

清华 大学

综合 论文 训 练

题目：手机动画效果的量化及卡顿感知模型的建立

系 别：计算机科学与技术系

专 业：计算机科学与技术

姓 名：王倩

指导教师：史元春教授

2018 年 6 月 21 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

· 签 名: 王伟 导师签名: 孙杰 日 期: 2018年6月21日

中文摘要

智能手机应用中的各种交互动画效果使用户体验自然、流畅。然而，由于软硬件因素的影响，这些动画效果中往往会出现卡顿现象，影响用户体验。本研究一方面采用计算机视觉的方法，识别交互动画中的卡顿现象；另一方面开展大规模用户实验，经统计分析得到卡顿感知模型，最终建立卡顿分析评价体系。本研究在交互动画中卡顿的识别、多维度卡顿感知评价方面的探索，对理解用户需求、软硬件及系统优化，有重要指导价值。

关键词：智能手机交互；卡顿；用户感知；动画提取

ABSTRACT

Transitional animation on smartphones greatly improves user experience. However, due to limitations of both software and hardware, lagging happens during these animations, which on the contrary degrades user experience. This work seeks to model lagging perception and offer practical tools to evaluate an animation. First, we identify lagging from a video of transitional animation using algorithms in computer vision. Then we run a large-scale user experiment to study what factors affect users' experience and perception of lagging, and in what way do they affect. Then we combine these two parts and build a system that can evaluate a short video of transitional animation. This study is important for understanding users, and offers informative knowledge to those who intend to solve the problem of lagging.

Keywords: smartphone interaction; lagging; user perception; animation extraction

目 录

第 1 章 引言	1
1.1 研究背景	1
1.2 研究现状	2
1.3 本文研究内容	4
第 2 章 卡顿检测	6
2.1 视频获取	6
2.1.1 高速成像系统	6
2.1.2 动画任务的自动生成	7
2.2 动画提取	7
2.2.1 相关知识	8
2.2.2 平移动画	9
2.2.3 缩放动画	10
2.3 常见卡顿分析	12
2.4 卡顿检测	15
2.4.1 预处理	15
2.4.2 卡顿检测	16
第 3 章 卡顿感知用户实验	18
3.1 实验介绍	18
3.2 实验设计	19
3.2.1 设计细节	19
3.2.2 最终实验流程	22
3.3 技术实现	23
3.3.1 实验环境	23
3.3.2 卡顿生成方式	23
3.4 预实验	25
3.5 实验配置规则	26

第 4 章 卡顿感知模型的建立	27
4.1 背景知识	27
4.1.1 逻辑回归	27
4.1.2 有序逻辑回归	28
4.1.3 朴素贝叶斯	28
4.1.4 方差分析	29
4.1.5 交叉熵	29
4.2 数据预处理	30
4.3 数据观察	31
4.4 单因子分析	32
4.4.1 配置检验	33
4.4.2 多元变量分析	34
4.4.3 单卡顿单因子分析	35
4.4.4 多卡顿单因子分析	38
4.5 多因子分析	40
4.5.1 数据准备	40
4.5.2 模型评估指标	41
4.5.3 模型选取	42
4.5.4 模型分析	43
第 5 章 总结与展望	46
5.1 本文总结	46
5.2 本文展望	47
插图索引	49
表格索引	50
参考文献	51
致 谢	54
声 明	55
附录 A 外文资料的书面翻译	56
附录 B 各任务对应的模型参数	64

第 1 章 引言

1.1 研究背景

智能手机中的交互动画^①往往能有效地提升手机的用户体验。有研究表明，交互动画会影响用户的整体体验^[1]、对时间的感知^[2]、对界面的理解^[3]、甚至是完成任务的效率与正确率^[4]。然而，由于硬件性能、系统资源调度、软件实现、网络环境等因素的影响，这些动画效果中往往会出现卡顿^②的现象。本文所讨论的卡顿现象包括动画过程中短暂出现的“帧延迟”与“帧丢失”现象。其中，“帧延迟”也包含系统对用户交互的响应延迟。

可以想像，动画中的卡顿会给用户体验带来很大的负面影响，也有相关研究支持这一观点^[5]。业界也越来越重视交互动画流畅度的优化。例如 [6] 中就详细讨论了上下滑动、左右滑动以及其他类型交互动画的设计和优化。因此，智能手机中流畅性的用户体验（或卡顿的严重程度）亟需一个量化的指标。这一指标可以被用来检验手机流畅度，并依此指导卡顿问题的解决方案。

目前，业界存在一些流畅性指标，这些指标是 UI（User Interface）性能测试中的重要部分^[7]，主要包括平均响应延迟时间、刷新帧率^③等综合数据。此外 Android 还提供了一些包含更具体信息的工具用来辅助性能测试。比如 GPU 监测工具、overdraw^④分析工具^[8]、janky frames 报表^[9]、每一帧具体的时间记录等。然而，现有的评价指标和分析方法体现的都是系统本身客观的运行状态。而由于卡顿成因众多、发生的随机性大、对用户体验的影响复杂，其带来的用户体验并不能被上述的性能评价指标直接准确地度量。而真实的用户体验才是我们提升性能、解决卡顿的终极目标。特别是，当我们想要找到并解决最严重的卡顿问题时，用户体验最差的部分不一定是上述指标表现最差的部分。

目前，为了获得真实的用户体验，业界有时候会采用人工分析以上数据的方法来判断一个 App 的用户体验是否满足可以发布的要求^[10]，但少数几个人的判断显然不能够作为可靠的标准，而对每一次流畅性评价都召集大量用户又成

① 指智能产品中 UI(User Interface，用户界面) 元素位置、尺寸、形态等的变化。

② 卡顿对应的英文为 jank 或者 lag，jank 是安卓体系中开发人员对由于 CPU 计算能力不足所引起的“帧延迟”或“帧丢失”现象的表示。

③ Frame Per Second，FPS。手机理想状态下的刷新速度通常设定为 60 fps。

④ 由于代码原因，页面中的元素被重复绘制多次的现象。

本太高。此外，如果使用人工分析的方法，需要测试的条目太多。比如对于一个 App，需要测试其在不同机型上的表现；又比如对于一款手机，需要测试其运行不同 App 或不同数量 App 时的表现；而由于卡顿的出现具有较强的随机性也是重要原因原因之一，即便是同一款 App 也需要进行多次测试。因此可以说，人工分析的方法不仅效率较低，而且成本较高，收效甚微。我们亟需一种科学的方法能够高效地给出可靠的用户体验评价。而这首先依赖于对“用户对卡顿的感知模式”的了解，而目前我们相关的知识非常欠缺。

因此，本研究从了解用户对卡顿的感知模式开始，首先建立起卡顿感知模型，然后依据模型中的输入因素确定获取相关因素的技术方法，从而建立一套完整的卡顿评估方法和系统。研究中的主要难点在于：

1. 由于可能影响人对卡顿感知的因素繁多，传统的实验室用户实验的方法能够采集到的样本量过小，无法进行有效评估，因此大规模无监督用户实验的实验方法和设计非常重要，并具有较大的挑战。
2. 基于与上一点相同的原因，数据分析、数据有效性检验，以及合理的建模方式都具有一定的难度和探索性。
3. 由于需要提取实际 App 使用过程中的动画，我们只能通过计算机视觉的方法对画面进行分析，这过程中涉及到较多的算法。而为了获得与人看到的画面一致的动画，我们需要采用外部摄像机，由此带来的光环境、显示屏逐行更新^①等问题都会增加算法的难度与复杂度。

1.2 研究现状

我们对相关研究进行检索时^②，发现以往的研究集中于响应延迟（latency）与帧率（FPS）这两个话题，而几乎没有关于动画中突然出现的“帧延迟”或“帧丢失”的相关研究。因此，本节首先从多个方面介绍响应延迟与低帧率相关的现有研究成果，然后总结现有研究的不足。

人的感知阈值 响应延迟是一个比较经典的话题，在其感知方面已经有大量相关研究^[11-16]。人对乐器声音延迟的感知阈值依据不同的乐器分别为 10~100 ms 不等^[13]，而对图像反馈延迟的感知则有更多不同的研究。例如，使用不同设备时的感知不同：在使用鼠标时对延迟的感知阈值为 60 ms 左右^[14]，佩戴 HMD 时对

^① 显示屏上电子元件颜色的更新是逐行的，因此拍摄到的画面可能一半在新的状态，一半在旧的状态；此外，刷新不是瞬间完成的，因此拍摄到的画面可能是前后两帧重叠在一起的过渡阶段画面

^② 主要使用 lag, dropped frame, delayed frame, jank, smartphone, animation 等关键词进行检索

延迟的感知阈值为 3 ms^[15]，使用触摸屏时对延迟的感知阈值可以低至 2 ms^[16]。完成不同任务时的感知也不同：绘图书写任务（Sketching / Inking）中对延迟的感知阈值为 50 ms 左右，非绘图任务中对延迟的感知阈值为 2 ms ~ 7 ms 左右^[13]，拖拽一个部件时感知阈值为 6 ms 左右，而拖拽行为的结束阶段（land-on）的感知阈值则为 64 ms 左右。帧率也是影响流畅性体验的重要因素。电影行业通常采用 24 fps 作为标准，而智能手机通常采用 60 fps 作为标准，因为大多数人无法区分 60 fps 以上的帧率之间的差异^[17]。在上述研究中，为了寻找感知阈值，大多数工作采用了 Just-Noticeable Difference (JND) 原则，即通过同时或连续给用户两个不同参数的任务，让用户判断能否察觉其中的区别。

人的优良体验阈值 除了对响应延迟的感知阈值之外，也有相关工作给出了人能够忍受的响应延迟范围^[13] 以及影响其感知的因素（比如在书绘任务中，参照物是否可见有决定性影响，而绘制的内容与参照物的距离则影响不大^[13]）。对于帧率，相关研究^[18] 表明，在电子设备中，21 fps ~ 24 fps 以上的刷新速度能够让用户感到舒适，而低于 20 fps 的刷新速度则会被用户抗拒。

人被影响的方式 现有工作也从多个角度探究了响应延迟对人的影响，比如不同类型的游戏中的延迟^[12,19-20] 等对用户行为和表现的影响；选择或指向任务中的延迟对用户行为的影响^[21]；数据查询中的响应延迟对用户行为和知识获取能力的影响^[22]。

人的交互行为 由于手机交互动画效果中最常见的几种包括上下滑动、左右滑动、点击放大等，我们也针对人激发这些动画的行为特点进行了调研。一项针对触摸屏中上下滑动（scroll）行为的研究^[23] 有多项以往未被卡顿研究者重视的发现，比如，用户使用食指滑动屏幕的速度高于使用拇指；手机屏幕较大会导致滑动行为时间变长，但并不会影响滑动速度；用户希望越快速滑动时^①，离手（clutch）^②的时间会越短；向下滑动时离手的时间比较短。

检测方法与解决方法 相关研究也涉及了响应延迟的检测方法，比如，使用机械臂模拟人进行操作，使用摄像头拍摄手机屏幕^[16,23]；又如，使用震动传感器检测触摸行为，光电二极管检测屏幕亮度变化^[24]。此外，无论是学界还是工业界也都在尝试对这一问题进行解决，比如 [16,25] 都提出了一些通过设计解决触摸交互中低延迟问题的方法，[26] 也提出了 AR 与 VR 中延迟的检测和解决方法。

① 实验将任务要求按照速度分为“浏览速度”，“比较快速”，“尽可能的快速”三个级别

② 指一次滑动完成后，手离开屏幕，移动到初始位置从而开始第二次滑动的时间间隔。

总的来说，近年来，智能设备用户体验的改善，特别是流畅度体验的改善已经逐渐受到业界^[8-9,17]和研究人员的重视。虽然有大量研究是关于流畅度体验的，但总体来讲，关于动画中卡顿的知识还是非常有限，且存在不足。首先，大多数已有工作评估的是均匀的延迟对用户感知的影响，对于随机、突发的复杂卡顿的用户感知还有待探索。第二，现有的研究采用的指标基本上只关于时间，但人因方面的已有研究让我们看到，可能影响用户体验的因素非常之多，包括但不限于不同卡顿模式、不同任务类型、用户处在不同交互状态等。因此，我们需要综合考虑更多的因素来探究人对卡顿的感知模式。第三，现有关于感知的研究大多只为了获得一些阈值，比如可以察觉到延迟的时间阈值或可以忍受的延迟的时间阈值，而对具体的感知模式了解很少。

这三点不足也导致了目前提高流畅度的主要措施是提高整体性能，但实际上，对卡顿模式的详细了解可能带来更高效的提升流畅度方式。比如，目前动画设计领域已经有一些方法试图减弱卡顿对人的影响，但一方面这些方法很少公开细节，我们也并不知道方法究竟是基于严谨的理论，还是只是设计师的天才创意；另一方面，我们对这些方法的效果并不知晓。因此，为了高效地提高交互流畅度，仍有很多值得探索的工作。

当然，现有工作的发现为我们提供了重要的参考，以检验我们的研究成果；此外，现有的人因方面的研究也给了我们一些启发；现有工作中激活动效和动效检测的方法也为本研究提供了极有价值的参考。

1.3 本文研究内容

我们结合研究的目的，参考现有研究的经验，关注现有研究的不足，最终确定了详细的研究方案。本研究的目的是了解用户对手机动效中卡顿的感知模式，其中涉及的卡顿包括“帧延迟”和“帧丢失”两类。我们选取三种常见的手机交互动画作为切入点，建立基于用户感知的卡顿评价系统。这三种交互动画为：上下滑动（比如翻看一个列表）、左右滑动（比如在桌面翻页）、缩放动画（比如打开一个App）。该卡顿评价系统主要有三个环节：动画提取、卡顿识别、感知模型，如图1.1所示。

动画提取部分利用计算机视觉技术，从录制的视频中提取出动画曲线。然后，以此为基础，对一些常见的手机App中出现的动画进行分析与探索，找到常见的卡顿模式，从而辅助卡顿识别环节以及用户实验环节。比如，用户实验中

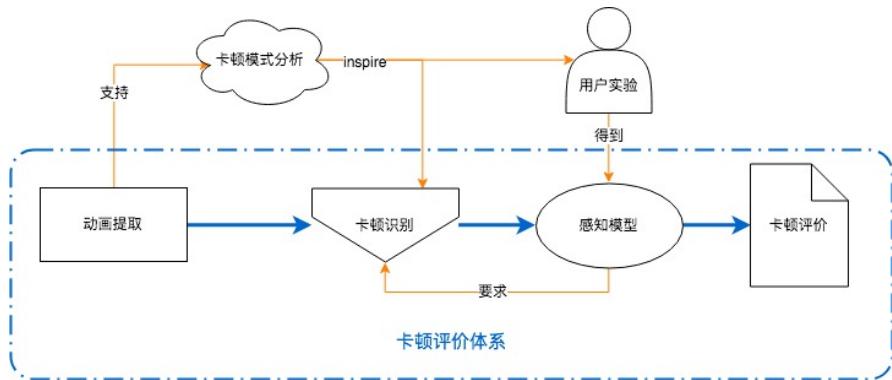


图 1.1 卡顿评价系统的流程图

以何种频率何种方式人为插入卡顿需要参考实际发生的卡顿的特征。卡顿识别部分的工作为从动画曲线中找到卡顿，并依据卡顿模型的要求检测每个卡顿对应的参数。感知模型部分则首先依赖于用户实验收集到的数据建立模型。这部分为了避免传统实验室研究样本量小的局限，采用在线用户实验的方式扩大实验规模。实验通过 Android App 的方式采集用户在不同使用情景下和不同类型的交互动画中，对不同卡顿的感知和评价，然后通过统计分析和模型训练形成通用、可解释的卡顿感知模型。在实际卡顿评价体系中，模型以卡顿检测的输出作为输入，从而得到卡顿评价报表，其中包含用户群体对卡顿的评价的分布等。

本文的内容组织如下：

- 第二章介绍动画提取、卡顿识别部分，以及卡顿模式分析。我们首先介绍视频获取方式以及这样选择的依据，然后介绍动画提取的两类算法、难点以及最终效果，再介绍常见卡顿分析的发现，最后介绍卡顿提取的方法。
- 第三章首先介绍实验探究的因素，然后介绍实验平台的设计细节以及技术实现方式，最后介绍两轮实验的参数配置以及实施情况。
- 第四章首先介绍统计学相关的背景知识，然后介绍数据预处理方法以及初步的数据观察，再通过统计学的方法进行单因子分析，汇报各个因素的影响并排除完全不相关的因素。最后，我们利用机器学习的工具进行多因素分析，得到感知模型，汇报模型的效果，并对模型参数进行分析与检验。
- 第五章总结全文的工作，分析工作的不足，并展望未来的工作内容和方向。

第 2 章 卡顿检测

2.1 视频获取

实验中，我们首先自然地考虑用录屏软件来获得屏幕内容。但是我们发现，现有的录屏软件（如安卓录屏大师）和我们自己开发的录屏工具（读取显示内容的缓存）都无法避免在录制的过程中随系统一同卡顿。这会导致丢帧的现象，并且软件无法获知自己是否丢帧，从而导致录制视频的时长小于实际时长。也正是因此，我们平时录屏时得到的视频帧率不是 60 fps (frames per second)，而是 40 fps ~ 60 fps 不等。此外，手机录屏也会占用 CPU 资源，对手机性能有影响。综合以上的考虑，以及相关研究的做法，我们决定采用外部摄像机拍摄的方式来获取视频。

2.1.1 高速成像系统

由于一般手机显示的理想刷新频率为 60 fps 左右，因此，为了能够捕捉超过 1 帧的卡顿，用于拍摄的摄像机应有至少超过 60 fps 的拍摄帧率。同时，为了方便数据处理，帧率最好为 60 的整倍数，如 120 fps 或 180 fps。本实验的高速成像系统采用 FLIR BlackFly S 摄像头。拍摄时选择 180 fps 的拍摄速度、 1600×900

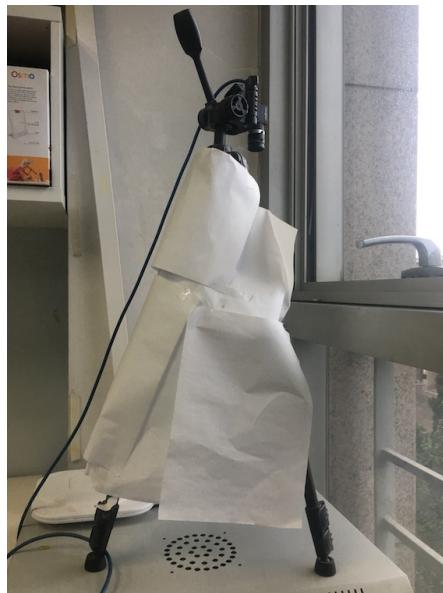


图 2.1 搭载 FLIR BlackFly S 摄像头的成像系统

的分辨率，拍摄结果以 H.264 压缩保存，比特率为 15 Mbps。拍摄时，摄像机被固定在三脚架上，拍摄其正下方的区域。手机平放在三脚架中间，周围用纸遮挡以减少手机屏幕反射画面带来的干扰。成像系统如图2.1所示。

2.1.2 动画任务的自动生成

拍摄时，如果直接用手在屏幕上操作，手的遮挡会增大视频分析的难度，影响动画提取的效果。因此我们尝试了如下三种生成交互的方式：

1. 机械臂模拟交互操作：有研究^[23]采用用机械臂来模拟交互操作，因此我们也尝试了用机械臂直接在手机屏幕上进行点击、滑动等操作。然而，我们发现机械臂的模拟有效果较差、不自然、不稳定、价格昂贵等多种缺点，因此这种方式没有被最终采纳。
2. 程序自动模拟交互操作：这一方法采用 Android SDK 中 UIAutomation 提供的 API 实现，UIAutomation 提供的主要操作如表2.1所示。我们使用了其中的点击手势和移动手势，并将相关信息记录在日志中。这一方法的缺点是：模拟的交互不像用户实际的操作的那么自然，有一定区别，特别是在翻看列表这样需要反复多次操作的情况下。
3. 程序先记录再复现用户交互操作：这一方式首先通过我们开发的 Android App 记录用户的交互行为，即通过 Android SDK 的 onTouchEvent 监听用户手势事件，根据设备界面响应，约每 16.6 ms 记录用户手指所在坐标 ($point_x, point_y$)，然后通过 UIAutomation 提供的 API 复现交互过程。

综合比较，程序先记录再复现用户交互操作这一方式效果最符合我们的需求。因此，使用评价体系时，首先需要使用我们的 Android App 记录一段用户操作，然后打开需要测试的 App，将待测试手机放到摄像头下，通过脚本自动开始复现交互操作。此外，这种方法的可复现性使得它也可以使用同一个交互操作测试比较不同设备或 App 的流畅性。

2.2 动画提取

获得视频后，我们首先需要从视频中提取交互动画的曲线。对于平移类运动，我们最终采用的算法是基于特征提取与匹配的；对于缩放类运动，我们采用的算法则是基于图像差异的。动画曲线提取的难点主要在于：1) 手机内容更新需要一定的时间，因此在高速摄像机拍摄的内容中存在两帧内容渐变重叠的现

表 2.1 UIAutomation 提供的主要操作

任务	方法
打开指定 App 界面	运行 <code>uiautomatorviewer.bat</code> 获取界面元素信息，使用 <code>InstrumentationRegistry</code> 启动指定的 App 界面，调用 <code>UIObject.UiSelector</code> 寻找界面元素进行确认。
点击手势	调用 <code>UIDevice.click(point x, point y)</code> 短点击指定坐标。
缩放手势	调用 <code>UISelector</code> 寻找指定界面布局，调用 <code>UIObject.pinchOut() / pinchIn()</code> 模拟双手指手势缩放屏幕。
图标拖拽	调用 <code>UISelector()</code> 寻找指定图标或其他元素，调用 <code>UIObject.dragTo()</code> 将指定元素拖拽到指定坐标
左右滑动	调用 <code>UIAutomation</code> 自带的 <code>swipe</code> 方法。
触摸移动	调用底层的 <code>MotionEvent</code> 方法，建立 <code>touchMove</code> 、 <code>touchdown</code> 、 <code>touchUp</code> 方法并自定义 <code>swipe</code> 方法。与 <code>UIAutomation</code> 自带的 <code>swipe</code> 方法相比具有以下优势： <ul style="list-style-type: none">• 跳过设备卡顿造成的被无视的轨迹点；• 实现不受设备卡顿影响的且更加符合设备帧率的用户滑动；• 能记录坐标所对应的当前手势。

象，这对特征提取的算法造成了一些干扰；2) 手机页面刷新是逐行进行的，因此拍到的画面中可能有一半是前一帧图像，有一半是后一帧图像，这对特征匹配算法和卡顿提取算法都造成了一定干扰。

2.2.1 相关知识

在具体介绍动画曲线提取算法之前，本节首先对其涉及的计算机视觉与图像处理相关知识进行简要介绍。

SIFT (Scale Invariant Feature Transform，即尺度不变特征) 算法^[27] 常被应用于图像匹配，即找到两张图片里相似的部分并将其匹配。算法假设这两幅图中相似的部分主要有旋转、缩放、尺度、亮度或对比度等方面的变化。算法的主要思路是首先找到图像上的特征点，再将不同图像上的特征点进行匹配。SIFT 算法主要包括以下步骤：

1. 保证尺度不变性的前提下，计算图像的局部极值；
2. 对特征点进行过滤，即删除不稳定的特征点；
3. 对每一个特征点计算其特征描述符，然后求得其方向；
4. 用上述特征描述符在图像中寻找匹配点；
5. 求最终整体变换的参数。

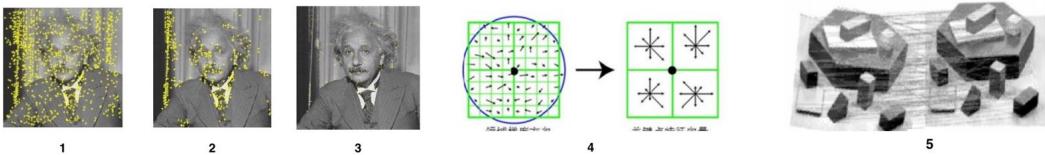


图 2.2 SIFT 算法各步骤的效果示例

各个步骤的效果示例如图2.2所示。

SURF (Speeded Up Robust Features) 算法^[28]与 SIFT 算法原理相似，也是一种图像匹配算法，只不过其中使用的运算比较简单（如 Harr 特征），因此运算效率较高。OpenCV 实现的 SURF 算法甚至可以达到实时运算的速度。SURF 算法与 SIFT 算法的一个不同之处在于尺度空间的不同。二者均通过金字塔变换方法实现不同尺度的特征提取，但 SIFT 算法按照高斯金字塔的截面积变化趋势来缩放图像，同时保持图像过滤器的尺寸不变，从而获得不同大小图像的特征点；而 SURF 算法保持图像大小不变，按照同一比例改变过滤器的大小。

SURF 算法与 SIFT 算法的有效性也符合人类学知识。人类对物体的视觉描述也是局部化的，因此，基于局部不变特征的图像识别方法与人类自身的视觉机理非常相似。通过局部化特征的组合，获取对图像的整体印象，这为二者方法提供了生物学上的解释。

RANSAC (Random Sample Consensus, 即随机抽样一致性) 算法^[29]是从包含异常数据的样本数据集中获取有效样本数据集的算法。算法的大致思路为每次随机从数据样本中选择一些数据，进行检验，如果符合要求则结束算法，否则反复迭代。

2.2.2 平移动画

平移动画提取算法的基本思路是利用计算机视觉技术在图像中找到特征点，在不同帧之间进行匹配，从而得到图像的位置变化信息。实现上采用了 OpenCV 中的 SURF 算法实现。算法要求画面上有大面积、特征明显的内容，其主要流程如下：

1. 特征点提取及匹配：提取前后帧画面的 SURF 特征点，穷举匹配两帧中的特征点，记录符合比值审敛法^①条件的点对，以便进行进一步处理。
2. 平移距离的计算：计算上一阶段提取的每一对特征点的平移向量及其长度，统计移动方向上分量的大小，并取其中位数作为平移距离的估计。

^① 即 ratio test，是判别级数收敛性的一种方法，又称为达朗贝尔判别法。



图 2.3 平移动画提取结果

算法结果举例如图2.3所示。图中为 iPhone 手机桌面的平移动画。左侧窗口为当前帧的画面，绿色框中是程序识别出正在发生运动的部分。右侧窗口是前后两帧中匹配的特征点对。算法最终结果精确到 ± 1 像素，并在平移类动画的曲线提取上表现良好，正确率可以达到 90% 以上。此外，经过简单的参数调整，也可以排除画面上某些区域的内容，用于检测复杂动画中局部的变化，或检测用户真实交互时（比如有手指挡住部分屏幕）的动画。此外，我们还使用 GPU 加速，使得处理速度达到每分钟 50 帧。

2.2.3 缩放动画

我们最初尝试使用 SIFT 算法对缩放动画进行提取。实现上采用 OpenCV 提供的 `findHomography` 函数，该函数通过 RANSAC 算法筛选图中找到的 SIFT 特征，并进行匹配，从而依据匹配点的位置求得图像放大的比例。但结果上述算法在图片较小的时候会出现不准确的情况，因此我们尝试了另一种更容易理解的算法。这种算法主要利用帧与帧之间的差值进行计算。算法分为两个阶段，其流程如图2.4所示，具体如下：

1. 采样与校准：

- 采样：从视频的前 1 秒中采样 40 张图，使用前 20 张图的平均值作为彩色版背景图，然后对彩色版背景图进行基本处理，得到最终背景图。基本处理包含两个步骤，第一步将图片转为灰度图，第二步对图片进行模糊处理。这里为了更好的保留边缘信息，模糊处理选择边缘模糊算法。

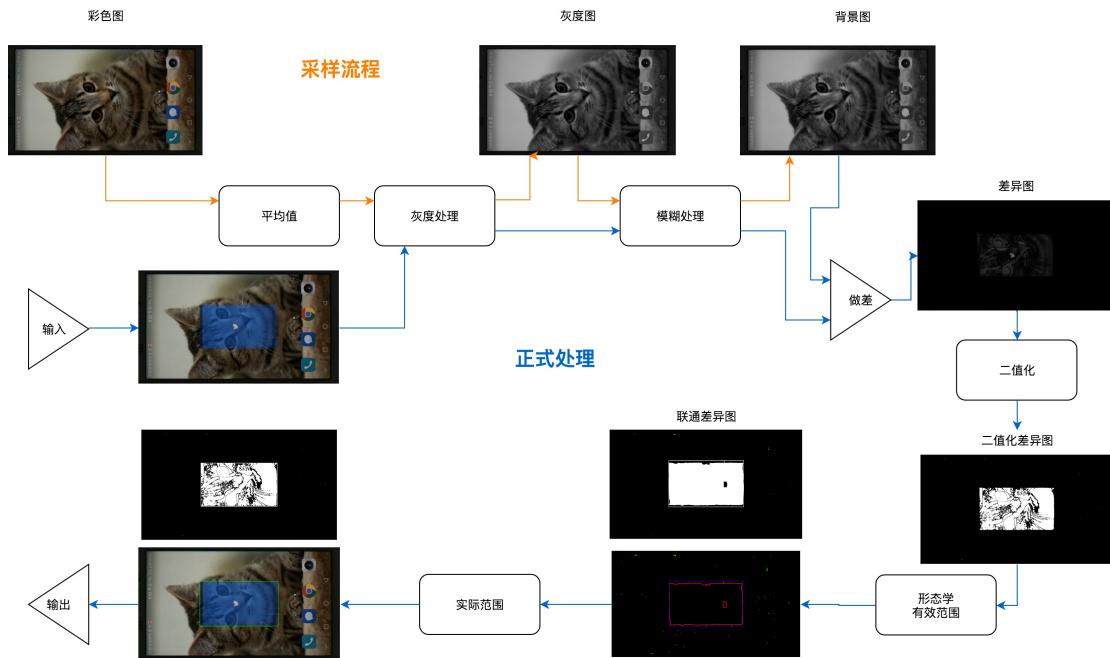


图 2.4 缩放动画提取算法流程图

(b) 校准：这里主要利用 20 张背景图校准差异阈值。在正式处理中，我们首先会对每一张图进行基本处理，然后算出该帧图像与背景图的差值图，再根据差异阈值将差值图进行二元化处理以方便后续操作。这一步中，我们对上述阈值进行校准。校准方法为首先初始化该阈值，然后按照正式的流程求出二元化差异图。接下来计算差异占比，如果差异占比大于程序设定的阈值（实际算法中为 0.05%），则增大阈值再次尝试，否则认为该差异阈值符合要求。

2. 正式处理：

- (a) 基本处理与初步筛选：首先对每一张图进行基本处理，然后算出该帧图像与背景图的差值图，再根据差异阈值将差值图进行二元化处理得到二元化差异图。接下来计算差异占比，如果差异占比小于程序设定的阈值（实际算法中为 0.05%），则忽略这一帧，否则进入下一步。
- (b) 获得有效范围：这一步的假设是，图中新出现的部分集中在同一区域。因此这一步的主要功能是连通相邻的部分，并找到面积较大的区域共同构成有效范围。具体算法是：首先对上一步得到的二元化差异图进行形态学变换，变换方式采用 OpenCV 提供的闭变换方式，即先膨胀再腐蚀，得到连通的差异图；然后从连通差异图中找到面积比较大的

部分，画出恰好包含这些部分的方框作为有效范围。

- (c) 获得实际范围：有效范围虽然包含了有变化的区域，但动画曲线的提取需要比较精确的边缘位置。因此我们依据有效范围框，从步骤 2 得到的二元化差异图中再次筛选，选出在框内且面积大于某一阈值的连通部分作为有效部分，再选取有效部分的包含方框作为实际范围。

2.3 常见卡顿分析

利用动画曲线提取程序，我们首先进行了常见卡顿分析。我们使用华为 Mate 9 和 iPhone 两款手机在国内外多个常用 App 上进行了测试和探索。发现卡顿主要有跳帧和延迟两种类型，iPhone 的卡顿以跳帧的形式居多，Android 以延时的形式居多。此外，卡顿容易发生在加载图片的位置。有些 App 中卡顿会改变动画曲线的走势，或者引发速度短时间内的突变，我们认为这是不同 App 对卡顿不同的处理和优化，其中包括前置补偿、后置补偿、减速、匀速等策略。下面以几个例子介绍我们在各个常用 App 的处理优化方式中的发现。

微博

摄像机拍到微博的画面与位移变化图如图2.5所示。微博热门页的页面滑动中，一旦在加速阶段出现卡顿立即进入减速阶段，我们称之为减速策略。也或许因此加速阶段相比其他 App 比较短。另外，有多处突然增大的位移和短位移依次出现，特点为有视频和图片加载处，目前认为这一现象由加载导致。大位移加小位移的组合我们称之为前置补偿策略，即卡顿时先进行大位移，得到运算结果后再进行小位移补齐。

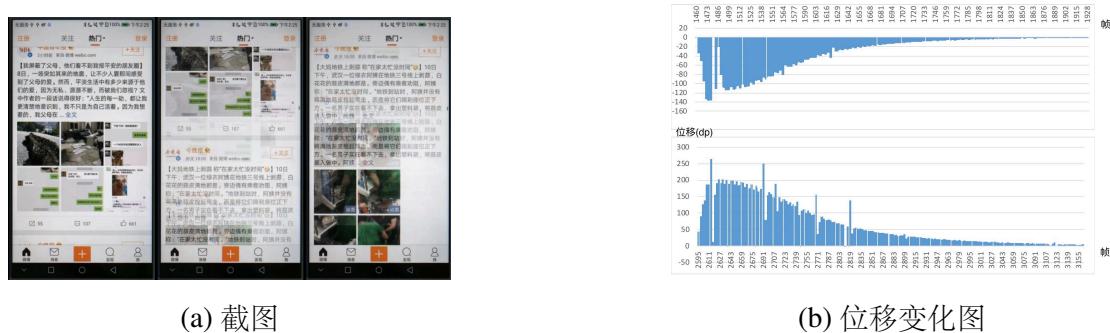


图 2.5 微博的动画模式

大众点评

摄像机拍到大众点评的画面与位移变化图如图2.6所示。大众点评页面滑动在加速阶段容易出现卡顿或位移不稳定的现象。卡顿后继续加速，因此加速阶段显得比较长。几处大位移发生在小位移之后，我们称之为后置补偿，即知道卡顿发生时先进行小位移，得到运算结果后再进行大位移。这一段交互在视觉上有强烈的画面残留现象，体验较差。

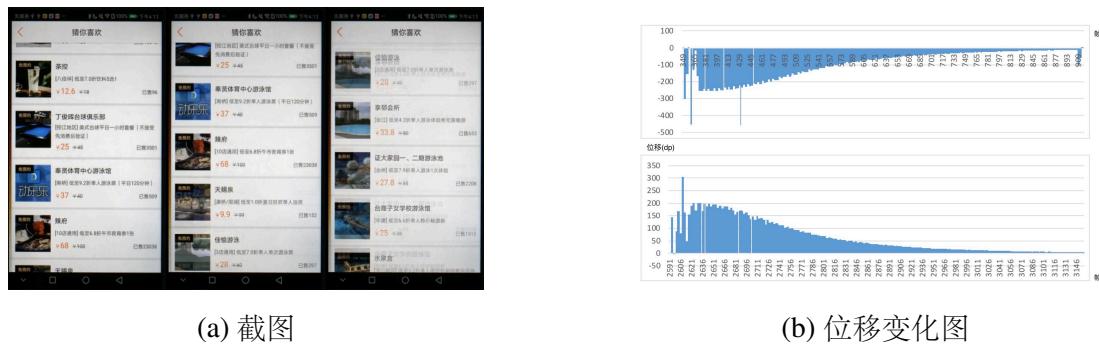


图 2.6 大众点评的动画模式

今日头条

摄像机拍到今日头条的画面与位移变化图如图2.7所示。数据显示，在加速期间出现了卡顿。结合视频观察发现，动画中出现了形似卡顿，但实际上有小位移填补，让画面趋于完整的现象，我们称之为低速策略。卡顿后并未延续之前的速度，而是有从较慢的速度重新迅速加速的现象。我们称之为重新加速策略。Youtube也使用了这一策略。

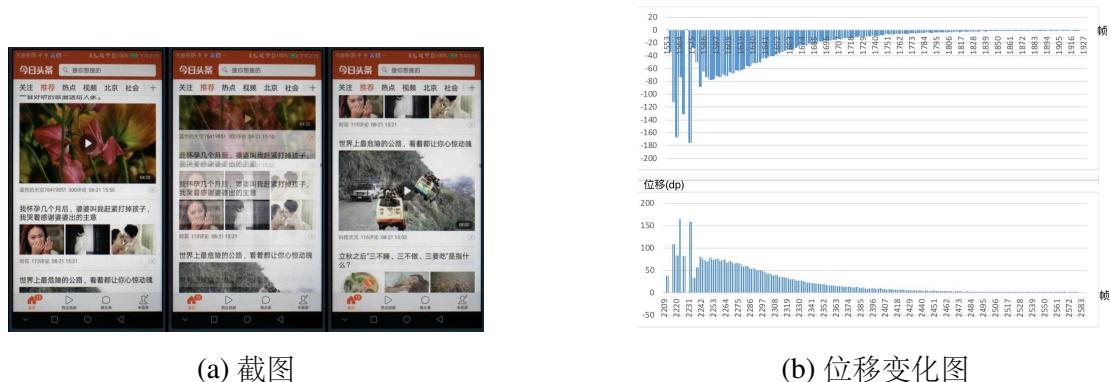


图 2.7 今日头条的动画模式

CNN

摄像机拍到 CNN 的画面与位移变化图如图2.8所示。和微博一样，CNN 的动画曲线在加速时遇到卡顿会立即进入减速阶段。此外，快速滑动时图片显示为灰白色，此时整体滑动比较平滑。这是其他 App 中比较罕见的。整体上看，CNN 的动画曲线平稳，体验较好，可能有被策略掩盖的动画设计。

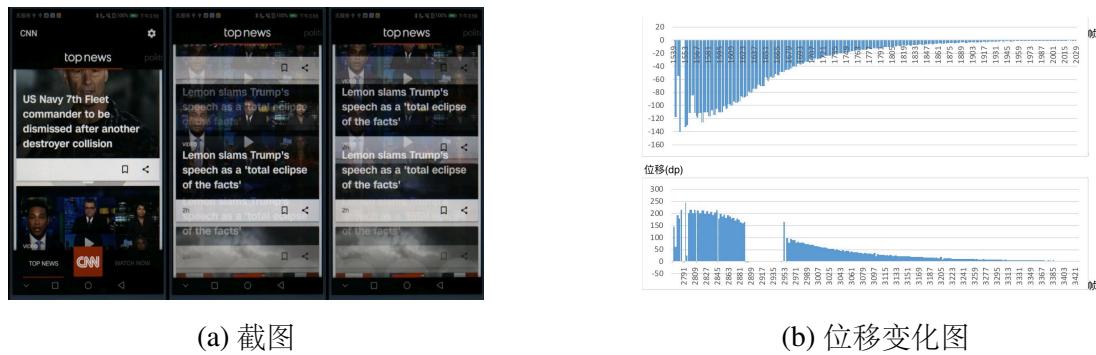


图 2.8 CNN 上下滑动的动画模式

在 CNN 的左右滑动中发现了一个特有的模型，如图2.9所示。在减速阶段有一帧的停顿，多次尝试后，我们认为是专门设计的动画效果。在视觉上观察不到这一瞬间的停顿，是因为停顿位置和手指抬起相关，这一设计带来的感受是滑动时有强烈的跟手感，更有力度。

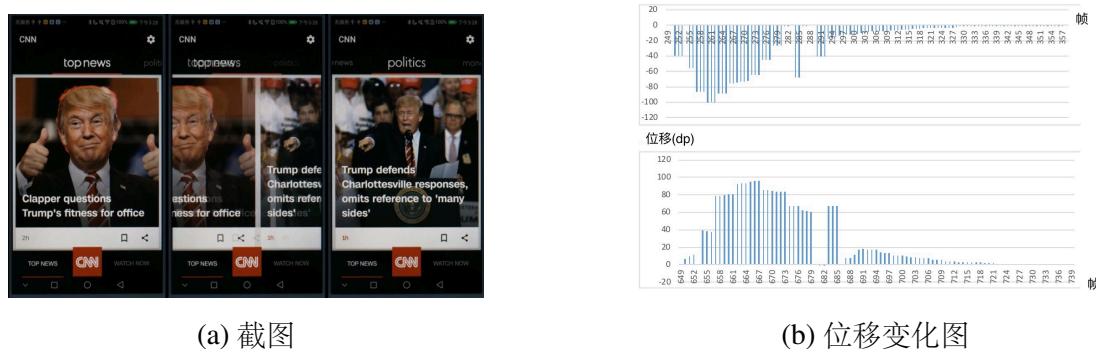


图 2.9 CNN 左右滑动的动画模式

Instagram

摄像机拍到 Instagram 的界面与位移变化图如图2.10所示。我们在 Instagram 中极少观察到卡顿。Instagrgam 的动画曲线的一个特点是，当滑动速度约为 110

或 335 像素点每帧时，会触发匀速运动。

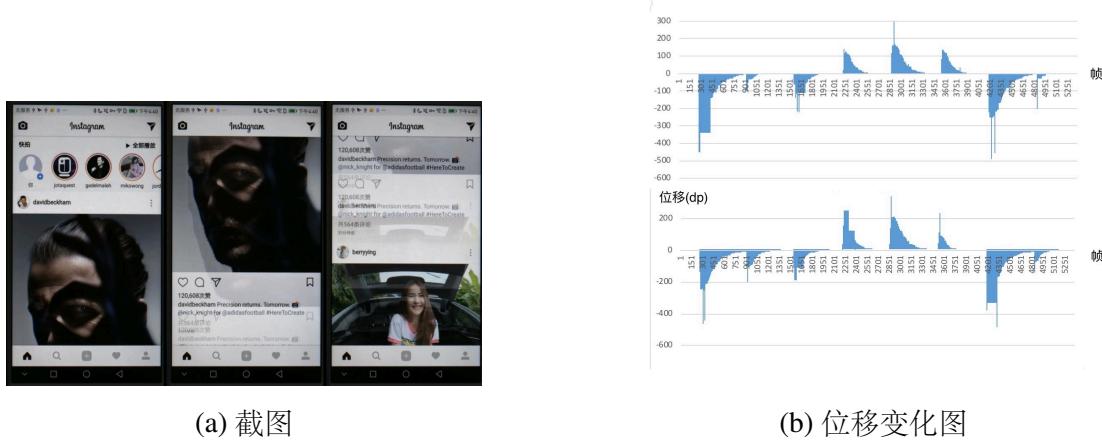


图 2.10 Instagram 的动画模式

2.4 卡顿检测

卡顿检测的目的是对第2.3节动画曲线提取的结果进行处理，找到其中的卡顿，并且结合第2.1节保存的交互记录分析得到每个卡顿的属性。

2.4.1 预处理

卡顿检测前，首先需要对所需动画曲线以及交互记录进行预处理。这一步骤的主要目的是将相应信息进行匹配，并转换到手机的帧空间上方便分析。动画曲线中的帧被称为视频帧。我们将视频帧与手机帧匹配的流程分为两步，下面分别进行介绍。

第一步为帧对齐。由于视频帧率高于手机刷新帧率，我们首先需要将视频帧与手机帧对齐。假设视频帧率为手机帧率（通常为 60 fps）的 k 倍，那么视频中的连续 k 帧就会对应一个手机帧。帧对齐的目的就是找到这连续 k 帧的开始。具体方法是，首先对所有视频帧从 0 开始编号，接下来对视频的前若干帧^①进行对齐检验，即判断从当前帧开始的连续 k 帧是否属于同一帧。设当前为第 i 帧，如果检验成功，则认为第 i 帧是一个手机帧的开始，同时第 $i \bmod k$ 帧也是一个手机帧的开始，因此给 $i \bmod k$ 进行一次标记。完成所有的判定后，我们将有标

^① 实现中，我们对视频的前 48 帧进行检验。

记次数最多的数字视为实际的手机帧的开始。对齐性检验的具体方法是，判断这连续 k 帧中每两个帧之间的位移是否都小于设定的阈值。

第二步为帧验证。计算平移类动画的位移时，首先需要对数据进行初步筛选。平移类动画检测算法的输出中记录了提取到的特征中最显著的两个，我们在本段以纵向位移 y 为例进行说明。假设这两种特征对应的点对数分别为 c_1 与 c_2 ，对应的位移量分别为 y_1 与 y_2 。我们按照以下方法验证第一步中对齐的结果是否可信：

1. 考虑手机帧对应的连续 k 个视频帧的数据，如果 y_1 或 y_2 小于某一阈值，则将其设为 0。
2. 计算连续 k 个视频帧中每一帧 c_2/c_1 的值，若此值小于设定的阈值，则认为此视频帧为正常帧；若此值大于等于设定的阈值，则认为此视频帧是非正常帧。若连续 k 帧中非正常帧超过一半，则认为这一手机帧不可信。
3. 对于一个可信的手机帧，如果位移为 0 的视频帧数超过一半，则认为位移为 0，否则认为位移为非零位移的均值。

2.4.2 卡顿检测

接下来便是从动效曲线中找到卡顿，并提取相关属性。提取哪些属性依赖于第四章感知模型的要求。对每一个卡顿需要提取以下五个属性：卡顿类型、卡顿时长、卡顿时速度、距后一个卡顿时间、卡顿时状态。然后从中选出卡顿时长最长的 6 个卡顿（但必须包括第一个卡顿），记录其原始次序作为第六个属性，再按照卡顿时长进行重新排序，并依据新的次序将各个卡顿的六个属性合并成为一个向量。此外，还需要依据次序为第一个的卡顿推断这一段记录是否为响应延迟类型，作为另一个属性，与前一步算出的向量合并，作为感知模型的输入。

以上属性中，响应延迟与卡顿时状态结合交互记录简单比对即可获得，其余的属性需要通过动画曲线的分析得到。以下为其中几个比较复杂的属性的提取思路介绍：

- **卡顿类型：**对于一次卡顿（即位移为 0 的手机帧区域），将卡顿段视为 1 帧。我们向前寻找两帧正常帧，向后寻找两帧正常帧（两帧不要求相邻）。首先计算当前帧位移与前一帧位移之比 t_1 ，以及后一帧位移与当前帧位移之比 t_2 。若 t_1/t_2 在我们设定的区间范围之内，这一帧为跳帧；否则此帧为普通延迟帧。若这一帧的匹配信息缺失，则视为跳帧。

- 卡顿时速度：卡顿前一个正常帧的速度。对于上下滑动与左右滑动，以 mm/ms（毫米每毫秒）为单位；对于点击打开动画，则以每 ms 变化的百分比为单位。
- 距后一个卡顿时间：计算两帧的时间之差。注意，对于最后一帧而言，此属性为运动停止的时间与帧时间之差。若视频结束时运动仍未停止，则依据最后一帧的速度估计停止时间。

第3章 卡顿感知用户实验

3.1 实验介绍

本实验的目的是收集用户在不同动效中对不同卡顿组合的感知与评价。收集到的数据会被在下一阶段用来分析得到卡顿感知模型。

实验主要研究三种常见的动画类型：上下滑动、左右滑动、点击放大。上下滑动动画常见于页面和列表的浏览，如阅读新闻、网页、朋友圈、微博等；左右滑动动画常见于标签页之间的切换、应用之间的切换或多页桌面之间的切换；点击放大动画常见于打开一个手机应用或打开某个小页面。实验中选取的三个典型场景分别为：阅读朋友圈（对应上下滑动），桌面页的切换（对应左右滑动），打开微信（对应点击放大）。

由于实验涉及的因素较多（卡顿时长、卡顿类型、卡顿时用户交互状态、卡顿的场景、用户握持手势等），用户实验预计招募被试 2000 人，每人的实验时长预计为 15 ~ 20 分钟。这样规模的实验难以在实验室环境下完成，因此，我们将用户实验的手机 App 发布到网络，在线分发实验配置和收集数据。

实验的主体由任务组成，每个任务中，我们通过 Android App 模拟三种交互情景之一，在进入情景前引导用户进行要求的操作，在用户交互的过程中依据实验配置加入卡顿，在用户完成交互任务后要求用户对其体验与卡顿感知进行评价。

实验记录的所有变量大致分为四类，如表3.1所示。此外，实验也记录了整个交互流程。

表 3.1 实验中收集的所有信息

类别	信息			
用户属性	年龄	性别	重做次数	手持姿势
设备属性	屏幕尺寸	设备型号	安卓版本	
实验可控	动画类型	卡顿类型	卡顿分布方式	卡顿次数
	卡顿时长	是否为响应延迟	运动方向	任务速度
实验不可控	卡顿时速度	卡顿时状态		



图 3.1 实验中的评分界面

3.2 实验设计

3.2.1 设计细节

由于实验通过网络进行，我们无法监督和引导用户，故用户很可能不按照我们的预期进行交互。而实验本身研究的就是交互，因此数据质量可能会受到比较严重的影响。所以，实验的设计非常重要，并且具有一定挑战。我们进行了反复的迭代，以保证实验平台的鲁棒性和易用性，从而引导用户按照期望进行操作，尽可能地提高数据质量。本节将介绍实验的部分设计细节。

用户评价 评价部分设计了两个问题分别询问用户对卡顿的感知以及对任务的整体体验，详细内容如图3.1所示。

由于用户的评价是我们收集的最重要的信息，我们反复迭代了评价页面的设计，以减少不认真答题的情况。问题一为单选题，用来在分析数据时找到感知的临界值，与已有研究进行比较。如果问题一选择没有感受到卡顿则无法回答子问题，这样可以减少数据出现难以解释的冲突。问题二的答案是简单的好与差，因此我们选取了星型评分条，在两段添加了固定的说明文字，从而减少需要阅读的文字，使得评价比较直观。

值得一提的是，问题一的子问题虽然和问题二一样都是对程度的评价，但经过尝试和讨论后发现问题一子问题中的程度比较容易产生不同理解，因此需要比较复杂的描述。这样一来，如果仍用星形条表示，就只能在选择某一级别时才看得到对应的内容，体验较差，并且文字容易被忽视。所以我们最终没有使用



图 3.2 实验中的确认事项

星型评分条，而是直接描述了三个相对明确且易于区分的量度，以选择题的形式呈现，以督促用户阅读每一个程度对应的文字。

之所以设计多程度的评分，而不是如已有研究一样采用 JND (Just-Noticeable Difference) 原则设计实验，原因有二：1) 本实验并不只是想找到可分辨的卡顿的临界条件，而是想了解用户对卡顿感知的整体规律，但是依照 JND 方法收集的数据无法用于了解用户对各种情况下的卡顿的感知；2) 用户间存在个体差异，所以本实验更期望得到用户评价的分布而不是一个临界点，从而需要更丰富的评价，而不只是对比有无卡顿。

确认事项 有研究^[30] 表明，用户在浏览网页时，即便是总文字量小于 100 词，阅读的内容也不会超过 50%；而当文字总量上升时，这一比例快速下降^①。因此，对于移动端，为了保证用户能够完全获取重要的信息，我们应当以这一研究的结果为下界。由于我们的实验说明有 300 多字，我们必须采用其他方式帮助用户获取重要信息。否则，如果用户没有阅读须知就直接开始实验，不仅可能产生困扰，而且会影响实验的结果。我们将几件重要的事情单独写成确认项，如图3.2所示，要求用户逐条点击确认。其中，确认联网是为了以避免后面获取实验配置时用户的等待；确认清理后台程序是为了提高手机自身流畅度。

固定手持姿势 考虑到不同的手持姿势可能影响用户的操作速度和流畅度感知^[23]，我们在实验开始前让用户选择一种习惯的手势，并在之后的每一次任务中提醒用户使用该手持姿势。

任务说明设计 由于任务周期比较短，因此任务说明页面的阅读时间也不会很长，这导致用户能够获取的信息量很小。我们需要尽力减少用户的阅读负担，使

^① 当一页平均文字内容为 593 个英文单词时，用户即便所有的时间都在阅读文字，也只会读其中 20% 的内容。即便是对于文字量小于 100 词的网页，阅读内容所占的比例也不超过 50%。



图 3.3 实验中的任务说明

用户快速获取重要信息，避免任务中的错误操作带来的负面感受，从而保证用户的评价主要原因为卡顿，而非其它。于是，我们将任务描述中的关键信息标红，如图3.3。此外，对较为复杂的交互任务，我们制作了操作示意动画^①放在任务说明页，方便用户快速理解，如图3.3下部所示。这样做很重要的原因之一是：已有研究做类似人因实验的时候发现，当我们指引用户向下滑动列表，用户既有可能手指向下滑动，也有可能手指向上滑动使得列表向下滑动。

任务中的提示 用户需要评价卡顿程度，故我们需要通过设计来帮助尽量减少任务中的信息提示带来的被打断的不流畅体验。我们将提示信息分成强制型和非强制型。强制型提示如图3.4(a)，一旦出现，任务会中断，用户必须重新完成；非强制型提示如图3.4(b)，主要用于提示用户任务是否完成。非强制型提示足够明显，能够起到提示作用，但只有在任务完成一段时间后用户仍然试图操作的情况下才出现。这样可以避免由于该提示出现的时间与任务中的卡顿时间间隔太短，从而干扰用户对卡顿的感知和判断。

流程控制 有研究^[30]表明，返回键是用户最常点击的按钮第三名，并且被业界称为“用户的生命线”。因此，我们需要避免用户由于点击返回键所引起的问题，但同时也要支持返回按钮以提升用户体验。如果用户体验不佳，则可能导致用户不认真完成任务，从而带来数据质量问题。最终程序规定，用户无法通过系统级的返回操作回到之前的页面，取而代之，点击返回键会触发提示框以提供需要的信息；处于评分阶段，尚未进入下一个任务时，用户可以返回并重做任务；一旦进入下一个任务，便无法再重做上一个任务，只能重新开始实验或者对上一个任务的反馈进行修改。

^① 用户研究表明，举例对于用户理解指引有巨大的帮助^[31]。



图 3.4 实验中的任务提示



图 3.5 实验流程图

朋友圈列表长度 由于实验要求用户用三种阅读速度进行操作，因此在不同的速度要求下列表的长度不同。这样做的目的有二，一是为了防止用户滑动速度过快导致列表在滑动中触底，从而影响用户对卡顿的判断；二是为了防止用户滑动速度较慢时，没有注意到任务的结束，而阅读比较久，导致卡顿带来的负面体验被“稀释”或“加强”。举个例子，如果一开始感受到了严重的卡顿，但是之后一段时间内的动画流畅，那么用户对之前卡顿的评价可能会偏高。

3.2.2 最终实验流程

最终确定的实验流程如图3.5所示。

正式实验开始后用户首先进入第一种动画类型的练习模式。练习模式的界面和操作与正式任务完全相同，但不插入任何卡顿。这一方面是为了让用户熟悉界面与操作（比如滑动反向、速度不合适等错误可以在练习阶段就得到提示和修正）；另一方面是为了给用户一个基准印象，让用户对没有卡顿的动画有基本的认知，从而减少因为设备本身卡顿或用户个人习惯所导致的评价偏差。练习任务结束后进入正式任务，正式任务结束后进入下一种动画的练习模式，直到实验结束。结束后，用户通过界面填写对整体实验的反馈信息。

3.3 技术实现

3.3.1 实验环境

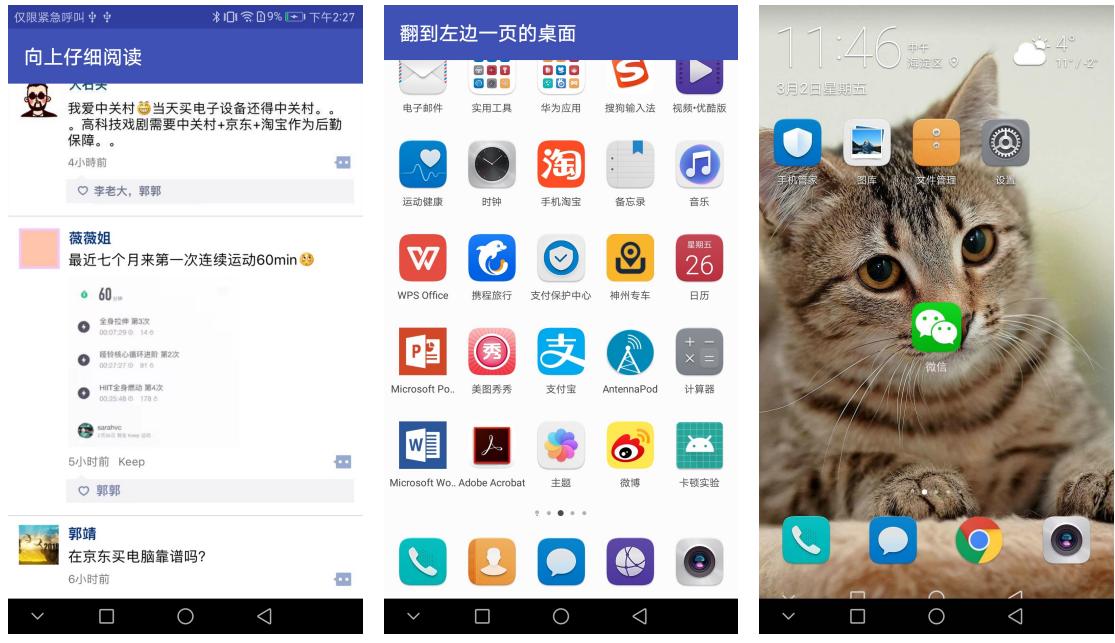
实验平台采用服务器-客户端模型。客户端可运行于系统为 Android 5.1 及以上版本的手机上。实验任务在服务器端按实验要求随机生成，下发给客户端。客户端按照服务器的指令，产生相应的卡顿效果，并采集用户主观评分。全部实验任务完成后，客户端自动上传实验数据至服务器。

3.3.2 卡顿生成方式

本段将分别描述三种动效中人工插入卡顿的实现方式。设计技术方案时，主要考虑支持延迟和跳帧两种不同的卡顿类型。

上下滑动 对于上下滑动的动画，实验模拟阅读微信朋友圈的情景。因此页面的主要构成是一个列表，采用 Android 提供的 `Listview` 实现。列表的每一行是一张图片，图片内容是一条常见的朋友圈样式，如图3.6(a)所示。列表采用 `ListAdapter` 方式填充内容。由于浏览列表过程中往往有多次滑动操作，且操作的频率与模式对于不同用户可能很不一样。比如，有的人习惯快速多次滑动，甚至双手接替滑动，而有的用户倾向于滑动一次等到动画速度变慢再滑动第二次。因此实验并未依据用户的操作来插入卡顿，而是在列表中的特定位置插入卡顿。

上下滑动的“帧延迟模拟”通过在 `ListAdapter` 中的 `getView` 函数中插入 `sleep(int lag_time)` 来实现，达到的效果是卡顿后列表恢复原来的速度，并从卡顿的位置开始继续按照系统设计的速度曲线运动；“帧丢失模拟”则通过在 `ListView` 的 `onScroll` 函数中插入 `sleep(int lag_time)` 来实现，达到的效果是列表整体的运动时间、位置和速度都不受影响，只有在卡顿期间画面不刷新。



(a) 上下滑动：微信朋友圈 (b) 左右滑动：桌面翻页 (c) 点击放大：打开微信

图 3.6 三种动画效果的模拟场景截图

左右滑动 对于左右滑动的动画，实验模拟桌面翻页的情景。页面的主要构成是 3 张桌面的截图，如图3.6(b)所示。页面采用 Android 提供的 Pager 实现，这样滑动的运动曲线保持 Android 系统级的设计。每次实验开始的时候将页面置于中间一页，也就是第二页，然后用户根据引导翻到左边一页或者右边一页。

根据 Android 的实现，翻页时只会在 ScreenSlidePagerAdapter 的 getItem 函数中进行一次资源的获取。为了在滑动过程中插入“帧延迟”，我们在每一页插入了一个不可见的列表，其长度超过一页，并在需要插入卡顿的地方修改列表停留的位置。该列表和上下滑动中的列表采用了相同的实现方式，这样当列表位置被修改时，需要获取之前不在页面内的条目，列表的 Adapter 的 getView 函数就会被调用。我们在其中插入 sleep(int lag_time) 即可达到延迟卡顿的效果。“帧丢失”的实现方式和上下滑动类似，也是在 Pager 提供的 onPageScrolled 函数中插入 sleep(int lag_time)，这样可以实现与上下滑动动画中相同的卡顿效果。

点击放大 对于点击放大的动画，实验模拟打开一个 App 的情景。页面的构成是一张桌面的截图和一个可点击的微信图标，如图3.6(c)所示。用户点击图标后，微信的开始页面会淡入并逐渐放大直到铺满屏幕。为了更真实地模拟，这一任务下顶部的任务引导和系统顶部的状态栏也会被隐去。打开应用的透明度和大

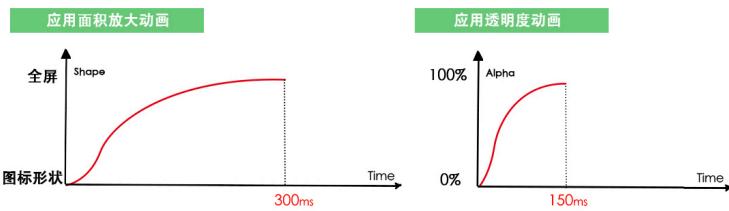


图 3.7 点击放大动画的曲线

小变化曲线采用特定参数的贝塞尔曲线，如图3.7所示。实现上，我们使用另一个线程每隔 16.6 ms 刷新一次图标的透明度和尺寸。

“帧延迟”的实现方式为直接在 Android 的 Thread 类的 run 函数中插入 sleep(int lag_time)。“帧丢失”则实现为：在 sleep(int lag_time) 的同时修改了记录当前帧数的变量 current_frame，在下一次更新时直接将页面设置为之后的状态。

三种动效实现后均进行了检验，以保证效果符合要求。此外，为了尽可能减少实验中非人工插入的卡顿，我们对程序进行了优化，以保证其占用的系统资源较少。

3.4 预实验

我们首先进行了小规模的预实验，参与实验的人数为 8 人，每人每种动画进行了 8 个任务，三种动画一共收集到 $3 \times 8 \times 8 = 192$ 条记录。我们根据数据分析确定了正式实验时的实验配置。以下为影响实验配置的几个发现，以及我们采取的对应措施：

1. 只有大约 25% 的用户可以感受到卡顿，因此我们依照数据提高了卡顿时间长度的范围。
2. 阅读朋友圈的任务整体评分高于另外两种动画，因此我们最终设定阅读朋友圈的任务的卡顿时间上限高于另外两种任务。
3. 在用户评价没有感受到卡顿的数据中，卡顿时长的均值为 2 帧左右，因此在一大多小的多个卡顿的配置中，我们将小卡顿时长的上限设置为 3 帧，以观察将大卡顿拆分出小卡顿。

表 3.2 实验配置规则

类型	因素	上下移动	左右移动	点击放大
适用各种类型	移动方向	上下各 50%	左右各 50%	-
	多个卡顿发生的时间范围	卡顿开始后 1000 ms 内随机	卡顿开始后 300 ms 内随机	卡顿开始后 300 ms 内随机
	无卡顿的比例	10%	10%	10%
	卡顿类型	跳帧和延迟各 50%		
普通单个卡顿	卡顿的时长	1 ~ 10 帧随机	1 ~ 8 帧随机	1 ~ 8 帧随机
响应延迟类 单个卡顿	卡顿的时长	-	1 ~ 10 帧随机	1 ~ 30 帧随机
	卡顿类型	-	延迟	延迟
一长多短的 多次卡顿	大卡顿的时长 t_1	2 ~ 10 帧随机	2 ~ 8 帧随机	2 ~ 8 帧随机
	小卡顿的长度 t_2	1 ~ min(3, $t_1 - 1$) 帧随机		
	总卡顿次数	最小 2 次, 最大 $\lceil 10/t_2 \rceil$		
均匀分布的 多次卡顿	卡顿的时长 t	1 ~ 6 帧随机	1 ~ 5 帧随机	1 ~ 5 帧随机
	总卡顿次数	最小 2 次, 最大 $\lceil 10/t \rceil$		

3.5 实验配置规则

整体实验配置规则如表3.2所示。实验程序可控的因素包括移动方向、卡顿类型、卡顿的时长、次数、多个卡顿发生的时间、是否为响应延迟，以及多次卡顿的分布模式。我们针对这些因素将卡顿模式分为四种类型，即非响应延迟的单个卡顿、响应延迟的单个卡顿、一长多短的多次卡顿，和均匀分布的多次卡顿。上下移动动画的实验配置不包含响应延迟类的单个卡顿，另外三种配置的数据量各 1/3；另外两种动画都包含所有四种类型，四种类型的配置为各 1/4。

第 4 章 卡顿感知模型的建立

4.1 背景知识

为了方便本章的陈述，本节中首先介绍一些所需要了解的统计学背景知识。

4.1.1 逻辑回归

统计学中，逻辑回归模型（Logistic Model / Logit Model）是一种通常用来分析二值因变量的数据的统计分析模型。逻辑回归适用的条件有：

1. 因变量为二分类的分类变量，其通常被表示为 0 或 1。
2. 因变量服从二项分布。
3. 自变量可以是一个，也可以是多个，但要求各自变量之间相互独立。
4. 自变量与逻辑概率是线性关系。

下面具体解释第 4 条条件。在逻辑回归模型中，假设某事件发生概率 p 的对数发生比（log odds，也称为 logit）是其对应自变量的线性组合，也因此逻辑回归模型属于广义线性回归模型的一种。Logit 的定义为：

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) \quad (4-1)$$

经过转换可以得到：

$$y = \frac{1}{1 + e^{-x}} \quad (4-2)$$

其中 $x = \text{logit}(p)$, $y = p$ 。公式4-2也被称为 Sigmoid 函数。逻辑回归模型的输出通常包含每个自变量以及常量对应的回归系数，以及他们分别对应的标准误差、 z 值，和 p 值。 p 值通过 Wald 检验得到，用来检验该回归系数的显著性。当仅有一个自变量时，逻辑回归模型的公式为：

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \quad (4-3)$$

其中 $p(x)$ 表示事件 x 发生的概率， β_0 为逻辑回归模型输出中的截距， β_1 则是模型输出中 x 对应的系数， e 为自然常数。多自变量时，逻辑回归模型的公式则为：

$$P(Y_i = 1 | \mathbf{X}_i) = \text{logit}^{-1}(\boldsymbol{\beta} \cdot \mathbf{X}_i + \beta_0) \quad (4-4)$$

此外，逻辑回归模型也可以被扩展到因变量为多值的情况。如果因变量是分类变量（Categorical Variable），则可以使用多分类逻辑回归（Multinomial Logistic Regression）；如果因变量不仅是分类变量，而且还是有序的，则可以使用有序逻辑回归模型（Ordinal Logistic Regression）。虽然逻辑回归模型本身并不是一个分类方法，但可以被用来做分类器，比如，通过设定阈值来得出结论。

4.1.2 有序逻辑回归

有序逻辑回归（也称为有序多分类逻辑回归）常被用来分析因变量为有序的数据，比如，我们在调查问卷中常见的“好”、“中”、“差”的评价。有序逻辑回归分析是从二元逻辑回归分析上衍生出来的，其原理是将因变量的多个分类依次分割为多个二元的逻辑回归。模型假设拆分后的几个二元逻辑回归模型的自变量系数相同，仅常数项不等。也就是说因变量各等级事件的 logit 为等差数列，换句话说，因变量各等级事件的发生比（odds）为等差数列。因此，有序多分类的逻辑回归模型中，必须对自变量系数相等这一假设进行检验（即平行线检验）。

4.1.3 朴素贝叶斯

朴素贝叶斯分类器（Naive Bayes Classifier）是一种常见的分类方法，它基于贝叶斯定理与特征条件相互独立的假设，是一种简单的分类器。其基本思想，即贝叶斯定理（Bayes' theorem），是根据某些事件发生的先验概率计算某事件的后验概率的方法，即如下式：

$$P(A | B) = \frac{P(A) \times P(B | A)}{P(B)} \quad (4-5)$$

其中 $P(A | B)$ 是在事件 B 发生的情况下事件 A 发生的概率，也被称为 A 的后验概率； $P(A)$ 是 A 的先验概率。

对于自变量个数大于 2 的情况，依然可以将贝叶斯定理推广，得到下式：

$$P(C_k | x_1, x_2, \dots, x_n) = \frac{P(C_k)P(x_1, x_2, \dots, x_n | C_k)}{P(x_1, x_2, \dots, x_n)} \quad (4-6)$$

其中 n 个自变量的值分别为 x_1, x_2, \dots, x_n ，模型需要将自变量所代表的事件分类为 K 个之一， $P(C_k)$ ($k \in \{1, \dots, K\}$) 代表分类为 C_k 这一事件的概率。

由于模型假设特征（自变量）相互独立，上述公式右边的因子部分 $P(x_1, x_2, \dots, x_n | C_k)$ 又可以转化为 $P(x_i | C_k)$ 的乘积，因而可以通过数据得到某事件发生的概率。朴素贝叶斯具有对缺失数据不太敏感的优势，算法也比较简单，因此得到了比较广泛的应用。

4.1.4 方差分析

方差分析 (Analysis of Variance, 即 ANOVA, 也称为变异数分析) 是用来检验数据中多个部分的均值的差异显著性的算法。方差分析方法认为不同类别之间的平均值差异来源于随机误差 (也被称为组内误差) 或者实验条件误差 (也称组间误差) 之一。方差分析假设：

1. 各组样本的数据分布是正态分布。
2. 各组样本必须相互独立。
3. 各组方差相等。

方差分析首先利用平方和与自由度分别计算组内与组间的均方。具体来说，按照如下公式分别计算总变异量 (TSS)、组间变异量 (BSS)，以及组内变异量 (WSS)：

$$TSS = \sum_i \sum_j (\mathbf{Y}_{ij} - \bar{\mathbf{Y}}_{total})^2 \quad (4-7)$$

$$BSS = \sum_i n_i (\bar{\mathbf{Y}}_i - \bar{\mathbf{Y}}_{total})^2 \quad (4-8)$$

$$WSS = \sum_i \sum_j (\mathbf{Y}_{ij} - \bar{\mathbf{Y}}_i)^2 \quad (4-9)$$

其中 $i (i = 1, 2, \dots, I)$ 为组别， $j (j = 1, 2, \dots, J)$ 为观测值个数， \mathbf{Y}_{ij} 为第 i 组第 j 个观测值， $\bar{\mathbf{Y}}_{total}$ 为所有观测值的平均数。 n_i 为 i 组内观测值总数， $\bar{\mathbf{Y}}_i$ 为第 i 组的平均数。

由于以上的值会受到观测数量的影响，因此再用 BSS 和 WSS 分别求得平均组间变异量 BMSS 和平均组内变异量 WMSS：

$$BMSS = BSS/(k - 1) \quad (4-10)$$

$$WMSS = WSS/(N - k) \quad (4-11)$$

其中 k 为分类个数， N 为数据总量。两个均方的比值 $BMSS/WMSS$ 也就是我们所说的 F 值。较大的 F 值代表组间差异远大于组内差异，也就是说各组代表的平均数存在明显差异。

4.1.5 交叉熵

交叉熵 (Cross Entropy) 的概念来自于香农的信息论，用来度量两个概率分布之间的差异。在介绍交叉熵之前我们首先介绍信息熵 (Entropy) 的概念。

信息熵 在信息论中，信息熵用于表示每个消息中包含的信息量，反映了一个系统的有序化程度。这一概念由香农提出，因此也被称为香农熵。其定义如下：

$$H(X) = \mathbb{E}[I(X)] = \mathbb{E}[-\ln P(X)] \quad (4-12)$$

其中 $\mathbb{E}[\cdot]$ 代表期望， X 为信息， P 为 X 的密度函数，可以理解为概率。当样本有限时，熵也可以表示为：

$$H(X) = \sum_i P(x_i)I(x_i) = -\sum_i P(x_i) \log_b P(x_i) \quad (4-13)$$

交叉熵 交叉熵的概念基于信息熵。交叉熵最早用于语言模型领域，用来评估每个词平均要用多少比特位来进行编码，从而找到压缩比最大，也就是所需位数最少的编码长度期望值。具体地说，对于各个词出现的真实概率分布 p 和一个非真实分布 q ，首先按照真实分布 p 来计算样本所需编码长度的期望值 $H(p)$ ，也就是 p 的信息熵：

$$H(p) = \sum_i p(i) \log \left(\frac{1}{p(i)} \right) \quad (4-14)$$

而如果采用错误的分布 q 来计算该期望，则得到 $H(p, q)$ ，我们称之为交叉熵：

$$H(p, q) = \sum_i p(i) \log \left(\frac{1}{q(i)} \right) \quad (4-15)$$

相对熵 相对熵（relative entropy）又称为 KL 散度（Kullback–Leibler divergence，简称 KLD）。相对熵可以用来表示两个概率分布的差异的非对称性，其定义为：

$$D(p\|q) = H(p, q) - H(p) = \sum_i p(i) \log \left(\frac{p(i)}{q(i)} \right) \quad (4-16)$$

即 p 与 q 的交叉熵减去 p 的信息熵。

相对熵可以衡量两个分布的差异，但在机器学习中通常使用交叉熵作为损失函数用来评估模型的预测分布 q 与数据分布 p 的差异。这主要是因为，数据分布固定且难以计算，因此在只需要比较两模型的性能时，比较其交叉熵即可。

4.2 数据预处理

用户实验通过某手机爱好者论坛发布，最终共收集到 2184 位用户的数据。我们首先对数据进行了基本的清洗：我们舍弃了缺失交互记录的数据，以及疑

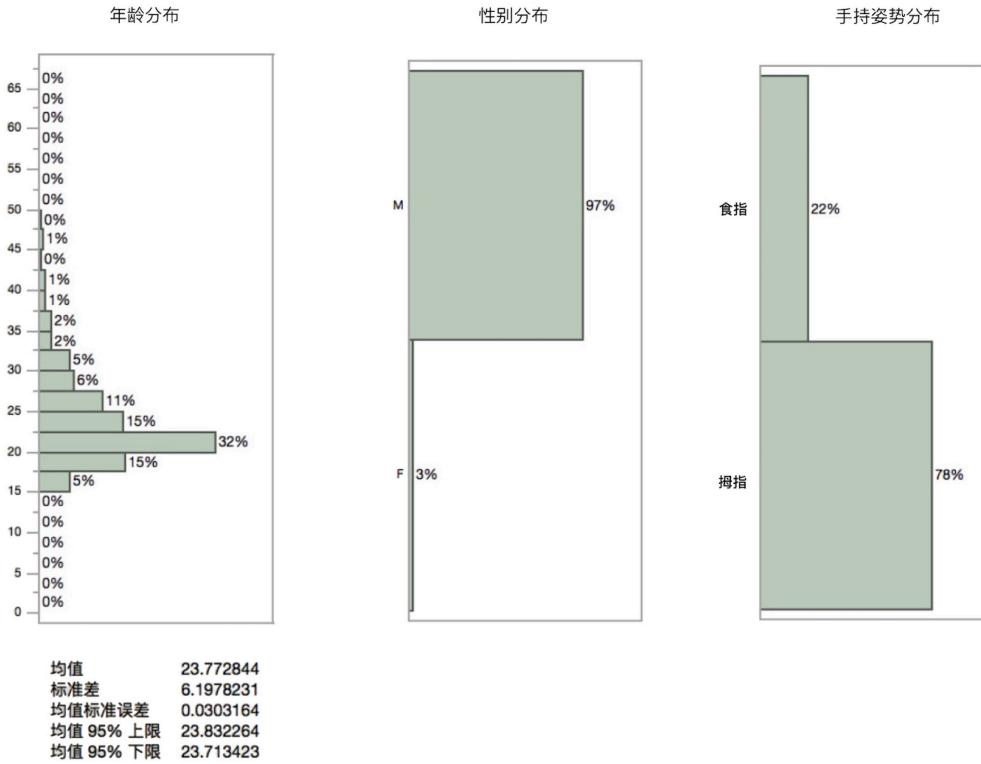


图 4.1 用户属性的分布

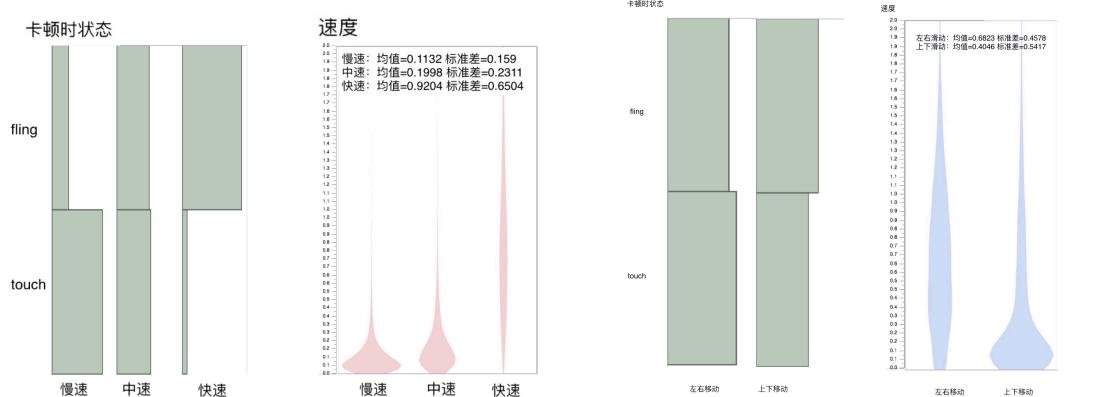
似恶意评分的数据。我们对恶意评分的定义是用户对同一类型动画中每一个任务的卡顿程度或流畅度体验都给出了相同的评分。数据清洗后有 41905 条记录，约合 1746 人次。

数据清洗后，我们还发现有些用户的手机尺寸异常，比如为 2.4 寸。依据记录的手机机型上网检索后，发现记录中有些手机尺寸并不正确。我们认为上述现象是用户修改了显示比例所导致，因此我们依据手机型号更新了所有记录中的手机尺寸一项，并依据旧尺寸与新尺寸的比例更新了滑动速度属性。

4.3 数据观察

我们首先对实验收集的所有数据分别进行了用户属性、设备属性、实验控制属性与实验不可控属性等方面的观察，并将发现与已有研究做了比较，结论基本一致。

用户属性 在用户属性中，我们发现，两种手持姿势中，使用食指的用户与使用拇指的用户的比例大约为 4:1，如图4.1所示。



(a) 对比上下移动任务的不同速度要求 (b) 对比上下移动任务与左右移动任务

图 4.2 实验不可控属性

实验不可控属性 实验不可控属性为卡顿时速度与卡顿时状态，观察中我们有两个发现：1) 用户操作越快，卡顿发生时速度越快；2) 用户操作越快，卡顿发生时状态为 fling^①的可能性越大，这也意味着用户手接触屏幕的时间变短。这两个发现依据图4.2(a)的数据。需要注意的是，已有研究^[23]中发现用户想要滑动速度越快，clutch^②的时间约短，但这并不与我们的结论冲突，因为 clutch 是手从终点方向向起点方向运动的时间，而 fling 则还包含了用户滑动动作后半程手指离开屏幕的时间。

另外，如图4.2(b)所示，左右滑动的任务中卡顿发生时速度的均值大于上下滑动任务中卡顿发生时速度的均值，介于中速上下滑动和快速上下滑动之间。左右滑动任务的卡顿时状态中 touch 与 fling 的占比则比较均衡。这说明左右滑动任务中手触摸屏幕的时间占比比较大。

设备属性与实验可控属性 设备属性的分布并无特别发现，实验可控属性的分布与实验配置的预期一致，因此不再赘述，具体分布请参考第3.5章的实验配置规则。

4.4 单因子分析

此部分的分析主要为定性分析，目的是更好的理解每一个因素是如何影响人对卡顿的感知的。分析历程大致分为 3 个步骤：

① 即列表仍在运动，但手已经离开屏幕。
② 用户完成一次滑动动作后回到起始位置，准备开始下一次滑动动作。

- 多元变量分析：简单挖掘各个自变量之间的相互关系。
- 单卡顿的单因子分析：以第一步中结论为依据，对每一个变量与用户评价的关系进行一对一的相关关系分析。
- 多卡顿的单因子分析：分析多个卡顿不同分布模式对用户感知的影响。

4.4.1 配置检验

配置规则的分类除了卡顿个数以外还依据了动画类型以及是否为响应延迟这两个条件。本节中，我们验证上述分类的合理性，即判断这两个条件是否会影响用户对卡顿的感知。

动画类型 检验动画类型对卡顿感知的影响时，我们选取只有一个卡顿且卡顿时间长度为1~8帧的数据，进行了不同动画类型的评分之间的横向比较。我们分别计算了每种动画每种评分的均值和标准差，并进行了方差分析。结果认为：上下滑动动画类型的三种用户评价指标均显著高于另外两种动画类型。具体数据汇总在表4.1中。这一发现与我们在预实验中的发现一致，也应证了我们将三种动画配置的参数修改为不同的合理性。

表 4.1 动画类型的方差分析

评价数据	项目	动画类型		
		上下移动	左右移动	点击放大
体验评分 ^①	均值	4.02	3.78	3.71
	标准差	0.018	0.019	0.017
	方差分析 R^2	0.013		
	结论	显著		
卡顿感知 ^②	均值	0.54	0.46	0.44
	标准差	0.008	0.008	0.007
	方差分析 R^2	0.0076		
	结论	显著		
卡顿评分 ^③	均值	3.32	3.06	2.97
	标准差	0.015	0.016	0.015
	方差分析 R^2	0.022		
	结论	显著		

① 体验评分分值为1~5。

② 卡顿感知分值为0或1，0代表有卡顿，1代表无卡顿。

③ 卡顿评分分值为1~4。1~3对应问题中三个选项，1为最严重的卡顿。当用户前一题选择无卡顿时，卡顿评分分值则被设为4。

表 4.2 响应延迟类型的方差分析

动画类型	项目	体验评分	卡顿感知	卡顿评分
点击打开	均值 (响应延迟)	4.26	0.67	3.6
	均值 (非响应延迟)	3.66	0.39	2.94
	标准误差 (响应延迟)	0.048	0.02	0.039
	标准误差 (非响应延迟)	0.034	0.15	0.028
	F 比	102.3	120.9	191.9
	结论	显著	显著	显著
左右滑动	均值 (响应延迟)	4.17	0.62	3.51
	均值 (非响应延迟)	3.66	0.36	2.98
	标准误差 (响应延迟)	0.024	0.12	0.19
	标准误差 (非响应延迟)	0.059	0.3	0.46
	F 比	63.72	63.99	110.06
	结论	显著	显著	显著

响应延迟类型 为了检验响应延迟类型对用户评价是否有显著影响，我们对区分了这一属性的两种动画类型分别选出：除是否为响应延迟这一属性外配置均相同^①的数据，并对其用户评价的分布进行单因素方差分析。表4.2为单因子方差分析的结果。

可见，两种动画的三个评价指标在响应延迟与非响应延迟模式下的值均有显著差异，这证明我们将单卡顿数据分成两类的是合理的。也为我们之后多卡顿分析中增加“是否为响应延迟”这一属性提供了依据。

4.4.2 多元变量分析

多元变量分析的主要目的是探索多个因子之间的关系。需要注意的是，这并不是独立性检验的一部分。我们采用的做法是对所有因素中任意两个构成的一对进行基于线性回归的相关性分析，并据此绘制出各变量之间的相关性 r 矩阵、相关性概率 (F 检验的 p 值) 矩阵、以及包含相关性系数的数据图矩阵，如图4.3所示。

分析发现，三种动画类型中均有屏幕尺寸与手持姿势相关的现象。屏幕越大，越多比例的用户选择使用食指操作。这一结论符合常识。此外，在左右滑动的数据中，我们发现用卡顿时状态也与户手持姿势有关，使用食指时用户会有

^① 点击放大任务选择了卡顿时长均为 1 ~ 8 帧，且卡顿类型均为延迟类型的单个卡顿。筛选后检验了其他因子的分布均相同。左右滑动任务选择了卡顿时长均为 1 ~ 8 帧，卡顿类型均为延迟类型，且卡顿状态均为手指触摸屏幕的单个卡顿。筛选后检验了其他因子的分布均相同。



图 4.3 多元分析的示例

更多的时间处在触摸屏幕的时间。在上下滑动的数据中，卡顿时状态还与滑动方向有关，列表向上移动时，用户会有更多的时间处在触摸屏幕的时间。

4.4.3 单卡顿单因子分析

单因子分析中，我们首先依据多元变量分析的结果对数据进行筛选，然后依据数据类型分别进行方差分析或者数值拟合。以下为差异具有显著性的因素：

卡顿时长 总体来讲在每一种配置中卡顿时长都与用户评价显著相关。卡顿时长越长，用户评价越差。每种配置类型的数据中卡顿时长与评价的数值关系不完全相同。表4.3为最佳拟合的详细信息。图4.4分别绘制出了每种配置下的数据变化趋势以及拟合较好的几种关系。

此外，我们也计算了每种动画类型与响应延迟类型下用户能够感知到卡顿

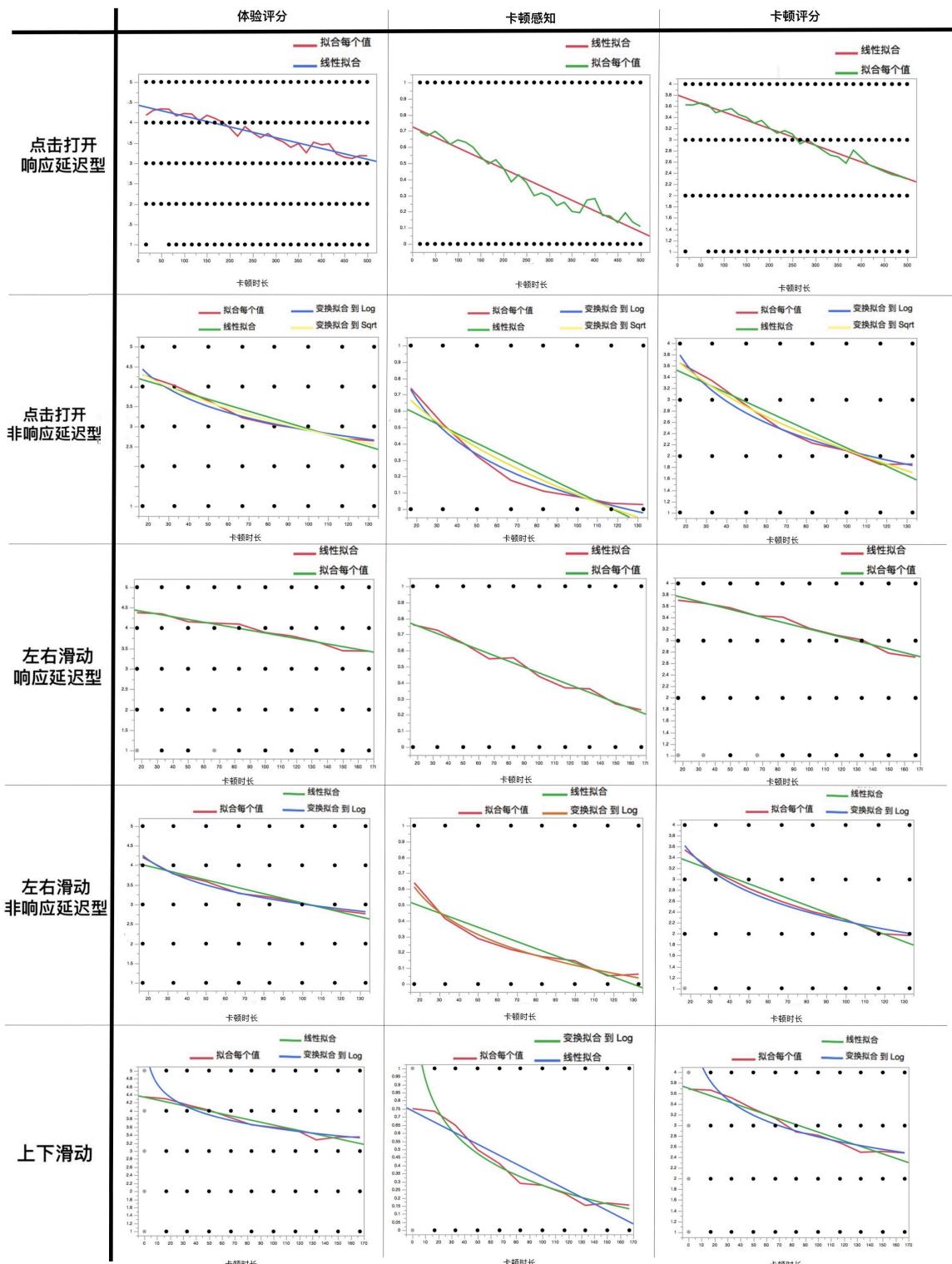


图 4.4 每种配置下的数据变化趋势

表 4.3 卡顿时长与各评价的关系

评价数据	配置类型	关系	R
体验评分	点击打开（响应延迟型）	线性	0.33
	左右滑动（响应延迟型）	线性	0.30
	点击打开（非响应延迟型）	对数	0.45
	左右滑动（非响应延迟型）	对数	0.35
	上下滑动	线性	0.37
卡顿感知	点击打开（响应延迟型）	线性	0.38
	左右滑动（响应延迟型）	线性	0.35
	点击打开（非响应延迟型）	对数	0.56
	左右滑动（非响应延迟型）	对数	0.39
	上下滑动	线性	0.47
卡顿评分	点击打开（响应延迟型）	线性	0.45
	左右滑动（响应延迟型）	线性	0.38
	点击打开（非响应延迟型）	对数	0.60
	左右滑动（非响应延迟型）	对数	0.48
	上下滑动	线性	0.50

的临界值^①。上下滑动的临界值为 50 ms 左右；非响应延迟型左右滑动的临界值为 17 ms，响应延迟型左右滑动的临界值为 100 ms；非响应延迟型点击打开的临界值为 33 ms，响应延迟型点击打开的临界值为 200 ms。

卡顿类型 在上下滑动类动画中，当卡顿类型为跳跃（即帧丢失）时，用户的三种评价都比较好。但在点击打开类动画中，当卡顿类型为延迟时，用户的三种评价都比较好。

卡顿时状态 在左右滑动以及上下滑动类动画中，当卡顿时状态为 touch 时，用户的三种评价都比较好。

卡顿时速度 在点击打开类动画中，卡顿时速度越快，用户评价越差。而在上下滑动类动画中，这一情况则相反，卡顿时速度越快，用户评价越好。

其他因素 此外，各个动画类型的数据中均可以看出：使用食指的用户整体的评价比使用拇指的用户好一些，但显著性概率在 $10^{-4} \sim 10^{-3}$ 级别。这或许可以说明使用食指时对卡顿的敏感度会降低。但由于这一变量是用户群体的一个属性，我们并不关心某一种手持姿势下的用户对卡顿的感知，而是更关注所有用

^① 我们假设卡顿时长低于临界值时用户无法感受到卡顿，高于临界值时可以感受到，那么使得分类错误率最低的取值被认为是临界值。

户群体对卡顿的感知，因此，在之后的分析中，我们并未对这一因素作考虑。而运动方向和任务速度在分析中并未呈现具有统计学意义的影响。

4.4.4 多卡顿单因子分析

实验任务中的多卡顿生成方式分为两种：均匀分布，以及一长多短。本小节主要汇报这两种方式对用户体验的影响。

均匀分布 我们选取卡顿总长度相同的数据分别进行分析，探究：如果将单个较长的卡顿拆分成多个较短的卡顿，是否能够提升用户体验。具体地，我们在单卡顿数据中选择卡顿时长为多卡顿中各个卡顿时长之和的数据，计算单卡顿的评价的均值与多卡顿的评价的均值的差，并使用方差分析检验其是否有统计学意义。数据分析如表4.4所示。

表 4.4 均匀分布的多卡顿分析

卡顿总时长	每个卡顿时长	评价之差的均值与显著性 ^①		
		体验评分	卡顿感知	卡顿评分
2	1	-0.106	-0.041	-0.095*
3	1	-0.031	-0.036	-0.062
4	1	-0.212**	-0.078**	-0.13*
5	1	-0.19**	-0.11**	-0.196**
6	1	-0.152*	-0.16***	-0.233***
4	2	0.035	-0.014	0.002
6	2	0.211***	0.024	0.084
8	2	0.107	0.002	0.078
10	2	-0.307**	-0.228***	-0.605***
6	3	0.081	0.043*	0.097*
9	3	0.425***	0.091***	0.449***
12	3	-0.162	-0.011	-0.173*
8	4	0.202***	0.059***	0.218***
12	4	0.007	0.062	-0.004
16	4	0.344	0.091	0.337
10	5	0.612***	0.135***	0.561***
12	6	0.042	0.062*	0.09
14	7	0.362*	0.135	0.401**
16	8	0.114	0.075	0.223

^① * 代表显著性 p 值 < 0.05 , ** 代表 p 值 < 0.01 , *** 代表 p 值 < 0.001 。

结果显示，将卡顿拆成多个小卡顿能够显著提高用户体验，而拆成比较大的卡顿则反而会降低用户体验。根据观察，区别上述小卡顿与大卡顿的临界值会随着卡顿总时长的增大而增大，但由于数据量不够，我们并不能确认此观察是否可靠。

一长多短分布 我们选取卡顿总长度相同的数据分别进行分析，探究：如果把单个较长的卡顿拆分成一长多短的多个卡顿，是否能够提升用户体验。具体地，我们在单卡顿数据中选择卡顿时长为多卡顿中各卡顿时长之和的数据，然后分别计算每一种拆分方式下多卡顿的评价的均值与单卡顿评价的均值的差，并用方差分析检验总卡顿时长相等的这些分布方式之间的差异是否显著。表4.5展示了有显著差异 ($p < 0.05$) 的数据。

表 4.5 一长多短分布的多卡顿分析

卡顿总时长	长卡顿时长	短卡顿时长	评价之差的均值与显著性 ^①		
			体验评分	卡顿感知	卡顿评分
6	2	1	-0.153	-0.11***	-0.18*
7	2	1	-0.091	-0.121***	-0.144***
6	3	1	-0.124	-0.067***	-0.112*
7	3	1	-0.024	-0.037***	-0.14***
8	3	1	-0.104**	-0.095***	-0.178***
7	3	2	-0.036	0.057***	-0.048***
6	4	1	0.265	0.06***	0.207*
7	4	1	0.088	0.025***	-0.008***
8	4	1	-0.224**	-0.064***	-0.286***
6	4	2	0.112	0.042***	0.08*
8	4	2	0.2**	0.089***	0.249***
7	4	3	0.156	0.091***	0.225***
6	5	1	0.071	0.045***	0.102*
7	5	1	-0.239	-0.071***	-0.25***
8	5	1	-0.028**	-0.004***	0.03***
7	5	2	0.085	0.086***	0.134***
8	5	3	0.2**	0.048***	0.17***
7	6	1	0.146	0.072***	0.152***
8	6	2	-0.026**	0.046***	0.099***
8	7	1	-0.023**	0.039***	0.039***

① * 代表显著性 p 值 < 0.05 , ** 代表 p 值 < 0.01 , *** 代表 p 值 < 0.001 。

通过观察我们有以下两个结论：

1. 将卡顿拆成一长多短的分布时，长卡顿的时长对于用户体验的影响是最大的。长卡顿较短时，能够显著提升用户体验；而长卡顿较长时，则可能会使用户体验变差。长卡顿不超过 3 帧的时，对用户体验的影响基本上都是积极的。
2. 长卡顿的时长相同时，短卡顿越短，用户体验越好。

综上，将大卡顿拆分未必能提升用户体验。根据现有分析，我们可以给出如下的建议：要么将单个卡顿均匀拆分成很小的卡顿，要么拆分出一个比较小的卡顿和一些更小的卡顿。

4.5 多因子分析

单因子分析的主要目的是对各个因素的影响有一个基本的认知，多因子分析则是要整合所有的因子，训练一个模型，探究所有因素综合的效应，并能够对任意一个动画曲线给出体验评分、卡顿感知、卡顿评价的分布。

4.5.1 数据准备

为了方便说明，我们首先定义任务属性与卡顿属性两个概念。任务属性是指在实验中，对于一个任务^①只有一个值的属性，比如动画类型。卡顿属性则是指即便是连续出现多个卡顿，每个卡顿都有其自己对应的属性值，比如卡顿时长等。

由于每个任务中卡顿的总个数可能不同，因此我们首先需要将不同个数的卡顿转化为等长的向量，从而训练模型。我们采取的策略是：

1. 由于实验数据中大多数记录中卡顿个数都在 0 ~ 6 之间，因此，我们设定最多卡顿数量为 6。对于不足 6 个卡顿的记录，我们用一些默认值来填充。对于超过 6 个卡顿的记录，我们在保留第一个卡顿的前提下，选择长度比较长的 6 个卡顿。
2. 将选出的卡顿按照出现的先后编号为 1 ~ 6，并按照卡顿时间长度由大到小重新排序，然后按照新的顺序依次拼接每个卡顿的属性，再与任务属性拼接在一起。按照时间长度重新排序是为了保持一致性，因为当卡顿个数不足时，我们填充了一些时间长度为 0 的卡顿在向量尾部，因此尾部的卡

^① 在最终的评级系统中则是对于一个视频输入。

表 4.6 模型输入的默认值

类别	因素	默认值	常规值
卡顿属性	卡顿类型	1	0 (延迟)、2 (跳帧)
	卡顿时长	0	/
	卡顿时状态	-1	0 (静止)、1 (触摸)、2 (滑动)
	卡顿时速度	0	/
	距后一个卡顿时间	0	/
	原本次序	-1	0 ~ 卡顿个数 - 1
任务属性	是否为响应延迟	-1	0 (响应延迟)、1 (非响应延迟)

顿应当时间比较短。此外，根据单个卡顿中数据的粗略分析，卡顿时长的影响非常显著，解释比超过 50%，所以我们认为以时长为排序标准也是合理的选择。但同时，我们也记录了其原本的次序编号作为属性之一。

由于实验输出的是人群对卡顿的感知，因此我们不再关心用户属性的具体分布或者设备属性^①。由第4.4.3节的单因子分析得知“运动方向”与“任务速度”对用户评价均无显著影响，因此我们也不再关心这两个因素。剩下的因素中，我们不将动画类型作为模型输入，而是直接按照不同类型训练了不同的模型。卡顿次数和分布方式可以由模型的输入向量推得，因此不再单独作为模型的输入。“卡顿发生的时间”被转化成“距后一个卡顿时间”。再加上新增的原本次序属性，我们得到了作为模型输入的因素与其默认值，如表4.6所示。

最后，我们将数据按照 4:1 的比例随机分成了训练集与测试集，以便进行下一步操作。

4.5.2 模型评估指标

由于本实验的数据的因变量为用户的评价，有一定的主观性，同样的输入会对应多个不同的输出。因此，评价模型不应只以预测值为标准，而应当观测预测的分布是否与实际的分布相同。我们选择了以下三种评价指标：加权误差、分布重合率，与交叉熵。

假设共有 k 个类别 $1, 2, \dots, k$ ，对应评分为 $1 \sim k$ 。对于一个输入，模型预测的各个类别的概率为 p_1, p_2, \dots, p_k ，实际的分类为 c ；对于一个分组，组内实际分类的分布对应的概率为 p'_1, p'_2, \dots, p'_k 。那么三种评价指标定义为：

^① 所有设备因素请参见表3.1。

- 平均加权误差：

$$\mathbb{E} \left[\left| c - \sum_{i=1}^k i \cdot p_i \right| \right] \quad (4-17)$$

首先对每条数据计算出评分按照预测概率加权的平均数，然后计算其与实际评分 c 之差的绝对值作为加权误差，最后计算所有测试数据的加权误差的均值作为模型的平均加权误差。这种评价指标虽然不能完全体现预测分布的准确率，但相比平均误差等常用的评价指标，已经包含了更多分布相关的信息。

- 分布重合率：

$$\sum_{i=1}^k \min(p'_i, p_i) \quad (4-18)$$

首先按照输入将数据划分为若干较小的子集，然后再比较该子集内实际分类的分布与预测分布均值的重合率，记作分布重合率。重合率是指两个分布的交集部分所占的比率。但由于观测分布需要将数据依据输入分成大小合适的子集，而输入变量较多，且包含很多离散值，因此划分子集的标准、子集的大小、划分过程中舍弃的数据都会对模型的评价产生影响。所以，我们只在评价最终选定的模型时使用这一指标。

- 交叉熵：

$$\mathbb{E}[H(p, q)] \quad (4-19)$$

交叉熵的定义见第4.1.5节，我们基于这一指标来选择表现较好的模型。计算的过程中，为了避免数据划分子集带来的问题，我们将每一个记录的分类视为值为 c 的概率是 1，为其他值的概率是 0 的单值分布。使用单值分布求得的交叉熵的平均等同于用原分布求得的加权平均。

4.5.3 模型选取

我们以加权误差和交叉熵为依据，从以下模型中选择表现较好的：

- Ordered Logistic Regression AT^① (OLR_AT)
- Ordered Logistic Regression IT^① (OLR_IT)

^① 由 Python 的开源工具包 mord 提供。

- Ordered Logistic Regression SE^① (OLR_SE)
- Multiclass Logistic Regression^② (LR)
- Multinomial Naive Bayes^② (NB)

表4.7汇总了各模型的得分。根据结果，我们以交叉熵为主要依据、加权误差为次要依据选择了OLR_AT模型作为最终的模型。

表 4.7 各模型在任务上的得分

动画类型	模型	体验评价		卡顿感知		卡顿评价	
		交叉熵	加权误差	交叉熵	加权误差	交叉熵	加权误差
点击打开	NB	6.517	2.012	6.451	0.426	7.399	1.638
	LR	1.486	1.209	0.467	0.345	1.218	1.095
	OLR_AT	1.446	1.214	0.418	0.259	1.165	1.105
	OLR_SE	1.604	1.197	0.417	0.259	1.239	1.111
	OLR_IT	5.868	1.147	0.418	0.259	1.551	1.125
上下滑动	NB	5.543	1.226	3.269	0.351	5.452	0.861
	LR	1.438	1.079	0.588	0.428	1.260	1.003
	OLR_AT	1.448	1.109	0.551	0.365	1.225	1.095
	OLR_SE	1.465	1.074	0.551	0.362	1.263	1.070
	OLR_IT	1.517	1.076	0.551	0.365	1.328	1.061
左右滑动	NB	3.623	1.557	1.867	0.352	3.170	1.366
	LR	1.505	1.115	0.565	0.413	1.303	1.045
	OLR_AT	1.472	1.186	0.515	0.336	1.245	1.136
	OLR_SE	1.545	1.140	0.512	0.333	1.303	1.114
	OLR_IT	1.658	1.079	0.515	0.336	1.425	1.069

4.5.4 模型分析

除了第4.5.2节中计算得到的加权误差和交叉熵，我们还需要对模型的效果做进一步的分析和检验。我们首先分析了预测系数，得到一些有关单个因素的结论，并与已有知识比较，然后利用分布重合率指标对模型效果进行部分检验。

预测系数 我们首先观察模型对各个因子的预测系数，预测系数表见附录B。

- **响应延迟**：我们发现响应延迟的系数为负数，这意味着响应延迟类的用户评价比较好，这与我们单因子分析的结果相吻合。

① 由 Python 的开源工具包 mord 提供。

② 由 Python 的开源工具包 sklearn 提供。

- 卡顿时长：卡顿时长虽然对于卡顿次数越多的位置影响的方向和大小都越来越没有规律，但整体而言还是符合我们之前的单因子分析，即卡顿时间越长，用户评价越差。
- 卡顿类型：卡顿类型随着卡顿个数的推移系数的符号由负变正，这可能是对第一个卡顿类型系数的补偿。第一个卡顿类型的系数符合单因子分析的结果，即延迟类型的卡顿评价比较好。此外可以看出，卡顿类型对上下滑动类型的评价影响远小于另外两种动画类型。
- 卡顿时速度：在点击打开动画中，速度越大，评价越差，这听起来比较合理。在上下滑动的动画中表现为速度越大，评价越好，这与已有研究^[23]中的结论一致，即快速滑动时人比较不敏感。而在左右滑动动画中的影响相对而言很小。
- 距后一个卡顿的时间：距后一个卡顿的时间影响很小，但第一个卡顿的影响系数均为正，即间隔越大，评价越好。
- 次序：点击打开类型动画中，次序的系数为正，即意味着卡顿发生在靠后位置的评价比较好。猜测是因为点击打开的动画靠后的位置变化比较小，因此卡顿比较难以察觉。而在上下滑动的动画中，次序的系数为负，这意味着卡顿发生在靠后位置的评价比较差，可以理解为随着时间的推移，用户对之前发生的卡顿的负面评价会一点点被消解。而左右平移动画中比较没有规律，第一个参数为正。
- 卡顿时状态：卡顿时状态表现为对于左右移动动画，fling 阶段的卡顿会导致评价较差，而对于上下移动动画类型这种影响小很多。

分布重合率 对于每一种动画，我们按照显著因子分类，找出划分出的子集中数据量大于 30 的数据集，分别对每个子集计算出分布重合率。由于数据量比较小，当卡顿个数比较多的时候数据量就不足以支持拆分出数据量大于 30 的子集了，因此表4.8只列出了整体、无卡顿，和单个卡顿的信息。整体和无卡顿的结果图见图4.5。综合来看，整体预测的重合率较高，随着分类越来越细致，重合率有所降低，单个卡顿的平均重合率在 80% 左右。我们对这一结果较为满意。

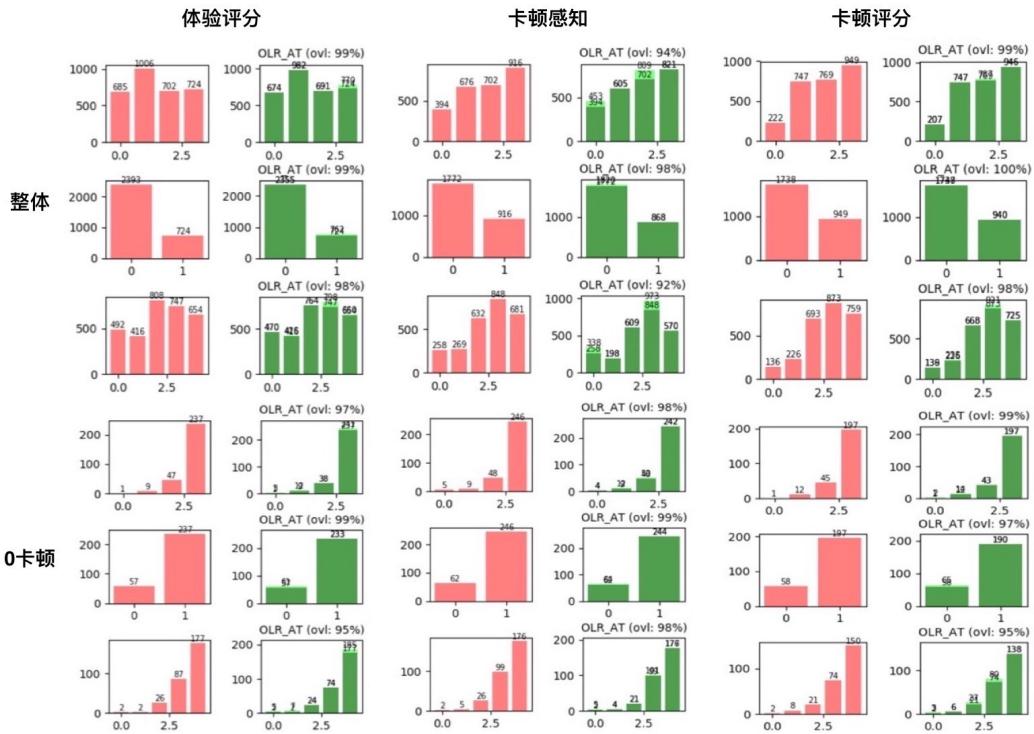


图 4.5 分布重合率^①

① 图中红色条形为实际数据中分类的分布，绿色为模型预测的分布。绿色中加深的部分为与红色重合的部分。

表 4.8 分布重合率

数据子集	动画类型	子集数量	重合率		
			体验评价	卡顿感知	卡顿评价
整体	点击打开	1	97.9	98.8	98.5
	上下滑动	1	97.8	99.7	99.3
	左右滑动	1	92.4	98.2	93.8
无卡顿	点击打开	1	97.2	98.9	94.8
	上下滑动	1	98.9	97.3	95.3
	左右滑动	1	98.2	99.1	97.8
单个卡顿	点击打开	23	81.3 ± 10.2	88.2 ± 7.6	80.3 ± 10.6
	上下滑动	7	87.0 ± 6.3	91.1 ± 5.8	79.5 ± 13.8
	左右滑动	11	80.4 ± 11.0	92.8 ± 4.8	78.2 ± 10.5

第5章 总结与展望

5.1 本文总结

智能手机用户体验的重要性与日俱增，而流畅度体验的评价方法往往并不能体现真实的用户体验。因此，本研究的目的是了解用户对卡顿的感知模式，从而为改善手机中的卡顿现象提供知识与工具。

现有工作的不足主要在于以下三点：1) 大多只关注整体的延迟或低帧率，鲜有工作关注随机发生的卡顿，而这反而是我们日常使用电子设备中最常遇到的卡顿；2) 只关注到了时间相关的因素，而没有关注到其他可能影响用户体验的因素；3) 只关注到了对卡顿或延迟感知的阈值，而没有关注到整体的感知规律。

本研究综合考虑目的与现有工作不足设定了如下目标：

1. 从三种常见的手机交互动画入手，通过大规模用户实验探究多种因素及其组合对用户体验的影响方式，并建立相应模型。
2. 依据感知模型设计并实现出一个完整的卡顿评价体系。
3. 用我们的卡顿评价体系观察不同手机上的不同 App 的表现。

最终，本研究的内容如下，基本达成了目标：

1. 完成了大规模无监督用户实验的设计及进行。我们探究了大规模无监督用户实验的设计难点以及解决方式，并通过实验收集了大量数据。无论是数据本身还是实验方法都可以对未来的相关工作提供参考。
2. 分析了各个因素对卡顿感知的影响。对已有知识涉及的部分，我们将实验结论与已有研究进行比对，基本没有冲突。此外，我们还得到了一些新的发现。
3. 建立了卡顿感知模型。我们的模型能够给出对三种用户体验指标的分布预测，诸如：有多少比例的人能够感知到卡顿，有多少比例不能。系数分析与分布重合率检验结果说明，我们的模型效果较好。
4. 设计并完成了整个卡顿评价体系，主要包括三个步骤：动画提取、卡顿提取、模型预测。详细地说，首先利用摄像机拍摄手机画面，然后用计算机视觉的方法从中提取动画曲线，再从中找出卡顿及其属性，并将其作为输入利用卡顿感知模型进行分析，得到用户评价的预测分布。本文用体验评

分、卡顿感知，与卡顿评分这3个指标来描述用户评价。这一卡顿评价体系不同于现在的常用流畅度评价方式，因此具有一定的创新性。

5. 对基于计算机视觉的动画提取进行了初步探索，完成了对上下滑动、左右滑动、点击放大三种效果的动画提取，并取得了不错的效果。
6. 利用动画提取工具对现有手机以及常用App进行了探索性分析，发现了现有App中出现的不同卡顿补偿策略，并初步分析了用户对这些策略效果的感知。

本文工作所产生的价值体现在以下几点：

1. 本文的发现可以填补“用户对卡顿感知”的知识空白，对理解用户体验有重要意义。
2. 本课题建立的卡顿感知模型以及卡顿评价体系可作为手机流畅度体验的评价方式，应用于智能手机的UI性能评价。这样直接反映用户体验的评价可以为系统优化的目标提供数据支持，对软硬件的优化方向、交互动画设计都有重要的指导价值。这样精准的定位问题也可以节约大量资源，带来巨大的经济效益，高效地提升用户体验。
3. 本课题收集的用户实验数据规模较大，其价值可供其他相关研究继续挖掘，数据收集方法也可供有类似需求的研究者参考。
4. 研究流程和方法都具有一定的探索性与创新性，可以为未来相关的研究提供参考。比如卡顿评价的方法，卡顿评价体系的设计，将模型用于用户体验评价的思路，大规模无监督用户实验的方法，动画提取的方法等。

5.2 本文展望

在本文的工作中，也存在一些不足，包括：

1. 卡顿检测部分：目前只检测了简单的卡顿，而没有考虑系统本身或者应用会进行的优化和补偿。有些补偿行为可能会导致没有明显的卡顿，但速度的突然变化可能也会对流畅度体验带来影响。此外，即便依然存在明显卡顿，补偿行为依然会对用户体验带来影响。日后的研究中还应综合这些发现进行。
2. 用户实验部分：
 - (a) 由于实验招募发布在某手机爱好者论坛，98%的被试都是男性。因此，我们期望在后续的工作中招募更多的女性被试来保证数据的有效性。

3. 常用 App 分析：目前的分析仅使用了动画提取算法，而没有使用整个评价体系对市面上的手机和应用进行探索。这将是接下来的工作中重要的一部分。

除了以上不足外，还有一些可以继续探索的方向，包括：

1. 动画曲线提取部分：动画曲线的提取目前只支持三种基本的动画，可以继续探索更多更复杂的动画的检测。
2. 用户实验部分：
 - (a) 目前恶意评分只是在分析数据时被去除，以后的实验可以考虑在程序中增加一些机制来检查用户是否有恶意评分的行为，如果有，及时提醒用户。
 - (b) 目前记录的卡顿信息虽然没有直接使用配置，而是用用户手机插入卡顿时所使用的参数，但这依然可能与真实的卡顿有微小的差别，我们可以在后续的工作中进一步完善这一部分的数据。

插图索引

图 1.1	卡顿评价系统的流程图	5
图 2.1	搭载 FLIR BlackFly S 摄像头的成像系统	6
图 2.2	SIFT 算法各步骤的效果示例	9
图 2.3	平移动画提取结果	10
图 2.4	缩放动画提取算法流程图	11
图 2.5	微博的动画模式	12
图 2.6	大众点评的动画模式	13
图 2.7	今日头条的动画模式	13
图 2.8	CNN 上下滑动的动画模式	14
图 2.9	CNN 左右滑动的动画模式	14
图 2.10	Instagram 的动画模式	15
图 3.1	实验中的评分界面	19
图 3.2	实验中的确认事项	20
图 3.3	实验中的任务说明	21
图 3.4	实验中的任务提示	22
图 3.5	实验流程图	22
图 3.6	三种动画效果的模拟场景截图	24
图 3.7	点击放大动画的曲线	25
图 4.1	用户属性的分布	31
图 4.2	实验不可控属性	32
图 4.3	多元分析的示例	35
图 4.4	每种配置下的数据变化趋势	36
图 4.5	分布重合率	45

表格索引

表 2.1	UIAutomation提供的主要操作.....	8
表 3.1	实验中收集的所有信息.....	18
表 3.2	实验配置规则	26
表 4.1	动画类型的方差分析	33
表 4.2	响应延迟类型的方差分析	34
表 4.3	卡顿时长与各评价的关系	37
表 4.4	均匀分布的多卡顿分析.....	38
表 4.5	一长多短分布的多卡顿分析	39
表 4.6	模型输入的默认值	41
表 4.7	各模型在任务上的得分	43
表 4.8	分布重合率.....	45
表 B-1	各任务对应的模型参数.....	64

参考文献

- [1] Huhtala J, Sarjanoja A H, Mäntylä J, et al. Animated ui transitions and perception of time: a user study on animated effects on a mobile screen[C]//Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. [S.I.]: ACM, 2010: 1339-1342.
- [2] Kuroki Y, Ishihara M. Manipulating animation speed of progress bars to shorten time perception[C]//International Conference on Human-Computer Interaction. [S.I.]: Springer, 2015: 670-673.
- [3] Schlienger C, Conversy S, Chatty S, et al. Improving users'comprehension of changes with animation and sound: An empirical assessment[C]//IFIP Conference on Human-Computer Interaction. [S.I.]: Springer, 2007: 207-220.
- [4] Shanmugasundaram M, Irani P. The effect of animated transitions in zooming interfaces[C]// Proceedings of the working conference on Advanced visual interfaces. [S.I.]: ACM, 2008: 396-399.
- [5] Khalid H, Shihab E, Nagappan M, et al. What do mobile app users complain about?[J]. IEEE Software, 2015, 32(3): 70-77.
- [6] Sillars D. High performance android apps: Improve ratings with speed, optimizations, and testing[M]. [S.I.]: O'Reilly Media, Inc., 2015
- [7] Liu Y, Xu C, Cheung S C. Characterizing and detecting performance bugs for smartphone applications[C]//Proceedings of the 36th International Conference on Software Engineering. [S.I.]: ACM, 2014: 1013-1024.
- [8] Developers A. Profile gpu rendering speed[EB/OL]. 2018[2018-06-05]. <https://developer.android.com/studio/profile/inspect-gpu-rendering>.
- [9] Developers A. Test ui performance[EB/OL]. 2018[2018-06-05]. <https://developer.android.com/training/testing/performance>.
- [10] Gómez M, Rouvoy R, Adams B, et al. Mining test repositories for automatic detection of ui performance regressions in android apps[C]//Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on. [S.I.]: IEEE, 2016: 13-24.
- [11] Ng A, Annett M, Dietz P, et al. In the blink of an eye: investigating latency perception during stylus interaction[C]//Proceedings of the 32nd annual ACM conference on Human factors in computing systems. [S.I.]: ACM, 2014: 1103-1112.
- [12] Armitage G. An experimental estimation of latency sensitivity in multiplayer quake 3[C]// Networks, 2003. ICON2003. The 11th IEEE International Conference on. [S.I.]: IEEE, 2003: 137-141.

- [13] Annett M, Ng A, Dietz P, et al. How low should we go?: understanding the perception of latency while inking[C]//Proceedings of Graphics Interface 2014. [S.I.]: Canadian Information Processing Society, 2014: 167-174.
- [14] Forch V, Franke T, Rauh N, et al. Are 100 ms fast enough? characterizing latency perception thresholds in mouse-based interaction[C]//International Conference on Engineering Psychology and Cognitive Ergonomics. [S.I.]: Springer, 2017: 45-56.
- [15] Jerald J J. Scene-motion-and latency-perception thresholds for head-mounted displays[D]. [S.I.]: The University of North Carolina at Chapel Hill, 2009.
- [16] Ng A, Lepinski J, Wigdor D, et al. Designing for low-latency direct-touch input[C]// Proceedings of the 25th annual ACM symposium on User interface software and technology. [S.I.]: ACM, 2012: 453-464.
- [17] Developers A. Android performance patterns: Why 60 fps?[EB/OL]. 2015[2018-06-05]. <https://www.youtube.com/watch?v=CaMTIgxCSqU>.
- [18] Braga Sangiorgi U, Motti V G, Beuvens F, et al. Assessing lag perception in electronic sketching[C]//Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design. [S.I.]: ACM, 2012: 153-161.
- [19] Claypool M, Claypool K. Latency and player actions in online games[J]. Communications of the ACM, 2006, 49(11): 40-45.
- [20] Sheldon N, Girard E, Borg S, et al. The effect of latency on user performance in warcraft iii [C]//Proceedings of the 2nd workshop on Network and system support for games. [S.I.]: ACM, 2003: 3-14.
- [21] Friston S, Karlström P, Steed A. The effects of low latency on pointing and steering tasks[J]. IEEE transactions on visualization and computer graphics, 2016, 22(5): 1605-1615.
- [22] Liu Z, Heer J. The effects of interactive latency on exploratory visual analysis[J]. IEEE transactions on visualization and computer graphics, 2014, 20(12): 2122-2131.
- [23] Quinn P, Malacria S, Cockburn A. Touch scrolling transfer functions[C]//Proceedings of the 26th annual ACM symposium on User interface software and technology. [S.I.]: ACM, 2013: 61-70.
- [24] Casiez G, Pietrzak T, Marchal D, et al. Characterizing latency in touch and button-equipped interactive systems[C]//Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology. [S.I.]: ACM, 2017: 29-39.
- [25] Deber J, Jota R, Forlines C, et al. How much faster is fast enough?: User perception of latency & latency improvements in direct and indirect touch[C]//Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. [S.I.]: ACM, 2015: 1827-1836.
- [26] Swindells C, Dill J C, Booth K S. System lag tests for augmented and virtual environments[C]// Proceedings of the 13th annual ACM symposium on User interface software and technology. [S.I.]: ACM, 2000: 161-170.

- [27] Lowe D G. Object recognition from local scale-invariant features[C]//Computer vision, 1999. The proceedings of the seventh IEEE international conference on: volume 2. [S.l.]: Ieee, 1999: 1150-1157.
- [28] Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features[C]//European conference on computer vision. [S.l.]: Springer, 2006: 404-417.
- [29] Fischler M A, Bolles R C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography[M]//Readings in computer vision. [S.l.]: Elsevier, 1987: 726-740
- [30] Weinreich H, Obendorf H, Herder E, et al. Not quite the average: An empirical study of web use[J]. ACM Transactions on the Web (TWEB), 2008, 2(1): 5.
- [31] Catrambone R. Following instructions: Effects of principles and examples.[J]. Journal of Experimental Psychology: Applied, 1995, 1(3): 227.
- [32] 薛瑞尼. THUThESIS: 清华大学学位论文模板[EB/OL]. 2017. <https://github.com/xueruini/thuthesis>.

致 谢

衷心感谢我的导师史元春教授和喻纯副教授，本研究的工作是在他们的精心指导下完成的。老师们精益求精的工作态度和科研热情鼓舞着我，他们的言传身教也将使我终生受益。

感谢实验室的钟鸣远同学，以及其他共同参与过这份工作的同学，他们给予了我很多帮助，与我一同讨论、探索、解决问题。

感谢所有参与到本研究的用户实验的用户们，以及在用户实验上给予我很多指导与帮助的吴思举、张宇博学长。

感谢我的父母、男朋友，以及其他在毕设期间关心着我的亲友们，是他们的关爱和支持造就了今天的我，他们也是我想要改善人在人机交互过程中的体验的原因和动力。

最后，感谢 L^AT_EX 和 THU⁺ESIS^[32]，帮我节省了不少时间。

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名: 王伟 日 期: 2018年6月21日

附录 A 外文资料的书面翻译

触摸滚动动画效果

摘要：触摸滚动系统使用一个传递函数将触摸屏上的手势转换为滚动输出。这些传输函数的设计非常复杂，因为它们必须能够让人直接控制界面上的内容，并且在数据量较大时支持快速滚动。然而，研究人员改进这两个功能的能力受到以下因素的影响：1) 对用户如何通过触摸手势表达滚动意图的了解；2) 缺乏关于转移函数的专业知识，这导致研究人员可能使用不符合事实的技术；3) 缺乏评价现有传递函数的工具。为了解决这些问题，我们研究了用户如何在人为因素实验中表达滚动的意愿；我们描述了对现有的“黑匣子”传递函数进行逆向工程的方法，包括使用精确的机器人手臂；我们使用这些方法来公开 Apple iOS 和 Google Android 的传递函数，公开数据和程序以使得我们的实验可以复制。我们讨论了获取的知识如何提高实验的严谨性并对改进触摸滚动效果提供帮助。

关键词：控制显示增益；轻弹滚动；滚动加速度；触摸界面；传递函数。

A.1 介绍

基于手势的“触摸滚动”允许用户通过模拟显示内容上的直接物理交互进行滚动。对于现实世界的隐喻——比如能够以一种速度使页面内容飞出去的能力和逐渐减慢它的摩擦力——有助于为用户提供一种被认为是“简单自然”的滚动机制。尽管基于手势的滚动感受起来很简单，但这一表现背后的机制却很复杂。举例来说，在苹果 iPhone 上进行轻轻滑动并仔细观察会发现，“轻滑”手势后的滚动速度并不总是遵循简单的物理定律，有时也会应用加速的滚动速度和快速减速。

触摸滚动技术的行为由转换函数 [8,17] 决定，该转换函数将人的输入操作转换为合成的滚动输出效果。这些函数想要把以下内容作为输入参数，如各个手势的属性（例如它们的长度，速度，压力或方向），手势序列的属性（例如一连串快速的手势的次数或时间间隔），页面上文档或其他可见物的属性（如文档的长度或内容的类型），以及当前系统状态和设置（如滚动速度或摩擦设置）。虽然任何或所有这些参数都可以用于任何特定的实现，但实际使用哪些参数的公

开信息以及它们的使用方式依然很少。此外，还缺乏关于用户如何选择通过手势表达他们对不同滚动速度的意图的基本的人因知识。

鉴于其作为触摸屏交互的核心组件的角色，支配触摸滚动行为的传输函数获得了很多研究的关注（除了之后说的内容）。其中一个原因是，现有技术及其载体是黑盒子，不会暴露其操作的关键信息。大多数平台仅公开并允许控制其内置滚动功能相关参数的一小部分。另一个关键原因是缺乏可以观测传输函数的方法。虽然最近的工作已经公开了观察鼠标指向 [4] 和滚轮操作 [17] 得到传递函数的技术，但这些方法依赖于公布的 USB 标准来与外部设备进行通信，因此它们不能用于触摸滚动。

缺乏关于触摸滚动传递函数的信息对研究人员来说是一个问题。它是实验不精确性的来源，因为精确的系统设置无从得知，不可复现，从而妨碍了触摸滚动系统的迭代改进，因为关键的部分（包括人和系统）是未知的。例如，Aliakseyeu 等人的研究 [1]，Lee 等人的研究 [13]，和 Tu 等人的研究 [19] 全部实现和评估触摸滚动传递函数，其中参照组实验控制条件与商业设备的行为不同，而这并不是他们的本意。

对于解决这些问题，本文工作有四个主要贡献。首先，我们提出一个人因实验来理解执行触摸滚动手势时的用户行为的关键特征和限制。其次，我们提出的方法有助于对触摸滚动传递函数所使用的参数进行逆向工程。其中包括受控软件的仿真，以及使用高质量选择性铰接式机器人手臂（SCARA）来精确控制输入手势动作。第三，我们使用这些方法比较 Apple iOS（iPhone 和 iPad）和 Google Android 所使用的传输函数，并汇报结果。结果显示出一些相似之处，但是它们的传递函数也存在实质性差异，这表明涉及手势滚动的任何实验都可能受到其进行平台的影响。第四，我们公开了用于提取，仿真和测试这些传输函数的工具的源代码；作为当前最新技术的记录和迭代改进的基础。

A.2 背景

术语“滚动”通常在 HCI 中非正式使用，指的是在界面显示的信息的任意几何转换，包括对角线或椭圆运动。然而，这种用法与“滚动”这个词的来源不一致，它一次限制运动到一个维度。在本文中，我们使用“滚动”一词来指代可见内容的一维移动。本节介绍滚动传输函数和现有技术的背景。

A.2.1 滚动传输函数

传递函数负责将人对设备的操作转换为输出效果 [8]。一些研究人员已经注意到滚动传输函数对用户行为的重要影响 [例如 7,20]，但由于缺乏分析，报告和控制其设置的方法，之前的滚动研究所使用的传递函数并不精准。

Casiez 和 Roussel[4] 最近对控制鼠标指向的传递函数进行了类似的调查，发现了不精确的“实验内容”。因此，Casiez 和 Roussel[4] 开发了 USB 鼠标仿真器 (EchoMouse) 和软件来解决这些问题。奎因等人 [17] 改编了 Echo-Mouse 来分析商业滚轮设备使用的滚动传输函数。图 2 来自 Quinn 等人的研究。[17] 显示了这种功能中可能发生的三个转换原则：1) 函数将设备单元转换为显示单元，例如毫米的手指位移为滚动平移的像素；2) 增益放大或衰减输入信号；3) 持久性允许状态记忆影响未来的行为，比如计算重复轻弹动作的次数，或者在时间上应用的效果（如动量）。传递函数也可以受用户设置和环境（例如文档或显示像素密度的长度）的影响。

奎因等人发现不同滚动轮驱动器供应商使用的参数与使用参数的方式之间存在很大差异。例如，有些应用横跨滚动方向的不同增益水平，并且在重复的滚动轮操作中累计应用增益，另一些仅涉及滚动轮速度，有些则关注输入持续时间。传输函数的多样性表明用一个滚轮来支持滚动操作其实十分复杂。

A.2.2 触摸滚动技术

通过触摸输入，除了可以通过滚动轮表达，还可以很轻易地表达其他参数：许多手势都是可能的——线性手势 [例如 1, 13]，椭圆 [例如, 2,18] 或振荡运动 [例如 14]；变化的压力 [例如 9, 16]；接触点的数量以及肢体接触的类型 [例如 6] 等。

在本文中，我们关注线性（或接近线性）手势。我们使用术语“拖动”来描述在触摸表面上移动手指的动作，与系统反馈无关，我们使用术语“轻拂”来描述在与触敏表面离开的位置快速移动手指的行为。线性手势的基本元素在概念上很简单，如图 1 所示：1) 用户与屏幕表面接触；2) 他们将手指拖到屏幕表面上；3) 他们拉回他们的手指（可能发起轻拂），这涉及将接触从表面提起以回到方便的位置以供后续的重复拖动动作；4) 完成后它们从表面脱离，并且可以：5) 通过将手指放回屏幕并保持它们的位置来抓住表面。尽管这些动作显然很简单，但它们使用多种参数来通过传递函数影响滚动输出。

A.2.2.1 接触与拖拽

在接触和拖动过程中，用户可以控制参数，包括手势长度，接触时间，速度，加速度，接触面积，接触面的方向，肢体使用，压力和位置。也可以使用多个接触（例如，两个手指滚动）和双手接触（例如，屏幕上的两个拇指像“跑步”一样交替滚动）。在商业和研究系统中有许多使用这些参数的传输函数示例：比如，滑动条和手指位置之间的垂直距离决定了 Apple iOS 的音乐应用中播放头的控制大小，研究人员也已经研究过使用压力控制焦点 [16] 和惯性速度 [3] 的方法。

用户对接触和拖拽/轻拂过程中传递函数行为的感知很可能受交互作用直接或是间接的影响。通过直接交互，触摸屏幕与输出屏幕重合，这让人觉得在对内容进行直接的物理操作。然而，这种操作的预期效果一直存在争议——用户的动作是否应该被解释为观察的窗口（将内容移动到与手指移动相反的方向），还是内容应当跟随手指移动 [11,12]——现在的技术赞成后者。这种感觉通过肢体行为得到了增强，例如能够用模拟动量和摩擦“甩开”内容。虽然这些行为通常使用从表面的接触移动到输出效果的 1:1 映射，但 Kwon 等人 [12] 表明，用户可以容忍映射的不连续性，特别是在滚动大距离时。

间接交互（如触控板滚动）涉及输入表面和显示内容之间的分离，这会减少或消除 1:1 映射的需要。实际上，触控板上的多指滚动通常反转了“直接”操作设备上使用的规则，向下拖动操作会滚动到文档的末尾而不是开头。输入和内容之间的不连贯可以用来让用户更精确地控制他们的滚动动作；例如，欣克利 (Hinckley) 等人 [9] 描述了几种方式，其中沿着触敏区域的运动可以用于滚动，包括响应于压力的自动滚动和响应于手指在条上的速度加速滚动。

A.2.2.2 回归并重复

回归和重复是相关的，因为方便的线性手势重复需要执行回归动作。回归涉及通过抬起手指暂时脱离触摸表面，然后将其重新接触屏幕上的初始位置附近（然而，某些类型的触摸交互并不需要显式脱离了 [例如 2,14,18]）。

先前使用滚轮进行的工作 [10,17] 已经使用了连续抓取动作的次数来增加滚动增益以促进长距离滚动。类比在重复手势序列上累积应用增益，这一方法同样可用于触摸滚动。此外，惯性和摩擦的隐喻可以用来支持手指不接触时的继续滚动 [15,21]。

A.3 用户实验：人手滚动机制

设计触摸滚动传输功能需要了解“人通过手势在设备上表达意图，执行输入”的特性和限制。手势较慢速度控制滚动的意图相对较为明显——即用户以每秒几毫米的速度缓慢拖动手指划过触摸屏表面时，用户的意图可能是缓慢滚动。然而，快速滚动的机制尚不清楚，其中包括以下未知因素：用户选择哪些操作参数来表示快速滚动的意图；他们的手指移动有多快；手势滑动范围有多大；手势重复的速度有多快？

本研究的主要目标是收集用户在触摸设备上提供的用于执行滚动的输入模式的数据，并回答上一段中的问题。此外，结果用于确定用机器人设备的研究 2 中一些参数的数值范围。

触摸滚动界面和滚动反馈的先前经验将影响用户对滚动行为的期望。为了减少这些因素对实验的影响，本实验没有提供任何滚动反馈。相反，参与者被要求采取任何觉得自然的手势来实现所要求的滚动速度而没有任何系统响应。

A.3.1 方法

当被要求设想以三种不同速度（浏览速度，舒适快速，最快速）和两个方向（上和下）滚动时，我们观察了参与者的姿势动作。分析了三种设备和姿势的组合下的数据：一只手握住 Apple iPhone 4S，另一只手用食指操作；用握着设备的手的拇指操作的 Apple iPhone 4S；和一只手握住苹果 iPad 2 并用另一只手的食指操作（图 3）。这几种方式代表了设备最常见的操作姿态。设备始终保持纵向。

A.3.1.1 被试和配置

12 名志愿者（4 名女性，平均年龄 26 岁）参加了实验。所有这些都是拥有并经常使用触摸屏设备（各种供应商）的右手使用者，他们都是计算机科学专业研究生学生。软件记录了所有的手势事件，视频捕捉了参与者的姿势和动作。

捕捉手势运动的量化方式将影响我们的结果，特别是在高速度运动和短时间手势的情况下。为了解这个严重的问题，我们分析了设备事件报告。iPad 和 iPhone 报告以 16 ms 的间隔移动。我们通过在显示屏边缘执行非常快速的拖动手势来测试显示屏边缘坐标的准确度，结果显示精确度达到 $\pm 1 \text{ mm}$ 。然而，第一次事件报告的坐标和时间在自上次接触以来已经过去了几秒之后（我们认为传感器面板的轮询间隔增加是为了节省电池电量）是可变的。同样，频繁断开联系的最终事件报告显示比先前报告低得多的速度。这可以解释，因为在设备

可以确认联系已经丢失之前，他会缓冲事件。因此，我们的结果描述了通过现有设备（在应用程序级别）观察到的手势中的人因因素。

A.3.1.2 流程

参与者被告知，实验涉及用户如何以不同的速度和方向在触摸屏设备上执行滚动手势。他们被告知他们不会看到他们的滚动动作的效果，但他们应该发出手势来控制三种速度：

- 浏览速度：“想象一下，您正在滚动浏览歌曲列表，寻找适合的播放内容。”
- 舒适快速：“想象一下，您想快速滚动到长页面或列表的底部。”
- 最高速度：“尽可能快地滚动。”

没有显示滚动反馈的一个结果是滚动方向不明确（试验性研究表明参与者在被指示向下滚动时，倾向于将其手指向下移动，而这将导致文档向上滚动）。因此，我们指示参与者将手指向下移动以进行“向下”任务，以及如何在每个姿势下握住设备。

实验软件显示了一个标题栏，其中包含下一个手势组的目标速度，方向和触摸屏上的数字计时器。显示屏的其余部分为纯黄色，但显示屏中央的“点击开始”按钮除外。点击按钮导致显示屏变绿；当参与者首次接触屏幕时，计时器从 6 秒开始倒计时。达到零时，显示器将变为红色 1 秒钟，然后返回到黄色“轻点以开始”状态。为了让参与者熟悉程序，我们加入了两次“浏览速度”的练习试验（这一过程中的数据被丢弃）。

每位参与者用三种设备与姿势的组合中的每一种完成了 18 次 6 秒的试验。试验包括三个方向和速度的重复；所有人都先完成一个姿势（姿势的顺序会随机以保证整体的均衡）的全部任务，再进入下一个阶段。在每个姿势中，所有的试验都先完成一种速度的全部任务（两个方向，在试验中交替进行），再进入下一个速度。

A.3.1.3 设计

重申一下，实验目标是描述触摸滚动中的人手势的操作，从而产生若干因变量。可达到的最大速度可以知道通过触摸滚动传递函数处理的值的范围，但是对于可以帮助设备区分用户不同滚动速度的意图的任何特征我们都会感兴趣，这包括回归时间（或手势频率）和手势长度。本研究店有关因素包括速度（浏览、舒适快速、最快速度）、设备与姿势（iPhone 与拇指、iPhone 与手指、iPad）、

方向（上、下）。其中，设备和速度使用拉丁方格的方法在被试中进行了交叉平衡；方向顺序是交替的。

A.3.2 结论和分析

速度的方差分析（使用格温豪斯-盖瑟球度校正）对速度有显著的影响 ($F_{1.76,19.34} = 35.5, p < .001, \eta_p^2 = .76$)，浏览、舒适快速，和最快速度对应的平均值分别为 178、399，和 560 mm/s。任何参与者在 iPad 上向上的姿势的最大速度不超过 1378 mm/s。设备与姿势的主要影响也是显著的 ($F_{1.22,13.36} = 42.1, p < .001, \eta_p^2 = .80$)，最慢的是 iPhone 与拇指的组合（平均 186 mm/s），然后是 iPhone 与食指 (390 mm/s)，然后是 iPad (560 mm/s)。一个显著的 *Speed×Device/Posture* 相互作用 ($F_{2.60,28.57} = 18.5, p < .001, \eta_p^2 = .63$) 最好归因于 iPhone 与拇指速度在目标速度之间的相对较小的差异。这种相互作用表明，速度对于区分滚动意图可能是一个很差的指标——虽然它可能成功用于区分手指，但单手拇指的手势也并不适用。方向没有影响 ($p = .05$)，也没有任何其他的相互作用。

回归时间分析（图 4，中间）表明，它是用户对不同滚动速度的意图的相当健壮的鉴别器。速度有显着影响 ($F_{1.30,14.25} = 16.4, p = .001, \eta_p^2 = .60$)，浏览时为 536 ms，舒适快速为 322 ms，最快速度为 160 ms。方向也显示出显着的效果 ($F_{1,11} = 5.6, p = .037, \eta_p^2 = 0.34$)，下行拖拽后的抓住（平均 320 ms）略快于上行 (339 ms)。没有其他主效应或相互作用是显着的。回归时间的结果表明用户选择在想要快速滚动时增加他们的手势频率（减少回归时间）。

最后，对手势长度的分析表明，设备外形对手势有重要影响，iPad 的较大尺寸比 iPhone 上的食指或拇指（分别是 46 mm 和 31 mm）鼓励更长的手势（平均 92 mm）—— $F_{1.39,15.31} = 75.12, p < .001, \eta_p^2 = .87$ 。速度没有显著影响 ($p = .35$)，这表明手势长度对于确定不同滚动速度的用户意图而言是一个较差的指标。没有其他影响是显著的。

A.4 讨论

这些结果展示了人们在不同设备和姿势下用于表达不同滚动速度意图的参数。结果还可以看出不同设备对应的用户表达的值的范围。

主要的发现是，回归时间是滚动速度意图的可靠指标，用户增加手势频率以表达对更快滚动的期望。手势速度也可以可靠地区分滚动速度的意图，但不能用拇指表示。预期速度之间的姿势长度变化不大。回归时间的一个直接应用

是传递函数可以通过在重复手势上应用累积增益来加速滚动。

此外，我们注意到了我们方法的一些限制。首先，如前所述，为了专注于用户的手势表达的自然方法，我们没有提供滚动反馈。然而，用户将不可避免地将他们的手势映射到输出行为，所以需要进一步的工作来检查这种关系。其次，我们的方法使用设备已知的手势类型和数据，因此量化效应会引入一些错误，特别是在高速度操作下。但无论如何，传输函数需要使用设备已知的数据进行操作，因此我们的结果反映了当前设备可以接收的内容。第三，该设备的物理特性将影响手势的表现，包括表面材料（例如玻璃与塑料），边框边缘或框架的存在以及设备尺寸等。

A.5 结论

触摸滚动是移动交互的一个基本组成部分，但很少有公开的其行为的实现方法。这对于研究人员来说是一个问题，因为他们缺乏可以用来改进交互的信息，而且由于无法知道实现在现有封闭系统中的具体参数，实验与当代系统的比较可能很难复杂。我们提供了可用于揭示触摸滚动传输函数的方法，并使用人因研究的结果来指导检测 Apple iOS 和 Google Android 的触摸滚动传输函数。我们在用户滚动信息发生变化的人机交互功能以及系统所关注的触摸输入属性中发现了有趣的特点。这些方法和结果，连同本文附带的数据表格和软件，为寻求改善触摸滚动传递函数并验证其有效性的研究提供了更坚实的基础。

原文索引

- [1] Quinn, Philip, Sylvain Malacria, and Andy Cockburn. Touch scrolling transfer functions. Proceedings of the 26th annual ACM symposium on User interface software and technology. ACM, 2013.

附录 B 各任务对应的模型参数

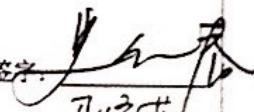
表 B-1 各任务对应的模型参数

因子		体验评价				卡顿感知			卡顿评价	
		点击打开	上下滑动	左右滑动	点击打开	上下滑动	左右滑动	点击打开	上下滑动	左右滑动
卡顿 1	是否响应延迟	-1.1243	-0.4469	-0.5232	-1.3393	-0.1119	-0.5911	-1.664	-0.4441	-0.5816
	卡顿时长	-0.0045	-0.0149	-0.0124	-0.0068	-0.0178	-0.0163	-0.0066	-0.0177	-0.0162
	卡顿类型	-0.2199	-0.1103	-0.2943	-0.2794	0.0318	-0.2553	-0.2876	-0.0641	-0.2711
	卡顿时速度	-0.1797	0.2555	0.0004	-0.1439	0.0645	-0.0022	-0.1873	0.1727	-0.0011
	距后一个卡顿时间	0.0007	0.0001	0.0002	0.0012	0.0001	0.0002	0.001	0.0002	0.0002
	次序	0.0709	-0.0282	0.0551	0.199	-0.0305	0.2426	0.0694	-0.083	0.0506
卡顿 2	卡顿时状态	-0.3608	0.1011	-0.1721	-0.0271	-0.0676	-0.2828	-0.3955	-0.0658	-0.2692
	卡顿时长	-0.0174	-0.0166	0.0005	-0.0428	-0.0118	0.006	-0.0249	-0.0162	0.0047
	卡顿类型	0.1812	-0.0343	0.0968	0.2079	0.0302	0.1709	0.216	-0.021	0.1638
	卡顿时速度	-0.0245	0.2849	0.0017	-0.0133	0.0986	-0.0013	-0.0355	0.2462	0.0006
	距后一个卡顿时间	-0.0001	0.0002	0	-0.0013	0.0002	0	-0.0005	0.0002	0
	次序	0.0865	-0.1297	-0.172	0.3273	0.0111	-0.0448	0.1439	-0.112	-0.2066
卡顿 3	卡顿时状态	-0.3608	0.1889	-0.0076	-0.0271	0.0959	-0.3048	-0.3955	0.1702	-0.1401
	卡顿时长	-0.0131	-0.0058	-0.0054	-0.0223	-0.0102	-0.0077	-0.0153	-0.0096	-0.0125
	卡顿类型	-0.1247	0.2122	0.2793	-0.0332	0.0477	0.2314	-0.0638	0.1915	0.3374
	卡顿时速度	-0.0064	0.2213	0.0026	-0.0025	0.1203	0.0006	-0.0052	0.2465	0.002
	距后一个卡顿时间	-0.003	0.0002	0.0001	-0.0004	0.0001	0.0003	-0.0026	0.0002	0.0004
										(接下页)

表 B-1 各任务对应的模型参数（续）

因子		体验评价						卡顿感知				卡顿评价	
		点击打开	上下滑动	左右滑动	点击打开	上下滑动	左右滑动	点击打开	上下滑动	左右滑动	点击打开	上下滑动	左右滑动
卡顿 3	次序	0.1683	-0.1051	-0.0532	0.0409	0.0196	-0.062	0.172	-0.0589	-0.1297			
	卡顿时状态	-0.3608	-0.0089	0.1555	-0.0271	0.0466	0.0468	-0.3955	0.039	0.0349			
	卡顿时长	-0.0295	0.0124	-0.0132	0.0061	-0.0067	-0.0116	-0.0276	0.0295	0.0133			
	卡顿类型	0.1633	0.3007	0.402	0.1896	0.063	0.2108	0.1566	0.2866	0.4117			
	卡顿时速度	-0.0056	0.1639	0.0017	-0.0025	0.0906	0.0005	-0.0044	0.1871	0.0012			
	距后一个卡顿时间	0.0008	0.0003	0	0.0018	0.0001	0	0.0003	0	0.0002			
卡顿 4	次序	0.0504	-0.1421	0.0647	-0.2049	0.008	0.0826	0.0163	-0.0942	-0.0139			
	卡顿时状态	-0.3608	-0.1882	-0.0189	-0.0271	-0.0184	0.0984	-0.3955	-0.1535	-0.0656			
	卡顿时长	0.0476	0.0763	0.0591	-0.0084	0.0159	-0.0428	0.054	0.0781	0.0349			
	卡顿类型	0.2287	0.3717	0.5477	0.1409	0.0943	0.2026	0.1912	0.3649	0.4868			
	卡顿时速度	-0.0016	0.0829	0.0006	-0.0002	0.0405	0	-0.0013	0.0906	0.0004			
	距后一个卡顿时间	0.0047	0.0003	0.0003	0.0022	0	0.0005	0.0059	0.0001	0.0009			
卡顿 5	次序	-0.3162	-0.2789	-0.0942	-0.1022	-0.048	0.0712	-0.3688	-0.2615	-0.1144			
	卡顿时状态	-0.3608	-0.3325	-0.2936	-0.0271	-0.0756	0.0141	-0.3955	-0.3257	-0.2537			
	卡顿时长	0.102	0.3032	0.0539	0.0085	0.0779	0.0004	0.1022	0.227	0.003			
	卡顿类型	0.2142	0.4181	0.5944	0.0593	0.1018	0.1507	0.3353	0.4068	0.5022			
	卡顿时速度	-0.0004	0.0383	0.0004	-0.0003	0.0123	0	-0.0005	0.0341	0.0002			
	距后一个卡顿时间	0.0036	-0.0001	0.0008	-0.0026	0.0001	0.0002	0.004	0.0001	0.0018			
卡顿 6	次序	-0.335	-0.3422	-0.2017	-0.0275	-0.0824	0.0492	-0.3322	-0.3646	-0.1916			
	卡顿时状态	-0.3608	-0.3879	-0.4158	-0.0271	-0.0971	-0.0126	-0.3955	-0.401	-0.341			

综合论文训练记录表

学生姓名	王倩					
学号	2014011319					
班级	计 42					
论文题目	手机动画效果的量化及卡顿感知模型的建立					
主要内容以及进度安排	<p>主要内容：</p> <ol style="list-style-type: none"> 1. 通过用户实验手机卡顿感知数据，建立数学模型 2. 使用高速摄像机拍摄手机交互过程，量化动效曲线 3. 完成曲线分析工具，结合感知模型给出卡顿评估 <p>进度安排：</p> <ol style="list-style-type: none"> 1. 寒假完成用户实验应用 2. 3-4月完成动效量化部分 3. 5月完成数据分析、验证 4. 6月完成论文撰写、工具制作 					
	<p style="text-align: right;">指导教师签字: </p> <p style="text-align: right;">考核组组长签字: <u>周伟芳</u></p> <p style="text-align: right;">2018 年 1 月 17 日</p>					
中期考核意见	<p>已完成文献调研、数据采集与整理工作，逻辑合理，为进一步深入研究打下基础。</p> <p style="text-align: right;">考核组组长签字: <u>周伟芳</u></p> <p style="text-align: right;">2018年4月10日</p>					

指导教师评语	<p>王倩同学为了手机动画卡顿建立 了数据计算模型，达到了综合论 文训练的预期目标。</p> <p>指导教师签字: <u>尹红霞</u></p> <p>2018年6月19日</p>
评阅教师评语	<p>论文对手机动画卡顿现象进行了探讨，利用 计数摄像机拍摄分析，建立了手机动画卡顿模 型理论与数据计算等方法，通过对手机动画卡顿进 行量化分析。论文提出的模型，内容丰富，字迹 规范，达到了本科综合论文学术训练的要求。</p> <p>评阅教师签字: <u>周红霞</u></p> <p>2018年6月19日</p>
答辩小组评语	<p>论文针对手机动画卡顿现象，提出了一种手 机动画卡顿数据模型，利用计数摄像机拍摄及 分析，对手机动画卡顿现象进行量化分析。论文立 题新颖，资料丰富，分析深入，结论准确，逻辑严密， 答辩过程回答问题合理、流畅，达到了本科综合 论文学术训练要求。</p> <p>答辩小组组长签字: <u>周红霞</u></p> <p>2018年6月19日</p>

总成绩: 88

教学负责人签字: 张永海

2018年6月20日