. Oliveto 和 D. Poshyvanyk，《API变更和故障倾向性对安卓应用用户评分的影响》，IEEE软件工程汇刊，第 41 卷，第 4 期，第 384–407 页，2015年4月。[15] A. Ciurumelea、A. Schaufelb¨uhl、S. Panichella 和 H. Gall，《分析移动应用的评论与代码，以改进发布规划》，载于第 24 届 IEEE 软件分析、演化与再工程国际会议论文集，系列：SANER’17，2017年2月，第 91–102 页。[16] A. Di Sorbo、S. Panichella、C. V. Alexandru、J. Shimagaki、C. A. Visaggio、G. Canfora 和 H. C. Gall，《用户会希望我的应用做出哪些改变？汇总应用评论以推荐软件变更》，载于第 24 届 ACM SIGSOFT 软件工程基础国际研讨会论文集，系列：FSE’16，美国华盛顿州西雅图，2016年，第 499–510 页。[17] F. Palomba、P. Salza、A. Ciurumelea、S. Panichella、H. Gall、F. Ferrucci 和 A. De Lucia，《基于用户评论为移动应用推荐并定位变更请求》，载于第 39 届软件工程国际会议论文集，系列：ICSE’17，美国新泽西州皮斯卡塔韦，2017年，第 106–117 页。[18] F. Palomba、M. Linares-V´asquez、G. Bavota、R. Oliveto、M. D. Penta、D. Poshyvanyk 和 A. D. Lucia，《众包用户评论以支持移动应用的演化》，《系统与软件杂志》，第 143–162 页，2018年。[19] F. Palomba、M. Linares-Vasquez、G. Bavota、R. Oliveto、M. D. Penta、D. Poshyvanyk 和 A. D. Lucia，《用户评论很重要！追踪众包评论以支持成功应用的演化》，载于 IEEE 软件维护与演化国际会议论文集，系列：ICSME’15，2015年9月，第 291–300 页。[20] D. Han、C. Zhang、X. Fan、A. Hindle、K. Wong 和 E. Stroulia，《通过分析厂商特定缺陷的主题来理解安卓碎片化》，载于第 19 届逆向工程工作会议论文集，系列：WCRE’12，美国华盛顿哥伦比亚特区，2012年，第 83–92 页。[21] L. Wei、Y. Liu 和 S. C. Cheung，《驾驭安卓碎片化：表征并检测安卓应用的兼容性问题》，载于第 31 届 IEEE/ACM 自动化软件工程国际会议论文集，系列：ASE’16，2016年9月，第 226–237 页。[22]《安卓碎片化统计数据 http://opensignal.com/reports/2014/android-fragmentation/》，2014年。[23] M. Linares-V´asquez、K. Moran 和 D. Poshyvanyk，《持续、演进且大规模：自动化移动应用测试的新视角》，载于 IEEE 软件维护与演化国际会议论文集，系列：ICSME’17，2017年，第 399–410 页。[24] C. Bernal-C´ardenas、N. Cooper、K. Moran、O. Chaparro、A. Marcus 和 D. Poshyvanyk，《将移动应用使用的视频录制转换为可重放场景》，载于 ACM/IEEE 第 42 届软件工程国际会议（ICSE’20）论文集，2020年，第 309–321 页。[25] I. Malavolta、S. Ruberto、T. Soru 和 V. Terragni，《Google Play商店中的混合型移动应用：一项探索性调查》，2015年8月。[26] J.-W. Lin、R. Jabbarvand 和 S. Malek，《通过语义映射在移动应用间迁移测试》，载于 2019 年第 34 届 IEEE/ACM 自动化软件工程国际会议[27] “Monkey https://developer.android.com/studio/test/ other-testing-tools/monkey”，2022年。[28] C. Bernal-C´ardenas、N. Cooper、M. Havranek、K. Moran、O. Cha- parro、A. Marcus 和 D. Poshyvanyk，“V2S 在线附录 https: //sites.google.com/email.wm.edu/video2scenario/home”，2022年。[29] A. Krizhevsky、I. Sutskever 和 G. E. Hinton，“使用深度卷积神经网络进行 ImageNet 分类”，载于第25届神经信息处理系统会议论文集，系列：NeurIPS’12，2012年，第1097–1105页。[30] K. Simonyan 和 A. Zisserman，“用于大规模图像识别的超深卷积网络”，载于国际学习表征会议论文集，系列：ICLR’14，第abs/1409.1556卷，2014年。[31] M. D. Zeiler 和 R. Fergus，“卷积网络的可视化与理解”，载于第13届欧洲计算机视觉会议论文集，系列：ECCV’14，瑞士查姆，2014年，第818–833页。[32] C. Szegedy、W. Liu、Y. Jia、P. Sermanet、S. Reed、D. Anguelov、D. Erhan、V. Vanhoucke 和 A. Rabinovich，“通过卷积深入网络”，载于计算机视觉与模式识别会议论文集，系列：CVPR’15，2015年。[33] K. He、X. Zhang、S. Ren 和 J. Sun，“用于图像识别的深度残差学习”，载于IEEE计算机视觉与模式识别会议论文集，系列：CVPR’16，2016年6月，第770–778页。[34] M. Tan 和 Q. Le，“EfficientNet：重新思考卷积神经网络的模型缩放”，载于第36届国际机器学习会议论文集，系列：机器学习研究论文集，K. Chaudhuri 和 R. Salakhutdinov 编，第97卷。PMLR，2019年6月9日至15日，第6105–6114页。 [在线]。可查阅：https://proceedings.mlr.press/v97/tan19a.html [35] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、L. Kaiser 和 I. Polosukhin，“注意力机制就是你所需要的一切”，载于第31届神经信息处理系统国际会议论文集，系列：NIPS’17。美国纽约州雷德胡克：Curran Associates Inc.，2017年，第6000–6010页。[36] B. Wu、C. Xu、X. Dai、A. Wan、P. Zhang、M. Tomizuka、K. Keutzer 和 P. Vajda，“视觉Transformer：面向计算机视觉的基于令牌的图像表示与处理”，CoRR，第abs/2006.03677卷，2020年。 [在线]。可查阅：https://arxiv.org/abs/2006 03677 。I. O. Tolstikhin、N. Houlsby、[37] A. Kolesnikov、L. Beyer、X. Zhai、T. Unterthiner、J. Yung、A. Steiner、D. Keysers、J. Uszkoreit、M. Lucic 和 A. Dosovitskiy，“MLP-Mixer：用于视觉的全MLP架构”，CoRR，第abs/2105.01601卷，2021年。 [在线]。可查阅：https://arxiv.org/abs/2105 01601 。R. Girshick、J. Donahue、T. [38] Darrell 和 J. Malik，“用于精准目标检测和语义分割的丰富特征层次结构”，载于IEEE计算机视觉与模式识别会议论文集，系列：CVPR’14，美国华盛顿哥伦比亚特区，2014年，第580–587页。[39] J. R. R. Uijlings、K. E. A. van de Sande、T. Gevers 和 A. W. M. Smeulders，“用于目标识别的选择性搜索”，《国际计算机视觉期刊》，第104卷，第2期，第154–171页，2013年9月。[40] N. Carion、F. Massa、G. Synnaeve、N. Usunier、A. Kirillov 和 S. Zagoruyko，“使用Transformer进行端到端目标检测”，CoRR，第abs/2005.12872卷，2020年。 [在线]。可查阅：https: //arxiv.org/abs/2005 12872 。S. Ren、K. He、R. [41] Girshick 和 J. Sun，“Faster R-CNN：借助区域提议网络实现实时目标检测”，载于第28届神经信息处理系统国际会议论文集，系列：NeurIPS’15，美国马萨诸塞州剑桥，2015年，第91–99页。[42] “Google Play 屏幕录制应用 https://play.google.com/ store/search?q=screen%20recording&c=apps”，2019年。[43] “安卓显示触摸选项 https://tinyurl.com/ show-touches”，2019年。[44] S. Ren、K. He、R. Girshick 和 J. Sun，“Faster R-CNN：借助区域提议网络实现实时目标检测”，《IEEE模式分析与机器智能汇刊》，第6期，第1137–1149页，2017年。[45] “Stackoverflow 安卓屏幕录制 https: //stackoverflow.com/questions/29546743/what-is-the-frame-rate-of-screen-record/44523688”，2019年。[46] “FFmpeg 工具 https://www.ffmpeg.org/”，2019年。

[47] K. P. Moran、C. Bernal-Cardenas、M. Curcio、R. Bonett 和 D. Poshyvanyk，“面向移动应用的图形用户界面机器学习原型设计”，《IEEE软件工程汇刊》，2018年。[48] “TensorFlow 对象检测 API https://github.com/tensorflow/ models/tree/master/research/object detection”，2019年。[49] “跟踪触摸和指针移动 https://developer.android. com/training/gestures/movement”，2019年。[50] “在视图组中管理触摸事件：使用视图配置常量 https://developer.android.com/training/ gestures/viewgroup#vc”，2019年。[51] “视图配置定义（安卓操作系统）https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/view/ViewConfiguration.java”，2006年。[52] L. Gomez、I. Neamtiu、T. Azim 和 T. Millstein，“RERAN：适用于安卓的时序与触摸敏感型记录和回放”，载于第35届软件工程国际会议论文集，系列：ICSE’13，2013年，第72–81页。[53] “多点触控（MT）协议 https://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt”，2009年。[54] K. Moran、C. B. Cardenas、M. Curcio、R. Bonett 和 D. Poshy- vanyk，“Redraw 数据集：一组安卓屏幕截图、图形用户界面元数据及图形用户界面组件标注图像 https://zenodo. org/record/2530277#.YrOFTy-B27M”，2018年。[55] T. Y. Lin、M. Maire、S. Belongie、J. Hays、P. Perona、D. Ramanan、P. Doll´ar 和 C. L. Zitnick，“Microsoft COCO：上下文中的常见物体”，第740–755页，2014年。[56] K. Moran、M. Linares-V´asquez、C. Bernal-C´ardenas、C. Vendome 和 D. Poshyvanyk，“自动发现、报告并复现安卓应用崩溃”，载于IEEE软件测试、验证与确认国际会议论文集，系列：ICST’16。IEEE，2016年，第33–44页。[57] K. Moran、M. Linares-V´asquez、C. Bernal-C´ardenas 和 D. Poshy- vanyk，“自动补全安卓应用的缺陷报告”，载于第10届欧洲软件工程会议与第23届ACM SIGSOFT软件工程基础研讨会联合会议论文集，系列：ESEC/FSE’15，意大利贝加莫，2015年8月至9月。[58] O. Chaparro、C. Bernal-C´ardenas、J. Lu、K. Moran、A. Marus、M. Di Penta、D. Poshyvanyk 和 V. Ng，“评估缺陷报告中复现步骤的质量”，载于第27届ACM欧洲软件工程会议与软件工程基础研讨会联合会议论文集，系列：ESEC/FSE’19，意大利贝加莫，2019年8月。[59] Z. Qin、Y. Tang、E. Novak 和 Q. Li，“Mobiplay：一种基于远程执行的移动应用记录与回放工具”，载于第38届软件工程国际会议论文集，系列：ICSE’16，美国纽约州纽约，2016年，第571–582页。[60] M. Fazzini、E. N. D. A. Freitas、S. R. Choudhary 和 A. Orso，“Barista：一种用于记录、编码和运行平台无关安卓测试的技术”，载于第10届IEEE软件测试、验证与确认国际会议论文集，系列：ICST’17，2017年3月，第149–160页。[61] “Google 搜索 https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox”，2019年。[62] “7-Eleven https://play.google.com/store/apps/details?id=com. sei.android”，2019年。[63] “Proximus https://www.proximus.be”，2019年。[64] C.-C. Lin、H. Li、X. Zhou 和 X. Wang，“Screenmilker：如何从安卓屏幕中榨取秘密”，载于网络与分布式系统安全研讨会论文集，系列：NDSS’14，2014年。[65] P. Krieter 和 A. Breiter，“分析移动应用使用情况：从移动设备屏幕录制生成日志文件”，载于第20届人机交互与移动设备及服务国际会议论文集，系列：MobileHCI’18，2018年，第1–10页。[66] L. Bao、J. Li、Z. Xing、X. Wang 和 B. Zhou，“scvripper：一种用于视频抓取的工具，可利用交互数据对开发者行为建模”，载于第37届软件工程国际会议论文集，系列：ICSE’15。IEEE Press，2015年，第673–676页。[67] C. Frisson、S. Malacria、G. Bailly 和 T. Dutoit，“Inspectorwidget：一种分析用户在其应用中行为的系统”，载于CHI计算系统人因会议扩展摘要，系列：CHI’16。ACM，2016年，第1548–1554页。[68] T. A. Nguyen 和 C. Csallner，“使用remaui对移动应用用户界面进行逆向工程”，载于第30届 *IEEE/ACM* 自动化软件工程国际会议 - ing，系列。ASE’15，美国华盛顿哥伦比亚特区，2015年，第248–259页。[69] T. Beltramelli，《pix2code：从图形用户界面截图生成代码》，载于ACM交互式计算系统工程研讨会论文集，系列：SIGCHI’18。ACM，2018年，第3页。[70] C. Chen、T. Su、G. Meng、Z. Xing和Y. Liu，《从UI设计图像到GUI骨架：用于启动移动GUI实现的神经机器翻译器》，载于第40届国际软件工程会议论文集，系列：ICSE’18。ACM，2018年，第665–676页。[71] G. Hu、L. Zhu和J. Yang，《AppFlow：利用机器学习合成稳健、可复用的UI测试》，载于第26届ACM欧洲软件工程会议与软件工程基础研讨会联合会议论文集，系列：ESEC/FSE’18，2018年，第269–282页。[72] N. Cooper、C. Bernal-C´ardenas、O. Chaparro、K. Moran和D. Poshyvanyk，《TANGO需要两人合作：结合视觉和文本信息检测重复的视频缺陷报告》，载于第43届IEEE/ACM国际软件工程会议，ICSE 2021，西班牙马德里，2021年5月22日至30日。IEEE，2021年，第957–969页。 [在线]。获取地址：https://doi.org/10 1109 43902 2021 00091 。/ICSE . . G. [73] Kordopatis-Zilos、S. Papadopoulos、I. Patras和I. Kompat-siaris，《通过聚合中间CNN层进行近重复视频检索》，第10132卷，2017年1月，第251–263页。[74] “Airtest项目”，2019年。 [在线]。获取地址：http://airtest.netease.com/[75] “Replaykit：用于安卓触摸屏事件录制、回放和镜像的命令行工具”，2019年。 [在线]。获取地址：https://github.com/appetizerio/replaykit [76] J. Jeon和J. S. Foster，《Troyd：安卓集成测试》，技术报告，2012年。[77] K. Moran、R. Bonett、C. Bernal-Cardenas、B. Otten、D. Park和D. Poshyvanyk，《安卓应用程序的设备端缺陷报告》，载于IEEE/ACM第4届移动软件工程与系统国际会议论文集，系列：MOBILESoft’17，2017年，第215–216页。[78] Y. Hu、T. Azim和I. Neamtiu，《面向安卓的通用而轻量级的记录与回放》，载于ACM SIGPLAN面向对象编程、系统、语言与应用国际会议论文集，系列：OOPSLA’15，2015年，第349–366页。[79] W. Lam、Z. Wu、D. Li、W. Wang、H. Zheng、H. Luo、P. Yan、Y. Deng和T. Xie，《安卓记录与回放：工业案例中我们是否已达到目标？》，载于第11届软件工程基础联合会议论文集，2017年，系列：ESEC/FSE’17，2017年，第854–859页。[80] D. Nurmuradov和R. Bryce，《Caret-HM：通过UI状态聚类记录和回放安卓用户会话并生成热力图》，载于第26届ACM SIGSOFT软件测试与分析国际研讨会论文集，系列：ISSTA’17，2017年，第400–403页。[81] J. Tuovenen、M. Oussalah和P. Kostakos，《MAuto：一种基于图像匹配方法的自动移动游戏测试工具》，Comput Game，第8期，第215–239页，2019年。[82] M. Halpern、Y. Zhu、R. Peri和V. J. Reddi，《Mosaic：面向碎片化安卓生态系统的跨平台用户交互录制与回放》，载于IEEE系统与软件性能分析国际研讨会论文集，系列：ISPASS’15，2015年，第215–224页。[83] D. T. Milano，《Android ViewServer客户端。通过在GitHub上创建账户，为dtmilano/AndroidViewClient的开发做出贡献》，2019年。[在线]。获取地址：https://github.com/dtmilano/AndroidViewClient [84] （2019）使用Espresso Test Recorder创建UI测试。[在线]。获取地址：https://developer.android.com/studio/test/espresso-test-recorder [85] （2019）HiroMacro自动触控宏——Google Play上的应用。[在线]。获取地址：https://play.google.com/store/apps/details?id=com.prohiro.macro&hl=en [86] （2019） [ROOT] 安卓机器人制作工具——Google Play上的应用。[在线]。获取地址：https://play.google.com/store/apps/details?id=com.frapeti.androidbotmaker paid&hl=en US&gl=US [87] M. Linares-V´asquez、C. Bernal-Cardenas、K. Moran和D. Poshyvanyk，《开发者如何测试安卓应用？》，载于IEEE软件维护与演化国际会议论文集，系列：ICSME’17，¯2017年9月，¯第613–622页。

[88] “Applause众包测试服务 https://www.applause.com/crowdtesting”，2019年。[89] “ Testbirds众包测试服务 https://www.testbirds.com”，2019年。[90] “ Mycrowd众包测试服务 https://mycrowd.com”，2019年。[91] M. Linares-V´asquez、M. White、C. Bernal-C´ardenas、K. Moran和 D. Poshyvanyk，《挖掘安卓应用使用情况以生成可执行的基于图形用户界面的执行场景》，载于第*12*届软件仓库挖掘工作会议论文集，系列：MSR’15，2015年，第111–122页。[92] M. Wan、N. Abolhassani、A. Alotaibi和W. G. J. Halfond，《安卓应用程序中用户界面实现的实证研究》，载于*2019*年*IEEE*国际软件维护与演化会议（*ICSME*），2019年，第65–75页。[93] S. Talebipour、Y. Zhao、L. Dojcilovi´c、C. Li和N. Medvidovi´c，《跨移动平台的UI测试迁移》，载于*2021*年第*36*届*IEEE/ACM*国际自动化软件工程会议（*ASE*），2021年，第756–767页。[94] F. YazdaniBanafsheDaragh和S. Malek，《Deep GUI：利用深度学习进行黑盒图形用户界面输入生成》，载于*2021*年第*36*届*IEEE/ACM*国际自动化软件工程会议（*ASE*），2021年，第905–916页。



Carlos Bernal-C´ardenas 目前在微软担任软件开发工程师。他于2012年获得哥伦比亚国立大学系统工程理学学士学位，并于2015年获得系统与计算工程硕士学位。他于2021年8月获得威廉与玛丽学院博士学位。他的研究兴趣包括软件工程、软件演化与维护、信息检索、软件复用、软件仓库挖掘和移动应用开发，以及用户体验。他曾在多个顶级同行评审软件工程会议上发表论文，包括：ICSE、ESEC/FSE、ICST 和 MSR。他还曾荣获 ESEC/FSE’15 和 ’19 以及 ICSE’20 的 ACM SIGSOFT杰出论文奖。更多信息请访问 http://www.cs.wm.edu/~cebernal/。



Nathan Cooper </style]是一位技术爱好者，2018年获得西佛罗里达大学软件工程理学学士学位。目前，他在威廉与玛丽学院攻读计算机科学博士学位，师从 Denys Poshyvanyk博士，并加入了 SEMERU研究组。他的研究兴趣包括软件工程、用于软件工程的机器学习/深度学习应用、信息检索，以及用于软件工程的问答应用。他曾发表论文于 ICSE 和 MSR 等顶级同行评审软件工程会议上。他还在 ICSE’20 上获得了 ACM SIGSOFT 杰出论文奖。更多信息请访问 https://nathancooper.io/#/。



Madeleine Havranek 是威廉与玛丽学院计算机科学系的一名本科生。她目前是SEMERU研究组的成员，正在撰写一篇本科荣誉论文，研究如何实现软件测试复现的自动化。她的研究兴趣包括将深度学习应用于软件工程和测试任务。Havranek是IEEE学生会员。



KevinMoran 目前是乔治梅森大学的助理教授，并负责领导SAGE研究组。他于2013年获得圣十字学院物理学文学学士学位，并于2015年8月获得威廉与玛丽学院的理学硕士学位。他于2018年8月获得威廉与玛丽学院的博士学位。他的主要研究兴趣是促进软件工程、维护和演进流程，重点关注移动平台。他曾在多本顶级同行评审期刊上发表论文，他曾为包括 ICSE、ESEC/FSE、TSE、USENIX、ICST、ICSME 和 MSR 在内的软件工程会议和期刊担任评审。他的研究获得了认可，在 ESEC/FSE 2019 和 ICSE 2020 上荣获 ACM SIGSOFT杰出论文奖，并在 CODASPY’19 上获得最佳论文奖。他是 ACM 和 IEEE 的会员。更多信息请访问 http://www.kpmoran.com



Oscar Chaparro 是威廉与玛丽学院的助理教授。他于2019年在德克萨斯大学达拉斯分校获得软件工程博士学位，导师为Andrian Marcus博士。他于2010年和2013年分别获得哥伦比亚国立大学系统工程与计算专业的工学学士和工学硕士学位。他的研究兴趣包括软件维护与演进、程序理解、代码重构、代码质量、开发者生产力，以及应用于软件的文本分析工程。Oscar曾在多个顶尖软件工程领域的会议上发表论文，例如ICSE、ESEC/FSE、EMSE和ICSME。他在ESEC/FSE’19和ICSE’20上获得ACM SIGSOFT杰出论文奖，并在ICSME’17上获得IEEE TCSE杰出论文奖。Oscar曾参与多个会议和研讨会的组织委员会及程序委员会工作，包括ICSE、ASE、ICSME、MSR、ICPC，以及NLBSE’22、DySDoc3和DocGen2研讨会。Oscar是IEEE和ACM会员。更多信息请访问：https://ojcchar.github.io/



Denys Poshyvanyk 是威廉与玛丽学院计算机科学系的教授。他分别于2003年和2006年在乌克兰基辅莫希拉国立大学和韦恩州立大学获得计算机科学硕士学位和文学硕士学位，并于2008年在韦恩州立大学获得计算机科学博士学位。他曾担任 ASE’21、MobileSoft’19、ICSME’16、ICPC’13、WCRE’12 和 WCRE’11 的程序联合主席。目前，他是 IEEE Transac- 编委会成员。《软件工程（TSE）》、《实证软件工程》期刊（EMSE，施普林格）、《软件：演化与过程》期刊（JSEP，威立）以及《计算机编程科学》。他的研究兴趣包括软件工程、软件维护与演化、程序理解、逆向工程和软件仓库挖掘。他的研究论文曾获得 ICPC’06、ICPC’07、ICSM’10、SCAM’10、ICSM’13 和 CODAPSY’19 的多项最佳论文奖，并在 ASE’13、ICSE’15、ESEC/FSE’15、ICPC’16、ASE’17、ESEC/FSE’19 和 ICSE’20 获得 ACM SIGSOFT杰出论文奖。他还曾获得 ICSME’16、ICPC’17、ICPC’20 和 ICSME’21 的最具影响力论文奖。他是美国国家科学基金会 CAREER 奖（2013）的获得者。他是 IEEE 和 ACM 的高级会员。更多信息请访问：http://www.cs.wm.edu/~denys/



AndrianMarcus 是德克萨斯大学达拉斯分校计算机科学与软件工程教授。他获得了肯特州立大学（美国）的计算机科学博士学位，此前还分别获得孟菲斯大学（美国）和巴贝什-博雅伊大学（罗马尼亚克卢日-纳波卡）的计算机科学和欧洲研究学位。他的研究兴趣涵盖软件工程，重点是程序理解和软件演化。他曾获富布赖特学者称号，并且

美国国家科学基金会 CAREER 奖获得者。他与学生及合作者开展的研究，曾在软件工程会议上荣获六项最佳/杰出论文奖和六项最具影响力论文奖。他还积极参与专业服务，担任过 ICSME 和 VISSOFT 指导委员会委员。他分别担任过 ICSME'11 和 ICSME'10 的大会主席及程序委员会共同主席，并担任其他会议（ICPC'09、VISSOFT'13、SANER'17）的程序委员会共同主席。他曾任《实证软件工程》期刊、《软件：演化与过程》期刊和 IEEE软件工程汇刊的编委会成员。更多信息请访问：https://personal.utdallas.edu/~amarcus/