

東 南 大 學

毕业设计(论文)报告

题 目： 基于虚拟现实的桌面手势交互
系统设计开发

学 号： 02018325

姓 名： 张瑞升

学 院： 机械工程学院

专 业： 机械工程

指导教师： 周小舟

起止日期： 2022.01-2022.06

东南大学毕业（设计）论文独创性声明

本人声明所呈交的毕业（设计）论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

论文作者签名: _____ 日期: ____ 年 ____ 月 ____ 日

东南大学毕业（设计）论文使用授权声明

东南大学有权保留本人所送交毕业（设计）论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学教务处办理。

论文作者签名: _____ 导师签名: _____
日期: ____ 年 ____ 月 ____ 日 日期: ____ 年 ____ 月 ____ 日

摘要

近年来，随着传感器设备的优化和计算机图像处理技术的提升，虚拟现实人机交互技术不断向着弱侵入性、高沉浸感的方向发展，以空中手势交互为主的自然手势交互成为虚拟现实最受欢迎的交互方式之一。然而，在佩戴头戴式显示器的坐姿作业状况下，用户长时间使用空中手势与虚拟界面产生接触式操作会引起手部疲劳不适，降低交互绩效，解决自然手势交互过程中的疲劳问题尤为重要。本课题面向虚拟现实坐姿作业环境，研发一种低疲劳、高效率、强灵活、多模态、适合长时间使用的桌面手势交互软硬件系统。课题主要完成的工作如下：

(1) 利用红外激光投影传感原理，搭建基于虚拟现实的桌面手势交互硬件系统，通过硬件选型、人因实验、尺寸设计、电路设计、产品设计、生产加工全过程，完成硬件系统的样机加工。

(2) 面向虚拟现实应用，开发了一套配合桌面手势交互硬件的桌面手势交互软件系统。基于计算机视觉技术开发桌面手势识别算法，通过分析桌面手势交互图像提取桌面手势交互信息；基于相机成像原理，对桌面手势交互点进行校正；采用射线隐喻的交互方式，实现桌面手势输入、交互系统响应的基本交互单元。

(3) 基于桌面手势交互软硬件系统，以用户显示控制相合性为导向，开发了三种选择型和四种操纵型桌面手势交互功能模块，覆盖了常用的交互功能控件，并实现和其他常见手势交互系统功能的兼容。

本课题提出了适用于虚拟现实的桌面手势交互技术，研发了桌面手势交互设备硬件系统，并将桌面手势交互软件系统封装成预制体供其他开发者使用。桌面手势交互技术有效解决了交互疲劳问题，对于降低自然手势疲劳问题、提升虚拟现实人机交互绩效、实现触摸手势在虚拟现实系统应用等方面具有指导意义。

关键词：虚拟现实，自然手势交互，手势识别，计算机视觉

ABSTRACT

In recent years, with the optimization of sensor equipment and the improvement of computer image processing technology, virtual reality human-computer interaction technology has been continuously developed in the direction of less intrusiveness and high immersion. One of the most popular ways to interact. However, in the sitting position of wearing a head-mounted display, the long-term use of aerial gestures and virtual interface to generate contact operations will cause hand fatigue and discomfort, reduce interaction performance, and it is particularly important to solve the fatigue problem in the process of natural gesture interaction. This topic is oriented to the virtual reality sitting working environment, and develops a low-fatigue, high-efficiency, strong flexible, multi-modal, desktop gesture interaction software and hardware system suitable for long-term use. The main tasks completed by the subject are as follows:

- (1) Using the principle of infrared laser projection sensing, a virtual reality-based desktop gesture interaction hardware system is built, and the prototype processing of the hardware system is completed through the whole process of hardware selection, human factors experiment, size design, circuit design, product design, production and processing.
- (2) For virtual reality applications, a desktop gesture interaction software system is developed that cooperates with desktop gesture interaction hardware. The desktop gesture recognition algorithm is developed based on computer vision technology, and the desktop gesture interaction information is extracted by analyzing the desktop gesture interaction images; based on the camera imaging principle, the desktop gesture interaction points are corrected; the interaction method of ray metaphor is used to realize desktop gesture input and interactive system response. basic interaction unit.
- (3) Based on the desktop gesture interaction software and hardware system, guided by the compatibility of user display and control, the desktop gesture interaction function module is developed, and the functions of other common gesture interaction systems are compatible.

This subject proposes a desktop gesture interaction technology suitable for virtual reality, develops a hardware system for desktop gesture interaction equipment, and encapsulates the desktop gesture interaction software system into a prefab for other developers to use. The desktop gesture interaction technology effectively solves the problem of interaction fatigue, and has guiding significance for reducing the fatigue problem of natural gestures, improving the

performance of virtual reality human-computer interaction, and realizing the application of touch gestures in virtual reality systems.

KEY WORDS: Virtual Reality, Natural Gesture Interaction, Gesture Recognition, Computer Vision

目 录

摘要	I
ABSTRACT	II
第一章 绪论	1
1.1 课题背景和意义	1
1.2 国内外研究现状	3
1.2.1 虚拟现实人机交互研究现状	3
1.2.2 自然手势交互技术研究现状	6
1.3 研究内容及组织框架	8
1.3.1 研究内容	8
1.3.2 组织框架	8
第二章 基于虚拟现实的桌面手势交互系统架构与技术	11
2.1 基于虚拟现实的桌面手势交互系统开发目标	11
2.2 基于虚拟现实的桌面手势交互的系统组成	12
2.3 基于虚拟现实的桌面手势交互系统的关键技术	13
2.3.1 红外激光投影传感技术	13
2.3.2 计算机视觉桌面手势图像处理技术	14
2.3.3 针对图像畸变的桌面手势交互点校正技术	15
2.3.4 桌面手势显控耦合技术	18
2.4 本章小结	19
第三章 基于虚拟现实的桌面手势交互硬件系统	20
3.1 硬件系统架构	20
3.2 硬件系统参数确立	21
3.2.1 硬件模块选型	21
3.2.2 基于人主观舒适交互区域的实验研究	22
3.2.3 硬件尺寸确立	25
3.3 硬件系统电路设计	26
3.4 硬件系统产品设计	27
3.4.1 硬件系统产品造型设计	27

3.4.2 硬件系统产品结构设计	28
3.4.3 硬件系统产品生产加工	32
3.5 本章小结	34
第四章 基于虚拟现实的桌面手势交互软件系统	35
4.1 软件系统架构	35
4.2 基于计算机视觉技术的桌面手势识别算法	36
4.2.1 桌面手势识别算法的一般流程	36
4.2.2 桌面手势识别过程噪声去除方法	37
4.3 基于成像原理的桌面手势图像畸变校正方法	39
4.3.1 基于张正友棋盘格标定法的相机内参求解	40
4.3.2 基于 OpenCV 库的相机外参求解	43
4.3.3 建立桌面手势交互点畸变校正关系式	45
4.4 基于射线隐喻的桌面手势显控映射交互算法	46
4.4.1 交互系统显控映射原理	46
4.4.2 基于 Unity3D 物理引擎的物理射线交互方法	47
4.4.3 基于 UGUI 系统的图形射线交互方法	49
4.4.4 交互方法总结	52
4.5 基于 Unity 的桌面手势交互预制体插件导出	52
4.6 本章小结	53
第五章 基于虚拟现实的桌面手势交互功能实现及展示	54
5.1 基于虚拟现实的桌面手势交互功能实现	54
5.1.1 选择型交互功能	54
5.1.2 操纵型交互功能	58
5.1.3 与其他交互方式协同型交互功能	60
5.2 桌面手势交互功能展示系统	60
5.2.1 实例开发工具简介	60
5.2.2 界面信息框架	61
5.2.3 界面布局形式	62
5.2.4 界面最终效果	63
5.3 本章小结	66

第六章 总结与展望	67
6.1 总结.....	67
6.1.1 课题主要内容.....	67
6.1.2 课题创新点.....	68
6.2 展望.....	68
参考文献	70
附录 A 桌面手势识别脚本源码.....	74
附录 B 桌面手势交互点速度求解源码	79
附录 C 物理射线隐喻交互源码	81
附录 D 图形射线隐喻交互源码	85
致 谢	92

第一章 绪论

1.1 课题背景和意义

随着近几年传感器设备的优化和计算机图像技术的提升，消费级虚拟现实硬件数量激增^[1]，虚拟现实（virtual reality, VR）越来越多地渗透进人的日常生活。一方面，伴随“元宇宙”概念的火热，将 VR 构建的沉浸式虚拟环境（virtual environments, VEs）与现实世界融合成为人机交互（Human Computer Interaction, HCI）领域研究者的关注焦点；另一方面，疫情影响下居家办公、居家学习的现状加速了非接触式文化的形成，解锁 VR 远程办公、VR 远程会议等新型桌面交互场景成为一项新的需求。VR 支持多模态的信息输入方式，包含手势、语音、手柄等。其中，由于语音输入对环境干扰较大，手势和手柄交互是目前最常用的信息输入方式，但相对于现实中最常使用的键鼠操控方式和触控交互方式，目前手势和手柄支持的交互复杂度与交互舒适度还远远不够。这些传统 VR 交互方式大都无法满足新型桌面交互场景在低疲劳、高效率、长时间使用等方面的要求，针对新型桌面交互场景探索人性化桌面交互方式的重要性日益增高。



图 1-1 Meta 公司提出的虚拟现实远程协作场景

图片来源：<https://oculus.com/>

交互疲劳问题^[2]是新型桌面交互场景需要克服的首要问题，同时也是人机交互环节的普遍问题。它是一个非常复杂的现象，受到环境、用户的健康状况、活力、恢复能力以及交互行为持续时间等诸多因素影响^[3]，具体表现为用户产生疲倦感、作业能力下降。由于交互疲劳问题直接影响到用户交互体验，避免交互疲劳在人机交互环节中极为重要，从人

的因素出发，能否降低用户交互过程中的交互疲劳，是评判一种交互方式优劣的重要指标。

现有交互方式中，自然手势交互凭借其不借助中间设备、直接人体输入的特点，成为虚拟现实人机交互^[4]中最主要的交互方式。自然手势主要有两种：空中手势交互和触摸手势交互。以下分别分析两种自然手势交互的疲劳问题。

空中手势交互作为一种六自由度（degrees of freedom, DOF）的自然手势交互方式，被广泛应用于三维空间人机交互中。空中手势交互利用实时传感器追踪识别的手部运动实现与虚拟现实系统的非接触式交互^[5]，可有效促进虚拟环境和现实环境融合，保证了虚拟现实人机交互过程中的沉浸感。然而，在需要频繁交互、长期交互的交互情景中，用户长时间使用空中手势交互，手臂因缺乏物理支撑和长期处于不良交互体位而承受较高的静肌负荷，进而导致较严重的交互疲劳问题，交互舒适度下降，交互绩效也大幅降低。此外，一些空中手势交互动作需要用户肢体在空中经历相对完整的运动轨迹之后再对手势语义进行判别，从而具有一定的延时性^[4]，无法满足高频率交互场景下的实时交互需求。因此，空中手势交互方式虽然是目前虚拟现实系统应用最广泛的交互技术，但易疲劳和延时性的缺陷决定其并不适合新型桌面手势交互场景。

触摸手势交互是另一种常用的自然手势交互方式。它是一种基于用户手指在物理表面触摸的三自由度交互方式，凭借手指运动灵活性的特点，触摸手势交互已经被广泛应用于二维输入设备，例如显示屏、触摸屏、手持移动设备等。近年来，触摸手势交互也开始应用于三维交互场景^[6]，例如 Mircosoft HoloLens 利用手指在手腕上的触摸唤起增强现实环境下的交互菜单。由于触摸过程存在物理表面支撑且手指触摸动作幅度较小，触摸手势交互相较于空中手势交互的交互速度要快得多，交互疲劳要小得多。此外，用户与物理表面接触时会获得触觉反馈，这提升了用户在交互过程中获取信息的通道数量，优化了虚拟现实人机交互体验。交互快速、不易疲劳、提供触觉反馈等特点使触摸手势交互成为虚拟现实桌面交互场景的潜在解决方案。

综上所述，传统虚拟现实交互方式存在交互效率低、交互速度慢、易疲劳等方面的缺陷，无法满足需要频繁交互、长期交互的新型桌面交互场景在低疲劳、高效率、长时间使用等方面的需求。本课题面向虚拟现实新型桌面交互场景的需求，在保留空中手势存在前提下，利用桌面触摸手势实现一种虚拟现实桌面手势交互软硬件系统，为传统虚拟现实交互技术在交互效率、交互疲劳等方面的缺陷提供解决方案，同时对触摸手势交互在虚拟现实三维空间交互中的应用具有指导意义。

1.2 国内外研究现状

1.2.1 虚拟现实人机交互研究现状

虚拟现实是一种利用三维显示、三维跟踪及其他技术让用户处于沉浸式虚拟环境中的计算机系统^[7]，它为用户提供了庞大的可互动三维空间，通过多感官的输入输出，创造出身临其境的交互体验。凭借其沉浸、真实的三维交互特点，虚拟现实已经在医疗^[8]、游戏^[9]、教育^[10]、军事^[11]、工业^[12]、科学^[13]等众多领域得到了广泛研究。

虚拟现实人机交互系统是虚拟现实系统的重要组成部分，也是研究者对于虚拟现实的重点研究领域。不同于传统的二维用户界面，虚拟现实的三维用户界面对人机交互提出了更高的要求，这意味着需要新的交互设备、新的显控技术、新的界面设计原则来满足更复杂的三维用户界面交互需求。虚拟现实人机交互要求“人-机-环”系统^[14]协同工作，如图1-2所示，“人”通过行为控制输入设备来改变三维用户界面以及沉浸式的虚拟环境，“环”通过输出设备来影响用户的感知与认知，而虚拟现实的输入输出设备组成的“机”是“人”与“环”之间的重要桥梁。本小节将从虚拟现实的输入环节与输出环节两方面分别探讨虚拟现实人机交互领域的研究现状。

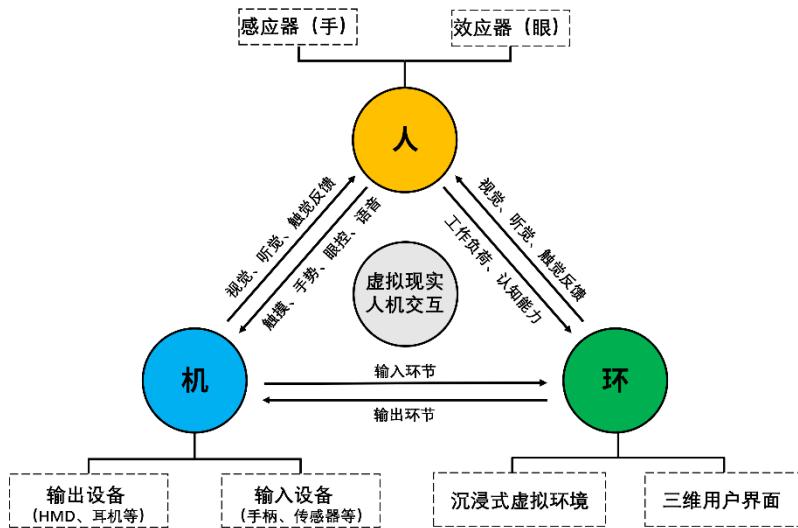


图1-2 虚拟现实人机交互流程

（1）虚拟现实的输入环节

虚拟现实为人们提供了庞大的可互动三维空间，三维空间的互动需要多自由度的三维输入设备与新型的三维输入方式，传统的二维输入方式（例如键鼠输入）已经无法满足三维空间的交互需求。目前，研究者已经探索出众多适用于虚拟现实的输入技术，这些输入

技术按照其输入的媒介可以分为裸手输入技术、可穿戴传感器输入技术、控制器输入技术和生理信号输入技术。表 1-1 整理了本文所提及的典型输入技术及其特点。

表 1-1 本文所提及主要输入技术

技术名称	输入类型	输入维度	交互效率	交互自然性	交互疲劳	是否应用于 VR
键盘鼠标	控制器	二维	高	低	低	否
Kinect ^[15]	裸手	三维	较低	高	较高	是
Leap Motion ^[16]	裸手	三维	较低	高	较高	是
MegaTrack ^[17]	裸手	三维	较低	高	较高	是
数据手套	可穿戴传感器	三维	较低	高	较高	是
TapID ^[18]	可穿戴传感器	二维	高	较高	低	是
HMD 手柄	控制器	三维	较高	低	低	是
VR 智能手表 ^[20]	控制器	三维	较高	低	低	是
VR 物理键盘 ^[21]	控制器	二维	高	低	低	是
HyFinBall ^[22]	控制器	三维	较高	较低	低	是
眼动仪	生理信号	三维	较高	高	较高	是
MASI-VR ^[26]	生理信号	三维	较低	高	低	是

裸手输入技术通常基于计算机视觉技术，采用低成本的三维相机捕获用户肢体特别是手部的运动图像，在虚拟环境中生成现实肢体的化身，最终以裸手的形式实现三维空间中的交互行为。常见的裸手交互设备有 Kinect 深度相机^[15]、Leap Motion 传感器^[16]等。目前也有一些裸手输入技术不借助外部设备，直接利用头戴式显示器摄像头完成裸手，例如 Meta Oculus 团队的 MegaTrack^[17]手势交互系统。这些裸手输入技术直接以人的肢体运动作为输入通道，符合人在现实世界的行为习惯，可以有效促进虚拟环境和现实环境融合，是应用最为广泛的输入技术之一。然而，现有裸手输入技术常需要大范围的肢体运动，受制于人有限的活动范围和精力，无法实现快速、低疲劳的交互。

可穿戴传感器输入技术利用可穿戴传感器监测用户肢体运动情况，来实现用户和虚拟环境之间的交互。传统的可穿戴传感设备有数据手套、力反馈手套等，近年来新型传感器的出现带动了新的传感交互设备诞生，例如 Meier 等^[18]开发的基于惯性传感器的 TapID 智能手环。这些基于可穿戴传感器的交互设备具有精度高、识别准确的优点，然而存在较强的侵入性，并且校准过程繁琐。

控制器输入技术类似于传统的键鼠输入，通过六自由度的控制器实现输入动作。一般头戴式显示器都配备有手柄控制器，这类控制器通常采用射线隐喻^[19]的方式，通过控制器发出射线与虚拟物体进行接触实现交互行为。除此之外，TickTockRoy^[20]交互技术使用智能手表作为控制器投射射线来完成交互；Knierim 等^[21]使用物理键盘为控制器实现 VR 高速打字；Cho 等^[22]使用按钮球作为控制器实现多模式输入。这些控制器有效降低了交互疲劳问题，提升交互速度，然而由于自身物理限制（例如有限的控制器按键），控制器输入技术不够自由，可扩展性不佳。

生理信号输入技术通过监测人的生理信号变化来实现交互行为，这类生理信号包括眼球运动^[23]、语音信息^[24]、肌电信号^[25]、脑电信号^[26]等。其中，眼控技术基于眼球运动的生理信号来实现交互行为，它具有非侵入、快速、直观的优点，也存在视疲劳、米达斯接触^[27]等问题。影响生理信号输入技术精度的因素很多，因此，这类输入技术主要作为其他通道输入的补充。



(a) LeapMotion 传感器



(b) Noitom 动捕手套



(c) Meta Oculus VR 手柄



(d) Pupil-lab 眼动模块

图 1-3 常用虚拟现实输入设备

图片来源：(a): <https://www.ultraleap.com/>

(b): <https://www.noitom.com.cn/>

(c): <https://www.oculus.com/>

(d): <https://pupil-labs.com/>

（2）虚拟现实的输出环节

虚拟现实常采用多感官、多通道信息输出激发用户对沉浸式虚拟环境的认知与感知。传统虚拟现实输出技术常依赖视觉听觉双通道完成信息输出：在视觉通道上，头戴式显示器利用两个凸透镜分别模拟人的左右眼结构，通过左右两个屏幕分别显示左右眼图像而从视觉上呈现给用户一个立体的三维虚拟世界；在听觉通道上，头戴式显示器自带的声音显示器与其他耳机为用户输出虚拟环境的立体声音。近年来有越来越多的研究者开始探寻虚拟现实触觉通道的反馈输出：Martinez 等^[28]采用聚集超声波远程刺激用户的手部；Born 等^[29]使用可穿戴触觉手套为视障人士提供触觉反馈。然而，这些触觉反馈解决方案成本远高于视觉、听觉输出的技术成本，仍然缺少低成本的触觉反馈输出技术。

综上所述，虚拟现实现有人机交互方式已经繁多。在输入方式方面，已经存在基于裸手、可穿戴传感器、控制器、生理信号等多种输入媒介的交互方式，然而现有输入方式仍无法兼顾交互自然性、交互高效率与交互低疲劳；在输出方式方面，虚拟现实已经实现视听触多通道的信息输出，然而目前的触觉通道反馈技术成本仍高。

1.2.2 自然手势交互技术研究现状

基于前一小节对虚拟现实交互技术的整理，裸手输入技术凭借其非侵入性、高自由度的特点，最大程度保证了虚拟现实交互过程的沉浸感，本小节将对基于裸手输入的自然手势交互技术进行更细致的调研。广义上，任何不借助中间设备、由人的肢体运动直接交互的交互技术都可以被称为自然手势交互技术。自然手势交互技术是虚拟现实中应用最为广泛的交互技术，它以人最自然、最贴近现实的肢体运动作为交互方式，能够最大程度地保持虚拟现实交互过程中的沉浸感。依据手势获取方式，自然手势交互技术可以分为非接触式手势交互技术和接触式手势交互技术。

（1）非接触式手势交互技术

非接触式手势交互技术以空中手势交互技术为典型，主要通过计算机视觉技术分析用户肢体图像来实现手势交互，前文中提及的裸手输入技术便属于空中手势交互。由于空中手势交互技术常依赖大幅度的手臂运动，交互疲劳问题随之产生，研究者开始探索低疲劳的空中手势交互改良方法。Lu 等^[30]在桌面虚拟现实手部追踪组合策略中，将手势识别传感器由最开始所在的头戴式显示器上转移到桌面上，减轻头戴式显示器重量以减轻颈部疲劳，同时通过桌面物理支撑来降低交互疲劳；Li 等^[31]研究基于手指和手部动作的 3D 微手势设

计，以避免较大幅度的前臂和上臂手势造成的手臂疲劳；Li 等^[32]提出了基于 Kinect 的手臂向下选择方法——Bracelet，通过手臂向下运动减轻手臂交互产生的“大猩猩手臂综合征”^[33]；Iqbal 等^[34]提出 ProxyHand 和 StickHand 技术，通过引入 3D 空间偏移来减小虚拟现实交互中的手臂疲劳。虽然这些交互方法一定程度上减缓了交互疲劳，交互效率低、缺乏触觉反馈仍然是以空中手势交互为主的非接触式手势交互技术的普遍问题。

（2）接触式手势交互技术

接触式手势交互技术以触摸手势交互技术为典型，主要通过传感器感知用户手指在平面上的触摸状况来实现手势交互，其中最为熟知的便是我们现实生活中最常使用的触控交互。相较于非接触式手势交互技术，接触式手势交互技术被探索得更早，且已经广泛应用于二维用户界面交互中。

在二维空间交互领域，接触式手势交互凭借其逻辑简单、交互方便的特点而得到了广泛的应用。Qing 等^[35]针对个人便携式电脑触控手势交互设计进行探索，收集用户使用最频繁的手势集并进行数据分析，将触摸手势分为三类：基本手势、符号手势和组合手势；凌云等^[36]对触摸手势在移动设备地图可视化系统方面展开调研，总结了移动设备上单点触摸和多点触摸基本交互手势，同时指出基于触摸的手势交互因常用手势表示多种语义并且手势数量过少而存在局限性；Suto 等^[37]将红外图像处理应用于多点触摸桌面系统，同时指出红外多点触摸存在遮挡问题同时容易受到阳光中的红外光线干扰。

在三维空间交互领域，不同于 3DOF 的二维操纵，接触式手势交互技术在 6DOF 的三维物体操纵及与三维空间的交互情形中存在诸多问题，例如：二维平面的交互信息由于缺少深度信息无法控制三维空间、较少的触摸手势数量也难以执行复杂的三维操纵。针对这些问题，研究者对于接触式交互技术实现三维操纵方向的探索越来越多。Martinet 等^[38]介绍“Z 技术”来提供更沉浸感的体验，“Z 技术”利用两根手指触摸，第二根手指向前向后移动来控制深度信息变化；Hancock 等^[39]介绍了一种叫 Shadow-depth 的三点触控交互技术，该技术可实现三维对象在 5 个自由度方向上的平移、旋转；一种名为 tBox^[40]基于触摸的交互技术出现，该技术将用户的平移、旋转、缩放的操作分离开来，使得一根手指就可以完成对三维对象的操纵；Liu 等^[41]开展了基于小型触摸显示屏的触摸交互研究，用两根手指手势实现在小型触摸显示器上的六自由度三维物体操纵，并且改进了 Hancock^[39]的方法，采用单手两根手指实现三维物体的平移旋转（包含三维物体操纵的 4DOF），余下 2DOF（控制两个方向旋转）则采用一根手指移动、另一根手指固定（1f-1m）的方法来实现。基于触

摸的交互在手持性移动增强现实中得到了广泛应用，Goh E 等^[5]全面总结了基于触摸的手势交互在手持性移动增强现实中的应用，并指出基于触摸的技术在手持性移动增强现实环境中由于视野遮挡而存在“胖手指问题”。

综上所述，空中手势交互技术仍缺少提高交互方式可用性的解决方案，而接触式手势交互技术以手指在物理平面小幅度移动为主要交互形式，提高交互效率，降低交互疲劳，可以有效弥补非接触式手势交互技术的缺陷。

1.3 研究内容及组织框架

1.3.1 研究内容

本文基于虚拟现实中的坐姿作业环境，研发一种低疲劳、高效率、强灵活、多模态、适合长时间使用的桌面手势交互软硬件系统。其具体内容包括：

- 1) 调研本课题的国内外研究现状，确定课题研究方法。介绍现有的虚拟现实输入设备并分析其各自特点，整理降低交互疲劳的相关技术，综述桌面手势交互的特点与研究现状，综述基于触摸的手势交互方法，并最终确立本课题的研究方法。
- 2) 基于虚拟现实的桌面手势交互硬件系统设计。基于红外激光投影相机原理，设计一款捕获手指触摸状态的桌面手势交互设备，包含设备人性化尺寸设计、设备硬件参数选型、设备电路结构设计、设备外观造型设计、设备实物制作。
- 3) 基于虚拟现实的桌面手势交互软件系统设计。基于上述桌面手势交互硬件系统，利用计算机视觉技术对桌面手势交互图像进行图像处理，获取桌面手势交互点信息；对相机图像畸变进行校正，更新校正后的桌面手势交互点信息；通过对桌面手势交互点信息进行分析，完成桌面手势识别并完成与虚拟现实系统的手势交互环节，建立“桌面手势交互硬件系统输入-虚拟现实交互系统响应”的交互单元。
- 4) 基于虚拟现实的桌面手势交互功能实现与展示。基于上述桌面手势交互软硬件系统，根据交互任务对桌面手势进行功能设计；最终通过开发桌面手势交互展示系统，设计典型虚拟现实人机交互界面，将本课题的系统投入应用实践。

1.3.2 组织框架

本文共分为六章，各章节组织框架如图 1-4 所示。

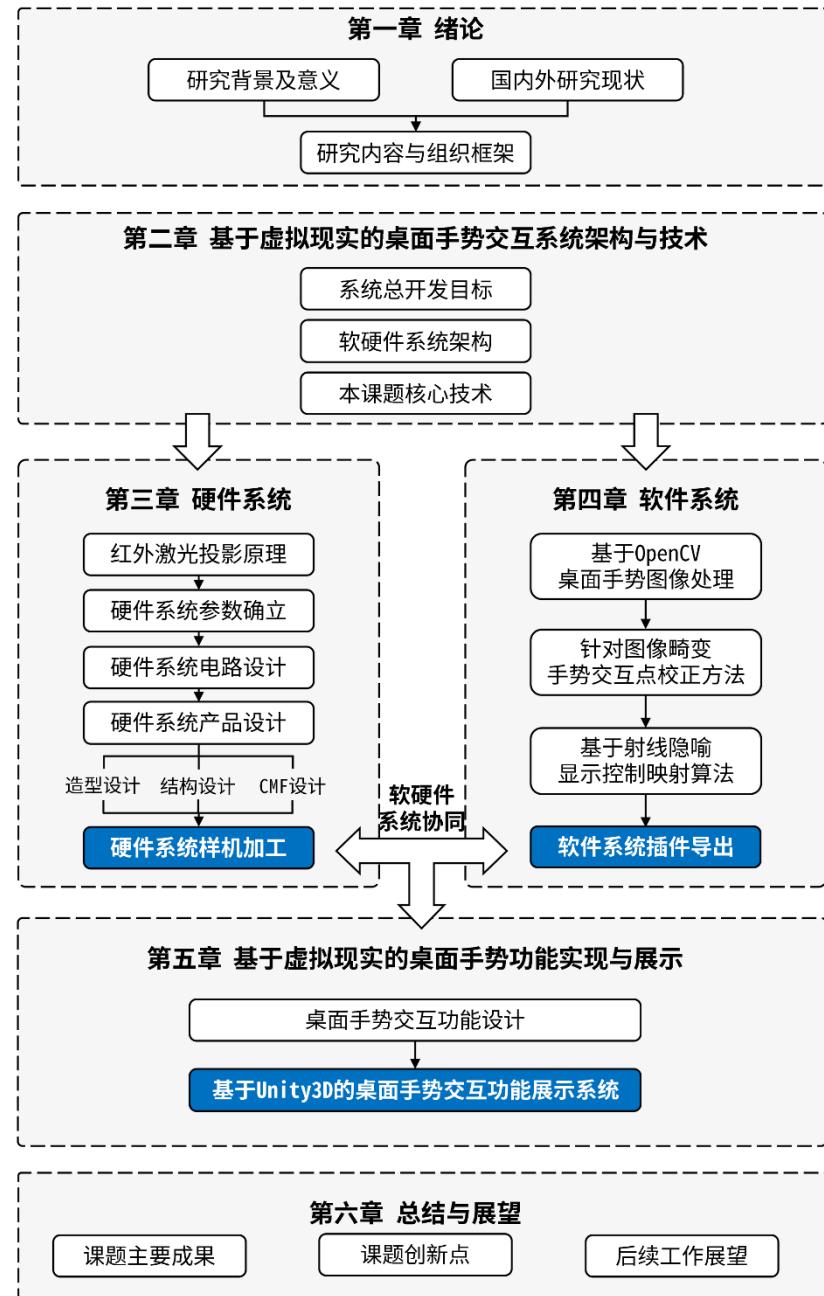


图 1-4 文章组织框架

第一章：绪论。本章阐释虚拟现实桌面手势交互系统的课题背景与意义，综述现有的虚拟现实交互技术与空中手势交互技术的国内外研究现状，引出本课题主要研究内容。

第二章：虚拟现实桌面手势交互系统总述。本章介绍了软硬件系统的开发目标与整体架构，总结了本课题软硬件系统所应用的核心技术。

第三章：虚拟现实桌面手势交互硬件系统设计。本章通过硬件模块选型、人因实验确立尺寸、电路设计、产品设计、生产加工等过程，完成硬件系统的设计与制造。

第四章：虚拟现实桌面手势交互软件系统开发。本章通过开发桌面手势识别算法、相机镜头校正方法、虚拟现实系统交互算法，完成桌面手势交互的基本交互单元。

第五章：虚拟现实桌面手势交互功能实现与展示。基于上述硬件系统与软件系统，本章对桌面手势功能进行设计和实现，并搭建桌面手势交互展示系统，展示桌面手势交互系统在虚拟现实中的具体应用效果。

第六章：总结与展望。本章总结桌面手势交互系统的研究成果与创新点，并总结目前存在的不足，指明接下来的研究思路。

第二章 基于虚拟现实的桌面手势交互系统架构与技术

传统空中手势交互技术存在显著的交互疲劳问题，为了解决这一痛点，本课题提出借助手指在桌面的触摸来降低交互疲劳的桌面手势交互方案。由于缺少合适的桌面手势识别设备，本课题必须自主搭建桌面手势交互硬件系统，并与软件系统建立稳定联系，最终实现与虚拟现实环境的交互功能。本章节将总述本课题的开发目标，展示桌面手势交互软硬件系统的总体架构，介绍本系统所涉及的关键技术。

2.1 基于虚拟现实的桌面手势交互系统开发目标

本课题面向虚拟现实中的坐姿作业环境，最终目标交互场景如图 2-1 所示。用户佩戴头戴式显示器坐在桌前，桌面上放置有便携性桌面手势交互硬件设备，用户通过手指在桌面上做出特定的桌面手势，完成与虚拟现实系统的交互任务。本系统有以下几点开发目标：

- (1) 设计桌面手势交互硬件系统，要求能够实时检测用户在桌面的手指触摸图像。硬件系统尺寸应尽可能小，易用性强，校正方便，有良好的便携性。
- (2) 开发桌面手势交互软件系统，要求能够实时识别用户在桌面上手指滑移、点击等桌面手势，建立桌面手势交互区域与虚拟环境的控制显示映射，实现“桌面手势输入-交互系统响应”的基本单元。软件系统应保证手势识别准确，兼容空中手势交互技术。
- (3) 设计虚拟现实人机交互界面，要求能够展示桌面手势交互系统的实际应用情况。界面设计应符合虚拟现实人机界面设计规范，满足人在三维空间的生理行为习惯。



图 2-1 桌面手势交互系统的概念图

2.2 基于虚拟现实的桌面手势交互的系统组成

基于上述开发目标，本课题桌面手势交互系统包含桌面手势交互硬件系统与桌面手势交互软件系统两大部分，软硬件系统总体架构如图 2-2 所示。

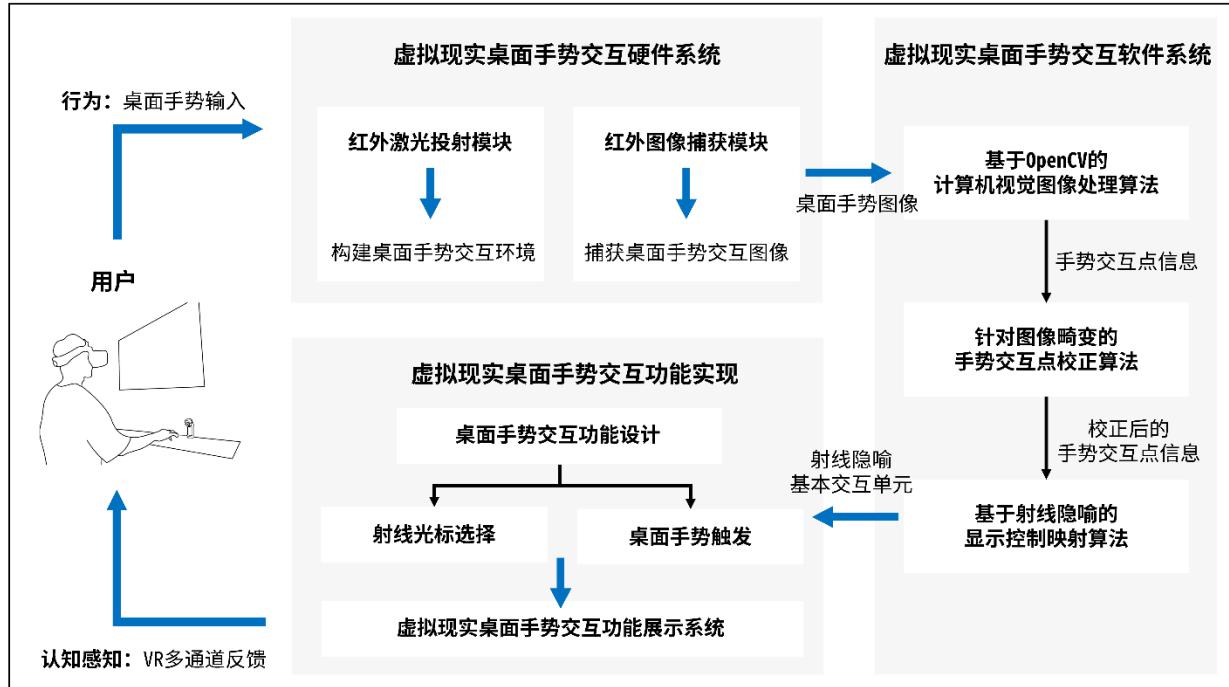


图 2-2 桌面手势交互软硬件系统架构

用户产生桌面手势输入行为时，虚拟现实桌面手势交互硬件系统利用红外激光投影原理，通过红外激光投射模块构建带有水平红外光层的桌面手势交互环境，通过红外图像捕获模块捕获用户手指在桌面触摸的桌面手势交互图像，并将图像传输给软件系统。

虚拟现实桌面手势交互软件系统接收硬件系统捕获的桌面手势图像，通过目前广泛采用的计算机视觉库 OpenCV 对桌面手势图像进行处理，提取桌面手指触摸的手势交互点。由于图像受成像原理及红外图像捕获模块倾斜等因素影响而存在畸变，软件系统需要对提取出的桌面手势交互点进行校正，保证手势交互点与用户手指在现实情况一一对应。之后，通过对桌面手势交互点的分析，建立与虚拟环境的映射关系，完成“桌面手势输入-虚拟环境响应”的基本交互单元。

软件系统的桌面手势控制显示映射结果最终作用于虚拟现实环境，本文以一个由 Unity 3D 引擎搭建的虚拟现实人机交互界面为例，通过对桌面手势的判别，完成虚拟环境中的射线光标移动或者一些交互控件的响应等交互任务，这些交互任务的结果最终通过视觉、听觉、触觉等通道又反馈给用户，引导用户产生对虚拟环境的认知与感知。

2.3 基于虚拟现实的桌面手势交互系统的关键技术

根据上述的系统架构，本课题的关键技术可以总结为：硬件系统的红外激光投影传感技术、软件系统的计算机图像处理技术、针对图像畸变的手势交互点校正技术、桌面手势显控耦合技术，见图 2-3。以下将分别介绍这些技术的基本原理。

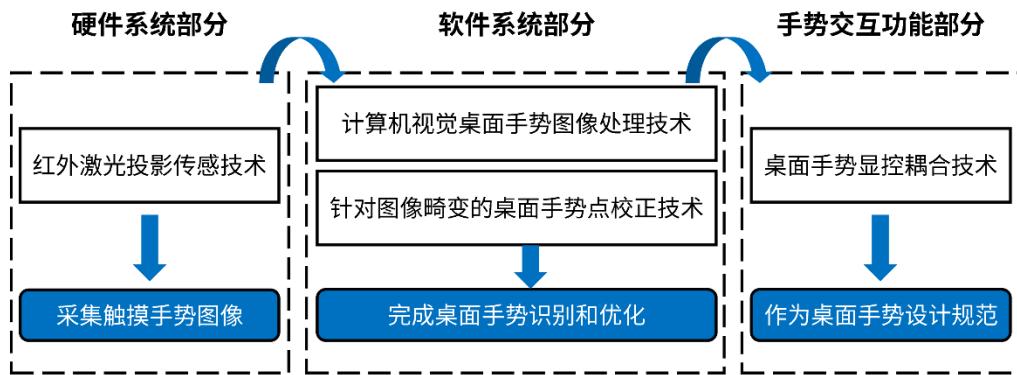


图 2-3 本课题的关键技术

2.3.1 红外激光投影传感技术

红外激光投影传感技术最早可追溯到 1992 年由 IBM 工程师发明的光学虚拟键盘，凭借其低成本、便携性、可靠性等优点，如今激光投影已经被广泛应用于人机交互中。红外激光投影传感技术的组成原理如图 2-4 所示，由红外激光投射模块与红外图像捕获模块组成。红外激光投射模块常采用一字型红外激光头并紧贴桌面放置，一字型光学镜头可以完成红外激光导向功能，使横向光线通过、纵向光线隔离，最终形成平行于桌面的红外激光层。红外图像捕获模块常采用红外摄像机，当用户手指触摸桌面时，手指在触摸处将原本平行桌面的红外光线拦截，红外光由于手指漫反射作用而反射向四面八方，最终反射到红外图像捕获模块而被捕获，在图像中形成一光点，如图 2-5。通过逐帧对光点的分析处理，即可实现实时辨别手指在桌面的触摸情况。

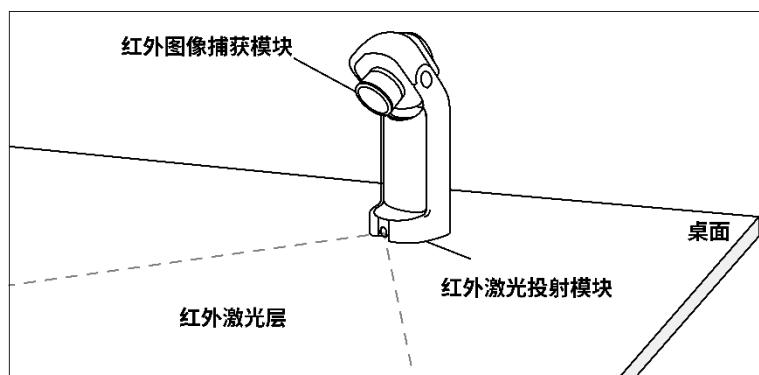


图 2-4 红外激光投影传感技术原理图



图 2-5 图像中的手指光点

2.3.2 计算机视觉桌面手势图像处理技术

数字图像是计算机视觉技术的基础，它们由照相机捕获，经过图像传感器将光信号转换成计算机能够理解的数字信号。摄像机捕获到的数字图像由一个个的像素组成，它们以矩阵的形式储存在计算机中，每个像素都携带着一定量的图像信息，例如图像的色彩、亮度等。通过对一张图像的全部像素进行全局或局部的扫描、采样、重写等操作，可以轻松实现对于图像的处理与兴趣点特征提取和分离。由于计算机视觉技术涉及范围非常广，本小节将主要介绍本课题所应用的图像处理方法。

(1) 计算机图像的预处理

计算机图像的预处理^[42-43]是图像处理特征提取的常见步骤，可以将其理解为复杂数据集的归一化，预处理通过对摄像头捕获的原始图像进行全局处理，使得图像变得更有利于后续的图像分析和特征提取。常见的预处理方法有灰度形态学与二值形态学。

灰度形态学方法将彩色图像(RGB三通道)变为仅表征图像亮度的灰度图像(单通道)。灰度图像的灰度值量化为256个等级，从0到255依次表示由黑到白。目前图像灰度化的方法有三种：最值法、平均值法、加权均值法。

在最值法中，图像每个像素的灰度值为原图像RGB通道最大值或最小值，即

$$GRAY = \max(R, G, B) \quad (\text{公式 2.1})$$

$$GRAY = \min(R, G, B) \quad (\text{公式 2.2})$$

在平均值法中，图像每个像素的灰度值为原图像RGB通道的平均值，即

$$GRAY = \frac{R+G+B}{3} \quad (\text{公式 2.3})$$

在加权均值法中，图像每个像素的灰度值为原图像 RGB 通道的加权均值，即

$$GRAY = \frac{k_R R + k_G G + k_B B}{k_R + k_G + k_B} \quad (\text{公式 2.4})$$

由于人对于红绿蓝颜色的敏感程度不同，对绿色最为敏感，对蓝色最不敏感。常用的加权均值模型为：

$$GRAY = 0.3R + 0.59G + 0.11B \quad (\text{公式 2.5})$$

二值形态学方法在灰度形态学的基础上，通过阈值分割灰度图像，使图像达到非黑即白的效果。一般的图像二值化方法设定阈值 T ，当灰度值 $GRAY > T$ ，灰度值置为 255（白色）；当灰度值 $GRAY < T$ ，灰度值置为 0（黑色）。此外，也有自适应阈值的方法，将该点像素的灰度值与周围像素平均值进行比较来完成灰度图像二值化，常用的有 4 连通域自适应阈值与 8 连通域自适应阈值。

（2）图像差值法

本课题要求实时检测桌面图像变化，并且桌面环境较为复杂，可能存在各种各样的杂物遮挡，图像差值法^[42-43]可以有效去除固定不动的桌面背景，该方法已经被广泛应用于固定摄像机的视频分析中。图像差值法通过计算当前帧图像与初始背景图像之间的差值，检测两图像中发生变化区域，再结合变化区域的时序信息，可以有效提取动态目标点的运动情况。

图像差值法的计算方法为两张图像各像素点的各个通道值做差，这要求两张图像需要相同大小且通道数一致。由于彩色图像差值法复杂度较高，所得结果不利于目标提取，故常采用彩色图像灰度化再进行图像差值，减小运算量并增强结果对比性。图像差值法实现简单，不容易受到环境光线变化影响，是静态背景目标检测的极佳方案。

2.3.3 针对图像畸变的桌面手势交互点校正技术

根据上述计算机视觉图像处理流程，桌面图像下的手指轮廓中心点原本就可以表征手指在桌面上的具体位置。然而，由于硬件系统红外图像捕获模块的相机倾斜、相机广角较大等原因，捕获到的桌面手势交互图像是存在畸变的，即手指在图像中的位置并不能准确表征手指在桌面上的实际方位。因此，针对图像畸变对桌面手势交互点进行校正是非常必要的，对于提高手势交互系统中手势识别准确性有重要意义。

图像畸变校正过程与相机成像原理密切相关。如图 2-6 所示，在红外图像捕获模块捕获到手指的过程中，手指的位置经过世界坐标系 $O_W - X_W Y_W Z_W$ 、相机坐标系 $O_C - X_C Y_C Z_C$ 、

图像坐标系 $O_1 - xy$ 、像素坐标系 $O_0 - uv$ 四个坐标系，最终转换成在桌面手势交互图像中的像素坐标。这四个坐标系的概念如下：

- (1) 世界坐标系 $O_W - X_W Y_W Z_W$ 是表征手指在桌面中实际位置的三维坐标系，该坐标系的坐标轴与坐标原点常根据实际情况变化，本课题取硬件系统红外激光投射模块的激光发射点为坐标原点，桌面水平方向为 x 轴，桌面竖直方向为 y 轴，垂直桌面的方向为 z 轴，单位为cm；
- (2) 相机坐标系 $O_C - X_C Y_C Z_C$ 是表征手指相对于相机相对位置的三维坐标系，以相机光心为坐标原点，以相机镜头光轴为 z 轴，以相对相机镜头水平方向为 x 轴，以相对相机镜头竖直方向为 y 轴，单位为cm；
- (3) 图像坐标系 $O_1 - xy$ 是表征相机成像平面上图像实际物理距离的二维坐标系，以桌面手势交互图像中心为坐标原点，图像水平方向为 x 轴，图像垂直方向为 y 轴，单位为cm；
- (4) 像素坐标系 $O_0 - uv$ 是以像素为单位表征计算机图像的二维坐标系，以桌面手势交互图像左上角为坐标原点，水平方向为 x 轴，垂直方向为 y 轴，单位为px。

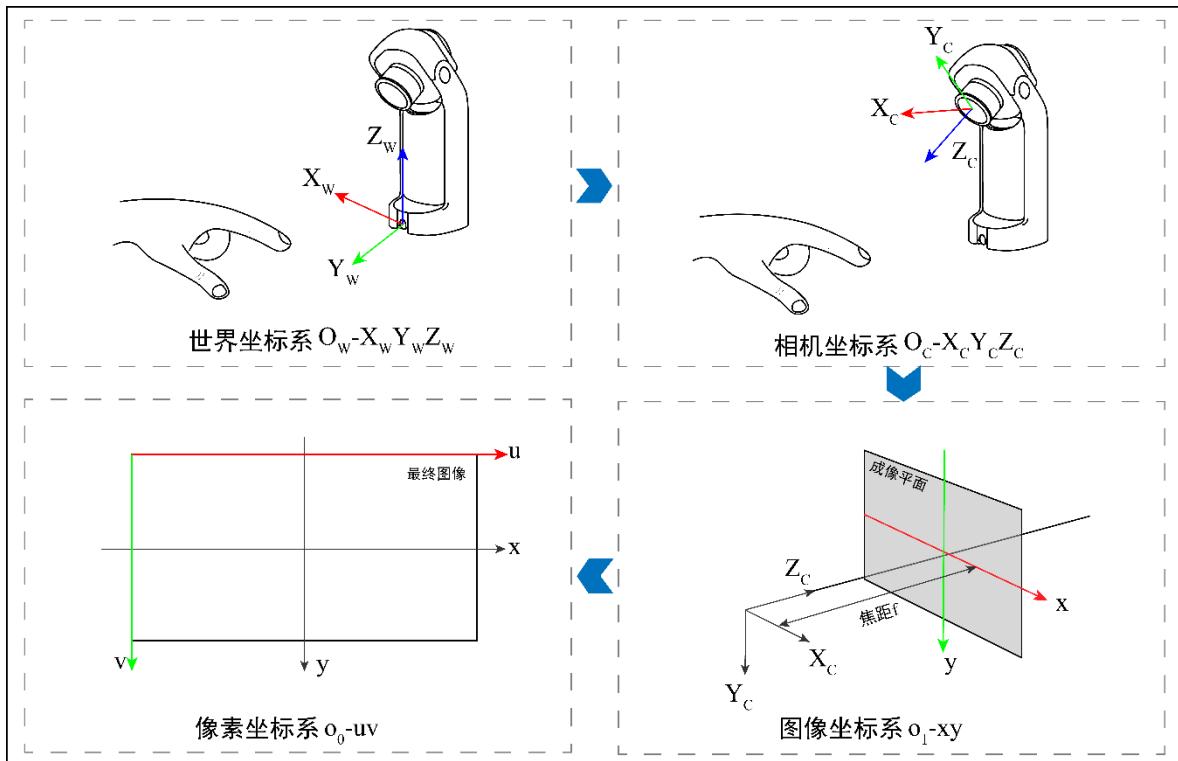


图 2-6 相机捕获手指成像过程坐标系转换

从用户手指触摸桌面产生手势交互点并被相机捕获、再到最终形成计算机图像整个坐标系转换过程及转换关系总结如下：

(1) 手势交互点世界坐标转换为相机坐标

用户手指与桌面触摸过程始终在一个平面进行，故桌面手势交互点世界坐标 $Z_W = 0$ 。桌面手势交互点世界坐标 $(X_W, Y_W, 0)$ 转到相对于相机的相机坐标 (X_C, Y_C, Z_C) 的转换关系为：

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = R \begin{bmatrix} X_W \\ Y_W \\ 0 \end{bmatrix} + T \quad (\text{公式 2.6})$$

其中， R 为三维坐标转换过程的旋转矩阵：

$$R = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \quad (\text{公式 2.7})$$

T 为三维坐标转换过程的平移矩阵：

$$T = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix} \quad (\text{公式 2.8})$$

(2) 手势交互点相机坐标转换为图像坐标

桌面手势交互点相机坐标 (X_C, Y_C, Z_C) 转换为图像坐标 (x, y) 基于小孔成像原理，二者之间转换关系为：

$$Z_C \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \quad (\text{公式 2.9})$$

其中， f 为相机焦距。

(3) 手势交互点图像坐标转换为图像坐标

桌面手势交互点图像坐标 (x, y) 迁移到像素坐标 (u, v) 的转换关系为：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & u_0 \\ 0 & k_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (\text{公式 2.10})$$

其中， k_x 、 k_y 为桌面手势交互图像的物理单位 mm 转换成像素单位 px 的比例因子， (u_0, v_0) 为桌面手势交互图像中心点的像素坐标。

(4) 手势交互点由实际位置转换为像素位置总转换过程

联立公式 2.6、2.9、2.10 可得桌面手势交互点世界坐标 $(X_W, Y_W, 0)$ 到像素坐标 (u, v) 的总转换关系：

$$\begin{aligned}
 Z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \begin{bmatrix} k_x & 0 & u_0 \\ 0 & k_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ 0 \\ 1 \end{bmatrix} \\
 &= \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ 0 \\ 1 \end{bmatrix}
 \end{aligned} \quad (\text{公式 2.11})$$

其中， $\begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ 为相机内参，是造成桌面手势交互图像畸变的内部因素，与摄像头广角、焦距等因素有关，可以通过棋盘格标定法求解； $\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$ 为相机外参，是造成桌面手势交互图像畸变的外部因素，与摄像头摆放位置、倾斜角度等因素有关，可以直接通过计算或利用 OpenCV 库提供的 solvePnP() 函数测量求解； Z_C 可通过公式 2.6 求解为 $r_{20}X_W + r_{21}Y_W + t_2$ ，仅与 X_W 、 Y_W 、旋转矩阵 R 和平移矩阵 T 有关。

综上所述，通过求解相机内参与相机外参，可以建立起桌面手势交互点在桌面上的实际位置 $(X_W, Y_W, 0)$ 与在图像像素中的位置 (u, v) 间的映射关系。

2.3.4 桌面手势显控耦合技术

桌面手势显控耦合技术旨在实现桌面手势交互控制与虚拟现实三维人机交互界面显示的相容性。相容性^[14]是人机交互界面显示控制研究的核心概念，一个交互系统的控制显示相容性越好意味着该交互系统拥有更高的交互绩效、更快的响应时间、更低的生理负荷。在人机交互界面研究中，相容性被分为空间相容性、运动相容性和概念相容性三部分^[44-45]。

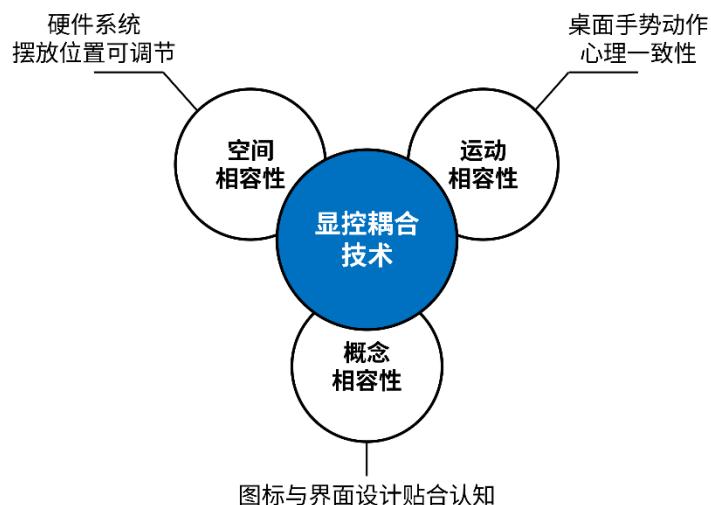


图 2-7 桌面手势显控耦合技术

空间相容性是指显示器与控制器的空间位置同人的行为习惯之间的心理一致性。在本课题中，为了满足不同优势手用户的交互需求，桌面手势交互硬件应设计为便携式可调节系统，以保证交互设备可以根据用户优势手被放置在身体左侧和身体右侧。

运动相容性是指显示器与控制器的运动位置同人的心理预期之间的心理一致性。在桌面手势功能设计过程中，应保证用户做出的手势交互动作与用户期望的显示结果，例如：用户手指滑移方向应与系统显示的光标方向重合，用户手指旋转方向应与交互目标旋转方向相一致。

概念相容性是指显示系统的信息编码符合与人的认知习惯之间的心理一致性。在桌面手势交互展示系统设计中，应保证人机交互界面各类图标的设计风格、颜色样式等设计要素贴合人的认知习惯，能够让用户仅通过观察就知道该图标的表示含义。

系统设计开发过程对上述三种相容性进行充分考量，从而提高用户的交互体验。

2.4 本章小结

本章节阐释了课题系统的开发目标与目标应用场景，对于桌面手势交互软硬件系统从系统架构层面进行了梳理。之后，从技术原理层面，依次介绍了系统设计开发过程中所采用的红外激光投影传感技术、计算机视觉桌面手势图像处理技术、针对图像畸变的桌面手势交互点校正技术和桌面手势显控耦合技术等核心技术。

第三章 基于虚拟现实的桌面手势交互硬件系统

良好的交互技术离不开侵入性低、准确性好、可用性强、性价比高的交互设备。一方面，由于桌面手势交互环境非常复杂，在复杂桌面环境精确提取出用户桌面手势显得尤为重要；另一方面，传统交互设备大多存在侵入性强、难以携带、适配困难等问题，用户交互体验需要有舒适、可靠、便携、易用的硬件系统设计作为支撑。本章节将详细阐释虚拟现实桌面手势交互硬件系统的设计历程，包括硬件参数确立、硬件电路设计、硬件产品设计的全过程。

3.1 硬件系统架构

本课题的桌面手势交互硬件系统采用第二章描述的红外激光投影传感技术，通过红外激光投射模块构建桌面手势交互红外环境，通过红外图像捕获模块捕捉桌面手势交互图像，用于后续软件系统分析。本课题的硬件系统架构如下所示：

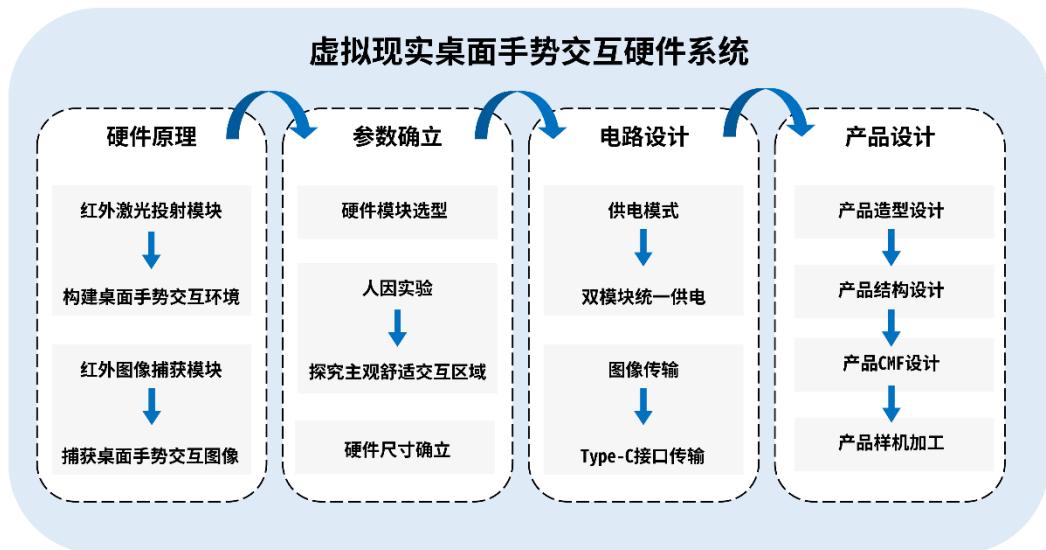


图 3-1 虚拟现实桌面手势交互硬件系统架构

硬件原理部分已在上一章进行了详细介绍。基于硬件原理，需完成相关参数确立工作。首先需确定红外激光投射模块和红外图像捕获模块的选型与参数。其次，考虑到不同用户主观上感受舒适的交互区域各不相同，本课题设定人因实验探究不同用户的主观舒适交互区域，以保证硬件系统能满足绝大多数人的交互需求。最后，为了保证设备足够小、足够便捷，需对各模块的相对位置和整个硬件系统的尺寸进行设计。

完成参数确立后，需要考虑到虚拟现实桌面手势交互系统的内部电路排布。为了便于

用户使用，本课题选择将两个模块统一供电并利用 Type-C 接口完成图像数据传输。

最终，根据上述尺寸参数与电路设计结果，对整个硬件系统进行产品设计。其中包含产品造型设计、产品结构设计、产品 CMF 设计及产品最后生产加工出样机。

本章接下来将详细讲述硬件系统各部分设计过程。

3.2 硬件系统参数确立

3.2.1 硬件模块选型

硬件模块选型包含红外激光投射模块选型与红外图像捕获模块选型两部分。

(1) 红外激光投射模块选型

红外激光投射模块选型如表 3-1 所示。为了准确捕获用户手指在桌面的触摸状态，红外激光投射模块需保证紧贴桌面，故激光头直径选择市场现有激光头的最小直径 6mm。红外激光波长采用 980nm 红外长波，红外长波具有强衍射能力，能够显著提高手势识别准确性。激光头功率选择 50mW，以保证红外光线有足够的强度同时减小红外光线对人体的伤害。工作电压选择 3V，降低功耗同时符合计算机的输出电压。发散角度选择 120°，以保证较大的红外光线覆盖范围。

表 3-1 红外激光投射模块选型参数

产品规格	规格参数
激光头镜头	一字型
波长	980nm
输出功率	50mW
发散角度	120°
工作电压	3-5V
激光头直径	6mm
激光头长度	13mm

(2) 红外图像捕获模块选型

红外图像捕获模块选型如表 3-2 所示。考虑到设备便携性，红外图像捕获模块需要在比较低的高度捕获尽可能大的桌面手势交互区域，这要求相机具有较大广角，本系统选择 120° 广角摄像头。为了准确捕获 980nm 红外激光，相机镜头选择添加 980nm 红外滤光片。由于三维虚拟空间渲染需要保持较高帧率来避免用户产生晕动症，故摄像头帧率选为

120Hz，便于后续高帧率的射线光标渲染。工作电压选择 5V 以满足计算机输出电压。

表 3-2 红外图像捕获模块选型参数

产品规格	规格参数
图像传感器	CMOS 传感器
摄像头分辨率	1920×1080 (1080P)
摄像头帧率	120FPS
镜头滤光片	980nm 滤光滤光片
镜头广角	120°
工作电压	5V
输出功率	2W

现有图像传感器有 CMOS 图像传感器和 CCD 图像传感器两大类，二者具有相似的特性。相比于传统 CCD 图像传感器，CMOS 图像传感器将图像传感模块和信号处理模块集成在同一芯片内，具有体积小、功耗低、集成度高、使用简单、价格低廉等诸多优点，故本课题的硬件系统选择 CMOS 图像传感器来完成图像光信号与数字图像信号的转换过程。

3.2.2 基于人主观舒适交互区域的实验研究

为了满足大部分用户对桌面手势的舒适性交互区域需求，本课题通过人因实验来确定桌面手势交互区域。实验目的是探究人在桌面单手交互情形中舒适交互区域大小和位置，实验方法采用主观测量法。本次实验共有 13 位参与者参与（5 位男生、8 位女生），被试身体状况均良好，优势手均为右手，且都拥有虚拟现实相关体验，实验均征得被试同意。

(1) 实验过程

人因实验场景如图 3-2 所示，在人因实验开始之前，主试首先告知参与者本次实验的目的，并测量被试臂长；让被试正坐在实验桌前，测量用户距离桌子的距离。实验正式开始，被试保持先前的坐姿状态，根据主试引导假想自己身处虚拟现实环境、桌面有一可单手触摸交互的手势交互区域，被试需用右手手指在桌面前后左右滑动，感受舒适手势交互区域的大小和位置，并最终指出舒适交互区域的左上、右上、左下、右下四个点。

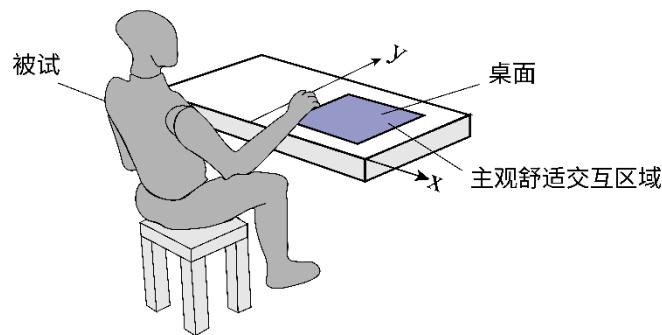


图 3-2 人因实验场景图

(2) 实验结果

在记录用户主观舒适的单手交互区域四个点时，课题选择以桌面下边缘为 x 轴，以用户所坐位置正中心的正前方为 y 轴，以用户正前方与桌面下边缘交点为坐标原点记录数据。实验所记录的用户臂长、坐下位置与主观舒适交互区域如表 3-3 所示。

表 3-3 用户主观舒适交互区域实验探究结果

被试序号	臂长/cm	距离桌面的距离/cm	舒适的桌面单手交互区域/cm			
			左上	左下	右上	右下
1	61	40	(6,17)	(0,0)	(32,16)	(41,0)
2	70	30	(19.5,20.5)	(23,10)	(29.5,20.5)	(38,10)
3	67	40	(1.5,20)	(5.5,4)	(21.5,14.5)	(25.5,2)
4	71	40	(10.5,21)	(10,8)	(30,20)	(29,7)
5	72	25	(4,23)	(6,13)	(33,23)	(33.5,12.5)
6	70	30	(9,22)	(9,9)	(20,21)	(20,9)
7	72	35	(18,23)	(16,12)	(34,24)	(32,11)
8	75	40	(8,19)	(9.5,11.5)	(22.5,19.5)	(21,12)
9	64	33	(11.5,18)	(11,5)	(31,19)	(31,3)
10	68	27	(15,23)	(14,14.5)	(20,20.5)	(20,14.5)
11	66	28	(2.5,13)	(7,8)	(14,17.5)	(20.5,12)
12	69	20	(4,21.5)	(9,10.5)	(39,22)	(30,11)
13	67	26	(2.5,31)	(2,15)	(33.5,27.5)	(30,16.5)

如图 3-3(a)所示，上述被试所指出的舒适桌面单手交互区域以 10% 透明度的四边形可视化表示，区域颜色越深，表示认为该区域舒适的被试数量越多。可以看出，不同被试所认为的主观交互区域位置分布较为分散，硬件系统的便携性与可调节性非常必要。

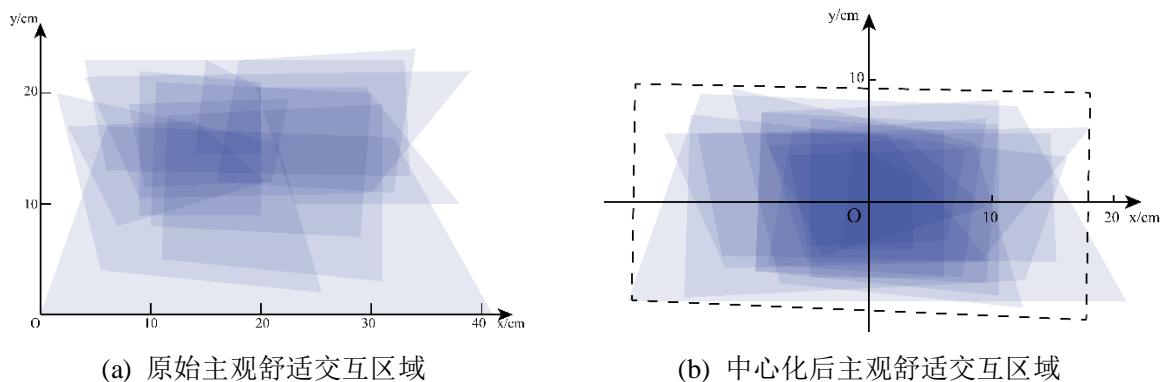


图 3-3 用户主观舒适交互区域可视化图

将这些交互区域的最小外接矩形中心点统一到坐标原点，中心化后的分布情况如图 3-3(b)所示。中心化后被试指出的舒适桌面手势交互区域坐标如表 3-4 所示。

表 3-4 中心化后的用户主观舒适交互区域四个点的坐标值

被试序号	左上点 x 坐标/cm	左上点 y 坐标/cm	右上点 x 坐标/cm	右上点 y 坐标/cm	左下点 x 坐标/cm	左下点 y 坐标/cm	右下点 x 坐标/cm	右下点 y 坐标/cm
1	-15	9	11	8	-21	-8	20	-8
2	-9.75	5.75	0.25	5.75	-6.25	-4.75	8.75	-4.75
3	-12.5	9.5	7.5	5	-8.5	-6.5	11.5	-8.5
4	-10	7.5	9.5	6.5	-10.5	-5.5	8.5	-6.5
5	-15.25	5.75	13.75	5.75	-13.25	-4.25	14.25	-4.75
6	-6	7	5	6	-6	-6	5	-6
7	-7.5	6	8.5	7	-9.5	-5	6.5	-6
8	-7.75	4	6.75	4.5	-6.25	-3.5	5.25	-3
9	-10	7.5	9.5	8.5	-10.5	-5.5	9.5	-7.5
10	-2.5	4.75	2.5	2.25	-3.5	-3.75	2.5	-3.75
11	-9.5	4.75	2	5.25	-5	-4.25	8.5	-0.27
12	-18	5.75	17	6.25	-13	-5.25	8	-4.75
13	-15.85	7.33	15.15	3.83	-16.35	-7.67	11.65	-6.25

表 3-5 柯尔莫戈洛夫-斯米诺夫(V)正态性检验

K-S 检验事项	左上点 x 坐标/cm	左上点 y 坐标/cm	右上点 x 坐标/cm	右上点 y 坐标/cm	左下点 x 坐标/cm	左下点 y 坐标/cm	右下点 x 坐标/cm	右下点 y 坐标/cm
统计	0.182	0.195	0.104	0.119	0.158	0.157	0.168	0.157
自由度	13	13	13	13	13	13	13	13
显著性	0.200	0.200	0.200	0.200	0.200	0.200	0.200	0.200

利用 SPSS 软件对中心化后的主观舒适交互区域四个点的横纵坐标值进行正态性检验，检验结果如表 3-5 所示，各组显著性水平均为 $0.2 > 0.05$ ，满足正态分布条件。取置信度 95%，根据标准正态分布函数表，分布函数

$$F(\mu + 1.65\sigma) = P(x < \mu + 1.65\sigma) = 95\% \quad (\text{公式 3.1})$$

其中， μ 为该组数据均值， σ 为该组数据标准差。中心化后的主观舒适交互区域四个点的横纵坐标值各组数据的均值、数据标准差、置信度 95% 的坐标值如表 3-6 所示。可以看出，由坐标值(-17.7,9.1)、(16.5,8.4)、(-17.8,-7.6)、(16.3,-8.9)围成的桌面手势交互区域可以满足 95% 的用户舒适度需求，其具体大小如图 3.3(b)虚线框所示。

表 3-6 中心化后的用户主观舒适交互区域四个点横纵坐标数据均值与标准差

计算指标	左上点 x 坐标/cm	左上点 y 坐标/cm	右上点 x 坐标/cm	右上点 y 坐标/cm	左下点 x 坐标/cm	左下点 y 坐标/cm	右下点 x 坐标/cm	右下点 y 坐标/cm
均值 μ	-10.7	6.5	8.3	5.7	-10	-5.4	9.2	-5.4
标准差 σ	-4.2	1.6	4.9	1.6	-4.8	-1.3	4.3	-2.1
$\mu + 1.65\sigma$	-17.7	9.1	16.5	8.4	-17.8	-7.6	16.3	-8.9

(3) 实验结论

通过对于用户主观舒适的桌面手势交互区域探究，可以得出结论：在主观舒适交互区域的位置方面，不同用户主观上舒适的交互区域位置较为分散，硬件系统需要具有较强灵活度与可调节性；在主观舒适交互区域的面积方面，95% 用户主观上舒适的交互范围集中在长约 34cm、宽约 17cm 的矩形范围内，硬件系统所能捕获的桌面区域需包含该矩形面积，以满足绝大多数用户的交互需求。

3.2.3 硬件尺寸确立

基于上述人因实验所确定的用户主观舒适的桌面单手交互区域，课题对红外激光投射

模块与红外图像捕获模块的相对位置关系进行规划，在保证红外图像捕获模块能够捕获到主观舒适单手交互范围的前提下尽可能降低模块间距离。两模块的尺寸及相对位置关系如图 3-4 所示，当摄像头倾斜角度与竖直方向夹角为 46° 且两模块高度差 92mm 时，可以覆盖上述人因实验探究所得 95% 用户主观上舒适的交互范围。

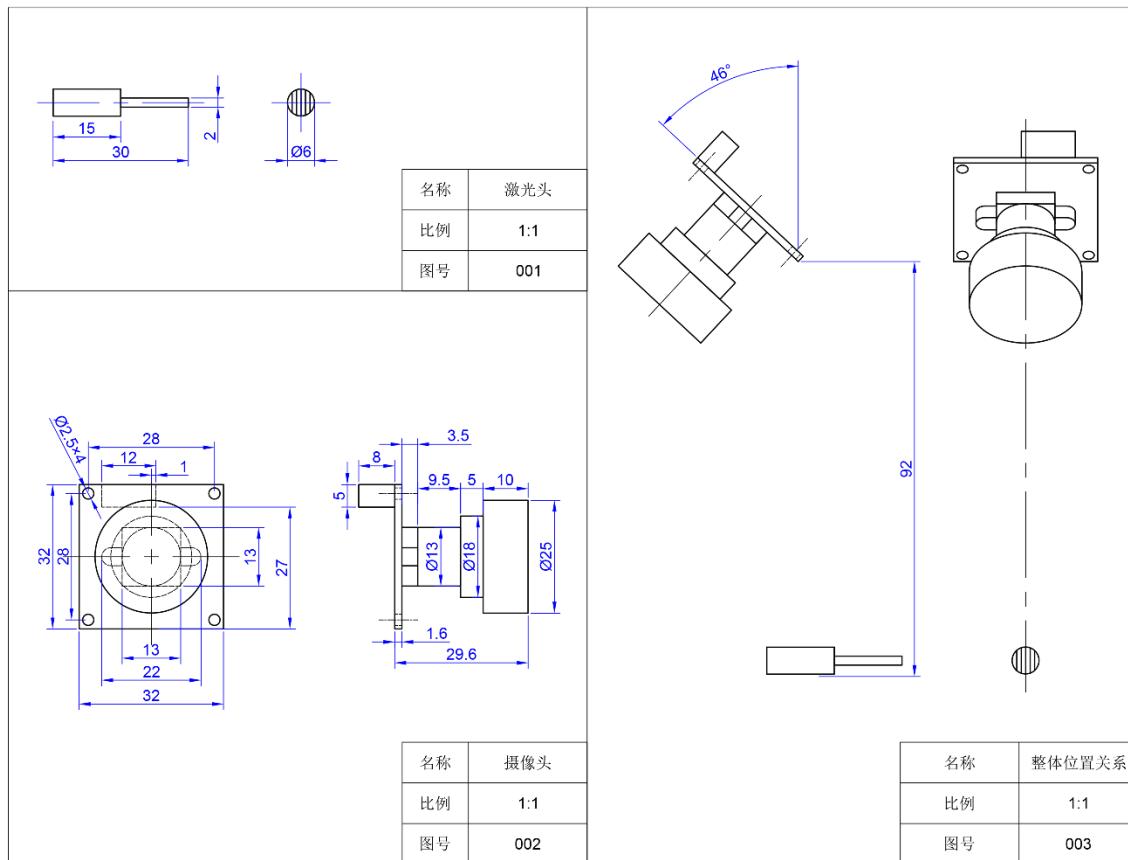


图 3-4 两模块尺寸及相对位置关系

3.3 硬件系统电路设计

为了方便用户使用，本系统电路设计采用一个 Type-C 接口完成两模块供电与图像信号传输，电路原理图如图 3-5 所示。红外激光投影模块的激光头通过降压模块连通到 Type-C 电源口，完成模块供电；红外图像捕获模块的摄像头直接连接 Type-C 电源口同时完成模块供电与桌面手势图像的信号传输。LED 指示灯在 Type-C 电源口接电时亮起，以提示用户硬件当前使用状态，增强用户对产品的信任感。在产品使用方面，用户仅需通过一根 Type-C 数据线将产品与计算机连接起来即可直接使用，省去更换电池的麻烦同时减轻设备重量。

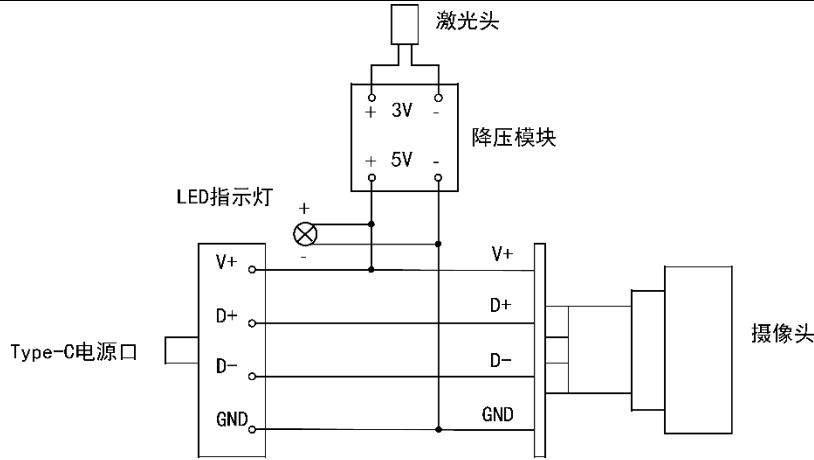


图 3-5 硬件系统电路原理图

3.4 硬件系统产品设计

硬件系统产品设计是硬件系统设计的最终环节，是硬件原理落地、系统体验优化的关键环节。在上述参数确立和电路设计的基础上，课题首先完成硬件系统的概念设计，从概念上确定桌面手势交互硬件系统的外观样式与造型风格；接下来，基于确定下的造型方案，进行硬件系统的结构设计，完成硬件系统的红外激光投射模块、红外图像捕获模块和电路结构的布局与固定，并基于镜头可调节性的需求设计红外图像捕获模块的摩擦自锁结构；在此之后，进行硬件系统的 CMF 设计，对硬件系统各部件的材料、加工工艺进行细化；最终将硬件系统产品设计方案交给加工方，加工桌面手势交互硬件样机。

3.4.1 硬件系统产品造型设计

桌面手势交互硬件系统采用 Rhino 三维造型软件进行曲面建模，通过 Keyshot 渲染后的产品造型设计如图 3-6 所示。在产品配色方面，硬件系统整体采用黑白配色，给用户以沉稳、简约、现代的视觉感受。在产品平衡感方面，硬件系统的摄像头需倾斜拍摄桌面图像，故采用前端小倒角平面、后端大圆角曲面的半圆形外壳结构，以平衡硬件的重心、防止产品向前倾倒。在视觉效果方面，产品采用前深色、后浅色的带状造型设计，以强调硬件系统的轻薄感；产品设计过程中倒角均为 0.12mm，空间造型 R 角的一致性保证了产品的理性观感；后端圆角设计提高了产品的视觉亲和力。



图 3-6 硬件系统概念设计图

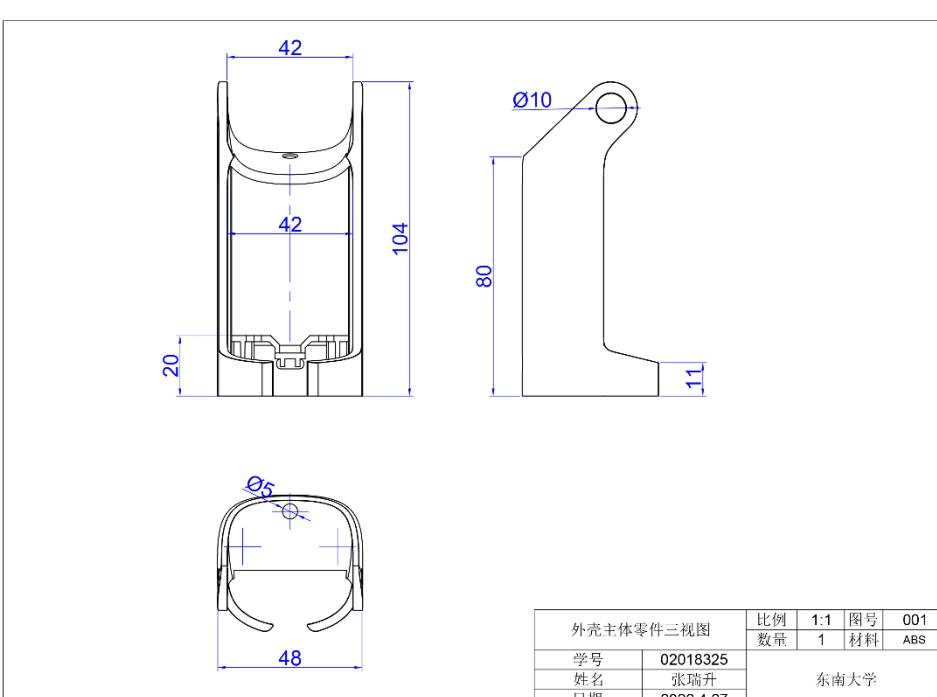
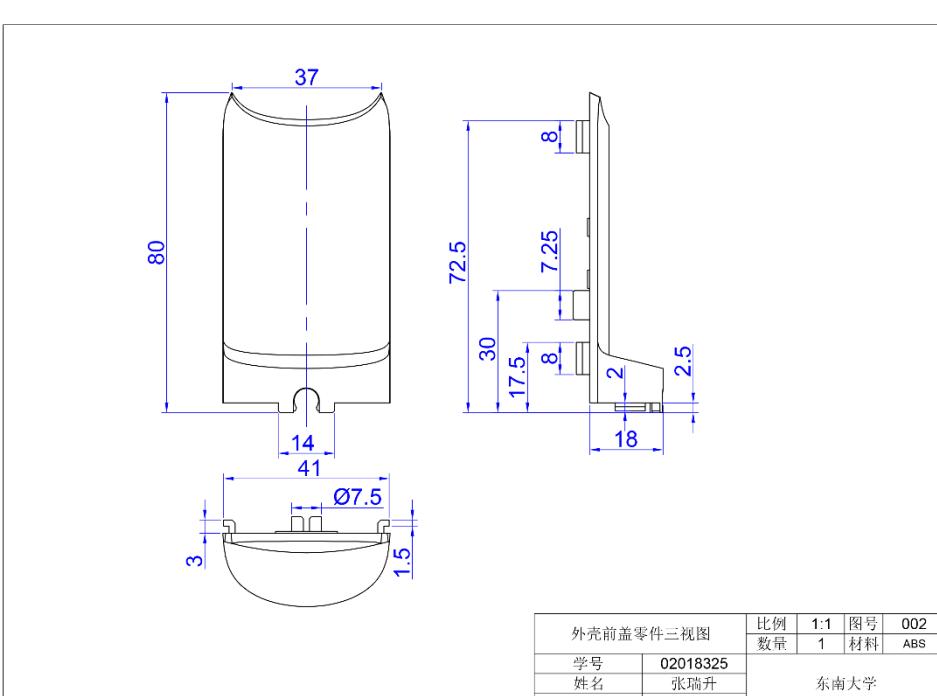
3.4.2 硬件系统产品结构设计

硬件系统产品爆炸图如图 3-7 所示。通过相机镜头前盖和镜头后盖完成红外图像捕获模块的固定与密封，并经由转轴与外壳主体相连。转轴与外壳主体的转轴孔间隙配合，利用摩擦自锁的原理保证摄像头在倾角 0~50° 范围内在无外力作用情况下保持固定。降压模块附着在外壳前盖上，外壳前盖通过滑轨滑入外壳主体，并通过底座将激光头和 Type-C 接口封装至外壳内部。硬件系统主要零件三视图如表 3-7 所示。



图 3-7 硬件系统产品爆炸图

表 3-7 硬件系统主要零件三视图

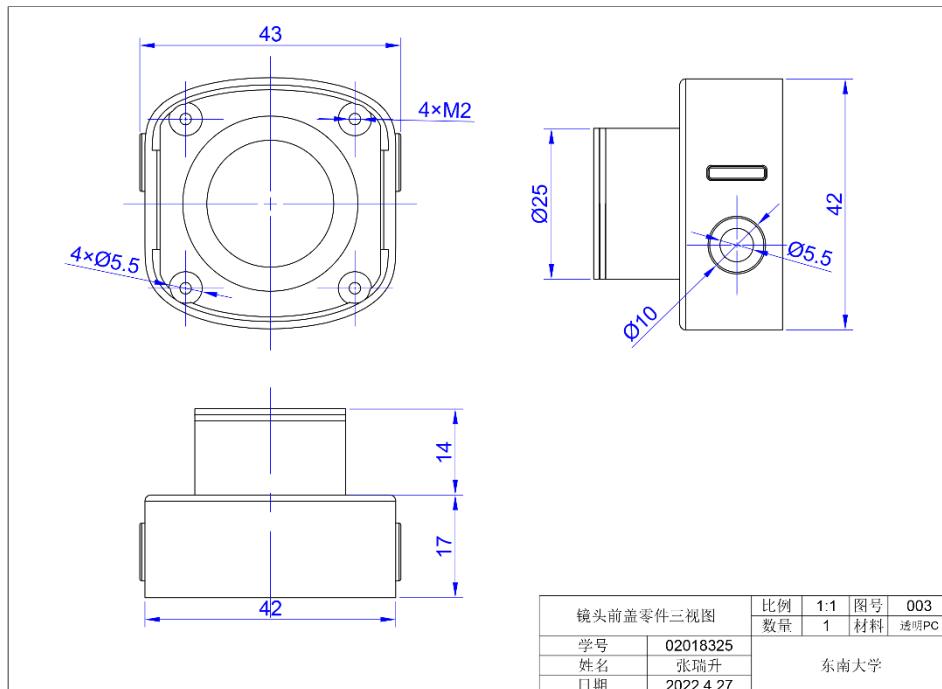
零件名称	零件三视图														
外壳主体	 <table border="1" data-bbox="952 932 1365 1033"> <tr> <td colspan="2">外壳主体零件三视图</td> <td>比例 1:1</td> <td>图号 001</td> </tr> <tr> <td>学号</td> <td>02018325</td> <td>数量 1</td> <td>材料 ABS</td> </tr> <tr> <td>姓名</td> <td>张瑞升</td> <td colspan="2" rowspan="2">东南大学</td> </tr> <tr> <td>日期</td> <td>2022.4.27</td> </tr> </table>	外壳主体零件三视图		比例 1:1	图号 001	学号	02018325	数量 1	材料 ABS	姓名	张瑞升	东南大学		日期	2022.4.27
外壳主体零件三视图		比例 1:1	图号 001												
学号	02018325	数量 1	材料 ABS												
姓名	张瑞升	东南大学													
日期	2022.4.27														
外壳前盖	 <table border="1" data-bbox="952 1729 1365 1819"> <tr> <td colspan="2">外壳前盖零件三视图</td> <td>比例 1:1</td> <td>图号 002</td> </tr> <tr> <td>学号</td> <td>02018325</td> <td>数量 1</td> <td>材料 ABS</td> </tr> <tr> <td>姓名</td> <td>张瑞升</td> <td colspan="2" rowspan="2">东南大学</td> </tr> <tr> <td>日期</td> <td>2022.4.27</td> </tr> </table>	外壳前盖零件三视图		比例 1:1	图号 002	学号	02018325	数量 1	材料 ABS	姓名	张瑞升	东南大学		日期	2022.4.27
外壳前盖零件三视图		比例 1:1	图号 002												
学号	02018325	数量 1	材料 ABS												
姓名	张瑞升	东南大学													
日期	2022.4.27														

续表 3.7

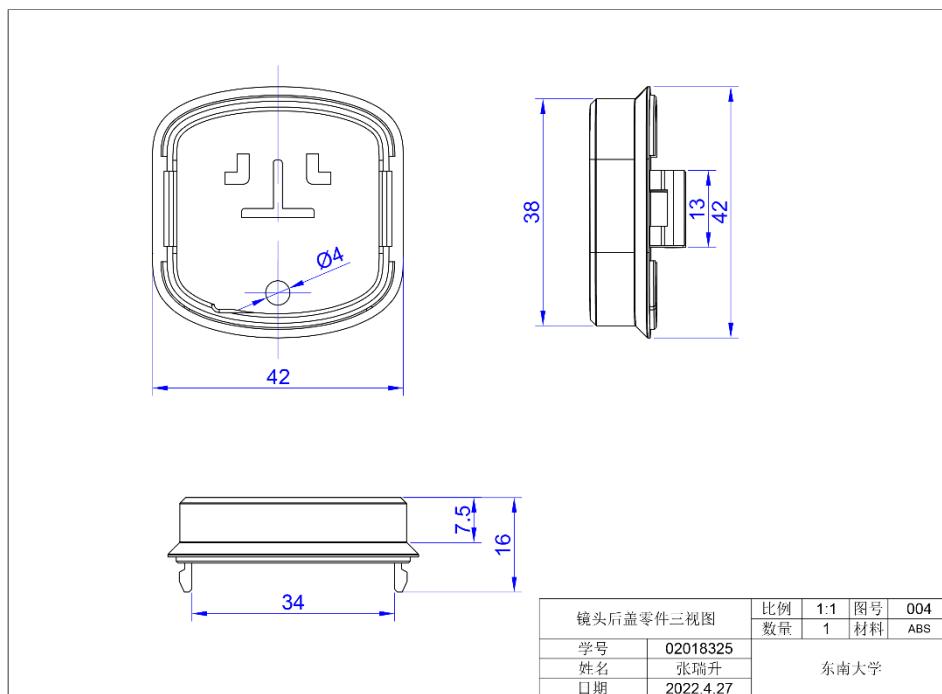
零件名称

零件三视图

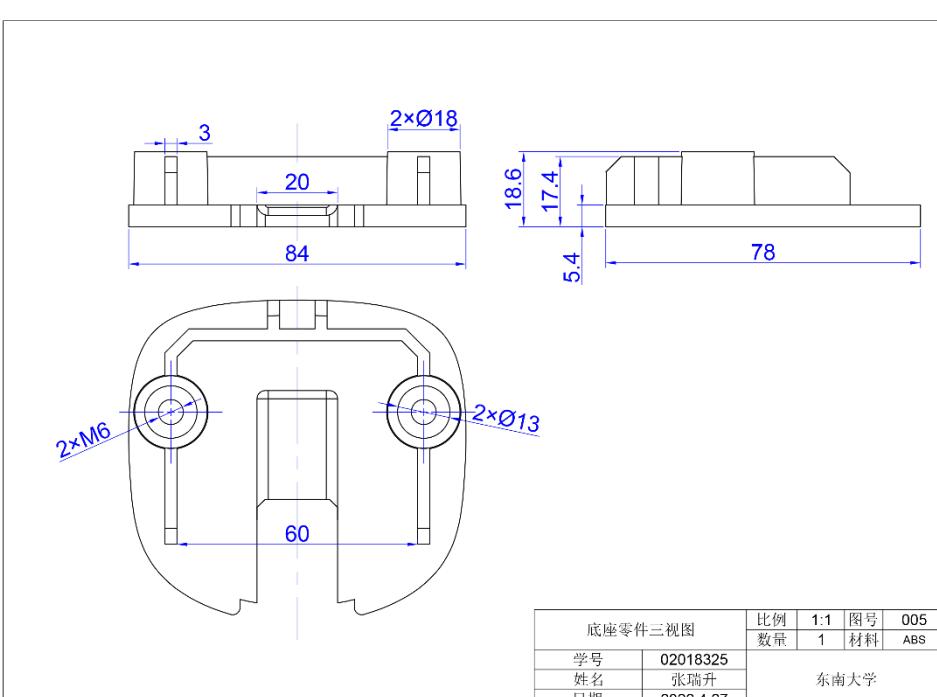
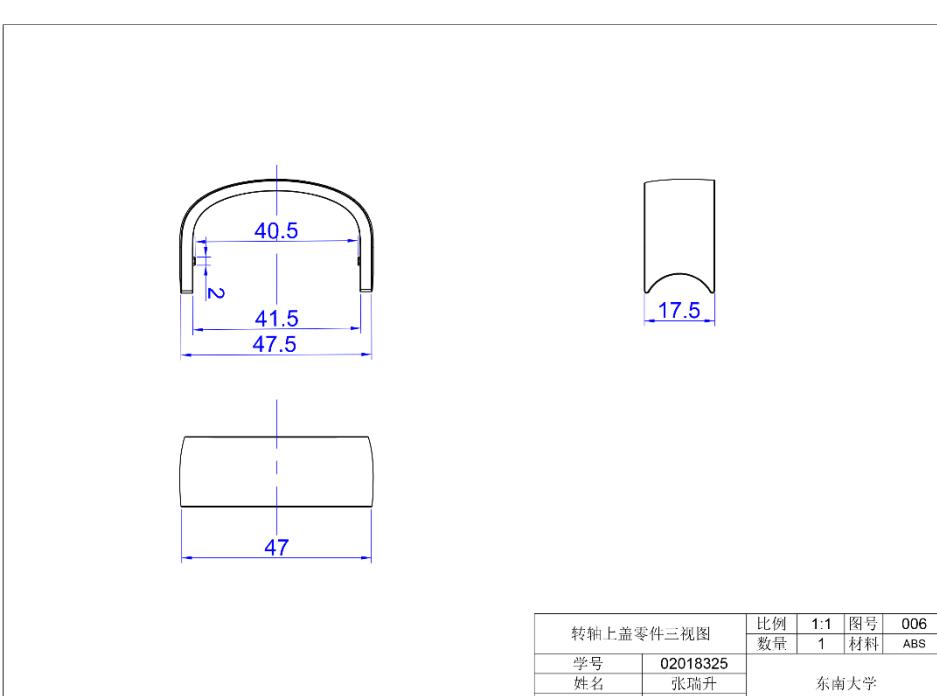
镜头前盖



镜头后盖



续表 3.7

零件名称	零件三视图																
底座	 <table border="1" data-bbox="952 932 1365 1044"> <tr> <td colspan="2">底座零件三视图</td> <td>比例 1:1</td> <td>图号 005</td> </tr> <tr> <td>学号</td> <td>02018325</td> <td>数量 1</td> <td>材料 ABS</td> </tr> <tr> <td>姓名</td> <td>张瑞升</td> <td colspan="2">东南大学</td> </tr> <tr> <td>日期</td> <td>2022.4.27</td> <td colspan="2"></td> </tr> </table>	底座零件三视图		比例 1:1	图号 005	学号	02018325	数量 1	材料 ABS	姓名	张瑞升	东南大学		日期	2022.4.27		
底座零件三视图		比例 1:1	图号 005														
学号	02018325	数量 1	材料 ABS														
姓名	张瑞升	东南大学															
日期	2022.4.27																
转轴盖	 <table border="1" data-bbox="952 1729 1365 1841"> <tr> <td colspan="2">转轴上盖零件三视图</td> <td>比例 1:1</td> <td>图号 006</td> </tr> <tr> <td>学号</td> <td>02018325</td> <td>数量 1</td> <td>材料 ABS</td> </tr> <tr> <td>姓名</td> <td>张瑞升</td> <td colspan="2">东南大学</td> </tr> <tr> <td>日期</td> <td>2022.4.27</td> <td colspan="2"></td> </tr> </table>	转轴上盖零件三视图		比例 1:1	图号 006	学号	02018325	数量 1	材料 ABS	姓名	张瑞升	东南大学		日期	2022.4.27		
转轴上盖零件三视图		比例 1:1	图号 006														
学号	02018325	数量 1	材料 ABS														
姓名	张瑞升	东南大学															
日期	2022.4.27																

3.4.3 硬件系统产品生产加工

硬件系统主要零件材质和加工工艺如表 3-8 所示。

表 3-8 硬件系统主要加工零件 CMF 汇总表

零件名称	零件图片	加工材料	加工工艺	喷漆种类	颜色要求
外壳主体		ABS 增强尼龙	CNC 数控加工	车漆	白色
外壳前盖		ABS 增强尼龙	CNC 数控加工	车漆	黑色
底座		ABS 增强尼龙	CNC 数控加工	车漆	黑色
转轴盖		ABS 增强尼龙	CNC 数控加工	车漆	白色
镜头前盖		透明 PC	CNC 数控加工	车漆	黑色
镜头后盖		ABS 增强尼龙	CNC 数控加工	车漆	黑色
转轴盖帽		ABS 增强尼龙	CNC 数控加工	车漆	白色

硬件系统最终的三视图如图 3-8 所示。产品全高 127mm，长 45mm，宽 47mm，高度低于市面上大部分手机，可以轻松携带在身。将产品各零件交给数控加工厂加工，将零件与电路模块拼接而成的实物图如图 3-9 所示。

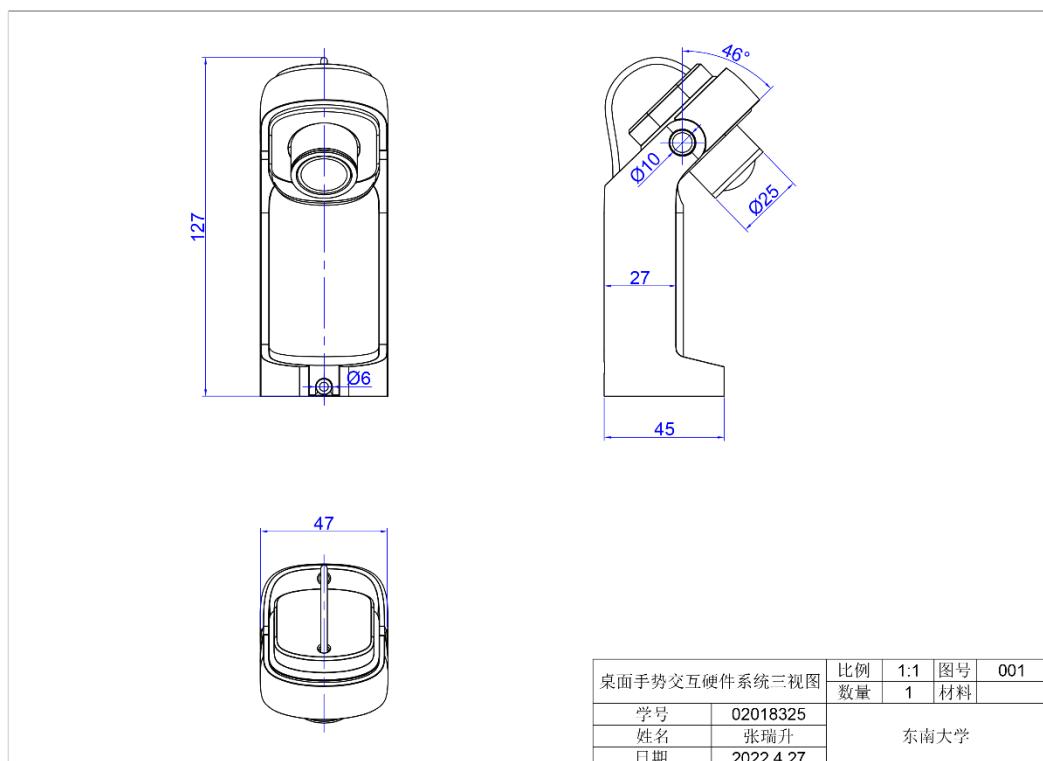


图 3-8 硬件系统产品三视图



图 3-9 硬件系统产品实物图

3.5 本章小结

本章节从第二章的红外激光投影传感技术出发，搭建虚拟现实桌面手势交互硬件系统。首先，确立各模块硬件选型参数，并对用户主观舒适的桌面手势交互区域展开人因实验探究，以人因实验探究结果为参考依据确定各模块的相对位置关系。之后，在硬件选型的基础上对硬件系统进行电路设计，连通红外激光投射模块与红外图像捕获模块。最终，通过桌面手势交互硬件造型设计、结构设计、生产加工等整个产品设计过程，完成桌面手势交互硬件设备的样机制作。

第四章 基于虚拟现实的桌面手势交互软件系统

前一章的硬件系统保证了高帧率且稳定的桌面手势交互图像捕获，接下来的重点是根据所捕获的桌面手势交互图像，准确提取用户做出的桌面手势，并建立用户桌面手势与虚拟现实系统之间的联系。本章节将详细阐释虚拟现实桌面手势交互软件系统的开发历程，包括利用计算机视觉图像处理技术分析桌面手势交互图像并提取桌面手势交互点、通过求解相机内参与相机外参对桌面手势交互点进行校正、采用射线隐喻的方式完成“桌面手势输入-虚拟现实系统响应”的基本交互单元、最终导出桌面手势交互预制体。

4.1 软件系统架构

本课题的桌面手势软件系统基于硬件系统红外图像捕获模块捕获到的桌面手势交互图像，主要建立用户输入的桌面手势与虚拟现实交互系统的交互层关联。软件系统架构图如图 4-1 所示。

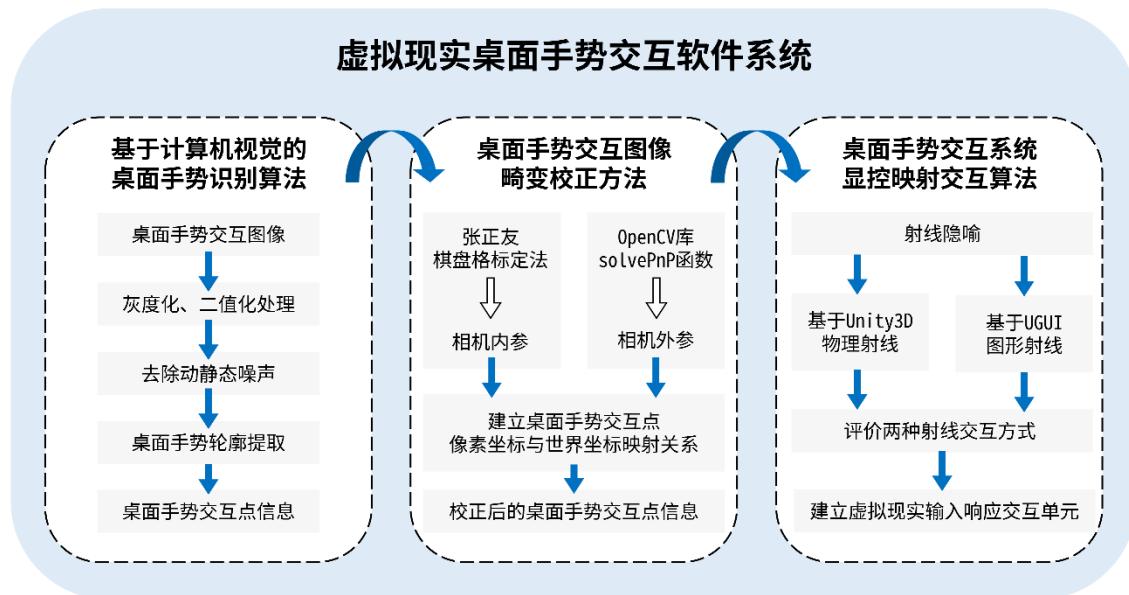


图 4-1 虚拟现实桌面手势交互软件系统架构

桌面手势识别算法采用 OpenCV 库作为计算机视觉图像处理工具，以分析硬件系统红外图像捕获模块捕获到的桌面手势图像，提取桌面手势交互点信息。OpenCV 库是目前采用最广泛的跨平台计算机视觉库，具有 C++、Python、JAVA、MATLAB 等编程语言接口。考虑到该算法后续将移植到 Unity3D 平台虚拟现实程序的 C#脚本中，本课题采用 OpenCV for Unity 插件作为 C#语言接口，实现基于 Unity3D 引擎的图像处理过程。

针对桌面手势交互图像存在的畸变问题，本课题基于第二章所述的桌面手势点校正技术，通过棋盘格标定法获取相机内参，通过 OpenCV 库的 solvePnP 函数获取相机外参，最终建立桌面手势交互点像素坐标到世界坐标的关系，通过桌面手势交互点在图像中的位置重建出用户手指在桌面的触摸实际位置。

桌面手势交互系统显控映射交互算法采用射线隐喻的交互方式，分别尝试了基于 Unity 3D 物理引擎的物理射线隐喻和基于 UGUI 系统的图形射线隐喻两种交互方式，建立用户手势与虚拟现实交互系统之间的联系，完成手势输入到虚拟现实系统响应的基本交互过程。

本章接下来将详细讲解软件系统各算法的开发历程。

4.2 基于计算机视觉技术的桌面手势识别算法

4.2.1 桌面手势识别算法的一般流程

硬件系统红外图像捕获模块捕获到桌面手势交互图像传递给计算机后，桌面手势识别算法对捕获到的图像进行逐帧分析，其一般流程如图 4-2 所示。

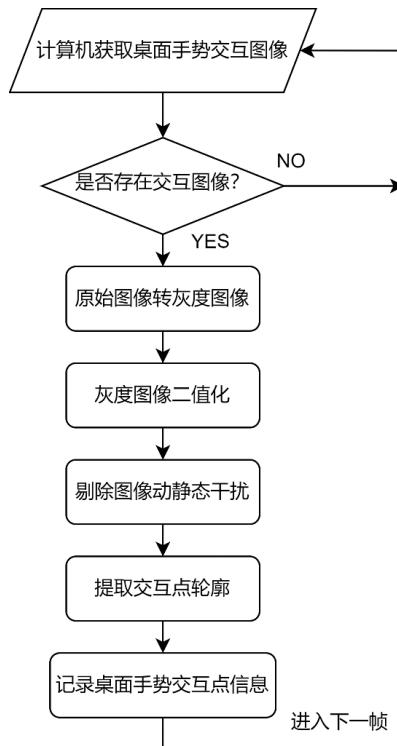


图 4-2 桌面手势识别算法一般流程

算法首先检测红外图像捕获模块的摄像头是否开启并捕获到图像；若检测到捕获的桌面手势交互图像，算法首先对桌面手势交互图像预处理，将图像转为灰度图像并进行二值

化，便于后续提取桌面手势交互点特征；此时由于复杂的桌面环境，图像仍然存在许多噪声，算法接下来对图像中的各种噪声干扰进行依次剔除，具体剔除过程将在下一小节详细说明；完成噪声干扰剔除后，算法将提取桌面手势交互点即手指轮廓，并将桌面手势交互点的交互信息储存在结构体 TouchInformation 类中，储存的交互信息有：当前帧手指触摸数量、当前帧各手指轮廓中心点像素坐标、当前帧时间、当前帧各手指轮廓面积。

TouchInformation 类的源代码为：

```
{
    public struct TouchInformation
    {
        public int touchNum; //当前帧手指触摸数量
        public List<Point> centerPoint; //当前帧各手指轮廓中心点坐标
        public float time; //当前时间
        public List<double> area; //当前帧各手指轮廓面积
    }
}
```

4.2.2 桌面手势识别过程噪声去除方法

桌面手势交互环境复杂多变，在提取桌面手势交互点时必须剔除复杂桌面环境中的噪声干扰以准确提取交互点目标。本算法根据各图像噪声干扰的运动特点，将干扰分为静态干扰和动态干扰并进行分别剔除。

（1）桌面手势识别过程的静态干扰

桌面手势识别过程中的静态干扰是指用户桌面环境等不随时间变化的固定图像噪音，常见的有桌面环境上杂物遮挡红外光线所产生的的图像噪点。静态干扰去除算法采用背景剔除的方法剔除静态干扰，具体过程为：以摄像头捕获到的第一帧稳定图像作为背景帧，通过当前帧与背景帧做差值的方法即可彻底去除桌面手势识别过程的静态干扰，具体代码如下：

```
{
    //待图像稳定后记录第一帧
    if (timeByFrame == 30)
```

```

{
    firstFrameMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
    Utils.webCamTextureToMat(webCam, firstFrameMat);
}

if (timeByFrame > 30)
{
    //原始图像转为 OpenCV 中的 Mat 类
    rgbMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
    Utils.webCamTextureToMat(webCam, rgbMat);
    //帧差法去除背景
    diffMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
    Core.absdiff(rgbMat, firstFrameMat, diffMat);
}
}

```

(2) 桌面手势识别过程的动态干扰

桌面手势识别过程中的动态干扰是指用户交互过程因肢体运动而产生的图像噪音，比如用户手臂移动过程中与桌面接触而产生的图像噪点。相对于静态干扰，动态干扰引入时间变量并且具有较高随机性，剔除难度加大。动态干扰去除算法通过提取桌面手势交互点的图形特征，对每项图形特征设定阈值以剔除不满足条件的图像噪点，具体实现方法有：通过对图形轮廓面积大小设定阈值剔除微小的杂乱噪点；通过对图形轮廓外接矩形长宽比设定阈值剔除太窄或太宽的长条状图像噪点；通过对手指轮廓与手指轮廓外接圆的面积比设定阈值剔除非圆形图像噪点。通过对不满足手势交互点图形特征的绝大多数噪点进行剔除，算法可以较为准确实现桌面手势交互点的识别。动态干扰去除代码如下：

```

{
    var contourArea = Imgproc.contourArea(contours[i]); //获取轮廓面积
    if (contourArea > 1000) //剔除小面积轮廓
    {
        var _rect = Imgproc.boundingRect(contours[i]); //获取外包矩形
        var leftOnPoint = new Point(_rect.x, _rect.y);
    }
}

```

```

var rightDownPoint = new Point(_rect.x + _rect.width, _rect.y + _rect.height);

double aspectRatio = _rect.width / _rect.height;//计算外接矩形长宽比

var _ellipse = Imgproc.fitEllipseAMS(contours[i]); //获取轮廓外接圆

var ellipseArea = _ellipse.size.area();

//计算手指轮廓与外接圆面积比

double ellipseAreaContourRatio = ellipseArea / contourArea;

//对外接矩形长宽比、轮廓外接圆面积比设定阈值，剔除干扰

if (aspectRatio < 3.3 && aspectRatio > 0.3 && ellipseArea ContourRatio < 1.5)

{

    Point point=new Point((_rect.x + _rect.width)/2, (_rect.y + _rect.height) / 2);

    double area = contourArea;

    touchInformation.centerPoint.Add(point); //获取手指中心点

    touchInformation.touchNum++; //获取手指数量

    touchInformation.area.Add(area); //获取手指轮廓面积

}

}

```

4.3 基于成像原理的桌面手势图像畸变校正方法

通过上一节图像处理过程，可以获得桌面手势交互图像中手指在图像中的像素坐标。由于硬件系统红外图像捕获模块的摄像头倾斜放置且相机广角为 120° ，这导致离相机镜头近的区域面积大、特征强，而离相机远的区域面积小、特征弱。图 4-3 可以明显看出，在现实中矩形区域摆放的四个基准点，在图像中则变成了一个近大远小的梯形区域，图像畸变的存在导致前一节识别出桌面手势交互点中心的像素坐标值并不能准确表征为其在桌面的实际位置，用户手指在图像中的滑移距离和滑移方向也并不等同于用户手指实际在桌面上的滑移距离和方向。因此，桌面手势图像畸变校正对于保证用户手势交互过程中一致性与提升手势识别的准确性十分必要。本小节通过建立桌面手势交互点在像素坐标和图像坐标的映射关系来获取用户手势在桌面的实际位置，克服图像畸变。

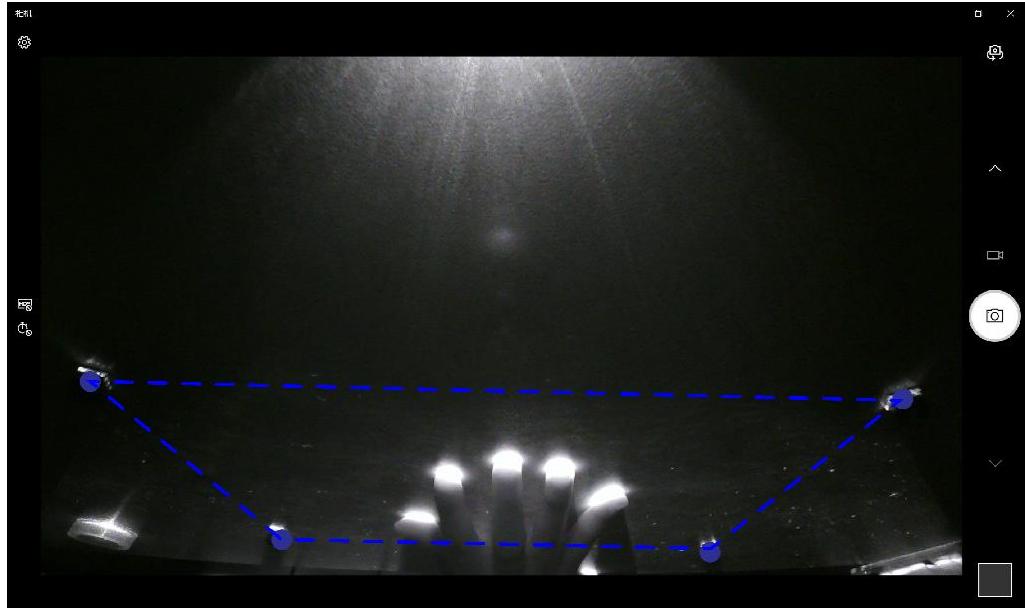


图 4-3 桌面手势交互图像中的图像畸变

桌面手势图像校正的基本原理如第二章中桌面手势交互点校正技术所述，要建立桌面手势交互点在表征图像位置的像素坐标和表征实际位置的世界坐标的映射关系，需要求解相机内参与相机外参，最后建立桌面手势交互点校正关系式。

4.3.1 基于张正友棋盘格标定法的相机内参数求解

张正友棋盘格标定法是目前图像畸变校正与相机标定中采用最为广泛的相机内参数求解方法。棋盘格标定法通过分析相机在不同角度拍摄到的一张黑白间距已知的棋盘格图像，将各角度图像的黑白棋盘格角点进行匹配，从而推测出相机的内部参数。本课题采用基于张正友棋盘格标定原理的 MATLAB 相机标定工具包 Camera Calibrator 进行相机内参数求解，其具体过程如下：

(1) 拍摄不同角度的棋盘格图像

硬件系统的红外图像捕获模块摄像镜头在设计制造过程中已添加了 980nm 红外滤光片，这导致相机只能采集 980nm 波长的红外光线，而无法采集其他波段的光线，普通室内环境已难以采集到清晰完整的棋盘格图像。考虑到室外环境太阳光作为天然光源提供了充足的红外线，课题选择在晴朗白天、太阳可直射的窗前使用桌面手势交互硬件系统拍摄打印好的黑白棋盘格图像。为了保证采集到的图像统一分辨率，课题基于 OpenCV 库编写截图算法，共采集到 10 张硬件系统在不同角度拍摄的棋盘格标定图像，棋盘格标定板截图采集算法如下所示：

```
{  
#include <opencv2\opencv.hpp>  
//该程序用于拍摄截图棋盘格标定板，用来相机标定  
  
using namespace cv;  
using namespace std;  
int main()  
{  
    Mat frame;  
    //读取视摄像头实时画面数据，0：默认笔记本的摄像头；1：外接摄像头  
    VideoCapture capture(1);  
  
    while (true)  
    {  
        capture >> frame;//读取当前帧  
        if (!frame.empty())//判断输入的视频帧是否为空的  
        {  
            imshow("初始画面", frame);//在 window 窗口显示摄像头数据画面  
        }  
        if (waitKey(10) == 'w')  
        {  
            imwrite("D:/OpenCV/Practice.opencv/Practice5/Project5/CUT.jpg",  
frame);  
        }  
        if (waitKey(10) == 'q') //延时 20ms,按 q 键退出程序  
            break;  
    }  
  
    capture.release();//释放摄像头资源  
    return 0;  
}
```

(2) 导入 MATLAB 相机标定工具包

将上述 10 张不同角度的棋盘格标定图像截图导入 MATLAB 相机标定工具包，输入打印的棋盘格标定板实际单元格边长 25mm。棋盘格角点提取后的图像如图 4-4 所示，可以看出提取黑白棋盘格角点准确，无显著差异。

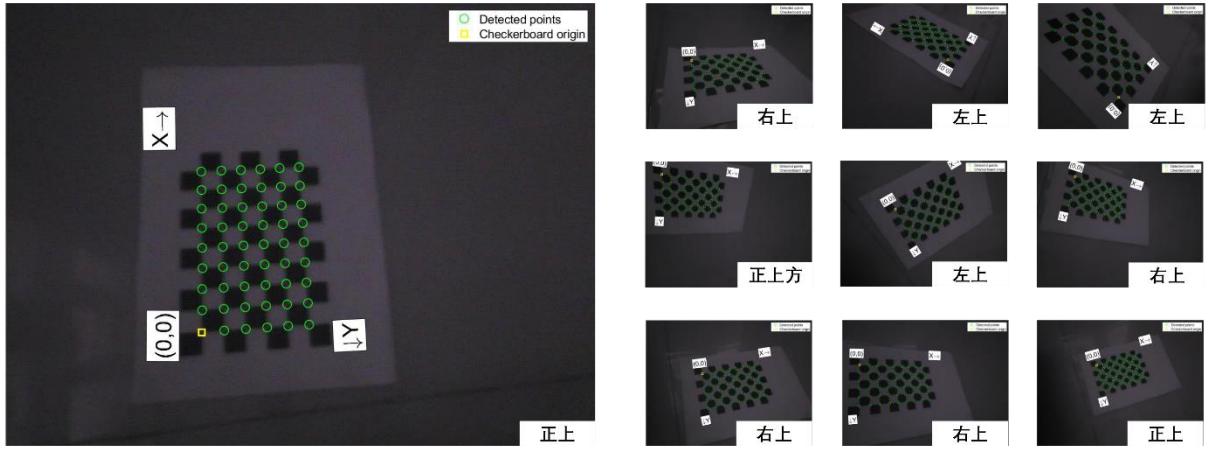
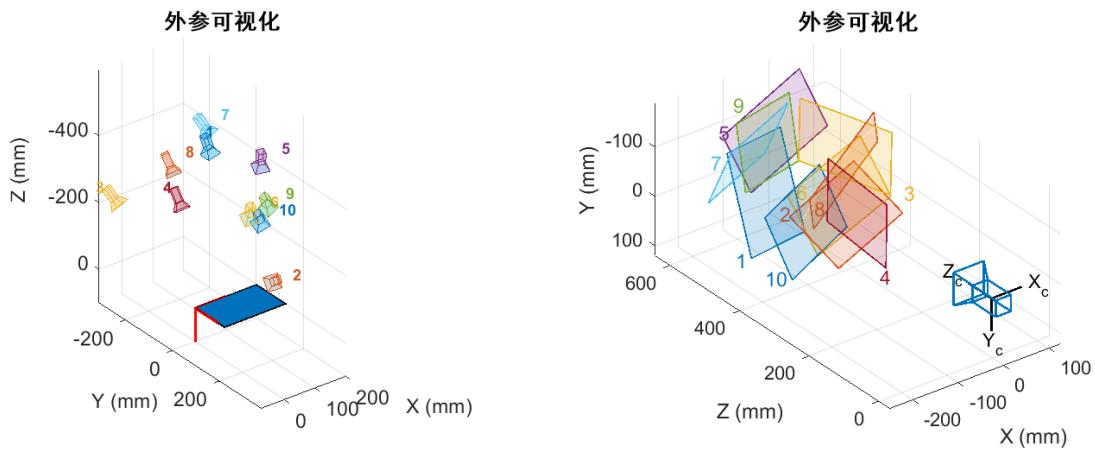


图 4-4 MATLAB 分析后不同方向下的棋盘格标定图像

(3) 正式标定过程

确认棋盘格角点提取准确无误后选择正式标定，工具包自动输出以棋盘格标定图为中心视角的可视化图像和以硬件系统摄像头为中心视觉的可视化图像，分别如图 4-5(a)和图 4-5(b)所示，可以看出，可视化图像和实际拍摄过程相符合。每张图像相机标定的重投影误差如图 4-6 所示，可以看出标定最大误差为 0.5 像素，平均标定误差 0.32 像素，说明相机标定结果较为准确。



(a) 以棋盘格标定图为中心视角

(b) 以硬件系统摄像头为中心视角

图 4-5 棋盘格标定过程三维重建可视化

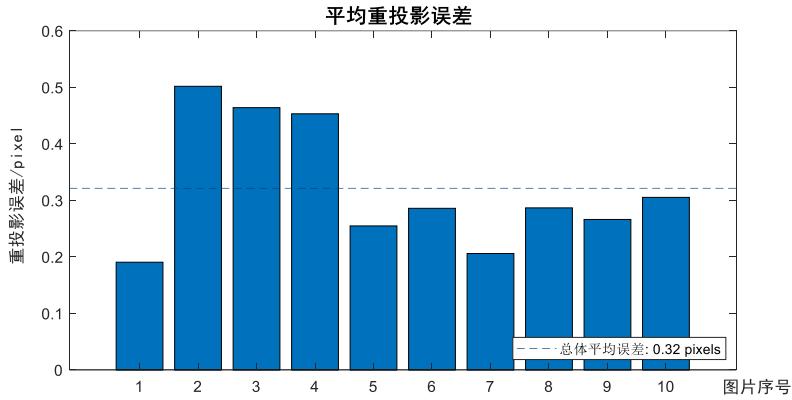


图 4-6 棋盘格标定误差

(4) 输出相机内参

经过上述相机标定，可以获得硬件系统红外图像捕获模块的摄像头相机内参矩阵 $cameraMatrix$ 为：

$$cameraMatrix = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 493.2978 & 0 & 328.0874 \\ 0 & 494.8968 & 244.1862 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{公式 4.1})$$

径向畸变矩阵 $radialDistortion$ 为：

$$radialDistortion = [0.0967 \quad -0.1000] \quad (\text{公式 4.2})$$

切向畸变矩阵 $tangentialDistortion$ 为：

$$tangentialDistortion = [0 \quad 0] \quad (\text{公式 4.3})$$

4.3.2 基于 OpenCV 库的相机外参数求解

在已知相机内参的情况下，OpenCV 库提供了 `solvePnP` 函数可求解相机外参，包括坐标系刚体变换过程中的平移矩阵和旋转矩阵。OpenCV 的 `solvePnP` 函数定义如下所示：

{

```
Bool cv::solvePnP(InputArray objectPoints,//世界坐标系中三维点的坐标
                   InputArray imagePoints,//三维点在图像中对应像素点的二维坐标
                   InputArray cameraMatrix,//相机的内参矩阵
                   InputArray distCoeffs,//相机的畸变系数矩阵
                   OutputArray rvec,//世界坐标系变换到相机坐标系的旋转向量
                   OutputArray tvec,//世界坐标系变换到相机坐标系的平移向量
                   bool useExtrinsicGuess=false,//是否使用旋转平移向量初值标志
```

```

    int flags=CV_ITERATIVE//选择解算 PnP 问题方法的标志
)
}

```

该函数需要在已知相机内参、已知至少 3 个点的世界坐标值和其对应的像素坐标值的条件下求解相机外参。本课题在桌面上选择了世界坐标为(0cm,0cm,0cm)、(9cm,0cm,0cm)、(0cm,3cm,0cm)、(9cm,3cm,0cm)四个点作为实验点并分别测量其像素坐标，将这四个点的世界坐标、像素坐标与上一节求解出的相机内参代入 solvePnP 函数中即可求解相机外参中的平移矩阵和旋转矩阵。具体算法代码如下所示：

```

{
int main()
{
    //定义内参矩阵和畸变系数矩阵
    Mat cameraMatrix = (Mat_<float>(3, 3) << 493.2978, 0, 328.0874,
                        0, 494.8968, 244.1862,
                        0, 0, 1);

    Mat distCoeffs = (Mat_<float>(1, 5) << 0.0967, -0.1, 0, 0, 0);
    //定义世界坐标和像素坐标
    vector<Point2f> imgPoints;
    imgPoints.push_back(Point2f(250, 337));
    imgPoints.push_back(Point2f(500, 330));
    imgPoints.push_back(Point2f(250, 419));
    imgPoints.push_back(Point2f(505, 418));
    vector<Point3f> worldPoints;
    worldPoints.push_back(Point3f(0, 0, 0));
    worldPoints.push_back(Point3f(9, 0, 0));
    worldPoints.push_back(Point3f(0, 3, 0));
    worldPoints.push_back(Point3f(9, 3, 0));
    //计算旋转向量和平移向量
    Mat rvec, tvec;

```

```

solvePnP(worldPoints,imgPoints,cameraMatrix,distCoeffs,rvec,tvec);

//通过罗德里格斯变换将旋转向量变换为旋转矩阵

Mat rvec_J;

Rodrigues(rvec, rvec_J);

cout << "世界坐标系变换到相机坐标系的旋转向量: " << rvec << endl;

cout << "世界坐标系变换到相机坐标系的旋转矩阵: " << rvec_J << endl;

cout << "世界坐标系变换到相机坐标系的平移向量: " << tvec << endl;

}

}

```

上述算法最终求解出桌面手势交互硬件系统相机外参为：

旋转向量 \vec{R} :

$$\vec{R} = (-0.046, 0.088, -0.03) \quad (\text{公式 4.4})$$

旋转向量 \vec{R} 经过罗德里格斯变换后的旋转矩阵 R :

$$R = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} 0.995 & 0.036 & 0.089 \\ -0.040 & 0.998 & 0.045 \\ -0.087 & -0.048 & 0.995 \end{bmatrix} \quad (\text{公式 4.5})$$

平移矩阵 T , 单位为cm:

$$T = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} -2.92 \\ 3.38 \\ 18.36 \end{bmatrix} \quad (\text{公式 4.6})$$

4.3.3 建立桌面手势交互点畸变校正关系式

通过已知相机内参与相机外参, 将公式 4.1、公式 4.5、公式 4.6 代入第二章公式 2.11 即可将上一节计算机视觉图像处理获得的桌面手势交互点的像素坐标 (u, v) 映射到实际的世界坐标 $(X_w, Y_w, 0)$ 。该方法可以有效将计算机视觉图像处理所得到的桌面手势交互点即手指轮廓中心点的像素坐标转换成表示手指在现实中实际位置的世界坐标, 保证后续桌面手势交互线性映射。然而, 目前本方法只能针对固定相机进行畸变校正, 一旦桌面手势交互硬件系统的摄像头被旋转到其他角度后, 必须重新对相机外参进行求解, 重新建立世界坐标和像素坐标的映射转换关系, 针对动态相机的畸变校正方法还有待进一步发掘。

4.4 基于射线隐喻的桌面手势显控映射交互算法

上述的桌面手势识别过程和桌面手势交互点的校正过程保证了准确的桌面手势交互点信息获取。然而，如何建立桌面手势交互点与虚拟现实系统之间的联系，是本课题的一大难点，也是本课题需要解决的关键问题。一方面，现有虚拟现实程序常基于 Unity3D 引擎开发，Unity3D 为开发者封装好了各类现有交互工具的交互接口，例如：鼠标接口、键盘接口、手柄接口，开发者只需在 Unity3D 的“黑箱”中将这些接口拿来使用即可，然而，Unity3D 并没有服务于本课题桌面手势交互的交互接口，这意味着课题必须独立编写一套适用于本系统的交互功能；另一方面，桌面手势交互作为二维平面内的交互方式，如何实现其对虚拟现实三维空间的控制，也是本课题需要考虑的一点。本小节将详细阐释建立桌面手势交互和虚拟现实环境控制显示关系的原理和实现方法。

4.4.1 交互系统显控映射原理

桌面手势交互无法像空中手势一样实现六自由度的三维空间交互，使用三自由度的桌面手势输入来控制六自由度的交互环境，则势必要舍弃深度方向的控制。射线隐喻作为虚拟现实常用隐喻类型，已经被广泛应用于虚拟现实远距离交互中，射线隐喻将深度方向的控制无限延伸，仅通过改变射线方向即可完成显控交互任务，成为桌面手势与虚拟现实环境建立显控交互关系的潜在解决方案。此外，Unity3D 系统的物理引擎和 UGUI 系统分别提供了射线交互接口，允许开发者通过投射射线的方式来完成和 3D 物体、UI 控件的碰撞响应。基于射线隐喻的交互特性与 Unity3D 系统的技术支持，课题决定采用射线隐喻的方式实现桌面手势显控映射交互方法。通过用户手指滑移控制射线投射角度，实现交互目标的选择，通过用户做出特定的桌面手势，实现交互行为的触发。

基于射线隐喻的桌面手势显控映射交互方法的实现思路如图 4-7 所示。从用户眼中心向虚拟环境投射出一条射线，射线初始状态指向用户视野正中央。在射线角度控制方面，用户可以通过单个手指在桌面滑移来控制射线投射角度的变化，此时射线角度变化基于用户手指的移动惯性或移动速度，是一种类似于笔记本触摸板的交互方式；用户也可以通过头部转动来控制射线方向随头部方向变化而变化，此时射线角度变化基于用户头部的旋转角度，是一种类似于眼控的交互方式。在交互任务触发方面，用户首先通过手指滑移和头部运动完成交互目标的选择，然后通过做出特定的桌面手势动作完成交互任务的触发，实现从桌面手势输入到虚拟现实响应的基本交互单元。

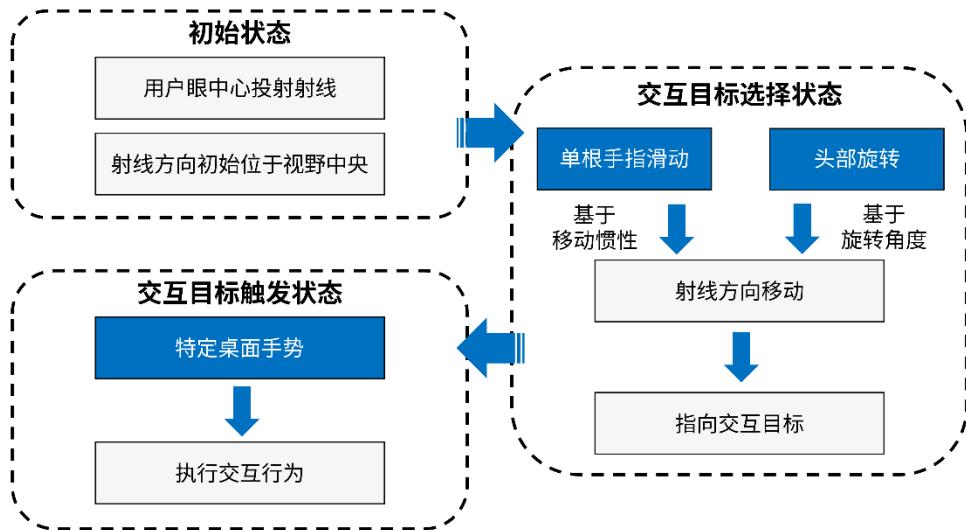


图 4-7 基于射线隐喻的桌面手势显控映射交互算法实现思路

用户使用单根手指滑移控制射线角度变化是基于运动惯性的，即射线角度的变化与用户手指移动速度相关，手指滑移越快，射线角度变化就越大。每帧射线角度水平方向变化 α_x 与竖直方向变化 α_y 随速度变化的关系如下所示：

$$\alpha_x = k v_x \quad (公式 4.7)$$

$$\alpha_y = k v_y \quad (公式 4.8)$$

其中， v_x 为当前帧用户手指在桌面水平方向的移动速度， v_y 为当前帧用户手指在桌面竖直方向的移动速度， k 为虚拟现实交互系统控制显示映射系数（C/D 比）。

在实现方法方面，本课题目前分别使用基于 Unity3D 平台物理引擎的物理射线和基于 Unity3D 平台 UGUI 系统的图形射线两种方式实现桌面手势显控映射交互。接下来的小节将详细阐释两种射线交互方法，并总结二者的特点。

4.4.2 基于 Unity3D 物理引擎的物理射线交互方法

Unity3D 物理引擎为开发者提供了模拟物理环境接口，开发者通过物理引擎可实现碰撞检测、重力效果、受力反馈等功能。基于 Unity3D 物理引擎，可以发射出一条带有碰撞体属性的物理射线（Physics. Raycast），物理射线的交互流程如图 4-8 所示。

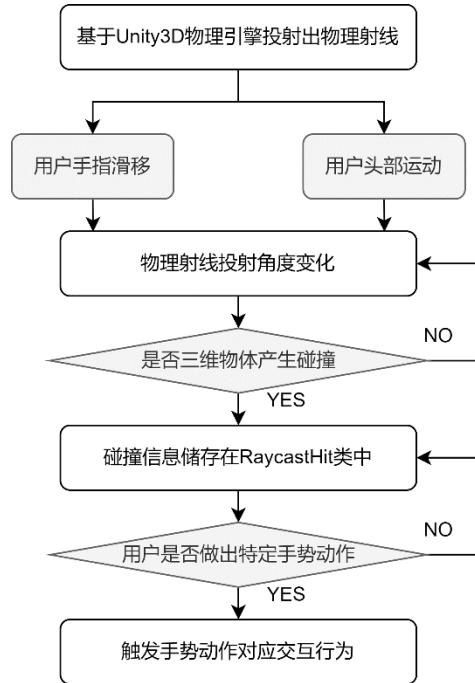


图 4-8 基于物理射线的桌面手势交互流程

通过物理射线与虚拟环境中带有碰撞体属性的三维物体产生碰撞，完成对三维物体的选择。被射线碰撞的三维物体的名称、所在图层、碰撞点的世界坐标等碰撞信息将被记录并储存在 RaycastHit 类中。用户可以通过单指点击、双指放缩等特定手势来触发三维物体选中、三维物体缩放等交互任务，实现三维物体交互的基本交互单元。

物理射线的实现代码如下所示：

```

{
    //设定射线发射方向
    vector.x = 10000*Mathf.Sin(verticalAngle) * Mathf.Cos(horizontalAngle);
    vector.y = 10000*Mathf.Cos(verticalAngle);
    vector.z = 10000*Mathf.Sin(verticalAngle) * Mathf.Sin(horizontalAngle);

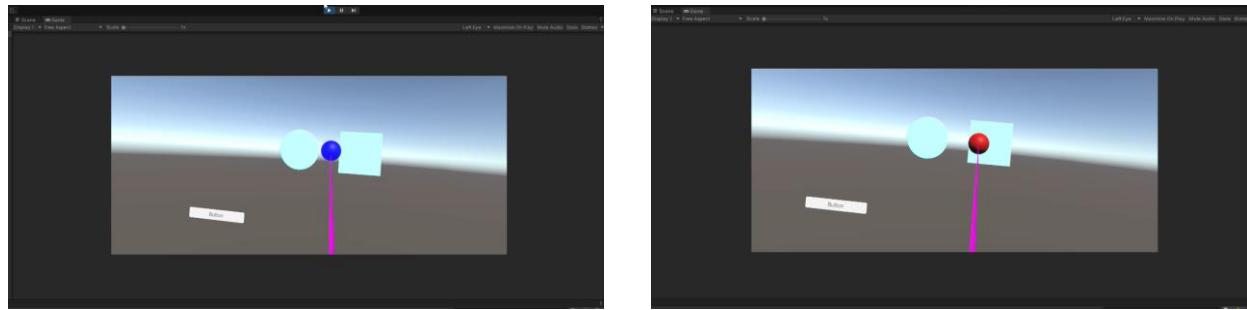
    //投射物理射线
    var ray = new Ray(this.GetComponent<Transform>().position, vector);
    //渲染物理射线
    lineRenderer.SetPosition(1, vector);
    //返回射线投射碰撞结果
    RaycastHit hit;
  
```

```

if(Physics.Raycast(ray,out hit)) //如果投射射线产生碰撞
{
    lineRenderer.SetPosition(1, hit.point);
    GameObject.Find("Cursor").transform.position = hit.point;
    SpriteRender spr=GameObject.Find("Cursor").GetComponent< SpriteRenderer>()
();
    spr.sprite = Resources.Load("Icon/Cursor2", typeof(Sprite)) as Sprite;
    hitName = hit.transform.name;
}
}

```

物理射线最终的交互效果如图 4-9 所示。当物理射线未与三维物体碰撞时，射线尽头的光标球处于未选择交互目标状态，颜色为蓝色；当物理射线与三维物体产生碰撞时，射线尽头的光标球变为红色，触发交互目标选择。目前该方法可以实现和所有带有碰撞体属性的三维物体的碰撞检测。



(a)物理射线未碰撞三维物体状态

(b)物理射线碰撞到三维物体球形光标变色

图 4-9 物理射线交互效果

基于 Unity3D 物理引擎的物理射线交互目标主要面向虚拟环境中的三维物体，同时也可实现和添加碰撞体属性的 UI 控件进行交互。然而，物理射线的交互结果只有碰到与未碰到两种状态，无法实现更复杂的交互行为；在与 UI 控件交互时，需要给每个 UI 控件设定碰撞体属性，增加了开发的复杂度；物理射线碰撞结束后，需要给每个三维物体添加其对应的交互响应效果，通用性较差。

4.4.3 基于 UGUI 系统的图形射线交互方法

Unity3D 的 UGUI 系统是 Unity 提供的用户图形界面系统，开发者可以利用该系统的

各种 UI 控件搭建复杂的人机交互界面。UGUI 系统包含了一套独立的图形射线交互接口，图形射线仅与 UI 控件产生碰撞，每类 UI 控件也有各自的触发效果。在二维图形界面中，使用鼠标点击按钮、使用鼠标拖动滑动条就是通过从鼠标点击处发射一条图形射线完成的。

基于 UGUI 系统的图形射线交互和上一小节物理射线交互方法类似。课题通过 UGUI 的 EventSystem 事件系统发射出一条仅与 UI 控件碰撞的图形射线，图形射线交互流程如图 4-10 所示。

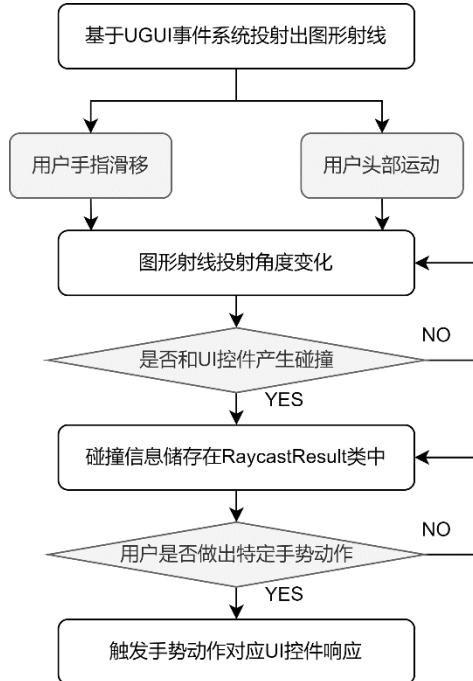


图 4-10 基于图形射线的桌面手势交互流程

通过 UGUI 系统的图形射线与虚拟现实环境中的 UI 控件产生碰撞，完成对 UI 控件的选择。被射线碰撞的 UI 控件名称、数量、图像射线碰撞点的世界坐标等碰撞信息被储存在 RaycastResult 类中。用户可以通过手指单击、手指双击、手指长按等特定手势触发 UI 控件的点击、拖动等交互任务，实现 UI 交互的基本交互单元。

图形射线的实现代码如下所示：

```

{
    //投射图形射线并获取投射结果
    List<RaycastResult> raycastResults = new List<RaycastResult>();
    m_eventSystem.RaycastAll(m_pointerEvent, raycastResults);
    //射线投射结果
    if (raycastResults.Count > 0)
  
```

```

{
    foreach (var result in raycastResults)
    {
        var newSelectable = result.gameObject.GetComponentInParent<Selectable>
();
        if (newSelectable)
        {
            if (newSelectable != m_excluded && newSelectable != m_currentSelectable)
            {
                Select(newSelectable); //鼠标移入
                m_currentRaycastResult = result;
            }
            break;
        }
    }
}

```

图形射线最终的交互效果如图 4-11 所示。当图形射线未与 UI 控件产生碰撞时，图形射线光标处于悬停状态；当图形射线与 UI 控件产生碰撞时，图形射线光标处于选择点击状态；当图形射线与 UI 控件产生长时间碰撞时，图形射线光标处于长按拖动状态。目前，该方法可以实现按钮、滚动条、框选槽、下拉菜单等绝大多数常用 UI 控件交互。



(a)未碰撞时光标悬停状态

(b)碰撞时光标选择状态

(c)长按时光标拖动状态

图 4-11 图形射线交互效果

基于 UGUI 系统的图形射线交互目标主要面向 UI 控件，同时也可以实现和带有 UI 属性的三维物体交互。图形射线的交互结果有悬停、点击、拖动等，可以实现更复杂的交互行为；在与 UI 控件交互时，可直接与基于 UGUI 系统的现有界面进行交互，交互响应效果也直接基于 UGUI 系统编写，简化了开发过程，同时提高通用性。

4.4.4 交互方法总结

上述两种射线交互方法均可以实现虚拟环境的交互行为，两种射线交互方法特点总结如表 4-1 所示。可以看出，图形射线相较于物理射线拥有更好的交互效果和更便捷的开发需求，故本课题在下一章的应用实例中采用图形射线为核心交互方法。

表 4-1 物理射线交互与图形射线交互特点总结

交互方法	交互对象	交互结果	开发便捷性	方法兼容性
物理射线 交互方法	三维物体； 带有碰撞体属性的 UI 控件	只有碰撞到与未 碰撞到两种结果	交互对象需添加碰撞体； 交互响应程序单独编写； 开发较为麻烦	可兼容空中手势 等其他交互方式
图形射线 交互方法	UI 控件； 带有 UI 属性的三维物 体	可以得到点击、拖 动、悬停等多种碰 撞结果	交互响应程序直接基 于 UGUI 系统编写； 开发比较方便	可兼容空中手势 等其他交互方式

4.5 基于 Unity 的桌面手势交互预制体插件导出

课题将桌面手势交互软件系统中的桌面手势识别算法与基于射线隐喻的桌面手势显控映射交互算法封装成一个预制体，并以插件的形式导出，以供其他开发者使用。如图 4-12 所示，预制体中包含：基于计算机视觉的桌面手势识别脚本、桌面手势交互点速度求解脚本、图形射线投射脚本、图形射线光标可视化脚本及单击、双击等自定义桌面手势触发脚本。开发者只需将该预制体拖到虚拟现实交互场景中，并且将计算机连接至本课题的桌面手势交互硬件系统，即可完成虚拟现实的桌面手势交互过程。开发者也可添加自己编写的自定义桌面手势触发脚本，扩展该预制体的手势交互功能。

预制体的建立简化了虚拟现实桌面手势交互应用的开发过程，保证绝大多数基于 Unity3D 开发的虚拟现实应用可以在不改变原有用户界面的基础上，轻松添加桌面手势交互功能，而不干涉其他现有交互方式，保证了交互方式通用性。

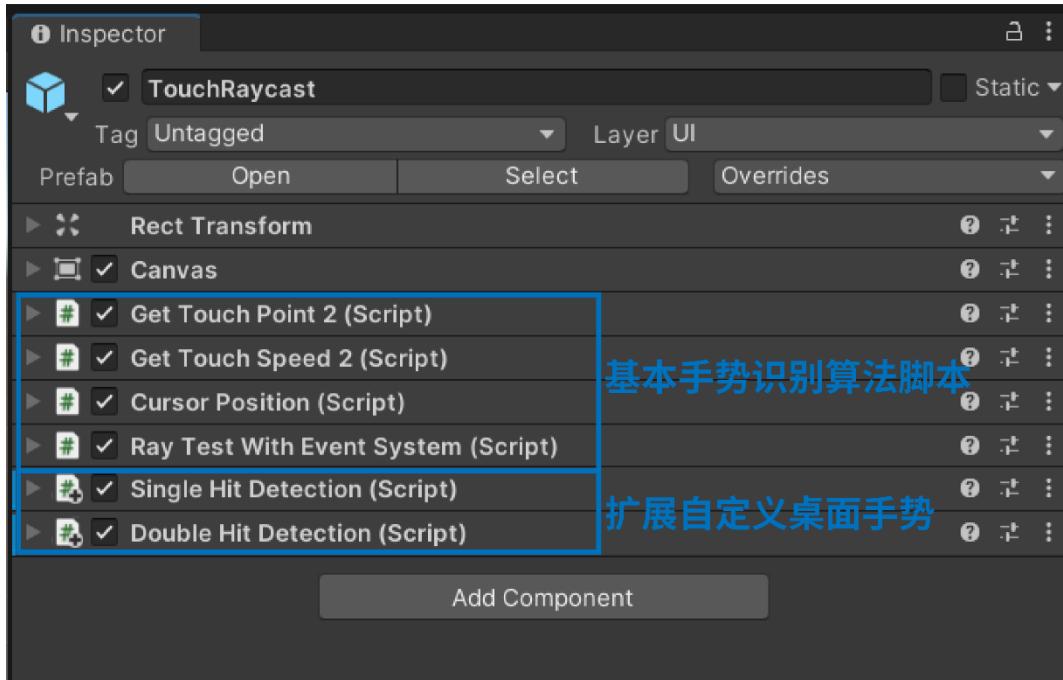


图 4-12 桌面手势交互预制体

4.6 本章小结

本章节基于第三章的桌面手势交互硬件系统，利用 OpenCV for Unity 插件作为计算机视觉图像处理接口，在 Unity3D 平台编写了桌面手势识别算法并提取桌面手势交互点信息。根据相机成像原理，利用 MATLAB 求解相机内参，利用 OpenCV 的 solvePnP 函数求解相机外参，最终完成了对桌面手势交互点坐标校正过程。最后，课题采用射线隐喻的方式建立桌面手势和虚拟现实系统之间的联系，实现了桌面手势输入到虚拟环境响应的基本交互单元，并将所有算法封装至预制体插件中供其他开发者使用。

第五章 基于虚拟现实的桌面手势交互功能实现及展示

前两章的软硬件系统协同工作保证了虚拟现实基本交互功能的正常实现，在基本交互功能实现的基础上，良好的人机交互方式应贴合人的生理心理特性，满足人的习惯定势。桌面手势交互系统应保证显控耦合性，即实现人机交互界面显示和桌面手势操作控制之间的心理一致性与相容性，以保证改善用户对外部信息处理能力，提高操作质量，减轻用户疲劳。本章将针对上述控制显示相容性问题进行桌面手势交互功能设计，并最终基于Unity3D引擎建立桌面手势交互功能展示系统。

5.1 基于虚拟现实的桌面手势交互功能实现

虚拟现实系统中的桌面手势交互功能可基于交互任务类型分为选择型交互、操纵型交互、与其他交互方式协同型交互三类，桌面手势交互功能架构如图 5-1 所示。

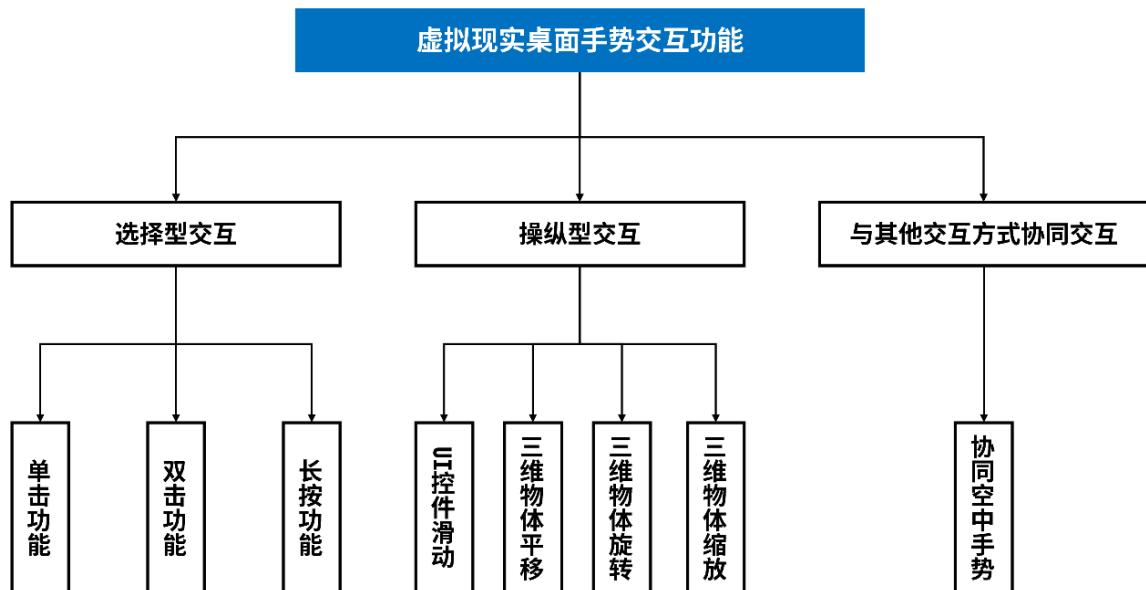


图 5-1 虚拟现实桌面手势交互功能架构

5.1.1 选择型交互功能

选择型交互功能指实现用户对于虚拟环境下 UI 控件和三维物体的选择任务。在二维界面交互中，采用最为广泛的选择型交互功能为手指点击与鼠标点击；在三维界面交互中，采用最为广泛的是选择型交互功能为按钮选择和手指捏合。无论是二维界面交互还是三维界面交互，选择任务都是以最简单直接的方式来实现。本课题采用单根手指在桌面单击、双击、长按的桌面手势实现对交互目标的选择功能。

(1) 单击功能

单击功能的实现逻辑如图 5-2 所示。系统实时记录单根手指按下的时间 t_1 和单根手指抬起的时间 t_2 ，当单根手指从按下到抬起的花费时间 $t_2 - t_1$ 小于设定的单击时间阈值 t_s ，则判定单击功能实现。选择型交互功能存在海森堡效应^[46]，即用户从指向交互目标到触发目标选择的过程中，选择动作本身可能会造成指向的交互目标发生偏移，进而导致交互目标选择错误。为了避免该类问题，设定回溯时间 t_0 ，在判定单击功能实现时，触发选择的交互目标为用户还未做出单击动作 t_0 时刻前所指向的交互目标，以保证选择手势动作控制和用户意图的运动相合性。

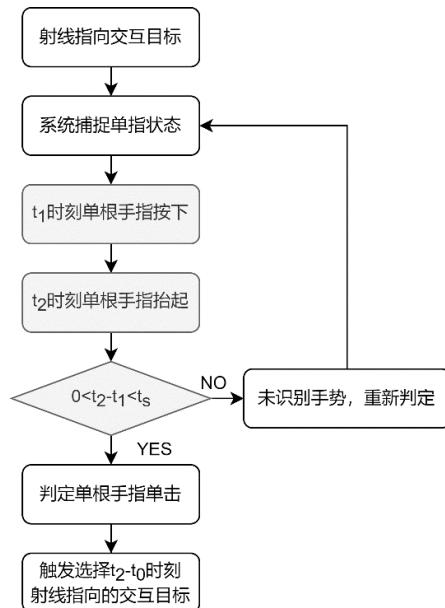


图 5-2 单击功能实现逻辑

单击功能的演示效果如图 5-3 所示，通过手指单击可以实现各种按钮、框选槽的选择。

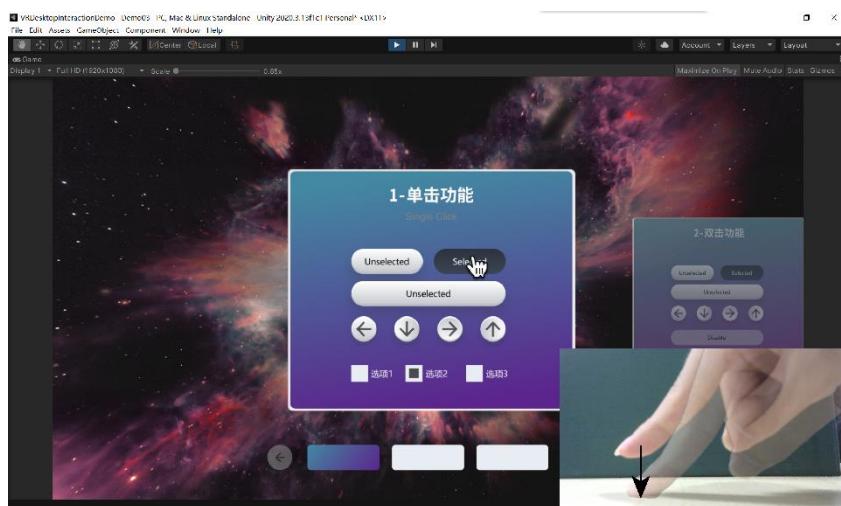


图 5-3 单击功能演示效果

(2) 双击功能

如图 5-4 所示，双击功能判定规则与单击功能类似。双击功能本质为两次间隔很短的单击动作，当第一次单击动作的时间间隔小于单击时间阈值 t_s 、第二次单击的时间间隔小于单击时间阈值 t_s ，且连续两次单击动作之间的时间间隔小于双击时间阈值 t_d ，则判定双击功能实现，触发 t_0 时刻前交互目标选择，以消除海森堡效应。

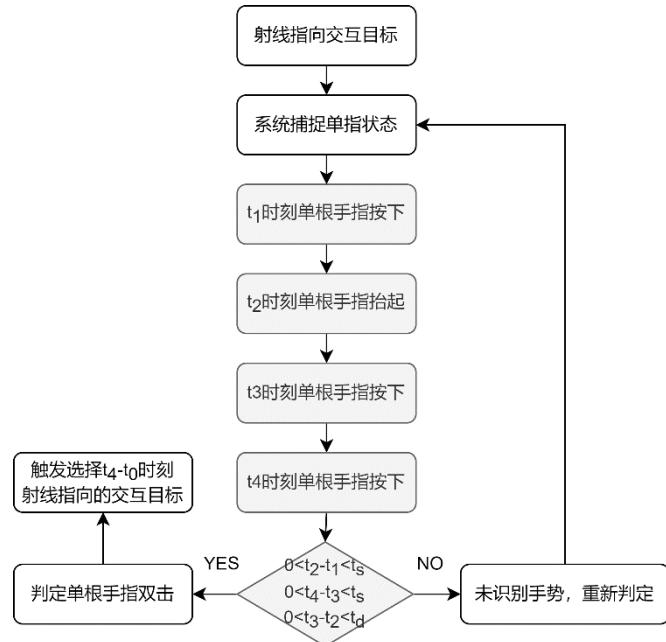


图 5-4 双击功能实现逻辑

双击功能的演示效果如图 5-5 所示，通过手指双击同样可以实现对各类按钮的选择。

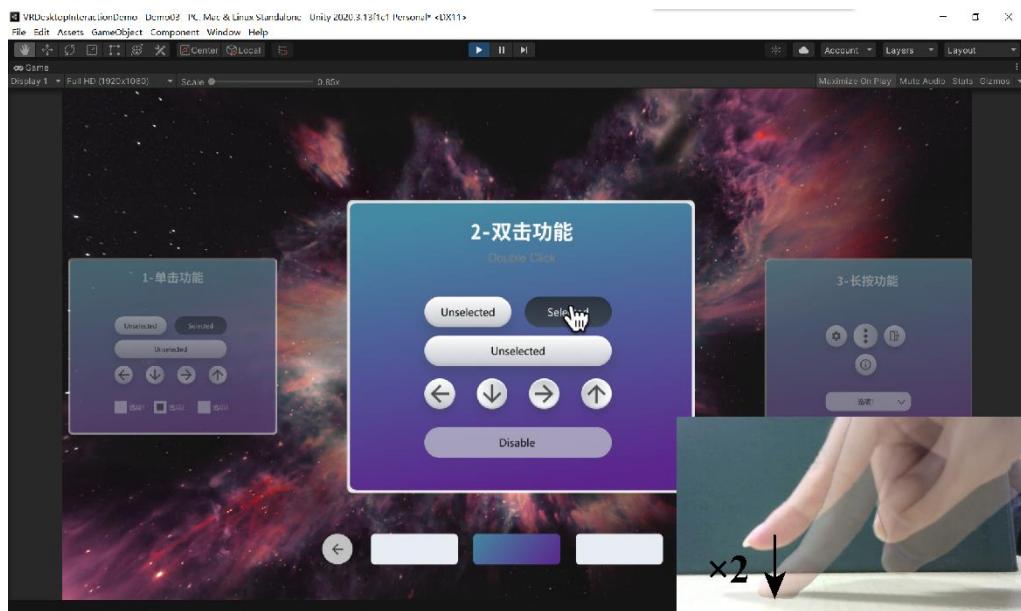


图 5-5 双击功能演示效果

(3) 长按功能

如图 5-6 所示，当用户单根手指按下持续时间超过长按判定阈值 t_l ，且在此期间桌面手势交互点位置变化未超过 x_l ，则判定为长按功能实现。长按功能将主要用于显示扩展信息，例如唤起控件说明、展开扩展菜单、打开下拉列表等。

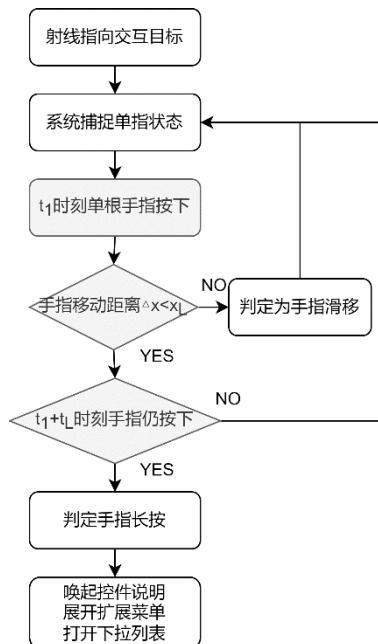


图 5-6 长按功能实现逻辑

长按功能的演示效果如图 5-7 所示，手指长按可完成弹出扩展菜单，张开下拉列表。

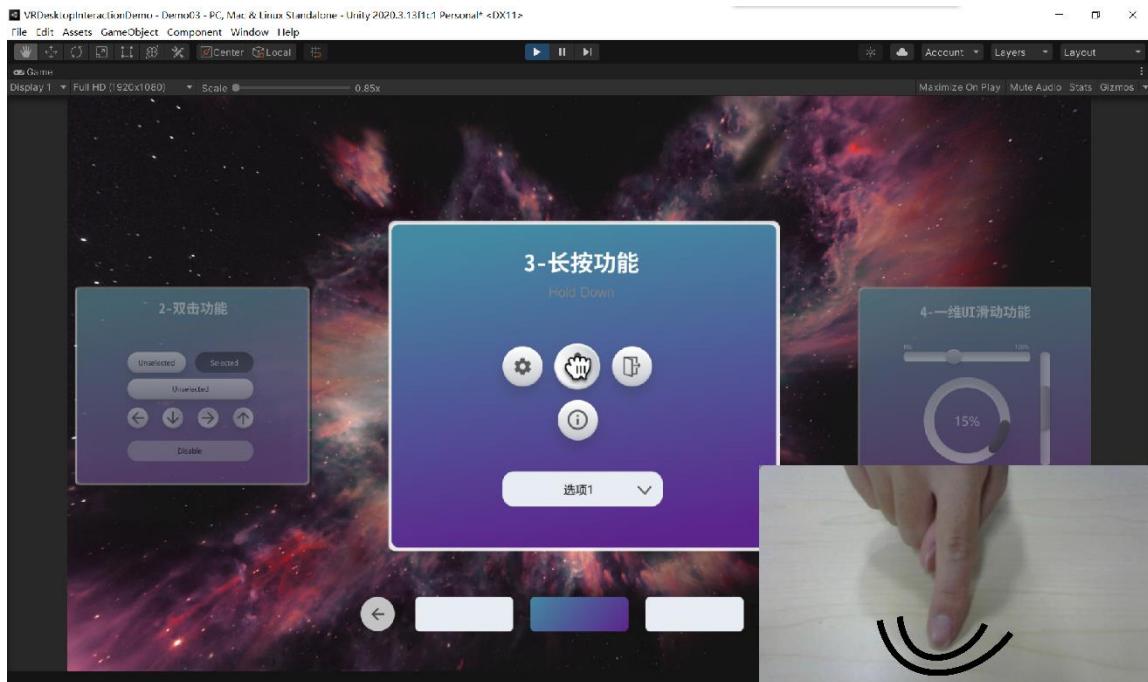


图 5-7 长按功能演示效果

5.1.2 操纵型交互功能

操纵型交互功能指完成用户对 UI 控件和三维物体的操纵任务。操纵型交互功能种类更为繁多，自定义特点更强，所需要的桌面手势也相对更为复杂。本小节将分别对一维 UI 控件滑动、二维 UI 控件滑动、三维物体平移、三维物体旋转、三维物体缩放等操作任务进行桌面手势设计。

(1) 一维/二维 UI 控件滑动

以滚动条为代表的可滑动 UI 控件常基于鼠标点击拖动或手柄指向拖动等交互动作触发，这些交互动作都是先选择后拖动的过程。本课题采用用户单根手指长按触发拖动状态，再通过手指滑移实现 UI 控件的滑动功能。UI 滑动演示效果如图 5-8、图 5-9 所示。

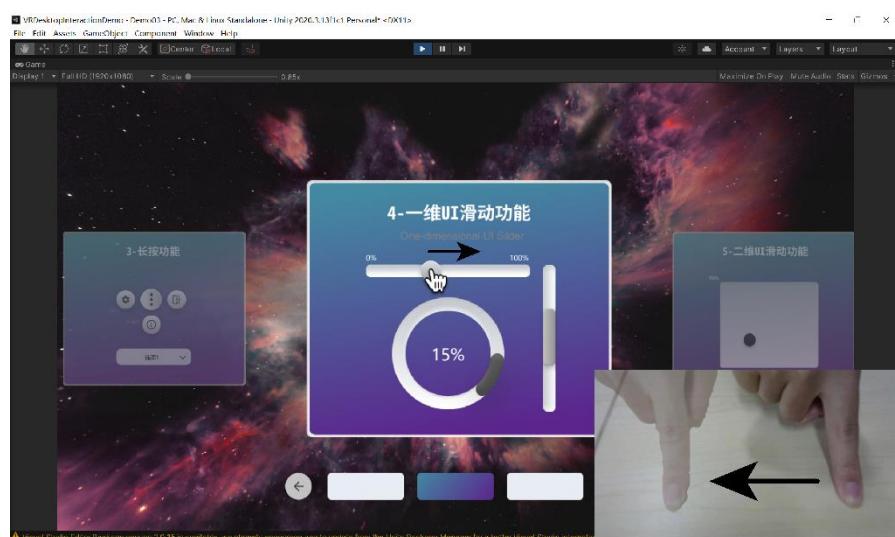


图 5-8 一维 UI 控件滑动功能

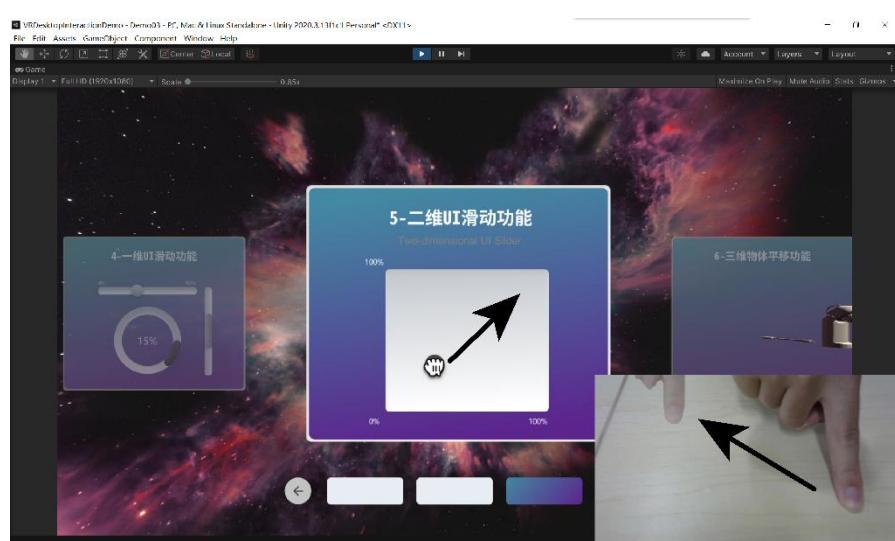


图 5-9 二维 UI 控件滑动功能

(2) 三维物体平移

如图 5-10 所示，三维物体平移与 UI 控件滑动类似，需要用户先通过手指单击、双击等选择型手势完成针对三维物体的选择，再通过单指滑移完成三维物体的移动。

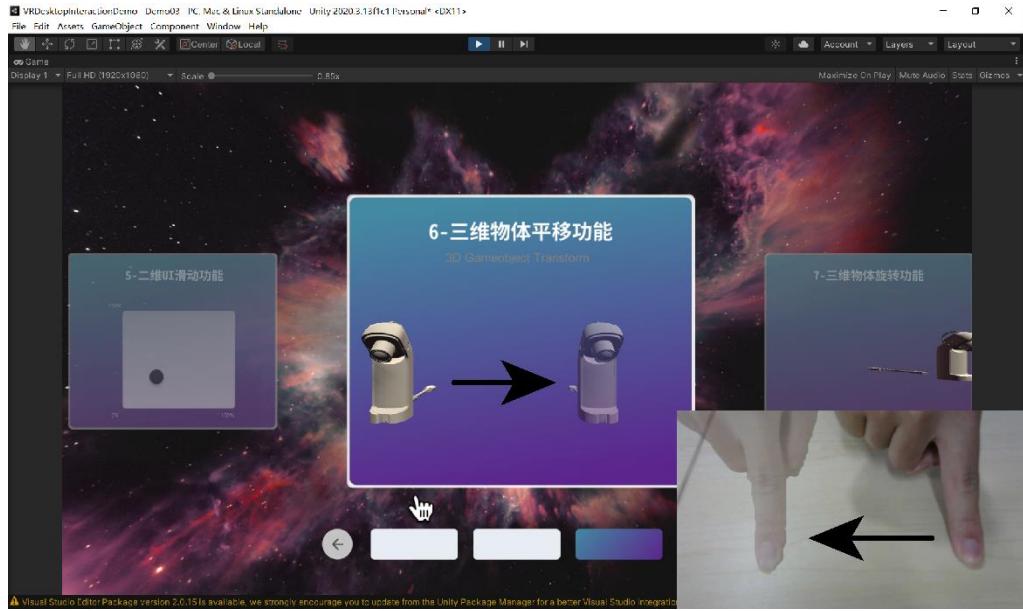


图 5-10 三维物体平移功能

(3) 三维物体旋转

如图 5-11 所示，三维物体的旋转需要用户先通过选择型手势选择目标三维物体，再通过一根手指固定、另一根手指旋转实现三维物体旋转。

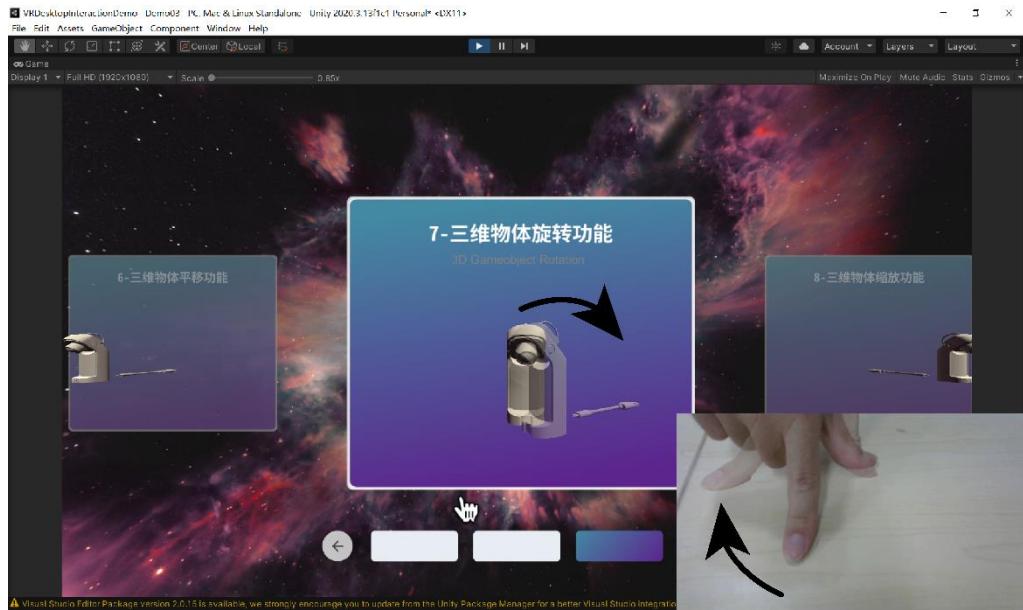


图 5-11 三维物体旋转功能

(4) 三维物体缩放

如图 5-12 所示，三维物体缩放要求用户选择三维物体后，通过两根手指张开回缩实现三维物体的缩放功能。

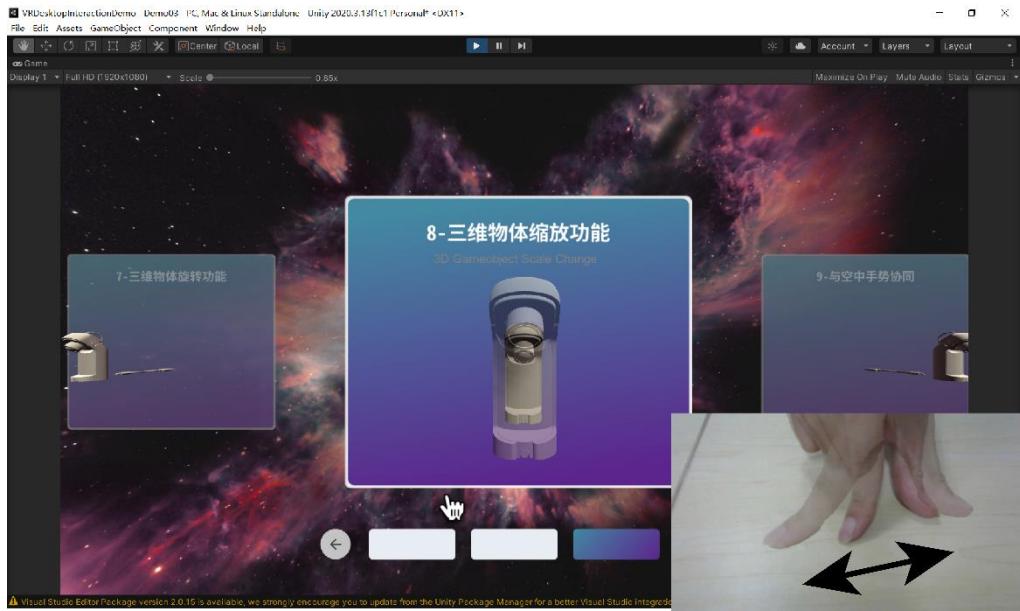


图 5-12 三维物体缩放功能

5.1.3 与其他交互方式协同型交互功能

由于桌面手势交互并不干涉空中手势等其他虚拟现实交互方式，多种交互方式协同运作成为可能。桌面手势交互输入速度快、疲劳程度低，但以二维平面交互控制三维空间始终存在局限性，可以通过高自由度的空中手势交互弥补该局限，实现两种交互方式互为补充，最大程度发挥各交互方式的优点。本课题选择将桌面手势交互动空中手势交互协同，共同完成交互任务。

5.2 桌面手势交互功能展示系统

基于上一节所实现的桌面手势交互功能，本节以桌面手势交互功能展示为主要内容设计虚拟现实人机交互界面，将本课题的手势交互系统投入到具体的应用实践中去。

5.2.1 实例开发工具简介

课题面向虚拟现实环境开发桌面手势交互功能展示系统应用实例。使用 Unity3D 引擎搭建三维虚拟现实环境、开发虚拟现实程序；使用 Visual Studio 2019 作为程序编辑器编写 C#脚本；使用 OpenCV for Unity 插件作为计算机视觉桌面手势识别软件；使用 Figma 完成

虚拟现实人机交互界面设计；使用 Oculus Intergration 插件作为虚拟现实接口，接入头戴式显示器并实现手部追踪。

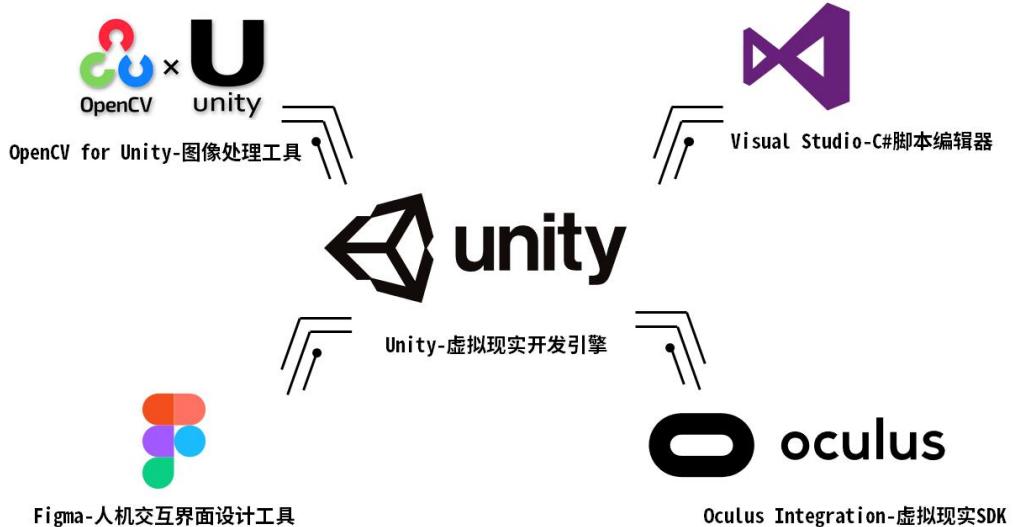


图 5-13 课题所使用的开发工具

5.2.2 界面信息框架

桌面手势交互展示系统应以展示本课题的虚拟现实桌面手势交互功能为主，引导用户体验本课题所实现的桌面手势交互功能。与此同时，展示系统应作为一种说明文档供开发者了解本课题软硬件系统的基本原理与使用方法。通过上述功能分析，建立界面信息框架如图 5-14 所示。

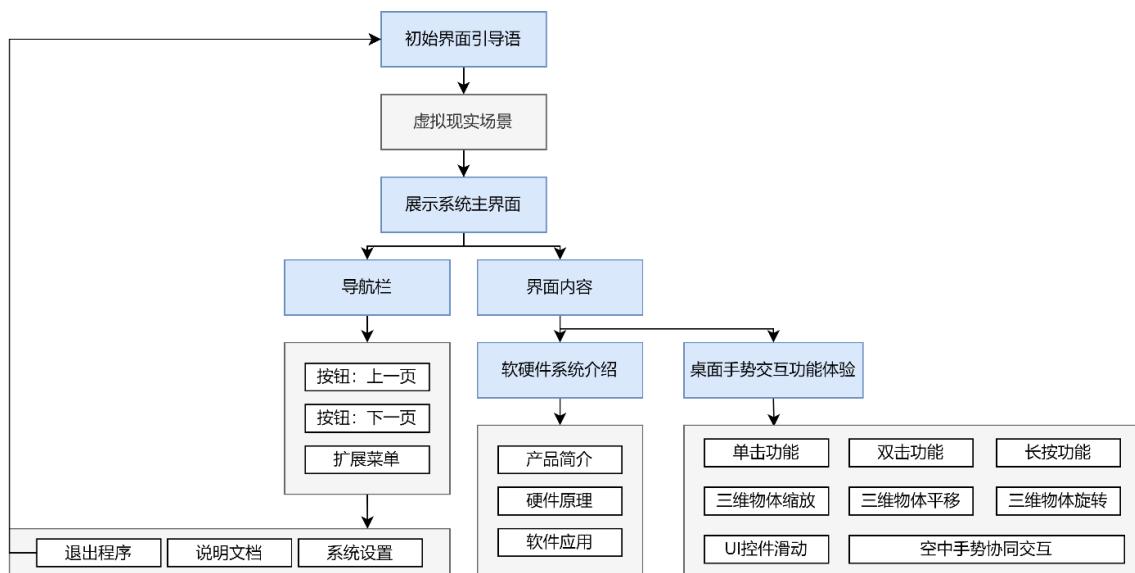


图 5-14 桌面手势交互功能展示系统界面信息框架

5.2.3 界面布局形式

虚拟现实的人机交互界面不同于传统二维人机界面，VR 界面不受屏幕矩形尺寸限制，而直接布局在无边际的三维虚拟空间中，故虚拟现实人机交互界面布局不能依赖传统二维人机界面设计范式，而应更多从人的行为习惯出发。针对虚拟现实坐姿作业状态，人的视觉行为习惯往往倾向于更低的自然视角、主要内容分布应在视野范围中央等。

在界面分布区域方面，虚拟现实头戴式显示器模拟人眼的生理习惯，将左右眼的视域进行叠加，由于头戴式显示器本身存在一定遮挡，所以用户在虚拟现实头戴式显示器中的视野范围相比与现实中的人眼视域更小，如图 5-15 所示。谷歌公司提出了 Google Daydream 虚拟现实界面设计规范，调研出用户在虚拟现实环境中的舒适视角大致为：上偏角 30°、下偏角 30°、左偏角 30°、右偏角 30° 的圆形视野范围，界面主要内容应分布在该视野范围内，以避免用户频繁转动头部获取信息。

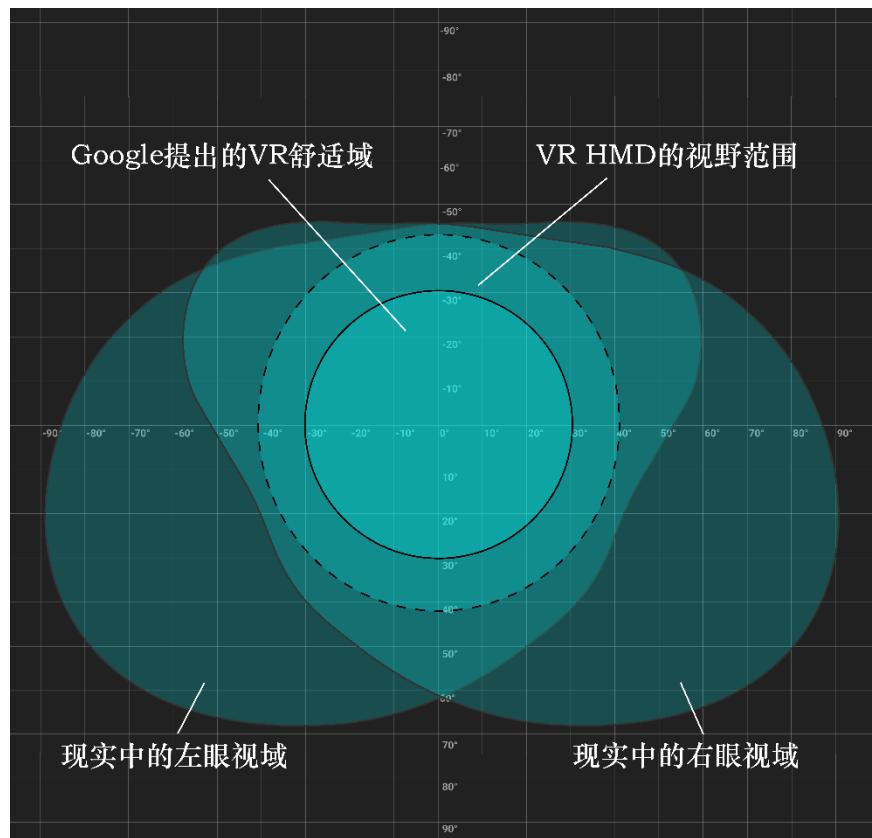


图 5-15 虚拟现实中的人眼舒适域

基于上述虚拟现实人机界面的特性和谷歌公司的 VR 舒适域，本课题通过 Figma 对桌面手势交互展示系统主界面进行初期布局，低保真图像如图 5-16 所示。

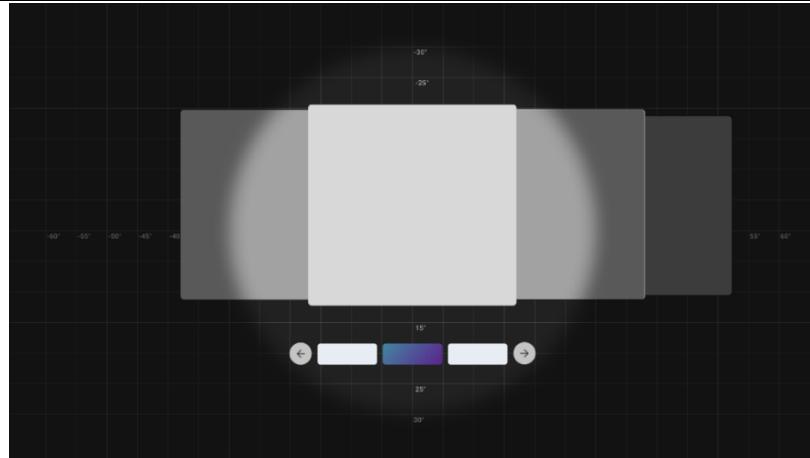


图 5-16 本课题展示系统的界面布局

展示系统的界面布局满足眼中心布局原则，主要内容显示在视野中央，并布局在人眼舒适域内。展示系统的主界面采用若干卡片组合成旋转木马式的交互效果，每张卡片上附带有各自的交互功能及其文字说明，用户可以根据文字说明的引导使用桌面手势完成交互任务。卡片与卡片之间大小深度不断变化，主页面卡片在最前端，其他页面卡片在后端且颜色更暗，从而构建交互界面的层次性，增强交互过程的沉浸感。导航栏作为次要内容布局在界面正下方，符合人更低视角的视觉行为习惯，确保用户在浏览主界面内容时更快速的利用导航栏导航。

5.2.4 界面最终效果

界面最终效果场景效果如图 5-17 所示。场景以实时旋转的星空为主要背景，构建沉浸式桌面手势交互氛围。在每个页面中都有可通过桌面手势交互的 UI 控件或三维物体。

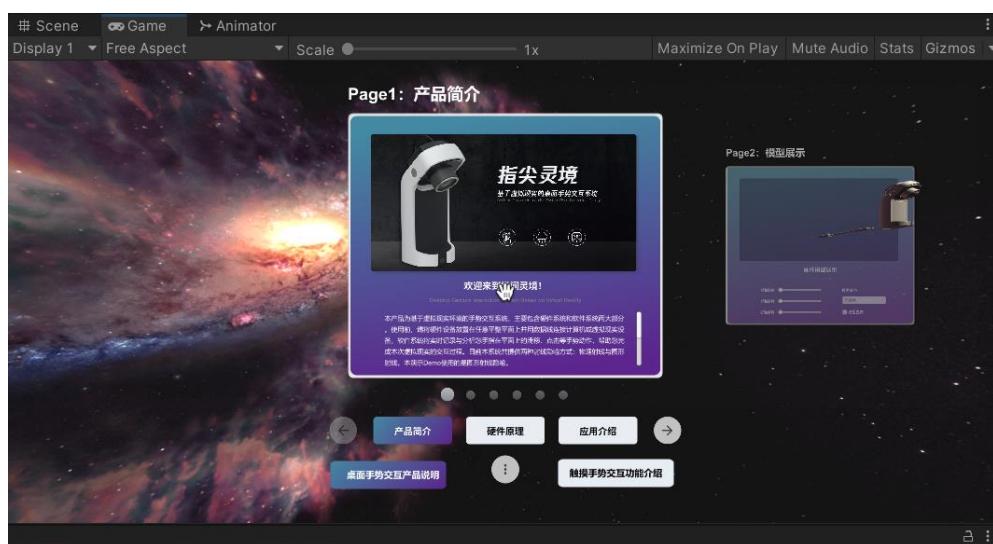


图 5-17 本课题展示系统的界面布局

(1) 导航栏交互控件效果

导航栏主要包含向上一页、向下一步的点击按钮和弹出扩展菜单的动态长按按键。上一页与下一页按钮采用左右方向键，通过方向性语义提醒用户按钮的功能，符合界面设计的概念相容性，当无上一页或下一页可滚动时，按钮置为不可用状态，相应的空间颜色变为灰色，透明度降低。弹出扩展菜单的长按按钮触发过程如图 5-18 所示，当用户控制滑移光标移入按钮范围且手指长按时，长按按钮周围有蓝色进度条表征目前的长按进度，长按进度到达阈值，通过 Unity 的 Animator 动画编辑器控制扩展菜单弹出动画，弹出扩展菜单，此时用户可以选择点击设置、说明文档和退出程序；当用户再次长按时，扩展菜单以同样的方式收回。扩展菜单的可视化动画流程图如图 5-19 所示。

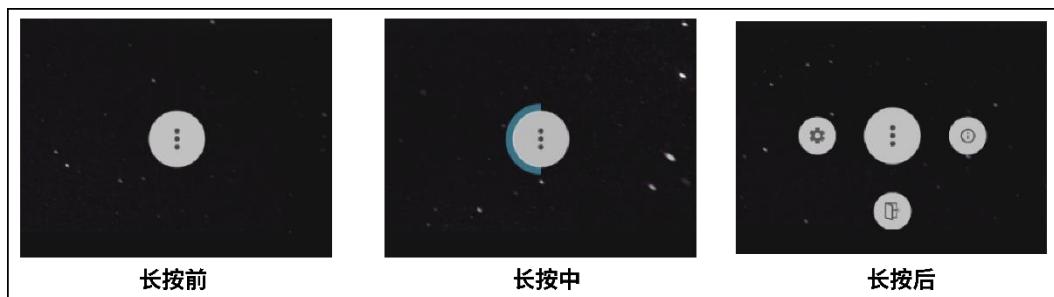


图 5-18 长按控件触发过程

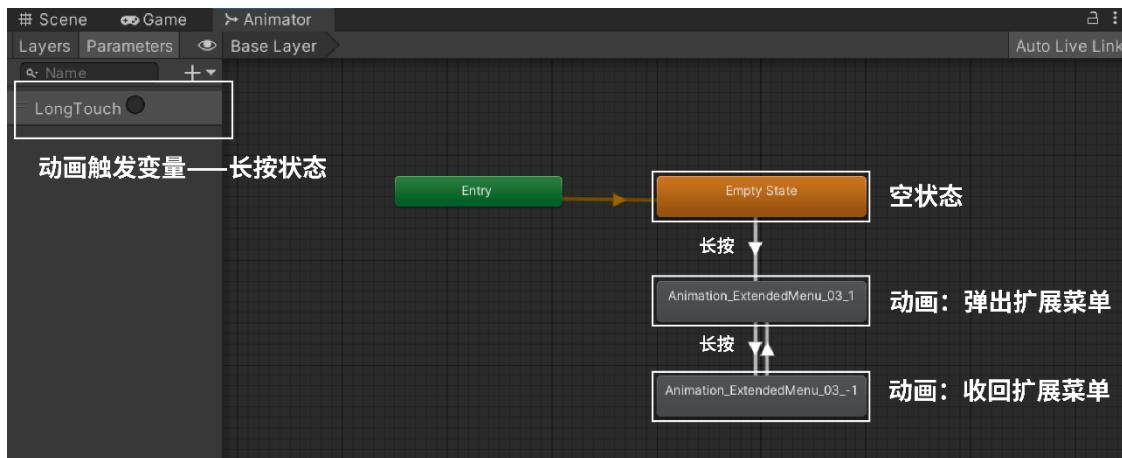


图 5-19 弹出扩展菜单的可视化动画流程图

(2) 主页面交互控件效果

展示系统主页面为旋转木马式交互页面，用户可以通过上一页、下一页按钮完成页面切换，也可以通过手指长按触发拖动状态，直接对页面拖动完成页面切换，如图 5-20 所示。



图 5-20 手指长按拖动页面中

主页面第一部分为桌面手势交互系统的系统简介、基本原理等说明性文字，以引导用户了解本系统的组成与基本操作方式。主页面第二部分为交互功能展示，用户可以与滚动条、框选槽、下拉菜单、下拉文本滚动条等 UI 控件交互，可以通过桌面手势控制三维物体进行选择、平移、旋转。可交互页面如图 5-21 所示，通过对滚动条的拖动，可实现三维模型各个方向上的旋转，通过下拉菜单选项完成模型着色，通过框选槽控制三维模型显示与隐藏，通过做出平移、旋转、放缩的桌面手势，可以实现模型的平移、旋转和放缩，通过空中手势可以将模型抓取到三维空间的任意位置。

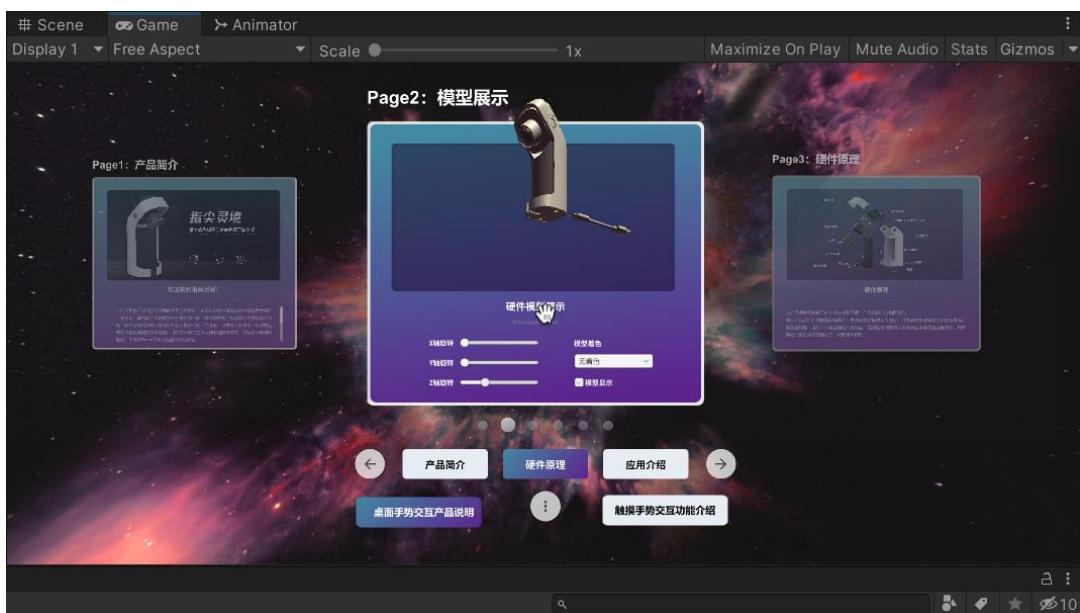


图 5-21 可交互页面中的 UI 控件与三维物体

5.3 本章小结

本章节基于前两章的软硬件系统，首先根据交互任务类型将桌面手势交互功能分为选择型、操纵型、与其他交互方式协同型三类交互功能，并从显控耦合性角度完成手势交互功能设计。在此之后，搭建桌面手势交互功能展示系统，通过设计虚拟现实人机交互界面，实现本课题桌面手势交互系统在虚拟现实界面中的应用实践，同时该展示系统也作为说明书向用户和开发者直观介绍本课题的原理及应用效果，便于用户更好地理解桌面手势交互的内涵。

第六章 总结与展望

6.1 总结

本课题从以空中手势交互技术为主的自然手势交互易疲劳、交互效率低等痛点出发，面向虚拟现实坐姿作业环境，设计开发了一种基于手指在桌面触摸的桌面手势交互软硬件系统，通过硬件设备设计、软件系统开发和桌面手势交互功能设计，实现了低疲劳、高效率、强灵活、多模态、适合长时间使用的交互效果，并最终借助该软硬件系统搭建桌面手势交互功能展示应用。本小节将总结本课题的主要成果与课题创新点。

6.1.1 课题主要内容

本课题所实现的主要工作内容包含以下几点：

- (1) 设计开发了基于红外激光投影技术的桌面手势交互硬件设备。该硬件设备可以高效率捕获手指在桌面的触摸状况，并且已完成产品级的样机生产加工，具有完整的产品造型和内部电路结构，可供所有需要进行桌面手势识别的开发者使用。
- (2) 探究了人主观感受舒适的桌面单手交互区域。课题通过开展人因实验探究用户主观舒适的桌面单手交互区域的大小和方位，为桌面手势设计的后来研究者提供数据参考。
- (3) 编写了基于计算机视觉的桌面手势识别算法。该算法有效剔除了桌面手势交互过程的动静态噪声干扰，实现准确的桌面手势识别。
- (4) 完成了桌面手势交互点校正方法。该方法基于相机成像原理，通过求解相机内参与相机外参，可以准确建立表征交互点图像位置的像素坐标与表征手指实际位置的世界坐标映射关系。
- (5) 完成了基于射线隐喻的桌面手势交互方法。该方法分别实现了基于 Unity3D 物理引擎的物理射线交互和基于 UGUI 系统的图形射线交互，并总结了两种射线交互特点，能够有效实现二维桌面手势对三维交互空间的控制过程。
- (6) 封装了基于虚拟现实的桌面手势交互预制体插件。该插件可以轻松实现在现有虚拟现实程序中添加桌面手势交互功能，也允许开发者对该插件进行二次开发。
- (7) 设计了一套具有强显控耦合性的桌面手势交互功能。该套功能基于任务类型进行详细的分类，实现了虚拟现实的交互目标选择、交互目标操纵、多种交互方式协同工作的交互任务，符合显示控制运动相容性的设计原则，具有良好的交互体验。

6.1.2 课题创新点

本课题在研究过程取得以下创新性成果：

(1) 提出了一种基于桌面手势的新型虚拟现实交互方式。传统虚拟现实交互技术虽然种类繁多，但相比现实中最常使用的触摸系统、键鼠系统，存在可用性较差、交互效率低、交互方式之间兼容性较差的问题。桌面手势交互将现实中的触摸输入同三维空间交互结合起来，通过小范围手指移动实现基本的交互任务，有效解决虚拟现实交互方式的痛点，提升了交互绩效，降低了交互疲劳，实现了与多种交互方式兼容共存，并为触摸手势在三维人机界面交互中的应用提供了指导价值。

(2) 设计了完整的桌面手势交互软硬件系统。本课题硬件系统已完成产品级样机加工，满足作为商品批量加工生产的条件；本课题软件系统核心算法已封装成预制体插件，可供其他开发者学习交流或为自己的虚拟现实程序添加桌面手势交互功能。软硬件系统协同工作状况良好，可以准确高效完成虚拟现实桌面手势交互过程。

(3) 设计了适用于虚拟现实的桌面手势集。通过对不同桌面手势交互功能设定对应的桌面手势，整理了满足虚拟现实基本交互需求的桌面二维手势集，为二维手势在虚拟现实环境中的应用提供参考。

6.2 展望

本课题已经完成了对于虚拟现实桌面手势交互技术的初步探索，并且已经实现了上述的研究成果。在硬件系统方面，课题还可以进一步实现硬件系统无线化，将软件系统集成到硬件中去。在软件系统方面，桌面手势识别程序基于传统的计算机图像处理框架，可以结合目前火热的机器学习实现更高准确度的手势识别；桌面手势交互点校正方法目前基于固定镜头、通过人工计算的传统方法进行校正，在面向动态相机进行自动化校正方法还有待探索。在手势设计方面，课题提供的三类桌面手势交互功能接下来需要通过人因实验进行交互绩效评估，建立一种合适的桌面手势评价方法。在未来应用方面，桌面手势交互除去和空中手势协同交互，还可以和眼控技术完成手眼协同技术，可以构建协同桌面手势、空中手势、眼控、语音、控制器的多模态人机交互系统。因此，基于虚拟现实的桌面手势交互系统还存在着巨大的开拓空间，可以从如下几个方面继续进行更深入的研究工作：

(1) 优化软硬件系统。对硬件系统进行升级，实现智能化、无线化、集成化的桌面手势交互设备，将软件算法通过单片机等形式集成到硬件设备中去，并建立蓝牙通讯；通

过搜集桌面手势训练集，搭建神经网络，通过机器学习的方式进一步提升桌面手势识别的准确性；通过编写桌面手势交互点动态校正算法。基于现有的图像成像原理和计算机图像处理工具，建立自动化的、针对动态镜头的桌面手势点校正方法。

（2）进行桌面手势交互效果的绩效评估实验并构建桌面手势评估方法。通过人因实验评估目前的桌面手势交互效果，对于桌面手势库进行更新和改进。

参考文献

- [1] Berg L P, Vance J M. Industry use of virtual reality in product design and manufacturing: a survey[J]. *Virtual reality*, 2017, 21(1): 1-17.
- [2] Ma F, Song F, Liu Y, et al. Quantitative Analysis on the Interaction Fatigue of Natural Gestures[J]. *IEEE Access*, 2020, 8: 190797-190811.
- [3] Zhang C, Zheng C X, Yu X L. Automatic recognition of cognitive fatigue from physiological indices by using wavelet packet transform and kernel learning algorithms[J]. *Expert Systems with Applications*, 2009, 36(3): 4664-4671.
- [4] 张凤军, 戴国忠, 彭晓兰. 虚拟现实的人机交互综述[J]. 中国科学:信息科学, 2016, 046(012): 1711-1736.
- [5] Koutsabasis P, Vogiatzidakis P. Empirical research in mid-air interaction: A systematic review[J]. *International Journal of Human–Computer Interaction*, 2019, 35(18): 1747-1768.
- [6] Goh E S, Sunar M S, Ismail A W. 3D object manipulation techniques in handheld mobile augmented reality interface: A review[J]. *IEEE Access*, 2019, 7: 40581-40601.
- [7] Laviola Jr J J, Kruijff E, McMahan R P, et al. 3D user interfaces: theory and practice[M]. Addison-Wesley Professional, 2017.
- [8] Székely G, Satava R M. Virtual reality in medicine[J]. *BMJ: British Medical Journal*, 1999, 319(7220): 1305.
- [9] Tingting Z, Feng T, Wei L. Survey of VR Application in Interactive Films and Games[J]. *Journal of Shanghai University (Natural Science)*, 2017, 23(3): 342-352.
- [10] Soliman M, Pesyridis A, Dalaymani-Zad D, et al. The application of virtual reality in engineering education[J]. *Applied Sciences*, 2021, 11(6): 2879.
- [11] 王崴, 李恒威, 刘海平, 等. 混合现实技术在军事中的应用综述[J]. *兵器装备工程学报*, 2021, 42(09): 15-25.
- [12] Berg L P, Vance J M. Industry use of virtual reality in product design and manufacturing: a survey[J]. *Virtual reality*, 2017, 21(1): 1-17.
- [13] Çöltekin A, Lochhead I, Madden M, et al. Extended reality in spatial sciences: A review of research challenges and future directions[J]. *ISPRS International Journal of Geo-Information*, 2020, 9(7): 439.
- [14] 郭伏, 钱省三. 人因工程学[M]. 北京:机械工业出版社, 2018: 314-327.

- [15] Zhao W. A concise tutorial on human motion tracking and recognition with Microsoft Kinect[J]. *Science China Information Sciences*, 2016, 59(9): 1-5.
- [16] Wozniak P, Vauderwange O, Mandal A, et al. Possible applications of the LEAP motion controller for more interactive simulated experiments in augmented or virtual reality[C]. In: Optics Education and Outreach IV. California: International Society for Optics and Photonics, 2016, 9946: 99460P.
- [17] Han S, Liu B, Cabezas R, et al. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality[J]. *ACM Transactions on Graphics (TOG)*, 2020, 39(4): 87: 1-87: 13.
- [18] Meier M, Streli P, Fender A, et al. TapID: Rapid Touch Interaction in Virtual Reality using Wearable Sensing. In: 2021 IEEE Virtual Reality and 3D User Interfaces (VR). Lisbon: IEEE, 2021: 519-528.
- [19] Yu D, Liang H N, Lu F, et al. Target Selection in Head-Mounted Display Virtual Reality Environments [J]. *J. Univers. Comput. Sci.*, 2018, 24(9): 1217-1243.
- [20] Kharlamov D, Woodard B, Tahai L, et al. TickTockRay: smartwatch-based 3D pointing for smartphone-based virtual reality[C]. In: Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology. 2016: 365-366.
- [21] Knierim P, Schwind V, Feit A M, et al. Physical keyboards in virtual reality: Analysis of typing performance and effects of avatar hands[C]. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 2018: 1-9.
- [22] Cho I, Wang X, Wartell Z J. HyFinBall: a two-handed, hybrid 2D/3D desktop VR interface for multi-dimensionaz visualization[C]. In: Visualization and Data Analysis 2014. Washington: SPIE, 2014, 9017: 150-163.
- [23] Clay V, König P, Koenig S. Eye tracking in virtual reality[J]. *Journal of eye movement research*, 2019, 12(1).
- [24] Siu A F, Sinclair M, Kovacs R, et al. Virtual reality without vision: A haptic and auditory white cane to navigate complex virtual worlds[C]. In: Proceedings of the 2020 CHI conference on human factors in computing systems. 2020: 1-13.
- [25] Rincon A L, Yamasaki H, Shimoda S. Design of a video game for rehabilitation using motion capture, EMG analysis and virtual reality[C]. In: 2016 International Conference on Electronics, Communications and Computers (CONIELECOMP). Cholula: IEEE, 2016: 198-204.
- [26] Bekele E, Wade J, Bian D, et al. Multimodal adaptive social interaction in virtual environment (MASI-VR)

- for children with Autism spectrum disorders (ASD)[C]. In: 2016 IEEE virtual reality (VR). Greenville: IEEE, 2016: 121-130.
- [27] Jacob R J K. What you look at is what you get: eye movement-based interaction techniques[C]. In: Proceedings of the SIGCHI conference on Human factors in computing systems. 1990: 11-18.
- [28] Martinez J, Griffiths D, Biscione V, et al. Touchless haptic feedback for supernatural VR experiences[C]. In: 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). Reutlingen: IEEE, 2018: 629-630.
- [29] Born F, Masuch M, Hahn A. Ghost sweeper: Using a heavy passive haptic controller to enhance a room-scale vr exergame[C]. In: 2020 IEEE Conference on Games. Osaka: IEEE, 2020: 221-228.
- [30] Lu S, Cai L, Ding X, et al. A combined strategy of hand tracking for desktop VR[C]. In: Pacific Rim Conference on Multimedia. Hefei: Springer, Cham, 2018: 256-269.
- [31] Li G, Rempel D, Liu Y, et al. Design of 3D Microgestures for Commands in Virtual Reality or Augmented Reality[J]. Applied Sciences, 2021, 11(14): 6375.
- [32] Li X, Han F, Sun X, et al. Bracelet: arms-down selection for Kinect mid-air gesture[J]. Behaviour & Information Technology, 2019, 38(4): 401-409.
- [33] Cheema N, Frey-Law L A, Naderi K, et al. Predicting mid-air interaction movements and fatigue using deep reinforcement learning[C]. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. 2020: 1-13.
- [34] Iqbal H, Latif S, Yan Y, et al. Reducing arm fatigue in virtual reality by introducing 3D-spatial offset[J]. IEEE Access, 2021, 9: 64085-64104.
- [35] Sheng Q, Liu T, Hou W, et al. The Research on Touch Gestures Interaction Design for Personal Portable Computer[C]. In: International Conference on Human Centered Computing. Kazan: Springer, Cham, 2017: 527-537.
- [36] Yun L, Yufen C, Jinliang K. Research on touch gesture based HCI of mobile device cartographic visualization system[C]. In: Proceedings of the 2011 International Conference on Information , Services and Management Engineering(ISME 2011) (Volume 3). Beijing: Scientific Research Publishing, 2011: 171-174.
- [37] Suto S, Watanabe T, Shibusawa S, et al. Multi-touch tabletop system using infrared image recognition for user position identification[J]. Sensors, 2018, 18(5): 1559.

- [38] Martinet A, Casiez G, Grisoni L. The design and evaluation of 3d positioning techniques for multi-touch displays[C]. In: 2010 IEEE symposium on 3D user interfaces (3DUI). Waltham: IEEE, 2010: 115-118.
- [39] Hancock M S, Vernier F D, Wigdor D, et al. Rotation and translation mechanisms for tabletop interaction[C]. In: First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'06). Adelaide: IEEE, 2006: 8 pp.
- [40] Cohé A, Dècle F, Hatchet M. Tbox: a 3d transformation widget designed for touch-screens[C]. In: Proceedings of the sigchi conference on human factors in computing systems. 2011: 3005-3008.
- [41] Liu J, Au O K C, Fu H, et al. Two - finger gestures for 6DOF manipulation of 3D objects[C]. In: Computer Graphics Forum. Oxford: Blackwell Publishing Ltd, 2012, 31(7): 2047-2055.
- [42] 冯振, 郭延宁, 吕跃勇. OpenCV4 快速入门[M]. 北京:人民邮电出版社, 2020: 27-52.
- [43] Krig S. Computer vision metrics: Survey, taxonomy, and analysis[M]. Heidelberg:Springer nature, 2014: 29-62.
- [44] Tsang S N H, Ho J K L, Chan A H S. Interface design and display-control compatibility[J]. Measurement and Control, 2015, 48(3): 81-86.
- [45] Wickens C D, Helton W S, Hollands J G, et al. Engineering psychology and human performance[M]. New York:Routledge, 2021: 61-87.
- [46] Wolf D, Gugenheimer J, Combosch M, et al. Understanding the heisenberg effect of spatial interaction: A selection induced error for spatially tracked input devices[C]. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. 2020: 1-10.

附录 A 桌面手势识别脚本源码

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

#if UNITY_5_3 || UNITY_5_3_OR_NEWER
using UnityEngine.SceneManagement;
#endif

using OpenCVForUnity;

public class GetTouchPoint2 : MonoBehaviour
{
    /// <summary>
    /// 获取相机图像
    /// </summary>
    static public WebCamTexture webCam;

    /// <summary>
    /// 获取设备名称
    /// </summary>
    public string deviceName;

    /// <summary>
    /// 定义第一帧图像、原图像、剔除第一帧背景的图像、灰度图像、二值化图像、提取
    /// 轮廓
    /// </summary>
    public Mat firstFrameMat;
    public Mat rgbMat;
    public Mat diffMat;
    public Mat grayMat;
    public Mat binarizeMat;
    public Mat hierarchy;
```

```

/// <summary>
/// 相机帧率
/// </summary>
public int fps = 30;

/// <summary>
/// 记录帧数，第 30 帧时作为相机背景
/// </summary>
public int timeByFrame = 0;

/// <summary>
/// 桌面手势交互点信息
/// </summary>
public TouchInformation touchInformation;

//初始化程序
void Start()
{
    //设置相机帧率，获取相机图像
    Time.captureFramerate = fps;
    WebCamDevice[] devices = WebCamTexture.devices;
    Debug.Log($"当前设备数: {devices.Length}");
    deviceName = devices[1].name;
    webCam = new WebCamTexture(deviceName);
    webCam.Play();
}

//每帧运行程序
void Update()
{
    //初始化桌面手势交互点信息
    touchInformation.touchNum = 0;
    touchInformation.centerPoint = new List<Point>();
    touchInformation.time = Time.time;
    touchInformation.area = new List<double>();
}

```

```
//运行桌面手势识别程序，进行计算机图像处理
ImageProcessing();
}

//桌面手势识别程序，负责计算机图像处理，并获取手势交互点信息
public void ImageProcessing()
{
    //待图像稳定后记录第一帧
    if (timeByFrame == 30)
    {
        firstFrameMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
        Utils.webCamTextureToMat(webCam, firstFrameMat);
    }

    if (timeByFrame > 30)
    {
        //原始图像转为 OpenCV 中的 Mat 类
        rgbMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
        Utils.webCamTextureToMat(webCam, rgbMat);

        //帧差法去除背景
        diffMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
        Core.absdiff(rgbMat, firstFrameMat, diffMat);

        //彩色图像灰度化
        grayMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
        Imgproc.cvtColor(diffMat, grayMat, Imgproc.COLOR_BGR2GRAY);

        //灰度图像二值化
        Imgproc.blur(grayMat, grayMat, new Size(3, 3));
        binarizeMat = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
        Imgproc.threshold(grayMat, binarizeMat, 150, 255, Imgproc.THRESH_BINARY);

        //提取二值化图像的光点轮廓
    }
}
```

```

hierarchy = new Mat(webCam.height, webCam.width, CvType.CV_8UC3);
List<MatOfPoint> contours = new List<MatOfPoint>();
Imgproc.findContours(binarizeMat, contours, hierarchy, Imgproc.RETR_LIST,
Imgproc.CHAIN_APPROX_SIMPLE);

//如果无轮廓
if (contours.Count == 0)
{
    touchInformation.touchNum = 0;
    touchInformation.centerPoint = null;
    touchInformation.area = null;
}

//如果有轮廓
for (int i = 0; i < contours.Count; i++)
{
    //获取轮廓面积
    var contourArea = Imgproc.contourArea(contours[i]);

    //剔除小面积轮廓
    if (contourArea > 1000)
    {
        //获取外包矩形，计算外接矩形长宽比
        var _rect = Imgproc.boundingRect(contours[i]);
        var leftOnPoint = new Point(_rect.x, _rect.y);
        var rightDownPoint = new Point(_rect.x + _rect.width, _rect.y +
_rect.height);
        double aspectRatio = _rect.width / _rect.height;

        //获取轮廓外接圆，计算轮廓与外接圆面积比
        var _ellipse = Imgproc.fitEllipseAMS(contours[i]);
        var ellipseArea = _ellipse.size.area();
        double ellipseAreaContourRatio = ellipseArea / contourArea;
    }
}

```

```
//对外接矩形长宽比、轮廓外接圆面积比设定阈值，剔除干扰
if (aspectRatio < 3.3 && aspectRatio > 0.3 && ellipseArea
ContourRatio < 1.5)
{
    //获取中心点
    Point point = new Point(_rect.x + _rect.width) / 2, (_rect.y +
_rect.height) / 2);
    double area = contourArea;
    touchInformation.centerPoint.Add(point);
    touchInformation.touchNum++;
    touchInformation.area.Add(area);
    //在原图像上绘制桌面手势外接矩形：
    Imgproc.rectangle(binarizeMat, leftOnPoint, rightDownPoint, new
Scalar(0, 0, 255), 3);
    //在原图像上绘制桌面手势中心点：
    Imgproc.circle(binarizeMat, centerPoint, 25, new Scalar(0, 255, 0),
-1);
}
}
timeByFrame++;
}
}
```

附录 B 桌面手势交互点速度求解源码

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GetTouchSpeed2 : MonoBehaviour
{
    /// <summary>
    /// 记录当前帧交互点信息
    /// </summary>
    public TouchInformation currentTouchInformation;

    /// <summary>
    /// 记录上一帧交互点信息
    /// </summary>
    public TouchInformation lastTouchInformation;

    /// <summary>
    /// 记录当前帧和上一帧之间的面积比，判断是否是按下和抬起的过程
    /// </summary>
    public double areaRatio;

    /// <summary>
    /// 记录当前帧触摸点移动速度
    /// </summary>
    public float touchSpeed_x;
    public float touchSpeed_y;

    // 初始化脚本
    void Start()
    {
        touchSpeed_x = 0;
        touchSpeed_y = 0;
        areaRatio = 0;
```

```
}

//每帧运行
void Update()
{
    currentTouchInformation = this.GetComponent<GetTouchPoint2>().touchInformation;
    if (currentTouchInformation.touchNum == 1 && lastTouchInformation.touchNum ==
1)
    {
        //手指 x 方向移动速度
        touchSpeed_x      =      (float)(currentTouchInformation.centerPoint[0].x -
lastTouchInformation.centerPoint[0].x) / Time.deltaTime;
        //手指 y 方向移动速度
        touchSpeed_y      =      (float)(currentTouchInformation.centerPoint[0].y -
lastTouchInformation.centerPoint[0].y) / Time.deltaTime;
        if (areaRatio > 1.1 && areaRatio < 0.9)
        {
            touchSpeed_x = 0;
            touchSpeed_y = 0;
        }
        else
        {
            touchSpeed_x = 0;
            touchSpeed_y = 0;
            //areaRatio = 0;
        }
        lastTouchInformation = currentTouchInformation;
    }
}
```

附录 C 物理射线隐喻交互源码

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

#if UNITY_5_3 || UNITY_5_3_OR_NEWER
using UnityEngine.SceneManagement;
#endif

using OpenCVForUnity;

public class RayTest : MonoBehaviour
{
    /// <summary>
    /// 记录当前帧交互点信息
    /// </summary>
    public TouchInformation currentTouchInformation;

    /// <summary>
    /// 记录射线指向的水平角和竖直角
    /// </summary>
    public float horizontalAngle;
    public float verticalAngle;
    public float touchSpeed_x;
    public float touchSpeed_y;

    /// <summary>
    /// 射线渲染
    /// </summary>
    public LineRenderer lineRenderer;

    /// <summary>
    /// 射线碰撞点
    /// </summary>
    public Vector3 hitPoint;
```

```
/// <summary>
/// 记录射线所指向游戏对象
/// </summary>
public string hitName;
public int hitLayer;

// 初始化脚本
void Start()
{
    //射线默认指向视野正中央
    horizontalAngle = Mathf.PI / 2;
    verticalAngle = Mathf.PI / 2;
    lineRenderer = this.GetComponent<LineRenderer>();
    Vector3 startLine = new Vector3(0, -10, 5);
    Vector3 endLine = new Vector3(0, 0, 500);
    lineRenderer.SetPosition(0, startLine);
    lineRenderer.SetPosition(1, endLine);
    lineRenderer.startColor = Color.red;
    lineRenderer.endColor = Color.red;
    lineRenderer.startWidth = 0.5f;
    lineRenderer.endWidth = 0.5f;

    hitName = null;
    hitLayer = -1;
    touchSpeed_x = 0;
    touchSpeed_y = 0;
}

//每帧运行
void Update()
{
    currentTouchInformation = this.GetComponent<GetTouchPoint2>().touchInformation;
    touchSpeed_x = this.GetComponent<GetTouchSpeed2>().touchSpeed_x;
    touchSpeed_y = this.GetComponent<GetTouchSpeed2>().touchSpeed_y;
}
```

```

if (currentTouchInformation.touchNum == 1)
{
    Vector3 vector = new Vector3(0,0,0);
    //水平角随手指移动的水平速度变化
    horizontalAngle += (-1) * touchSpeed_x * 0.1f * Mathf.PI / 1920;
    if(horizontalAngle >= 3 * Mathf.PI / 4)
    {
        horizontalAngle = 3 * Mathf.PI / 4;
    }
    if (horizontalAngle <= Mathf.PI / 4)
    {
        horizontalAngle = Mathf.PI / 4;
    }

    //竖直角随手指移动的竖直速度变化
    verticalAngle += touchSpeed_y * 0.1f * Mathf.PI / 1080;
    if (verticalAngle >= 3 * Mathf.PI / 4)
    {
        verticalAngle = 3 * Mathf.PI / 4;
    }
    if (verticalAngle <= Mathf.PI / 4)
    {
        verticalAngle = Mathf.PI / 4;
    }

    vector.x = 10000*Mathf.Sin(verticalAngle) * Mathf.Cos(horizontalAngle);
    vector.y = 10000*Mathf.Cos(verticalAngle);
    vector.z = 10000*Mathf.Sin(verticalAngle) * Mathf.Sin(horizontalAngle);

    //投射物理射线
    var ray = new Ray(this.GetComponent<Transform>().position, vector);

    //渲染物理射线
    lineRenderer.SetPosition(1, vector);
}

```

```

//返回射线投射碰撞结果
RaycastHit hit;
if(Physics.Raycast(ray,out hit)) //如果投射射线产生碰撞
{
    lineRenderer.SetPosition(1, hit.point);
    GameObject.Find("Cursor").transform.position = hit.point;
    SpriteRenderer spr = GameObject.Find("Cursor").GetComponent<
SpriteRenderer>();

    //改变光标状态
    spr.sprite = Resources.Load("Icon/Cursor2", typeof(Sprite)) as Sprite;
    hitName = hit.transform.name;
    if(hit.transform.gameObject.layer == 0)
    {
        //Debug.Log("光标移到 3D 物体");
        hitLayer = 0;
    }
    if(hit.transform.gameObject.layer == 5)
    {
        //Debug.Log("光标移到 UI");
        hitLayer = 5;
    }
}
else
{
    GameObject.Find("Cursor").transform.position = new Vector3(vector.x *
(500 / vector.z), vector.y * (500 / vector.z), 499);
    SpriteRenderer spr = GameObject.Find("Cursor").GetComponent<
SpriteRenderer>();

    spr.sprite = Resources.Load("Icon/Cursor", typeof(Sprite)) as Sprite;
    hitName = null;
    hitLayer = -1;
}
}
}
}
}

```

附录 D 图形射线隐喻交互源码

```
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.EventSystems;
using UnityEngine.UI;
using System.Collections.Generic;

public class RayTestWithEventSystem : MonoBehaviour
{
    /// <summary>
    /// 光标坐标
    /// </summary>
    public float cursor_x;
    public float cursor_y;

    /// <summary>
    /// 触摸手指数量
    /// </summary>
    public int curTouchNum;
    public int lastTouchNum;

    /// <summary>
    /// UI 选择状态
    /// </summary>
    Selectable m_excluded;
    Selectable m_currentSelectable;

    /// <summary>
    /// 图形射线投射结果
    /// </summary>
    RaycastResult m_currentRaycastResult;

    /// <summary>
    /// 检查 UI 交互过程中点击、拖拽事件
    /// </summary>
```

```
/// </summary>
IPointerClickHandler m_clickHandler;
IDragHandler m_dragHandler;

/// <summary>
/// 事件系统
/// </summary>
EventSystem m_eventSystem;

/// <summary>
/// 按钮点击事件
/// </summary>
PointerEventData m_pointerEvent;

/// <summary>
/// 手指长按事件、当前长按时间、长按判定时间
/// </summary>
public bool holdDownState;
public float m_holdDownTime;
public float holdDownTime = 2f;

/// <summary>
/// 手指单击事件
/// </summary>
public bool singleHitState;

/// <summary>
/// 手指双击事件
/// </summary>
public bool doubleHitState;

/// <summary>
/// 光标上一帧位置
/// </summary>
public float lastCursor_x;
```

```

public float lastCursor_y;

//初始化脚本
void Start()
{
    //初始化事件系统为当前系统、按钮点击事件为空、触发点击为鼠标左键
    m_eventSystem = EventSystem.current;
    m_pointerEvent = new PointerEventData(m_eventSystem);
    m_pointerEvent.button = PointerEventData.InputButton.Left;
    //初始化光标位置为视野中心
    cursor_x = this.GetComponent<CursorPosition>().cursor_x;
    cursor_y = this.GetComponent<CursorPosition>().cursor_y;
    lastCursor_x = cursor_x;
    lastCursor_y = cursor_y;
    holdDownState = false;
}

// Update is called once per frame
void Update()
{
    //从其他脚本中获取：当前手指按下数量、手指长按状态、手指单击状态、手指
    双击状态、光标位置
    curTouchNum=this.GetComponent<GetTouchPoint2>().touchInformation.touch Num;
    singleHitState = this.GetComponent<SingleHitDetection>().singleHitState;
    doubleHitState = this.GetComponent<DoubleHitDetection>().doubleHitState;
    cursor_x = this.GetComponent<CursorPosition>().cursor_x;
    cursor_y = this.GetComponent<CursorPosition>().cursor_y;
    //显示光标
    GameObject.Find("Cursor").GetComponent<RectTransform>().anchoredPosition =
    new Vector2(cursor_x, cursor_y);
    GameObject.Find("Cursor").GetComponent<RawImage>().texture = Resources.Load
    ("Icon/State or Style=Grab", typeof(Texture2D)) as Texture2D;
    m_pointerEvent.position = new Vector2(cursor_x, cursor_y);
}

```

```

//如果单击点击
if(singleHitState)
{
    m_pointerEvent.position=new Vector2(this.GetComponent<SingleHitDetection>()
().lastLoseTouchCursor_x, this.GetComponent<SingleHitDetection>().lastLoseTouchCursor_y);
}

//如果双击点击
if (doubleHitState)
{
    m_pointerEvent.position=new Vector2(this.GetComponent<DoubleHitDetection>()
().lastLoseTouchCursor_x, this.GetComponent<DoubleHitDetection>().lastLoseTouchCursor_y);
}

//投射图形射线并获取投射结果
List<RaycastResult> raycastResults = new List<RaycastResult>();
m_eventSystem.RaycastAll(m_pointerEvent, raycastResults);

//射线投射结果
if (raycastResults.Count > 0)
{
    foreach (var result in raycastResults)
    {

        var newSelectable = result.gameObject.GetComponentInParent<Selectable>()
();
        if (newSelectable)
        {
            if (newSelectable != m_excluded && newSelectable != m_currentSelectable)
            {
                Select(newSelectable);//鼠标移入
                m_currentRaycastResult = result;
            }
            break;
        }
    }
}

```

```

        }

    }

else

{

    if (m_currentSelectable || m_excluded)

    {

        Select(null, null); //鼠标移出
        Debug.Log("鼠标移出");

    }

}

//判断长按状态

HoldDown(m_currentSelectable, curTouchNum);

// Target is being activating

if (m_currentSelectable)

{

    GameObject.Find("Cursor").GetComponent<RawImage>().texture = Resources.Load("Icon/State or Style=Pointer", typeof(Texture2D)) as Texture2D;

    if (curTouchNum == 2 && lastTouchNum == 1) //如果两根手指点击
        //if(singleHitState) //如果单指单击
        //if(doubleHitState) //如果双指单击
    {

        if (m_clickHandler != null)
        {

            m_clickHandler.OnPointerClick(m_pointerEvent);
            Debug.Log("触发点击事件");

        }

        else if (m_dragHandler != null)
        {

            m_pointerEvent.pointerPressRaycast = m_current.RaycastResult;
            m_dragHandler.OnDrag(m_pointerEvent);

        }

    }

    if(holdDownState) //如果单指长按
}

```

```

    {
        GameObject.Find("Cursor").GetComponent<RawImage>().texture      =
Resources.Load("Icon/State or Style=Grabbing", typeof(Texture2D)) as Texture2D;
        if (m_dragHandler != null)
        {
            m_pointerEvent.pointerPressRaycast = m_currentRaycastResult;
            m_dragHandler.OnDrag(m_pointerEvent);
        }
    }

    lastTouchNum = curTouchNum;
    lastCursor_x = cursor_x;
    lastCursor_y = cursor_y;
}

//判断光标的移入和移出
void Select(Selectable s, Selectable exclude = null)
{
    m_excluded = exclude;
    if (m_currentSelectable)
        m_currentSelectable.OnPointerExit(m_pointerEvent);
    m_currentSelectable = s;
    if (m_currentSelectable)
    {
        m_currentSelectable.OnPointerEnter(m_pointerEvent);
        m_clickHandler = m_currentSelectable.GetComponent<IPointerClickHandler>();
        m_dragHandler = m_currentSelectable.GetComponent<IDragHandler>();
    }
}

//判断是否长按
void HoldDown(bool m_currentSelectable,int touchNum)
{
    if (m_currentSelectable && touchNum == 1)
    {

```

```
m_holdDownTime += Time.deltaTime;  
if (m_holdDownTime > holdDownTime)  
{  
    holdDownState = true;  
    Debug.Log("判定长按");  
}  
}  
else  
{  
    m_holdDownTime = 0;  
    holdDownState = false;  
}  
}  
}
```

致 谢

历时半年的毕业设计，历时四年的大学生活，历时十六年的求学生涯，一切都向着收尾奔去，恬惶的书生生涯渐渐到了中转站。回首毕业设计的时光，以及整段求学旅途，还有漫长又短暂的二十载人生，有一些话哽咽在咽喉，有一些感念激荡在胸口，难以用语言说出。就借此机会，向过去所有人和事道声感谢，背上行李继续准备接下来更漫长的路。

首先感谢我的指导老师周小舟老师。从第一次面对选题的手足无措，到渐渐懂得该怎么去做，再到最后有些许成果，我的毕业设计整个过程都离不开周老师的指导与帮助。感谢周老师在毕业设计过程迷茫时给我指明道路，感谢周老师在我参加互联网+比赛时给我的鼓励，感谢周老师在毕业设计硬件加工中为我忙里忙外，感谢周老师在百忙之中不厌其烦地解答我的每一个疑惑，感谢周老师忍受我丑陋不堪的设计并细心给我指导意见，感谢周老师带我第一次接触科研、第一次认识课题组的朋友们、第一次进入设计领域。从第一次见您也快有一年的时间了，一年来您的帮助和支持我终生难忘，谢谢您，周老师！

感谢为我的毕业设计硬件奔波忙碌的宗承龙学长。帮助从我零做一件产品级硬件绝非一件容易的事，感谢宗承龙学长从教我硬件的建模渲染、再到联系厂商加工、再到和工程师对接、再到产品最终装配喷漆整个过程的日夜辛劳。谢谢您，宗承龙学长！

感谢课题组的所有伙伴。感谢王逸雪学姐帮助我界面设计熬下的若干夜晚，感谢贾乐松学神为我 Unity 学习过程中解答的多项疑惑，感谢杜晓茜学神为我修改开题报告，感谢肖玮烨学长陪我唠嗑、帮助我使用 Oculus 并为我打开计算机图形学的大门，感谢小白学长为我的毕设拍摄的美妙照片和 UI 设计上的指导，感谢周雨晴学姐、徐君学姐为我引导界面设计中的思路方法，感谢郭一冰学姐、韩已臣学长和滕菲学姐教我正确使用筷子，虽然我到今天还是不太会用，但我会努力，感谢李荷露学姐解答我关于交互疲劳方面的问题，感谢曹源学姐、陈凯学长、秦昊学长、王柳清学长、谭涛涌学长为我一同分担毕业季的焦虑与烦恼，另外希望王柳清学长的腿以后要健健康康的，感谢景惠美学姐和李雅晴学姐解答我的各种问题。感谢 354 和校外工作室的所有朋友给我带来的欢乐和成长，谢谢你们！

感谢陪同我一起做毕设的小伙伴和我一起沟通学习，并忍受每天由我散布的焦虑，我今后也会更加克制自己的焦虑情绪。感谢章工为我装配硬件，在出现 1mm 干涉时用磨刀替我去料的整个上午。谢谢你们！

感谢东南大学机械工程学院四年对我的培养，感谢四年中遇见的所有老师、所有同

学，谢谢你们！

最后，感谢我的亲人。感谢我的爸爸妈妈、爷爷奶奶、外公外婆，感谢童年陪伴我的猫猫和狗狗，感谢我的故乡，有时候真的希望我能挡得住时间的巨浪，希望时间能停下，能再过得慢一些，但一个人总是要长大，总会变老。谢谢你们，你们是我一生永恒的光芒！