**大连理工大学本科毕业设计（论文）**

**基于头戴式VR设备的视觉延伸系统**

**A Vision Extension System Based on Virtual Reality Head Mount Display**

学 院（系）：电子信息与电气工程学部

专 业： 计算机科学与技术

学 生 姓 名： 杜佑宸

学 号： 201281476

指 导 教 师： 冯林

评 阅 教 师：

完 成 日 期： 2016年5月24日

大连理工大学

Dalian University of Technology

# 摘 要

虚拟现实的飞速发展，进一步增强了与现有各应用领域的结合性。将虚拟现实和摄像头拍摄结合在国内外虚拟现实领域、增强现实领域引起了大批研究者的兴趣并投入了大量的精力。但目前存在的一个问题是，大部分解决方案中视频采集设备与搭载平台过于紧密结合，无法作为单模块进行推广。

本文试图设计一种模块化的、方便被各种平台搭载的实时视觉延伸系统。通过两个步进电机为视频采集终端提供两个转动自由度，使用显示终端的陀螺仪数据与视频采集终端的陀螺仪数据之间的差异来计算并驱动视频终端的两个步进电机转动以匹配人体头部的转动。视频采集终端通过摄像头采集数据并进行实时发送，中间件服务器接收视频数据并进行压缩转码发送，显示终端接收转码后的视频数据并通过屏幕显示，使用者透过头戴式VR显示器进行观测。对整个系统，提供Wi-Fi网络以进行数据交换，在一定范围内保证连接的稳定性以及数据交换的及时性。在本文进行的实验结果中表明，系统对于人体头部转动数据的计算误差在5°以内，陀螺仪数据的传输延时在0.1秒以内，由于计算能力以及传输协议的限制，视频数据的延时被限制在0.5秒左右，整体系统网络带宽消耗在10Mbps以内，符合设计要求指标。

关键词**：**虚拟现实；摄像头；视觉延伸；Unity3D

**A vision extension system based on virtual reality head mount display**

# Abstract

The rapid development of virtual reality, has further enhance the binding with the various existing applications. A lot of researchers in the field of virtual reality and augmented reality have interested in the combination of virtual reality and camera. But here is the problem, in most of the solutions, video capture devices and mounted platform are too close to separate with each other, and are not able to popularize as a single module.

This paper attempts to design a real-time vision extension system which is modular and easy-mounted by a variety of platforms. Two stepper motors provide two rotational degrees of freedom for the video capture terminal, the difference between the gyro data of the display terminal and the gyro data of the video capture terminal is used to calculate and drive the two stepper motors to match the rotation of the user’s head. Video capture terminal collects video data and transmitted in real time, the middleware server receives the data and forwards the compressed and transcoded data, the display terminal receives the transcoded data and displays it with the screen, the user observes it through a virtual reality head mount display. For the whole system, a Wi-Fi network is provided for data exchange, ensure the stability of the connection and low delay of data exchange. In this paper, the experimental results show that the rotation data error of the user’s head which is calculated by the system is within 5°, the delay of the transmission for the gyro data is less than 0.1 seconds, due to computing power and transmission protocol, the delay of the transmission for the video data is limited to about 0.5 seconds, the consumption of the network bandwidth for the whole system is within 10Mbps, the results meet the design requirements.

**Key Words：**Virtual Reality；Camera；Vision Extension；Unity3D

目 录

[摘 要 I](#_Toc452727360)

[Abstract II](#_Toc452727361)

[引 言 1](#_Toc452727362)

[1 绪论 2](#_Toc452727363)

[1.1 课题研究背景及意义 2](#_Toc452727364)

[1.1.1 课题研究背景 2](#_Toc452727365)

[1.1.2 课题研究意义 3](#_Toc452727366)

[1.2 本文研究内容 3](#_Toc452727367)

[1.3 论文内容结构 3](#_Toc452727368)

[2 平台及工具介绍 4](#_Toc452727369)

[2.1 显示终端 4](#_Toc452727370)

[2.2 视频终端 4](#_Toc452727371)

[2.3 中间件服务器 5](#_Toc452727372)

[2.4 主要开发语言 6](#_Toc452727373)

[2.5 主要开发工具 6](#_Toc452727374)

[3 显示终端设计与实现 8](#_Toc452727375)

[3.1 系统结构 8](#_Toc452727376)

[3.1.1 姿态数据获取及平滑 8](#_Toc452727377)

[3.1.2 视频画面显示 9](#_Toc452727378)

[3.1.3 菜单的显示与控制 10](#_Toc452727379)

[3.1.4 连接的建立与数据发送 12](#_Toc452727380)

[3.2 基于FFmpeg的Unity3D插件 12](#_Toc452727381)

[3.2.1 逻辑结构 12](#_Toc452727382)

[3.2.2 与OpenGL ES配合 13](#_Toc452727383)

[3.3 本章小结 13](#_Toc452727384)

[4 视频终端的设计与实现 14](#_Toc452727385)

[4.1 步进电机模块 14](#_Toc452727386)

[4.2 姿态测算模块 16](#_Toc452727387)

[4.2.1 MPU6050 16](#_Toc452727388)

[4.2.2 姿态表示方式 16](#_Toc452727389)

[4.2.3 万向节死锁 16](#_Toc452727390)

[4.2.4 启动误差的消除 17](#_Toc452727391)

[4.3 视频采集发送模块 19](#_Toc452727392)

[4.4 本章小结 19](#_Toc452727393)

[5 中间件服务器的设计与实现 20](#_Toc452727394)

[5.1 视频流的处理与压缩转码转发 20](#_Toc452727395)

[5.2 姿态数据可视化 21](#_Toc452727396)

[5.3 通信模块 22](#_Toc452727397)

[5.4 本章小结 23](#_Toc452727398)

[6 完整系统的测试 24](#_Toc452727399)

[6.1 测试工具Instruments 24](#_Toc452727400)

[6.2 性能分析 25](#_Toc452727401)

[6.2.1 显示终端性能分析 25](#_Toc452727402)

[6.2.2 视频终端性能分析 26](#_Toc452727403)

[6.3 网络带宽占用 27](#_Toc452727404)

[6.4 数据传输延迟 28](#_Toc452727405)

[6.5 电机控制 29](#_Toc452727406)

[设 计 总 结 30](#_Toc452727407)

[参 考 文 献 31](#_Toc452727408)

[致 谢 33](#_Toc452727409)

# 引 言

视力所及就是眼界的边际，当人们想要见到远方的风景，或者进行深海探测、太空探测、现场救援等工作，由于环境、生理等各种各样的限制，人们不能或者不容易亲身前往，对于这类工作，往往都依靠一些辅助设备完成，摄像头就是其中一个。借助于摄像头，人们可以完成不亲身前往而得到视频图像进行观察，但是这样的视频图像是平面的，且与观察者不存在交互或者存在有限的交互，在一定程度上使得观察者对于环境的判断有一定的限制。近年来，以Oculus、HTC VIVE为首的桌面式虚拟现实头戴式显示设备和以Google Cardboard项目为基础的Samsung Gear VR等移动设备虚拟现实头戴式显示设备飞速发展，逐渐成熟，渐渐走入公众的视线。与此同时，虚拟现实技术与现有技术的结合出现了大量应用，而研究虚拟现实技术与视频采集的结合正成为一个热门方向。借助3D摄像头搭配虚拟现实头戴式显示器，通过计算处理实时生成虚拟现实画面能够欺骗人的大脑实现将人的视觉延伸至远方，达到非亲身又能走到远方的效果，而且视频采集终端可以搭载在很多设备上，比如小车，四旋翼飞行器等，同时结合提供一定的控制方式，将能完成平时人所难以完成的工作。将该系统应用于灾难救援、地下探测、深海探测等，将大大减少危险情况的发生，保证相关人员的安全。

本系统分为视频采集终端（以下称视频终端）、中间件服务器（以下称中间件）、用户显示终端（以下称显示终端）三部分，在本系统中需要解决的关键问题是视频数据的实时传输、电机的转动控制以及姿态数据的获取。视频数据从采集到接收到显示，需要经历编码、传输、解码的过程，在这一过程中，传输过程占用了大量的时间，所以通过适当的压缩转码降低数据量来减少传输时间是十分必要的，但是由于视频采集终端的计算能力限制，如果进行繁重的压缩转码操作带来的传输时间的减少远远低于压缩转码本身所需要的时间，如此操作得不偿失，所以加入了中间件，由拥有较高计算能力的计算机完成压缩转码操作。在电机转动的控制中，由于电机本身存在的机械电气误差，采用开环控制将会导致误差增大，所以系统对电机的控制采用闭环控制，通过不断比较目标姿态与当前姿态，来完成电机的控制。对于姿态数据的获取，显示终端提供了API接口来获取姿态数据，而对于视频终端则需要自己动手设计与实现，为了编程的方便，姿态数据统一使用了欧拉角表示，而在欧拉角中存在万向节死锁，单一的陀螺仪单元将会在特定的方向上损失一个转动自由度，所以在本系统中采用了双陀螺仪单元的设计，分别测算偏航角与俯仰角。

# 1 绪论

## 1.1 课题研究背景及意义

### 1.1.1 课题研究背景

虚拟现实（Virtual Reality），简称VR技术，也称人工环境，虚拟现实通过利用计算机技术模拟生成真假难辨的虚拟三维空间，使用者可以通过各种外设感知到触觉、听觉、视觉等模拟感觉，让使用者如同身临其境一般，可以实时地、没有限制地观察虚拟三维空间内的事物并且在一定限制下与其产生交互。使用者进行位置移动时，通过对动作姿态等的捕捉，计算机可以实时进行复杂的计算，生成准确的三维空间影像并传回以产生临场感。该技术集成了计算机仿真、显示、网络并行处理、计算机图形、感应、人工智能等技术的最新发展成果，是一种由多种计算机相关领域技术驱动的模拟系统[1]。虚拟现实这一定义的产生出现于上个世纪50年代，虚拟现实技术演化发展史大致上分为四个阶段，虚拟现实的第一阶段是在1963年以前，此时对虚拟现实的定义主要是对声音形象动作的模拟，虚拟现实的第二阶段在1963年至1972年之间，此时虚拟现实开始逐步萌芽，虚拟现实的第三阶段在1973年至1989年间，此时虚拟现实产生了比较成型的概念而且初步形成了较为完善的理论，而对虚拟现实理论进行进一步的完善和投入应用为第四阶段即1990 年至2004年[2]。

随着虚拟现实技术的不断发展，出现了类似于Oculus、HTC Vive等基于PC的头戴式VR设备，而随着智能手机的处理性能提高以及Google公司Cardboard项目的发布，Gear VR等基于移动设备的头戴式VR设备也成为了可能。基于PC的虚拟现实设备最大的优势就是更好的沉浸式体验，但是这类虚拟现实设备对于硬件的要求十分严格。基于移动设备的虚拟现实设备优势在于便携以及成本的低廉，但与此相对，基于移动设备的沉浸式体验较差。

借助摄像头等工具在一些难以到达的地方进行实时或者非实时拍摄已经有了很多相关研究，将其应用于救援、探测等领域已经有了很成熟的应用。2011年3月21日，中科院沈阳自动化所成功研究出了三款“废墟搜索与辅助救援机器人”，并顺利通过了中国地震应急搜救中心对其的测试，这三款机器人将于“十二五”期间应用到地震救援等复杂环境救援中。“洞穴搜救机器人”最大的特点就是可根据灾难废墟现场的地形进行变形，可以呈现条形、三角形等多种形状。机器人可以携带夜视摄像头和生命探测仪等救援设备进入废墟内部，为救援人员提供有效信息以供决策。

### 1.1.2 课题研究意义

随着虚拟现实技术的不断发展普及，结合虚拟现实技术与全景摄像技术，通过镜头捕捉的全景画面，通过虚拟现实效果进行展示，将会在一定程度上解决科研方面的某些棘手问题。由于摄像头的体积较小，可以去到人难以到达的地方，如倒塌现场、深海、太空等复杂环境，而通过三维重构处理显示传回来的视频画面，人的视觉系统可以敏锐地捕捉到在平面显示上无法得到的细节信息，为决策提供了良好的依据。

## 1.2 本文研究内容

本文试图设计一种模块化的、方便被各种平台搭载的实时视觉延伸系统。

硬件部分，设计搭建以树莓派为中心的简单电气机械结构以接受使用者的控制。通过树莓派的GPIO接口完成两个电机的实时控制，通过I2C总线驱动控制MPU6050芯片。

软件部分，通过调用树莓派摄像头完成视频采集，通过FFmpeg完成视频数据的发送、压缩转码、转发操作，使用Unity3D构建移动设备上的显示画面，编写基于FFmpeg和OpenGL ES的视频流解析模块解析视频流，获取并使用姿态数据驱动控制电机的转动。

## 1.3 论文内容结构

第一章为全文的绪论，简要介绍了该设计的研究背景意义，对论文的主要内容以及行文结构进行了说明。

第二章介绍了该设计中使用的开发平台、开发语言以及主要开发工具。

第三章对系统中显示终端的设计与实现进行了详细的论述，包括Unity3D场景的设计、解析模块的逻辑设计与实现等。

第四章对系统中视频采集终端的设计与实现进行了详细的论述，包括以树莓派为中心的控制部分以及跟其他芯片的接口交互设计实现。

第五章对系统中中间件服务器的设计与实现进行了简要的论述。

第六章对整个系统进行了集成测试以及从CPU利用率、能耗率、编解码性能、网络带宽占用等方面进行了详细的性能分析。

# 2 平台及工具介绍

## 2.1 显示终端

显示终端采用移动端VR设计，使用一部iPhone 6作为显示设备，采用卡尔·蔡司公司的VR ONE虚拟现实头戴式显示器完成光学处理。

VR One是由著名的镜头生产商卡尔·蔡司推出的一款使用手机作为基本显示设备的虚拟现实眼镜，与其他基于Cardboard概念的设备工作流程基本相同，透过透镜来观看被塞入镜片后特别制作的存放手机托盘中手机屏幕显示出来的内容，实际效果为虚拟现实的沉浸式场景。目前该设备只兼容和IPhone6三星Galaxy S5。

iPhone 6是美国苹果公司于2014年9月推出的一款智能手机，是苹果首次推出4英寸以上的大屏幕手机，其中iPhone 6屏幕大小是4.7英寸，搭载iOS 8系统及新一代Apple A8处理器，同属于iPhone系列的第八世代。iPhone 6使用了苹果自家设计的64位A8处理器，双核心主频为1.4Ghz，性能较A7提升了25%，图形处理提升了50%，并配备全新协处理器M8。A8采用了20纳米制程（A7为28nm），在这块处理器上集成了20亿个晶体管，相比较，A7仅有超过10亿个晶体管，部分Haswell架构处理器也仅有14亿个晶体管。

## 2.2 视频终端

树莓派（英语：Raspberry Pi），是一款基于Linux的只有信用卡大小的微型计算机。位于英国的树莓派基金会负责对树莓派进行硬件的设计以及开发，开发树莓派的目的是为了较低的硬件软件成本推动中小学中对于计算机科学技术的教育事业。

树莓派由Element 14/Premier Farnell和RS Components这两家获得生产许可的公司进行制造和销售。这两家公司都在网上出售树莓派。

第一代树莓派电脑的CPU为博通（Broadcom）公司设计生产的基于ARM架构700MHz BCM2835，该型号树莓派拥有256MB内存（在B型发布时内存容量已升级到512MB内存），使用SD卡当作存储数据，拥有一个以太网、两个USB接口、以及支持声音输出的HDMI和RCA端子输出支持。树莓派的面积只有一张信用卡大，厚度约为3厘米，在这样小的一台电脑上却可以流畅运行很多游戏和进行1080p高清视频的播放。树莓派支持多种系统，基金会提供了基于ARM架构的Debian、Arch Linux和Fedora等的发行版供大众下载，还提供支持Python作为主要编程语言，支持BBC BASIC(通过RISC OS映像或者Linux的"Brandy Basic"克隆)、Perl和C语言等编程语言。

GPIO（英语：General-purpose input/output），通用输入/输出的简称，功能类似单芯片微控制器8051的P0—P3，用户可以自由操控使用其引脚，PIN脚可根据实际情况调整，分别作为通用输入（GPI）或通用输出（GPO）或通用输入与输出（GPIO），如当clk generator, chip select等。通过读取某个寄存器的内容来确定引脚电位的高低以得到输入；通过写入某个寄存器来让这个引脚输出高电位或者低电位以完成输出；对于其他特殊功能，则有另外的寄存器来控制它们。

L293D是一款单片集成，高电压，大电流，4通道电机驱动。该芯片可以驱动直流电动机和高达36伏的电源，芯片可以提供每通道600mA电流的最大电流，L293D芯片也因是一款H型桥驱动而为人们熟知。

I²C（Inter-Integrated Circuit）字面上的意思是集成电路之间，它其实是I²C Bus简称，所以中文应该叫集成电路总线，它是一种串行通信总线，使用多主从架构，由飞利浦公司在1980年代为了让主板、嵌入式系统或手机用以连接低速周边设备而发展。

MPU6050为全球第一款集加速度传感器、三轴陀螺仪、温度传感器等部件为一体的6轴运动处理组件，跟其他多部件方案比较，MPU6050解决了协调陀螺仪与加速器中的一大问题——轴间差，同时高度集成的设计大幅减少了芯片的体积。MPU6050的角速度满量程支持±250、±500、±1000与±2000°/sec (dps)等多种方式，可以精确地对快速与慢速动作进行捕获，并且，用户可进行编程控制的加速度计满量程支持±2g、±4g±8g与±16g等多种方式。MPU6050数据的传输可通过频率高达400kHz的I²C总线或频率高达20MHz的SPI串行外设接口总线。MPU6050支持不同电压下工作，电源供电电压VDD允许2.5V±5%、3.0V±5%或3.3V±5%等多种方式，逻辑接口VDDIO供电为1.8V± 5%。MPU6050的封装尺寸仅为4x4x0.9mm(QFN)，且包含内建的温度感测器以及在正常工作中仅有±1%误差偏移的振荡器。

视频终端使用Raspberry Pi 2 Model B作为中心部件，通过其摄像头模块获取视频图像，通过扩展GPIO接口，使用两个L293D电桥驱动器分别驱动两个步进电机工作，通过GPIO接口上的I2C总线驱动两个MPU6050六轴运动处理单元分别获取摄像头模块偏航和俯仰的角度。

## 2.3 中间件服务器

中间件服务器由运算能力较高的个人PC担任，通过Wi-Fi使用TCP及UDP协议完成数据的交换。

## 2.4 主要开发语言

Python是一种面向对象、直译式的计算机程序语言，具有近二十年的发展历史。它包含了一组功能完备的标准库，由于Python语言的简洁性，它被用于完成很多常见的任务，Python使用对齐的缩进来表示语句块以代替其它很多程序语言使用的大括号。Python不像C语言需要手动管理内存进行垃圾回收，而是通过自动管理内存来进行垃圾回收，因此在系统管理和网络编程中，经常使用被当做脚本语言的Python，同时Python语言也能够轻松胜任各种复杂的任务。Python虚拟机由于良好的跨平台特性，Python语言几乎可以在所有支持Python虚拟机的系统中运行[3]。

C是一种通用的编程语言，广泛用于系统软件与应用软件的开发，相对于其他语言来说，C语言更接近底层，因此拥有较高的性能表现。Dennis M.Ritchie与Ken Thompson在1969年至1973年间于Bell Labs设计开发出了C语言。开发C语言的目的为了移植和开发UNIX系统，且C语言的开发以B语言为基础，C语言具有的特点很多，其中最重要的特点便是简洁紧凑、灵活方便、允许直接访问物理地址操作硬件、目标代码质量高、执行效率高等，在程序员中备受青睐，成为最近几十年来使用最为广泛、影响最大的编程语言[4]。与Python等解释型语言不同，C语言是一门编译型语言，目前，C语言编译器在常见的不同操作系统中都有实现，例如Microsoft Windows, Mac OS X, Linux, Unix等。C语言的产生对以后的编程语言的设计都有深远的意义，例如C++、Objective-C、Python、Java、C#等[4]。

C#是微软推出的一种面向对象的、基于.NET Framework框架的高级编程语言。C#的发音为“C sharp”，意思为C语言的升级版。C#的受到了C和C++的影响，在性能表现上一如既往的优秀，同时又以.NET Framework作为基石，拥有跟Visual Basic等语言相同的快速开发能力。C#由Anders Hejlsberg主持开发，微软在2000年发布了这种语言。

## 2.5 主要开发工具

Unity3D是一个用于创建实时三维动画、建筑可视化、三维视频游戏等多种互动内容类型的综合型制作创作工具。Unity3D类似于Virtools、Blender、Director或Torque Game Builder等利用交互式图型化开发环境为主要方式的软件，Unity3D支持跨平台工作，Unity3D编辑器支持Windows和Mac OS X等系统，可发布游戏至OSX、Windows、Wii以及移动端Android、iOS等平台。同时也可以通过Unity web player插件来设计制作网页游戏，支持Windows和Mac等系统下不同内核浏览器的网页浏览。它的网页播放器也被Mac widgets所支持。

FFmpeg是一套基于LGPL/GPL协议的自由软件，可以以多种格式完成音频和视频的录制、转码、流媒体处理等功能，其中包含了用于多种类型音频和视频的编码解码器库libavcodec，以及用于不同音频格式与视频格式之间的格式转换库libavformat等。

Xcode是包含一套由苹果公司开发的方便开发OS X、iOS、WatchOS和tvOS软件的开发工具的集成开发环境。苹果公司于2003年第一次公布了Xcode，截止到目前，最新的release版本是7.3.1，并且使用官方Mac App Store商店免费提供给所有的OS X El Capitan系统用户。开发者可通过苹果Developer网站下载Beta版本和套件的早期版本。

Xcode支持多种编程语言，主要有C、C++、Objective-C、Java、Python、Ruby、ResEdit(Rez)以及最新的Swift语言，具有多种编程模型，包括但不限于Cocoa、Carbon和Java。Xcode中可以建立通用的二进制文件，它允许软件在PowerPC和基于Intel（x86）的平台上运行，同时可以包括x86和x86\_64代码。使用iOS的SDK，Xcode中也可以用来编译和调试针对ARM架构的处理器上运行的iOS应用程序。Xcode中包括GUI工具Instruments，它运行于动态跟踪框架DTrace之上，DTrace由Sun Microsystems创建并作为OpenSolaris的一部分发布。

# 3 显示终端设计与实现

## 3.1 系统结构

### 3.1.1 姿态数据获取及平滑

由于iPhone上存在加速度、角度、磁力、重力等传感器，且提供了相关API接口以获取手机姿态的四元数，所以对于手机姿态欧拉角的获取十分方便。但是由于干扰和误差，获得的欧拉角数据是离散程度较大，而在实际情况中，数据应该是近似连续的，所以需要编写一定的平滑算法来平滑欧拉角数据，平滑公式如下：

 （3.1）

其中，为上次计算后的欧拉角数据， 为该次计算前从传感器获得的新欧拉角数据， 为 中 的一个，同理， 为预先定义的平滑阈值， 为本次计算后的欧拉角数据。

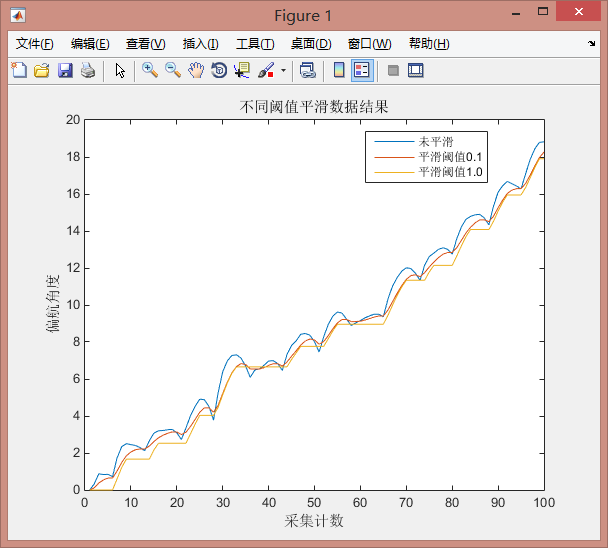


图3.1 平滑姿态数据前后

为了更为清晰的观测到平滑效果，取前100个姿态数据进行绘制分析，平滑结果如图3.1所示，可以看出，原始数据在未经平滑前波动较大，在平滑阈值为0.1的时候，大幅降低了数据的波动程度，在平滑阈值为1.0的时候，可以明显看出出现很多阶梯形状，这是阈值过大的表现，导致姿态数据出现跳跃式变化通过对不同阈值的实验，最后选定0.1作为整个系统的平滑阈值。

### 3.1.2 视频画面显示

借助于VR ONE提供的开发者工具包，系统通过Unity 3D构建生成iPhone上的左右眼的视频画面，如图2.1所示。

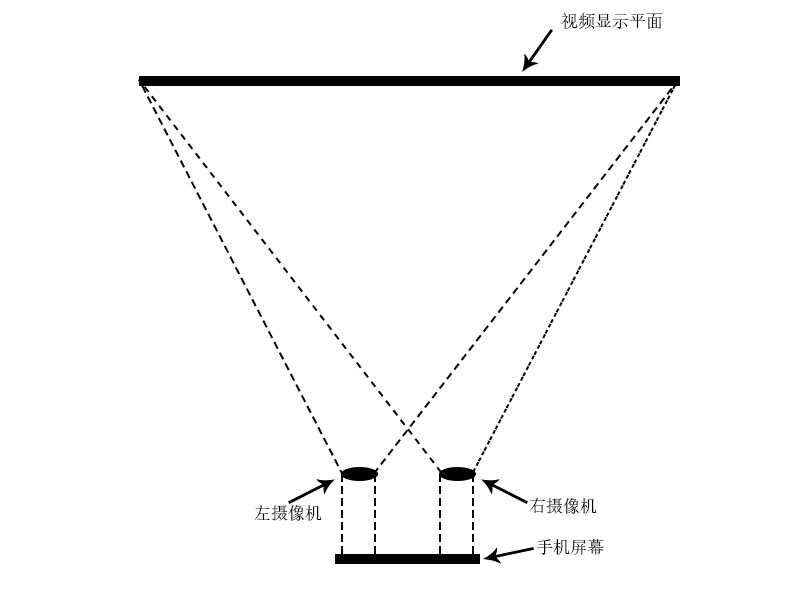


图3.1 视频画面的显示

视频显示平面显示画面为解析的视频流画面，具体解析过程下文将详述。摄像机是向使用者展示世界的物体，通过以一定的距离间隔构建左右摄像机，模拟人体眼睛的瞳距，对于左右摄像机，使用预先定义好的LUT表对摄像机得到的画面使用Graphics.Blit函数拷贝源纹理到目的渲染纹理，进行图像效果的处理，这样得到的左右摄像机的画面将会模拟一定的视差，最后，具有视差的手机屏幕的左右眼画面通过头戴式显示器的光学透镜进行成像，还原出真实图像。

### 3.1.3 菜单的显示与控制

在提供的VR ONE开发者工具包中，菜单的控制交互完全采用姿态数据进行动态调整，姿态数据的变化，将导致菜单的整体移动，同时根据视线落点进行菜单的选择等处理操作。对于本系统，菜单的控制通过追加菜单显示平面，采用半动态方式，如图3.2所示。

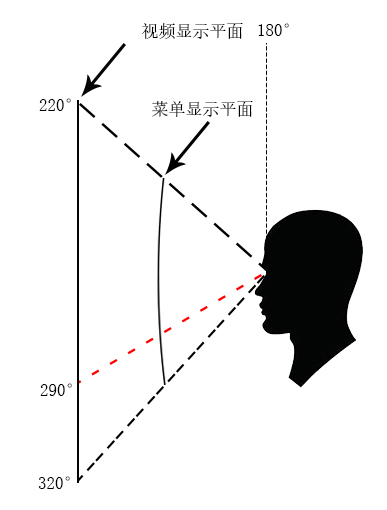


图3.2 菜单显示平面示意图

通过追踪偏航角、俯仰角数据，判断俯仰角是否大于290°，对满足俯仰角大于290°且当前不存在菜单，将会生成菜单显示，当俯仰角小于290°，将会隐藏所有菜单。通过对可视区域偏航角进行按菜单选项个数平均分区，使每一个区域对应一个菜单选项，通过追踪在可视区域中的偏航角的变化，视线在该区域停留时间长短来确定是否选中该菜单。

如图3.3所示，在菜单可视状态下，菜单选项将会按照顺序并排在视线区域内的下方，图中已有的三个图标代表的操作分别为重新校正中心点、关闭菜单、退出应用程序，在有需要的情况下，可以增加其他选项。

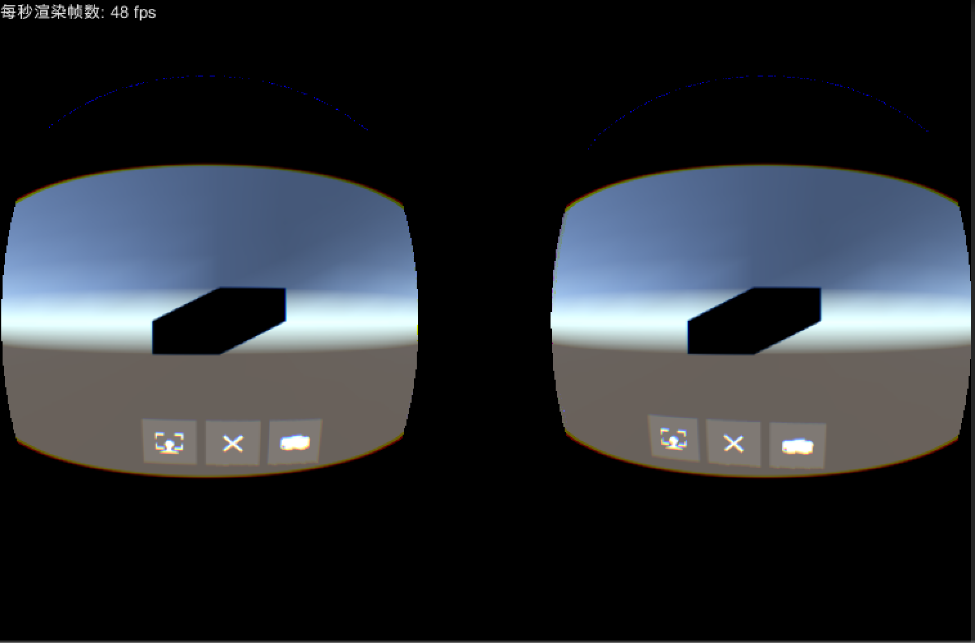


图3.3 实际生成的菜单选项

在实际的操作中，使用者在低头满足俯仰角大于290°后，左右旋转头部会依次选中菜单中的某一项，通过追踪使用者视线在该项的停留时间不断由下而上动态更新填充图标的进度条，当图标被完全填充后将在隐藏菜单的同时触发菜单选项的功能完成操作，图3.4所示为刚选中关闭菜单时出现的进度条。

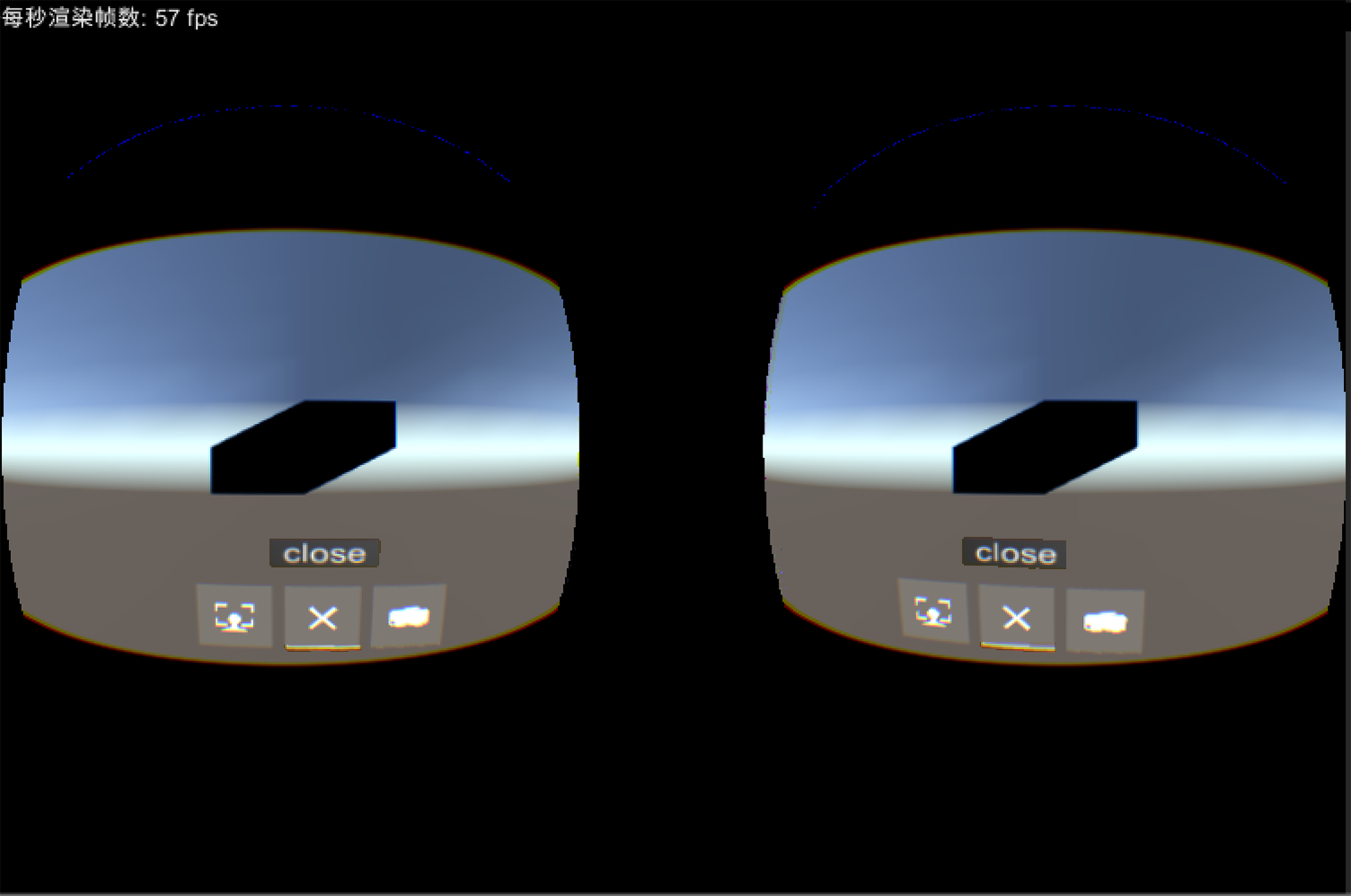


图3.4 选中菜单选项的动态效果

### 3.1.4 连接的建立与数据发送

在显示终端的设计中，主要需要处理两类数据：视频数据的接收以及姿态数据的发送。

在数据接收发送前需要先建立连接，考虑到降低使用者操作的需要，连接的建立采取了自动处理的方式。显示终端在启动后会初始化预先设置的端口以监听在局域网下的UDP广播（广播的产生由中间件服务器负责），根据接收到的UDP广播内容验证是否为中间件，在确定内容合法后，将会以TCP协议向中间件发出连接请求，在建立连接完成后将开始发送显示终端的姿态数据，同时以UDP协议接收视频流数据。

不同于视频流更新速率很快，人体姿态数据更新不需要达到每秒几十次的速度，如果使用UDP协议，在网络环境下产生的丢包情况将会使使用者明显的感觉到迟钝，因此对于姿态数据的发送，采用了TCP协议，如果产生了丢包，可以进行重发操作，更高的可靠性是在传输人体头部姿态数据中的第一考虑。

## 3.2 基于FFmpeg的Unity3D插件

对于使用解析的网络视频流内容更新材质，在免费版的Unity3D中并未提供相应功能，所以想要在Unity3D中完成以上功能，需要单独完成插件的编写工作。对于网络视频流的解析工作，使用了基于LGPL/GPL协议的FFmpeg完成，由于优秀的跨平台特性，在视频终端、中间件、显示终端都采用了这一解决方案来完成视频流创建、发送、压缩转码、转发等一系列操作。

### 3.2.1 逻辑结构

根据FFmpeg和Unity3D的相关教程以及文档，本插件需要完成根据传入的UDP视频流地址完成视频流的解析、根据解析的视频流数据生成相应的RGB帧数据、提供回调函数等待Unity3D渲染线程回调更新材质对象等操作，视频流解析流程如图3.5所示。

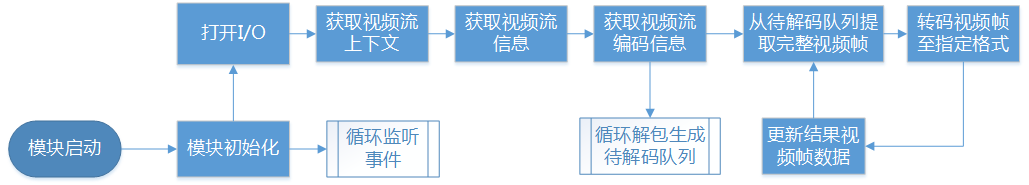


图3.5 视频流解析流程图

插件在初始化函数被调用后首先初始化所有FFmpeg的格式以及编解码器，完成所需内存空间的申请以及初始化操作，在完成初始化操作后，主线程将会开始循环监听事件，根据事件的不同响应对应操作，如退出事件将会完成所有子线程的终止以及内存回收等操作。在完成初始化操作后，主线程还将创建一个线程用于对接收到的网络视频流帧的解包操作，该线程将首先创建编解码线程，然后开始循环解包网络视频流帧，将得到的包含视频信息的包放入待解码队列中。编解码线程将会循环从待解码队列中提取包组成一个完整的视频帧，将该视频帧按照指定的格式（RGB24）进行转码，将结果保存并更新通知该帧数据已经可用，等待Unity3D渲染线程回调读取该帧数据。

### 3.2.2 与OpenGL ES配合

由于Unity3D引擎机制决定，很多的方法必须在主线程才能执行，所以在插件中无法完成在得到了指定格式的结果数据后直接更新Unity3D材质的操作，为了从插件完成渲染操作，应该在Unity3D脚本的Update函数中调用GL.IssuePluginEvent，这将使插件可以被渲染线程调用。回调函数根据传入的材质对象ID，通过OpenGL ES使用glTexSubImage2D函数完成材质对象的更新操作。

## 3.3 本章小结

由于显示终端的所有场景构建都必须通过一定的图像渲染引擎完成，在本系统中使用了Unity3D，所以本章的所有的工作都集中于扩展Unity3D上，无论是通过调用iPhone自带的传感器模块API得到手机的姿态还是编写基于FFmpeg的Unity3D插件来完成视频流的解析以及材质渲染操作都是如此。

# 4 视频终端的设计与实现

## 4.1 步进电机模块

由于需要对摄像头的偏航角、俯仰角进行实时控制，所以需要两个电机进行配合，考虑到精度以及误差的需要，选用步进电机来完成控制操作，因为步进电机能完成被精确定位以及支持连续转动。为了方便驱动步进电机的转动，在本系统中使用了L293D这款单片集成的高电压、高电流、4通道电机驱动芯片，芯片引脚如图4.1所示。

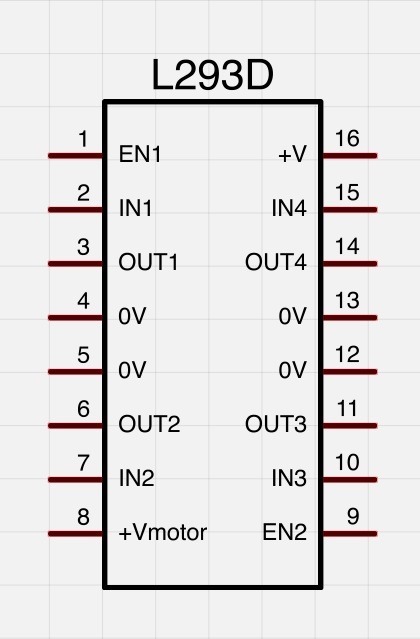


图4.1 L293D芯片引脚示意图

根据L293D芯片的说明书以及查阅的相关资料，使用步进电机的蓝、粉、黄、橙四个颜色的引脚分别依次接L293D芯片的14、3、10、6引脚，同时对L293D芯片的2、7、15、10引脚加高低电平信号，控制步进电机的正反转。

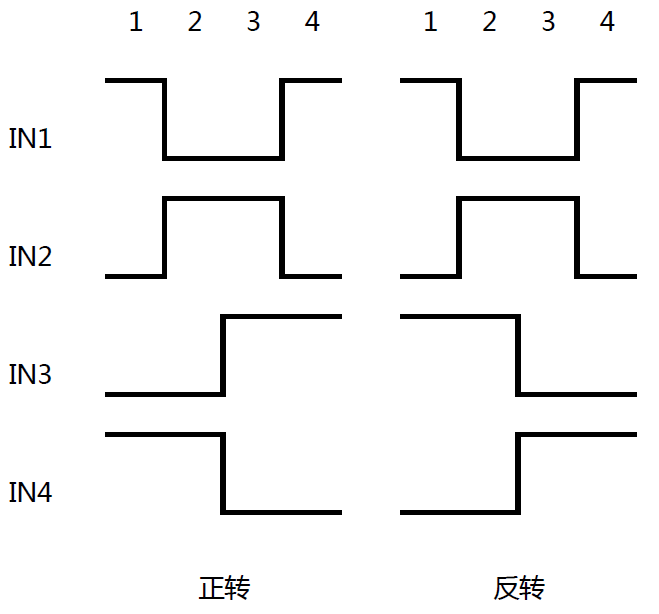


图4.2 L293D控制脉冲信号图

如图4.2所示，使用四个周期分别对IN1、IN2、IN3、IN4引脚加左边所示的电平信号，当完成电平跳转后，L293D芯片将会驱使步进电机正向转动一个步距角，反向同理。

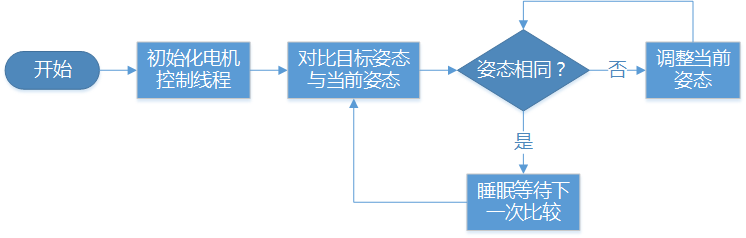


图4.3 电机模块工作流程

电机模块采用类封装，内部控制两个线程，根据传入的当前姿态和目标姿态实时比较，调整当前姿态以匹配目标姿态。当整个系统结束时，模块会自动调用函数完成内存的回收以及线程的终止等操作。

## 4.2 姿态测算模块

### 4.2.1 MPU6050

除了电机模块控制摄像头的转动外，摄像头姿态的获取在视频终端中也是十分重要的一个部分。在本系统中，对于摄像头的姿态测算主要集中于偏航角和俯仰角，所以需要测算摄像头姿态的两个旋转自由度，因此，选用了六轴运动处理单元MPU6050，使用欧拉角来表示计算摄像头的姿态。由于MPU6050内置了DMP处理单元，可以直接运算出四元数和姿态，而不再需要另外进行数学运算，可以减轻外围微处理器的工作负担且避免了繁琐的滤波和数据融合。

### 4.2.2 姿态表示方式

前文叙述到了欧拉角以及四元数这两种姿态表示方法，在实际使用中，还有旋转矩阵这一种姿态表示方法，对于这三种姿态表示方法，各存在优劣，在实际使用中按照需求采用不同的姿态表示方式，三种姿态表示方法的对比如表4.1所示。

表4.1 常见三种姿态表示方式部分对比

|  |  |  |  |
| --- | --- | --- | --- |
| 性质 | 旋转矩阵 | 四元数 | 欧拉角 |
| 在坐标系中旋转点 | 能 | 不能（必须转换为矩阵） | 不能（必须转换为矩阵） | |
| 连接或增量旋转 | 能，较慢 | 能，快 | 不能 | |
| 插值 | 基本不能 | Slerp提供了平滑插值 | 能，可能遭遇万向节死锁 | |
| 易用程度 | 难 | 难 | 易 | |
| 内存或文件中保存 | 9个数 | 4个数 | 3个数 | |
| 姿态表示唯一性 | 是 | 有一个逆元素 | 不是 | |

在本系统中，为了姿态测算的方便，采用了欧拉角这一方式来表示和计算所有的姿态，但是正如表4.1所示，使用欧拉角进行姿态的表示可能会遭遇万向节死锁的问题，所以必须针对万向节死锁这一问题进行处理。

### 4.2.3 万向节死锁

在三维空间中经常使用欧拉角坐标来表示物体姿态，欧拉角是利用绕三个轴旋转的角度即偏航、俯仰、旋转来表示针对物体坐标系轴的姿态。对于处于世界坐标系原点的物体，此时物体坐标系和世界坐标系重合。此时，规定物体的欧拉角为。

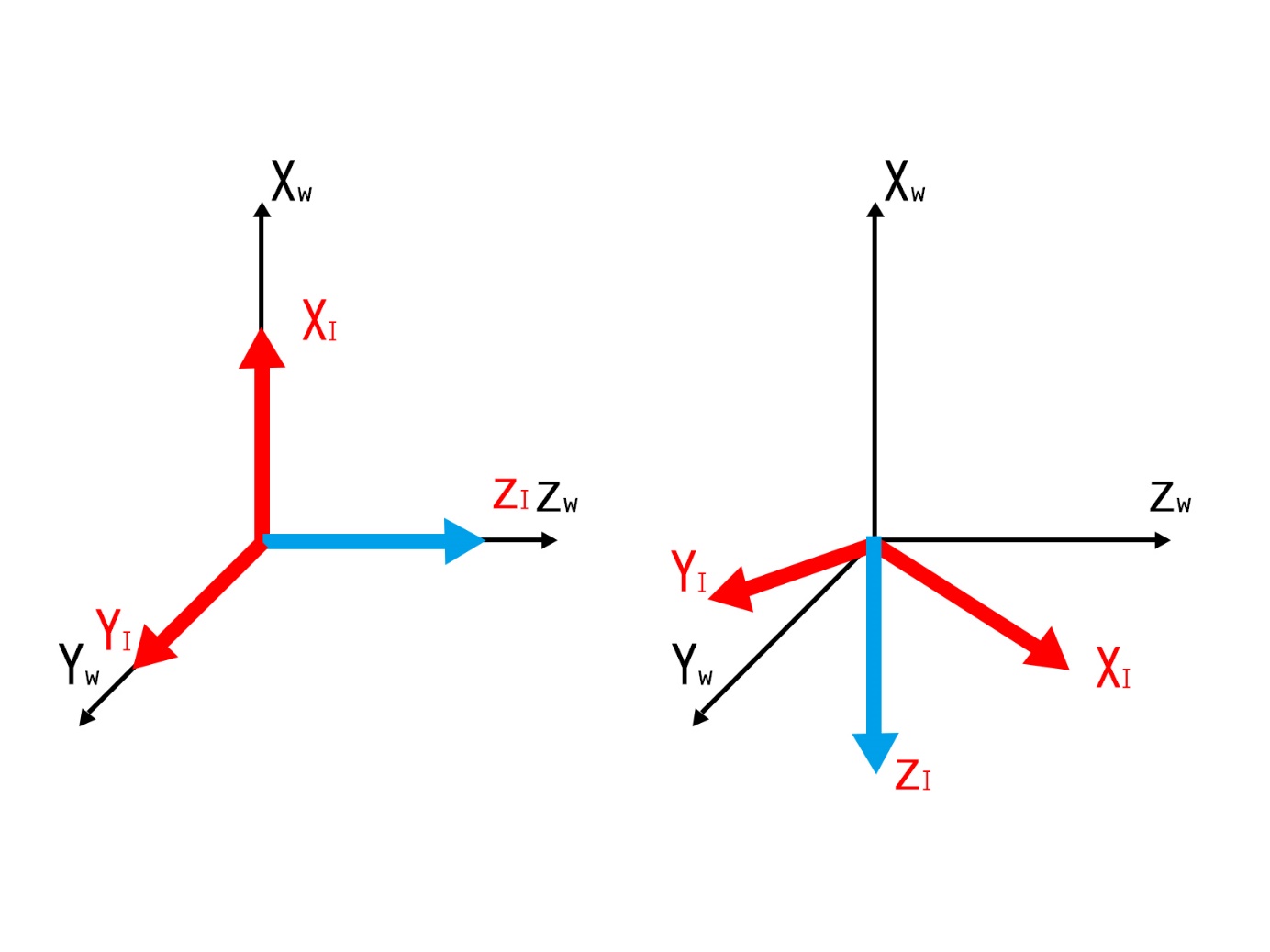


图4.4 万向节死锁的产生

如图4.4所示，左边部分表示物体的起始位置，此时物体的欧拉角，规定沿轴正方向看向负方向，顺时针旋转为正，然后首先使物体绕轴旋转30°，然后绕轴旋转90°，此时物体位置如图中右边部分所示，欧拉角为，而此时轴已与轴反向重合，再绕轴的旋转与第一次绕轴的旋转重复，此时的任何绕轴的旋转都可以在第一次旋转时通过额外旋转轴得到，换句话说，物体现在本质上只能完成绕两个轴的旋转，第三个自由度被丢失了，这就是三维空间旋转中使用欧拉角表示方向的万向节死锁问题。

在本系统中，为了解决万向节死锁这一问题，使用了两片MPU6050芯片分别测算摄像头的偏航角以及俯仰角。

### 4.2.4 启动误差的消除

由于MPU6050芯片从启动姿态数据测算到数据稳定需要一段时间，在此期间如果进行跟姿态有关的操作将会导致极大的误差，在系统设计时如果在所有模块加载完毕后再加载姿态测算模块，CPU需要闲置很长时间来等待数据稳定，因此在设计时使用了优先加载初始化姿态测算模块然后再加载其他模块的方法，在这种情况下，CPU在加载完姿态测算模块等待其数据稳定的时候，可以利用这一段时间加载并初始化其他功能模块，减少CPU的闲置等待时间。

在本系统中，偏航角数据是相对数据，每一次的基准点都是该次芯片启动的时候确定，对于相同的一个方向，不同次启动中得到的偏航角数据是不同的，而每次启动时摄像头的绝对方向是能够保证的（通过上次系统退出时的自动回位操作），因此，在MPU6050芯片启动达到数据稳定之后，需要校正稳定点的偏航角数据为0°，然后以此为基准点的转动符合整个系统的逻辑设计。

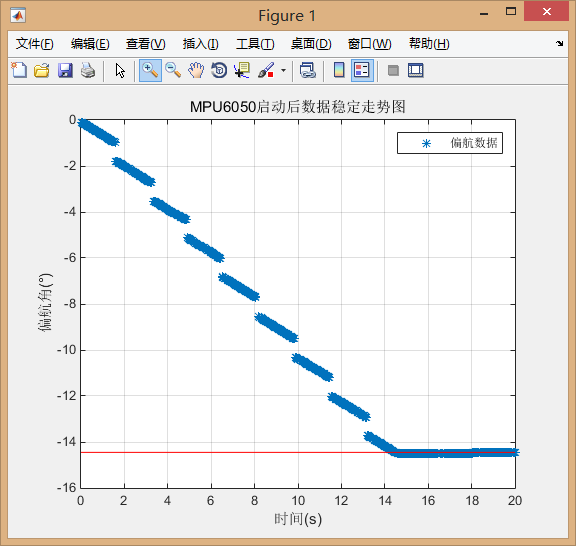


图4.5 MPU6050启动后数据稳定走势图

图4.5展示了MPU6050芯片启动后20秒内测得偏航角数据的变化，可以看出，在刚启动的前14.5秒左右偏航角数据是不稳定的，且近似呈现线性下降的趋势，在14.5秒以后系统达到稳定的状态，此时偏航角数据趋于稳定，红色参考线标注了稳定后偏航角数据的大致取值。在系统稳定后，使用该值对每次计算出来的偏航角原始值进行修正，得到系统的正确偏航角。对于在系统中需要使用的另一个角俯仰角，由于重力因素的存在，芯片能够自动进行纠正，因此无需在俯仰角上进行上述操作。

## 4.3 视频采集发送模块

视频采集发送模块是整个系统视频流的来源，如同上文所述的连接方式，视频终端通过监听当前局域网下的UDP广播与中间件建立连接，同时根据得到的地址启动视频采集操作，使用类Unix操作系统的管道机制，将视频采集进程的输出即视频流作为FFmpeg进程的输入，通过FFmpeg创建UDP视频流并向指定地址发送。

## 4.4 本章小结

本章详细介绍了视频终端的设计与实现，视频终端在与中间件建立连接之后，使用以上所述的步进电机模块、姿态测算模块、以及视频采集发送模块，与中间件发生数据的交换，使用TCP协议接收来自显示终端的姿态数据以及发送自身通过姿态测算模块得到的姿态数据，使用UDP协议来发送视频采集发送模块产生的视频流数据，然后根据显示终端的姿态数据以及自身的姿态数据，计算并通过步进电机模块闭环控制电机的转动，整个工作流程如图4.6所示。

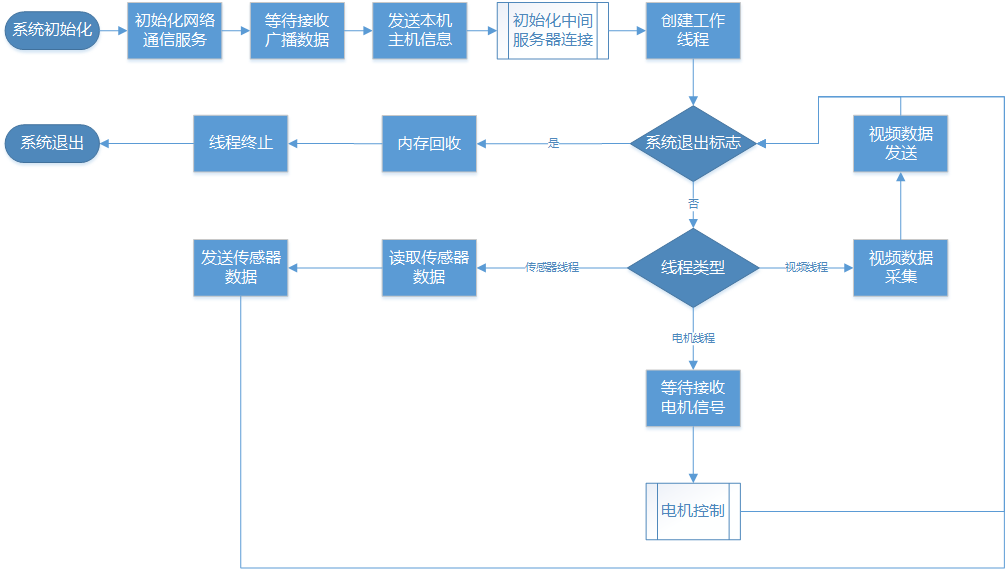


图4.6 视频终端工作流程

# 5 中间件服务器的设计与实现

在本系统中，中间件服务器的存在主要是为了提高视频的转码效率以缩短视频从采集到显示的延迟时间，理论上所有计算能力高于显示终端及采集终端的且满足符合本系统通信协议的处理设备都可以作为本系统中的中间件服务器，中间件服务器在本次设计中由预算能力较强的笔记本电脑担任。

## 5.1 视频流的处理与压缩转码转发

前文提到，视频流从产生到传输到接收显示的过程中，传输这一环节占用了大量的时间，如果能够减少视频流传输过程耗费的时间，理论上能够进一步减少视频从产生到显示的延迟。为了减少视频流传输过程耗费的时间，常见有两种方法，一种是缩短路径，即减少视频从被产生到被接收之间跳转的节点数量，另一种则是减少传输数据量的大小，即通过压缩转码将视频数据进行编码以减少需要传输的数据量，本系统采用的是第二种方式。压缩转码是一项十分耗费CPU时间的事情，在视频终端进行压缩转码带来的传输时间减少远远低于压缩转码本身消耗时间导致整体时间的增加得不偿失，因此采用计算能力较高的中间件服务器来完成压缩转码操作。

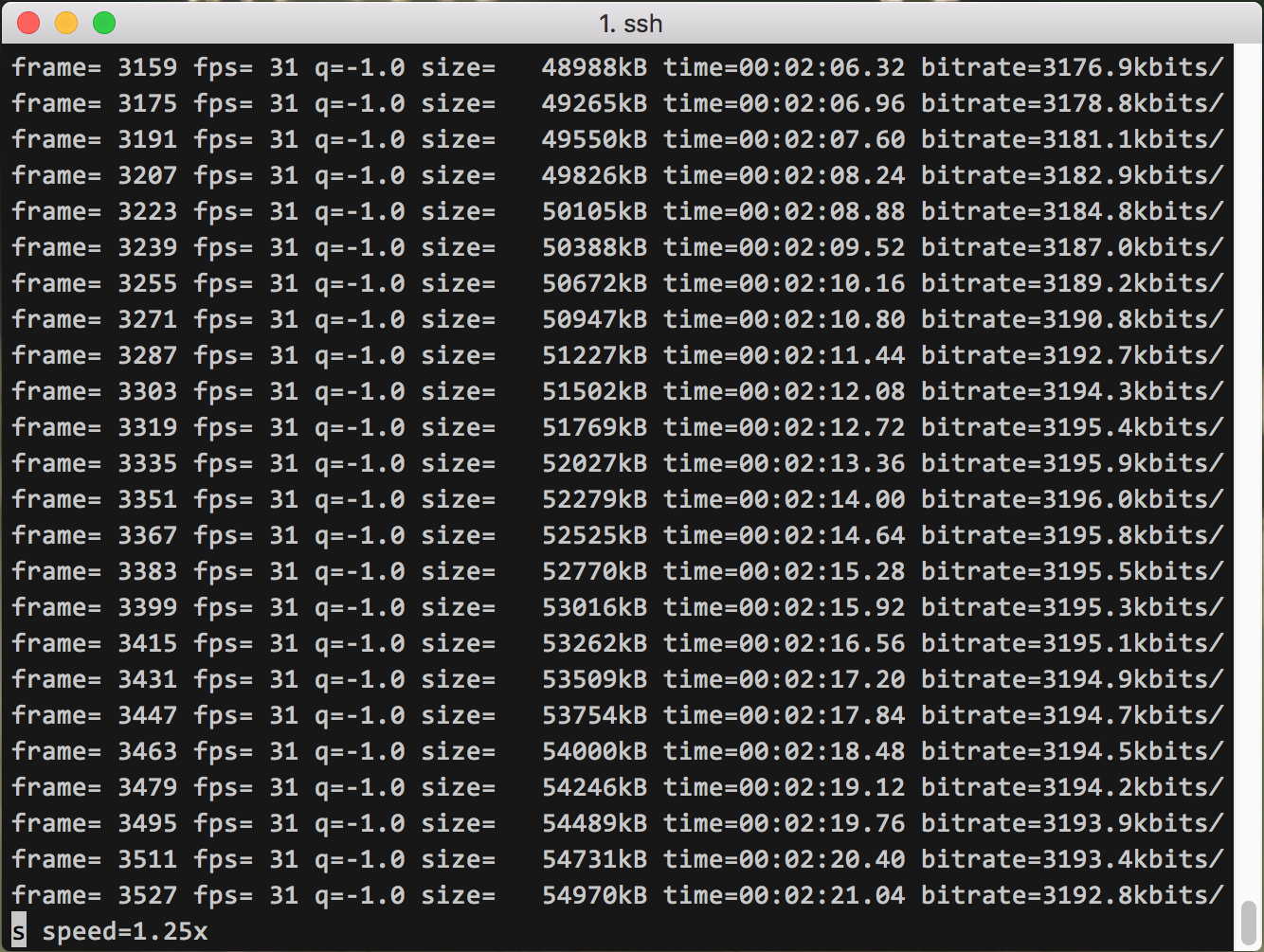


图5.1 原始视频流数据信息

图5.1展示了视频终端在启动了2分钟以后使用采集到的原始视频数据产生的视频流信息，由最后一项比特率可以大致得到每秒的数据量大小，在此图中为左右。

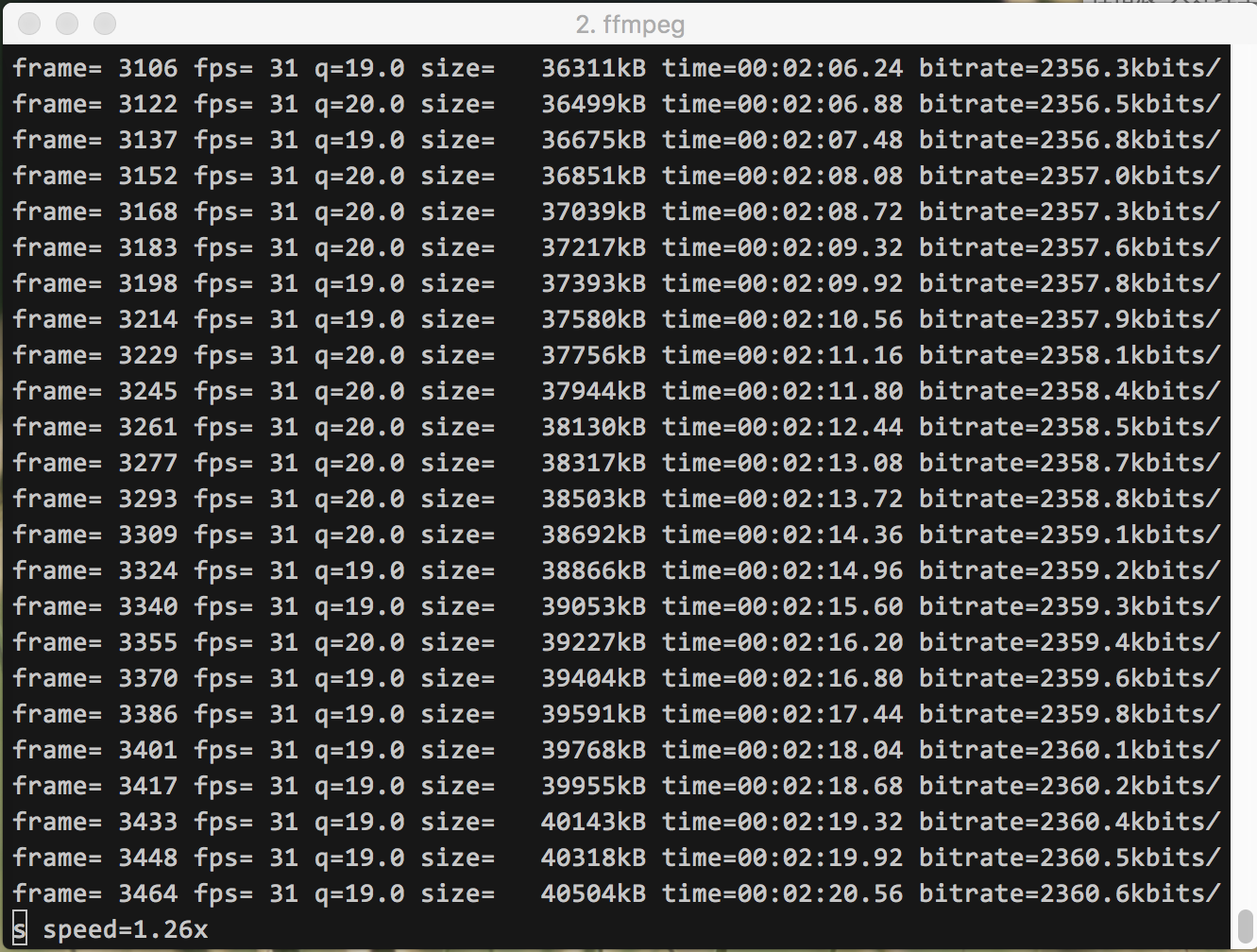


图5.2 压缩转码后视频流数据信息

图5.2展示了经过中间件服务器压缩转码后的视频流信息，此时最后一项比特率已经为左右，对比未经压缩转码的视频流信息，此时理论上带宽消耗以及传输时间均为原来的75%左右。

## 5.2 姿态数据可视化

中间件服务器在整个系统中充当了数据交换中心的作用，来自视频终端以及显示终端的姿态数据在这里进行转发，为了实时监控各终端姿态数据，中间件服务器将姿态数据进行了可视化处理，通过使用Pygame，实时地模拟显示了各终端姿态情况。

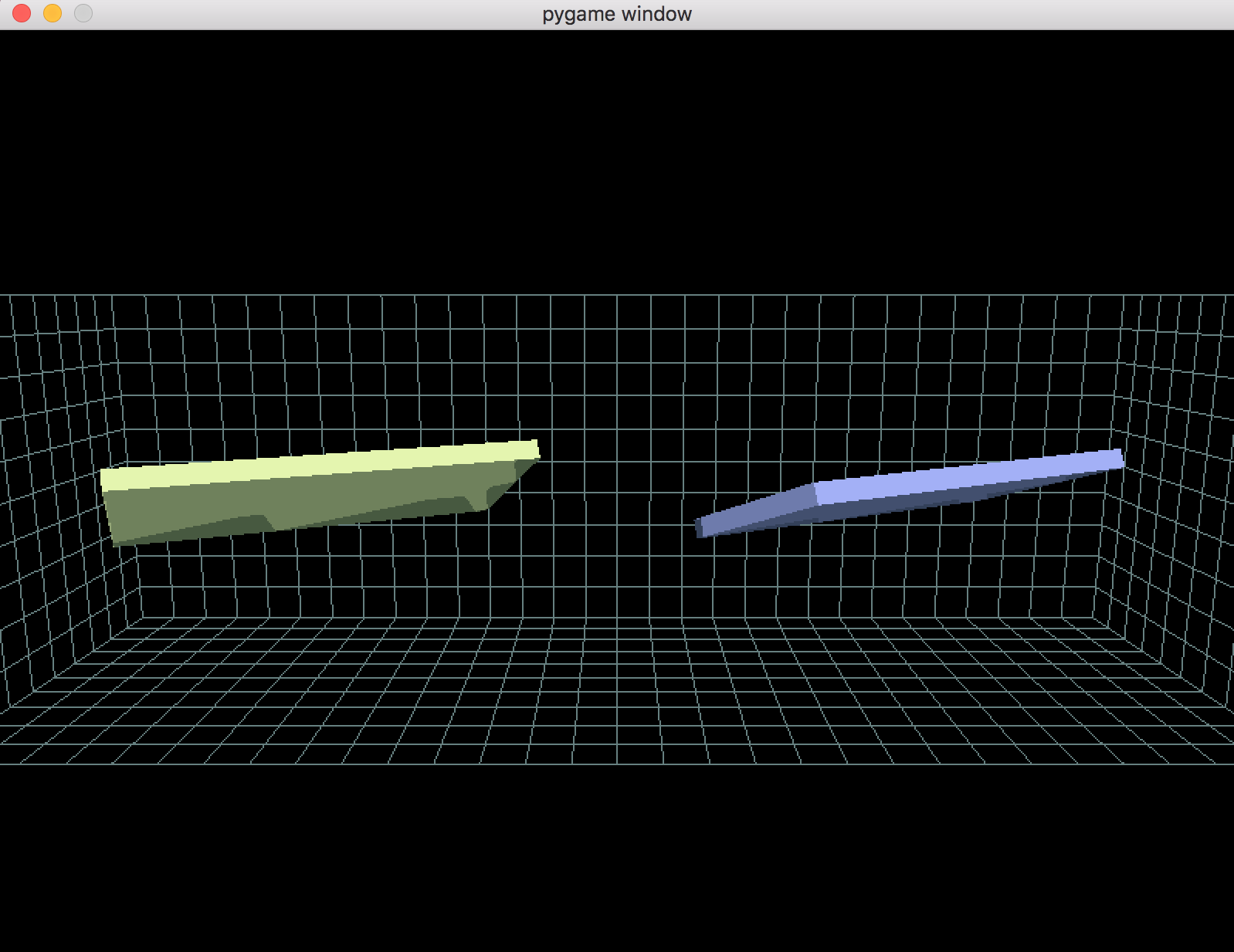


图5.3 中间件服务器对各终端姿态的实时可视化

图5.3展示了中间件服务器对各终端姿态的实时可视化情况，图中左边浅黄色方块代表了显示终端的姿态，右边蓝色方块代表了视频终端的姿态，所有流经中间件服务器的数据会被姿态可视化模块捕获然后调用OpenGL在三维空间中对方块进行旋转变换得到以上图片。

## 5.3 通信模块

本系统的所有通信服务都由中间件服务器维护，在中间件服务器启动以后，将会通过UDP广播端口8089循环向整个局域网发送UDP广播，广播内容为连接验证信息，通过使用Python的select模块，非阻塞地等待显示终端和视频终端以TCP协议分别跟中间件服务器的8092和8090端口进行连接，当检测到所有的终端连接成功后，通信模块将会以多进程单工管道的形式与姿态可视化模块建立通信，仍然使用select模块，非阻塞地从8090和8092端口得到各终端姿态数据，并且将得到的姿态数据通过管道传递给姿态可视化模块进行姿态可视化，特别的，在接收到显示终端的姿态数据后，将会立即向视频终端进行转发，保证数据的实时性。

## 5.4 本章小结

本章第1小节介绍了中间件服务器的视频流处理模块对于视频流的接收压缩转码和转发操作，第2小节介绍了姿态可视化模块数据的来源以及使用OpenGL在三维空间的绘图操作，第3小节介绍了通信模块UDP广播的产生以及运行时数据交换的过程。图5.4展示了中间件服务器的工作流程，需要注意的是，由于视频终端姿态数据和显示终端姿态数据到达中间件服务器的顺序不确定，当采用阻塞方式，规定视频终端姿态数据先于显示终端姿态数据被接收，如果视频终端姿态数据传输出现阻滞，将会导致显示终端姿态数据大量积累，造成显示终端姿态的误差较大。因此，在中间件服务器上对于姿态数据的接收使用了非阻塞的方式，并不强制要求姿态数据到达的先后顺序，这样能保证任意终端的数据不会因为其他终端的传输阻滞而出现大量积累导致误差。

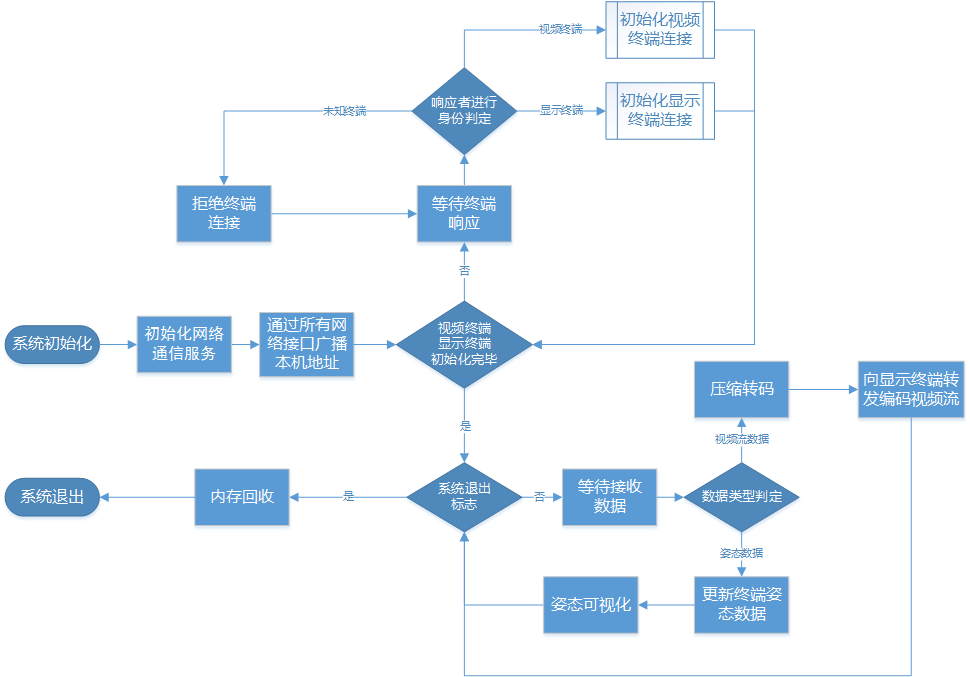


图5.4 中间件服务器工作流程

# 6 完整系统的测试

本章将会从CPU性能、网络带宽、传输延迟、电机控制等方面对系统进行完整的测试，从而对本系统的不同环境下性能表现能够作出一个合理的判断。

## 6.1 测试工具Instruments

在本次测试中，主要的测试内容主要集中在显示终端，而显示终端的主要部分为iPhone，苹果公司为方便开发者对开发的iOS程序的测试，提供了一套性能分析和测试工具Instruments。

Instruments是一套强大且灵活的性能分析和测试工具，它属于苹果公司Xcode开发套件的一部分，它被设计用于帮助开发者分析测试开发的OS X和iOS应用的行为和表现，帮助开发者在开发周期早期发现问题以减少后期开发不必要的时间。在Instruments中，使用被称为instruments的特殊工具，可以随着时间的推移来跟踪监视应用的运行，通过收集相关数据，Instruments可以为开发者呈现详细的分析结果供开发者使用。

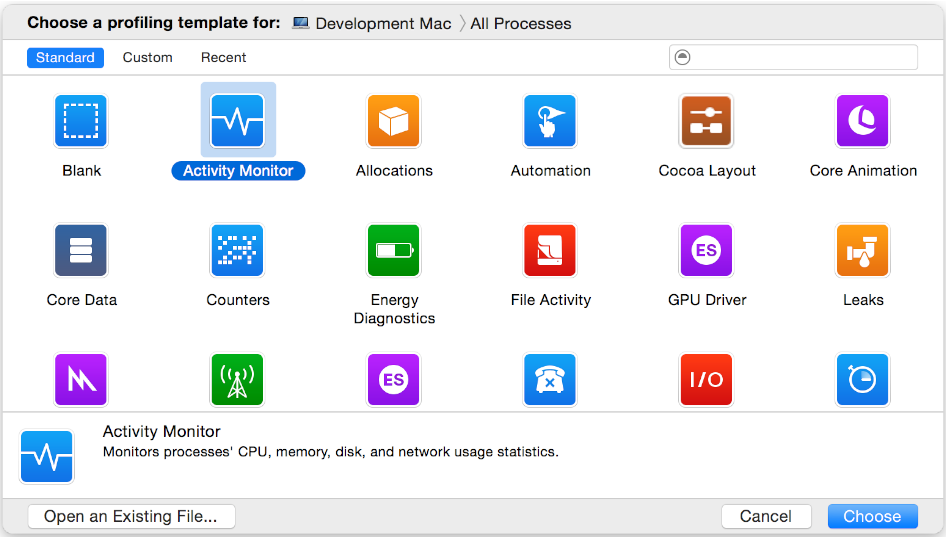


图6.1 Instruments启动界面

如图6.1所示，在默认情况下，Instruments提供了包括CPU、能耗、I/O、网络、GPU、位置信息等多项性能分析的instruments工具，开发者也可根据自己的实际需要定义不同的instruments工具，在本次测试中，主要使用到了CPU、能耗、网络等工具。

## 6.2 性能分析

### 6.2.1 显示终端性能分析

显示终端需要对接收的视频流进行解码转码操作，且通过Unity3D引擎构建整个画面，对CPU计算能力有较大的要求。

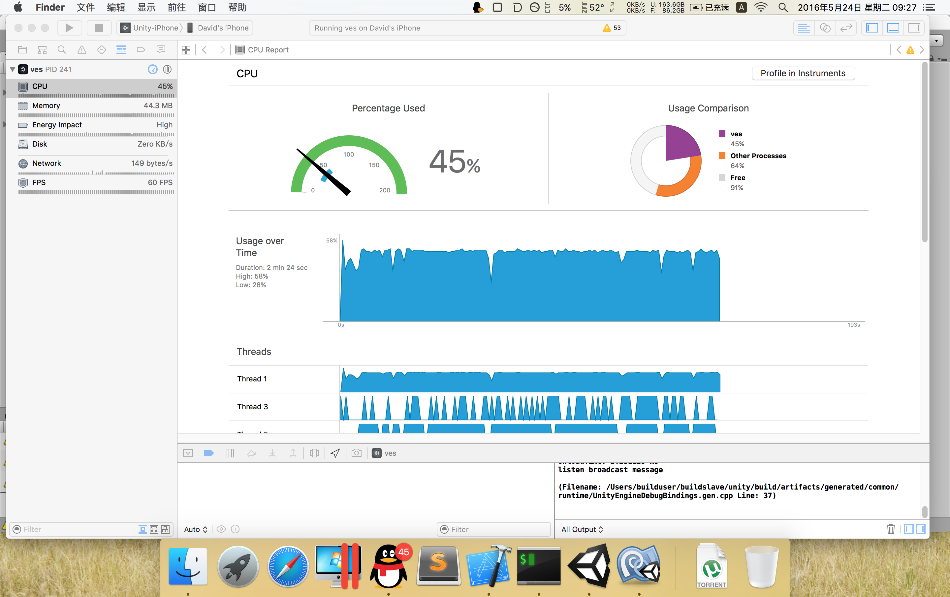


图6.2 启用解析视频流渲染画面前CPU使用量

图6.2展示了在启用解析视频流渲染画面前显示终端的CPU使用量，从图中可以看出，未启用视频流相关操作时CPU总体使用量在45%左右，因为此时并未启动视频流的解码转码操作，且OpenGL ES渲染画面操作也处于闲置状态，另外从右边也可以看出相对于设备上正运行其它进程，本系统的CPU使用量为其它进程的70%左右，处于一个较低水平。

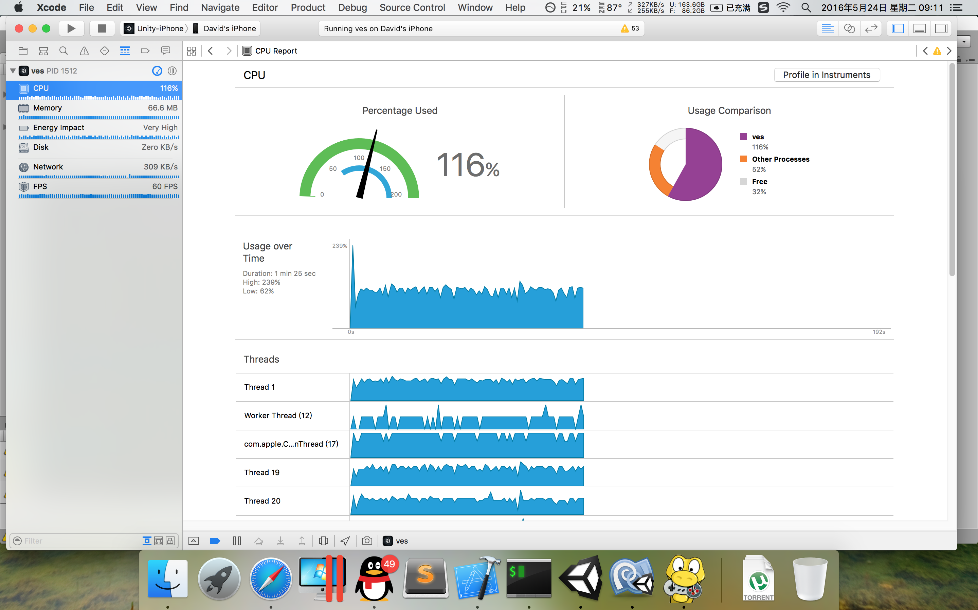


图6.3 启用解析视频流渲染画面后CPU使用量

图6.3展示了在启用解析视频流渲染画面后显示终端的CPU使用量，从图中可以看到，此时的CPU使用量已经达到116%，且对比其它进程，CPU使用量是其它进程的200%以上，由此可见，视频流的解析相关操作对于CPU来说是一项较为复杂的工作，但是由于iPhone出色的CPU性能，在如此繁重的工作下CPU仍能保持正常运转，并没有超出系统极限的200%，因此可以认为，在当前情况下，视频流的解析操作不存在CPU瓶颈。

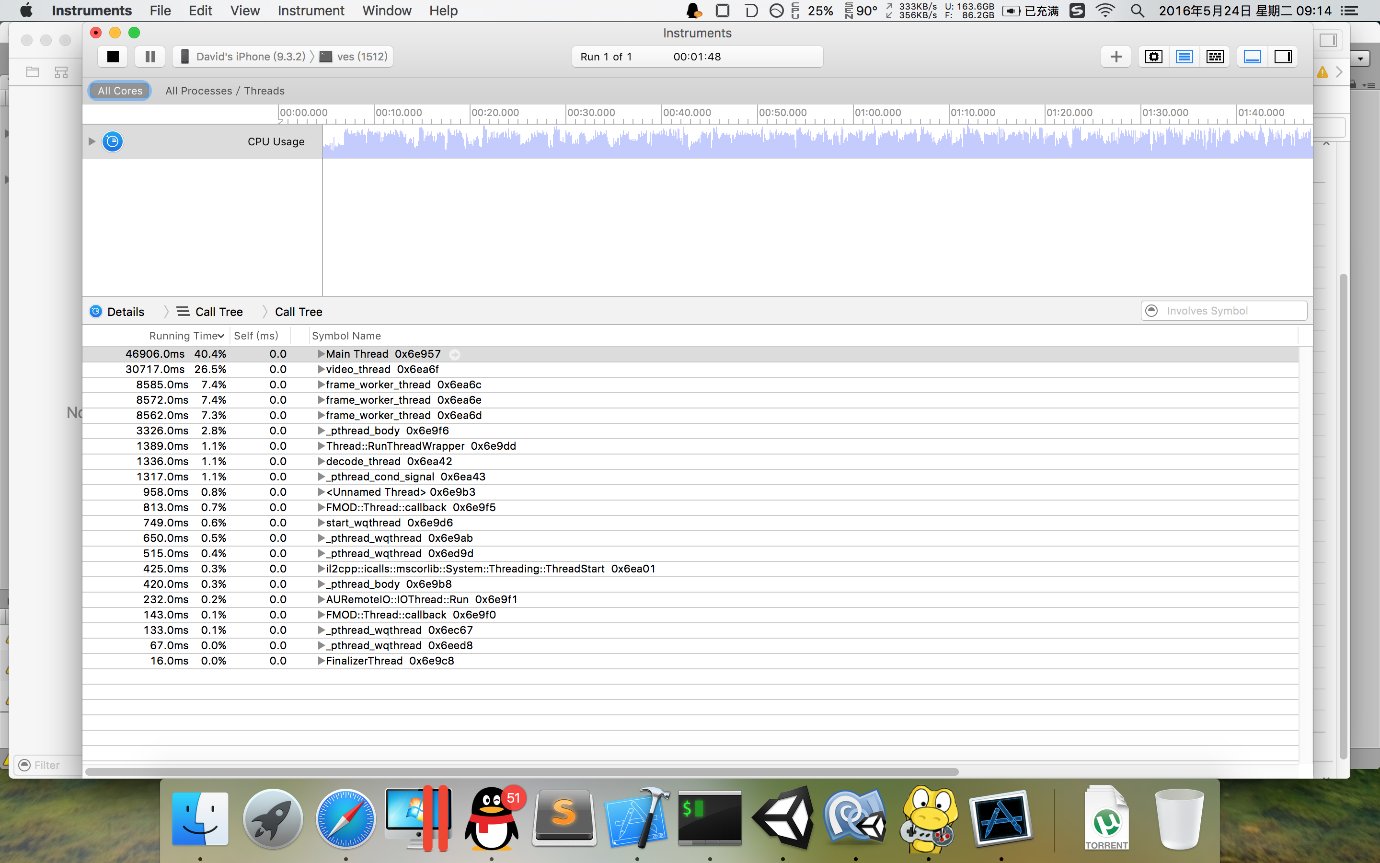


图6.4 Instruments中CPU详细使用情况

图6.4详细展示了在显示终端正常运行时，CPU的详细使用情况，由图可知，名为Main Thread的线程占用了近40%的CPU时间，根据相关文档得知，该线程为Unity运行时主线程，主要负责画面渲染相关操作。名为video\_thread的线程占用了26.5%的CPU时间，该线程为在前文提到编写的基于FFmpeg的Unity3D插件中的一个线程，负责将视频流从YUV420P格式转换为Unity3D材质使用的RGB24格式。这两个线程占用近70%的CPU时间也进一步说明视频流的解析相关操作对CPU来说是一项较为复杂的工作。

### 6.2.2 视频终端性能分析

视频终端需要通过摄像头捕获视频数据并生成视频流，相比于姿态获取、电机控制等模块，此项操作上会占用大部分CPU时间，由于Raspberry Pi运行基于Debian的Raspbian系统，可以使用Linux自带的一些工具对系统性能进行分析。通过使用两个SSH登录到视频终端，其中一个正常运行视频终端相关程序，另一个运行Linux的top命令，查看实时CPU使用情况。

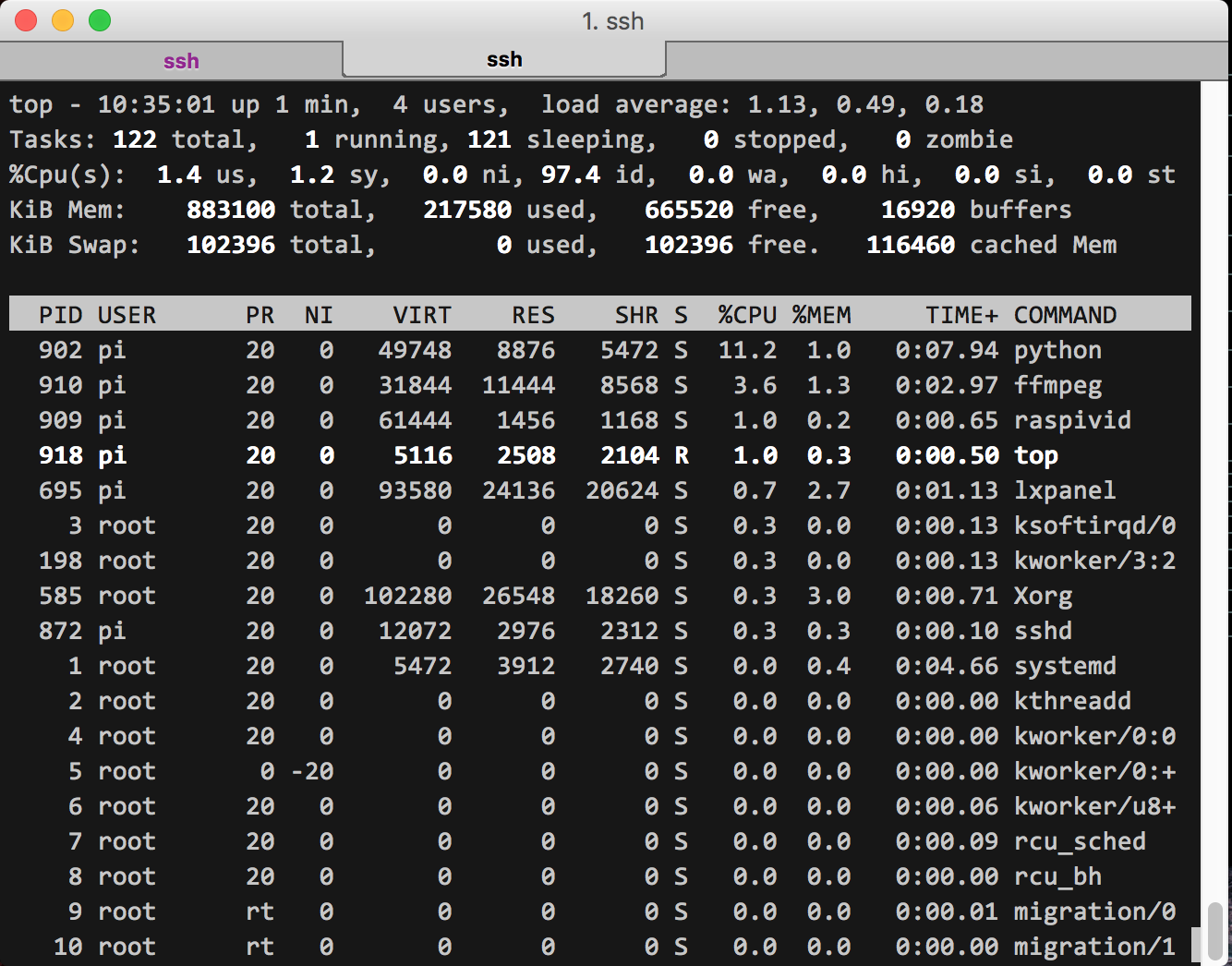


图6.5 视频终端运行时CPU使用情况

图6.5所示，与本系统有关的几个进程占用CPU比例排在前列，进程名分别为python、ffmpeg、raspivid，它们分别占用了11.2%、3.6%、1.0%的CPU使用量，据此可以分析，python进程作为整个软件部分的核心，负责整个视频终端的软件逻辑部分，且包含了姿态测算模块、电机控制模块等模块的所有线程，占用率达到11.2%合乎情理。raspivid是Raspberry Pi上的一个内置命令，负责捕捉静态图像或者动态视频，大部分时间在进行I/O操作所以CPU使用率不是很高。而ffmpeg进程负责将raspivid拍摄的视频进行处理并生成视频流进行发送，需要大量的CPU时间进行视频的处理，所以占用率在3.6%。因此，视频终端的CPU使用率与设计一致，符合要求。

## 6.3 网络带宽占用

本系统数据全部采用Wi-Fi的形式，对网络环境的稳定性要求较高，如果在信号较差或者信道较拥堵的Wi-Fi环境中，将极大地降低视频的流畅甚至大面积出现卡顿花屏的现象。因为采用了压缩编码的策略，视频流在Wi-Fi网络中传播所需要的带宽较低。

如图6.6所示，在本系统正常运行时，对于显示终端，下行速率为，上行速率为，从下方的活跃连接信息中可以看到，使用UDP协议的8093端口占用了绝大部分带宽，而8093端口在定义中即为视频流的流入端口。

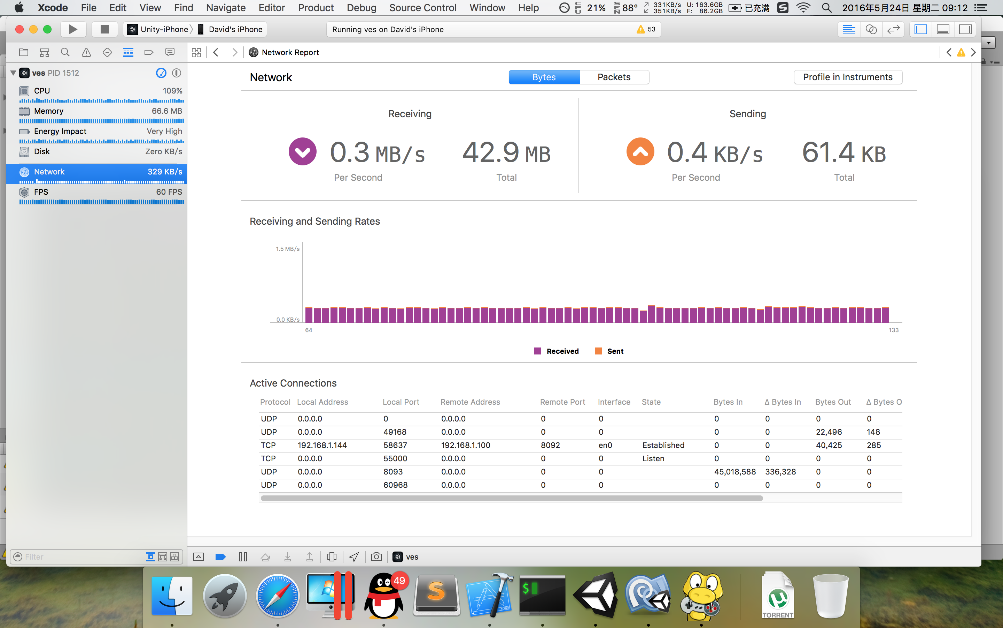


图6.6 系统运行时显示终端网络流量情况

根据中间件服务器的相关程序监测结果，中间件服务器的上行速率约为，而下行速率约为0.4，中间件服务器上行数据主要是提供给显示终端的转码视频流，因此中间件的上行速率和显示终端的下行速率几乎一致。

## 6.4 数据传输延迟

在本系统中，姿态数据的传输延迟由于数据量小且无复杂运算操作因此可以忽略不计，数据传输延迟主要在于视频流数据。



图6.7 摄像头拍摄的计时以及显示终端显示的计时

如图6.7所示，当显示终端显示计时器时间为，而此时实际计时器时间为，实际延后了约，根据多次测试结果，这一延迟时间约为左右，该延迟时间由于硬件、传输协议等限制，无法进一步减少。

## 6.5 电机控制

视频终端通过获得并处理中间件服务器传来的显示终端姿态数据，与自身姿态进行比较，通过脉冲控制电机的转动，如图6.8所示，红色星号为对显示终端姿态进行的采样数据，蓝色加号为对视频终端姿态的采样数据，在0-100段，系统刚完成所有初始化，显示终端和视频终端姿态不一致，视频终端通过调整自身姿态向显示终端姿态进行匹配。对初始化后的所有姿态变化调整时间进行量化分析，分析公式如下：

 （3.1）

其中，分别为显示终端的结束调整时间和开始调整时间，为视频终端的结束调整时间，为调整效率比，调整效率比理论最高为100%。

在320-350段以及480-520段，显示终端姿态小幅度变化，视频终端能够迅速调整自身姿态，调整效率比分别约等于81.8%和84.8%，在560-750段，显示终端姿态出现大幅度变化，视频终端仍能较好地完成调整姿态，但是此时的调整效率与之前小幅度变化相比下降很多，此时调整效率比为58.4%。由此可以得出，对于小幅度姿态变化，电机的控制能够较为快速完成反应，对于大幅度姿态变化，电机的控制调整较为迟钝。

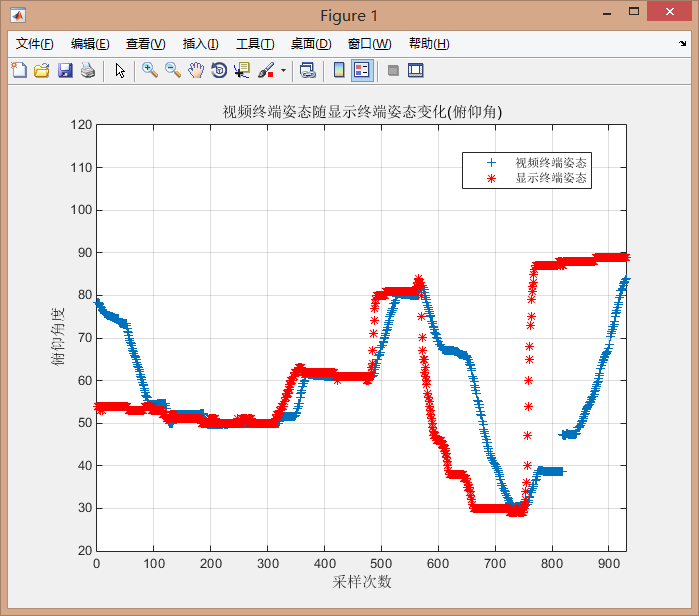


图6.8 视频终端姿态随显示终端姿态变化(俯仰角)

# 设 计 总 结

本系统在软件和硬件设计实现上均完成了设计指标。在显示终端方面，对姿态数据采用的平滑算法大幅降低了不必要的波动，为后面姿态数据的使用提供了方便。基于FFmpeg的插件使得在Unity3D中解析视频流变得方便，通过对显示终端的性能分析，该插件的解码性能以及解码表现也处于较为理想的水平。在视频终端方面，步进电机能够使用闭环控制较为准确的控制摄像头的姿态，姿态测算模块通过使用两片MPU6050芯片解决了使用欧拉角表示姿态带来的万向节问题，通过对MPU6050的追踪分析，在启动后的短时间内的启动误差也进行了修正，视频采集发送模块能够正确的根据得到的地址使用Raspberry Pi捕获视频数据并通过FFmpeg创建发送视频流。中间件服务器也正常的完成了视频流的处理以及姿态可视化的工作。

总的来说，本系统在功能上完成了预先设置的所有功能目标，但是仍然存在一些不尽如人意的地方，如电机的控制，在快速大幅度移动的情况下电机会出现移动较慢或者停转的情况，视频终端的姿态测算为了方便使用了欧拉角表示，而这带来的万向节死锁问题导致需要使用两个芯片解决，为进一步缩小硬件体积造成困难，整个系统目前只能工作在同一Wi-Fi网络下，对于一些特殊用途，这一条件限制颇多。

接下来的工作重点，将会进一步优化电机的控制，考虑采用直流电机等控制反应较快的电机，对视频终端的姿态测算提供更加精确简单的处理方式，封装整个系统的连接模块，对外提供接口，对内扩展连接方式，尝试提供跨局域网络的连接方式，考虑使用处理后虚拟现实效果更为明显的3D摄像头进行拍摄，使得画面具有更强的景深效果。

参 考 文 献

[1] Wikipedia. Virtual Reality [EB/OL]. 2016. http://en.wikipedia.org/wiki/Virtual\_reality.

[2] 百度百科. 虚拟现实 [EB/OL]. 2016. http://baike.baidu.com/link?url=G5FZmFiUwlTum20LG0kohzyMxyeFyQ18e9SbnbJ9iHaSvAYBe8IeaSz1ebYLKwdPVEwAkJCZJ5Fu3utkpPjLnq.

[3] Wikipedia. Python [EB/OL]. 2016. https://zh.wikipedia.org/wiki/Python.

[4] Wikipedia. C语言 [EB/OL]. 2016. https://zh.wikipedia.org/wiki/C语言.

[5] W. F. Abaya, J. Basa,M. Sy, A. C. Abad, E. P. Dadios. Low cost smart security camera with night vision capability using Raspberry Pi and OpenCV [C]. Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2014 International Conference on, 12-16 Nov. 2014:1-6

[6] S. N. Patel, V. Prakash. Autonomous camera based eye controlled wheelchair system using raspberry-pi [C]. Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on, 19-20 March 2015:1-6

[7] Goel, Saurabh. Cloud-Based Mobile Video Streaming Techniques [J]. International Journal of Wireless & Mobile Networks5.1 (Feb 2013): 85-92.

[8] Renaldas Zioma,Aras Pranckevičius. Unity: iOS and Android: cross platform challenges and solutions [C]. SIGGRAPH '12 ACM SIGGRAPH 2012 Mobile,2012.

[9] Google Inc. Cardboard SDK for Unity [EB/OL], 2014. https://github.com/googlesamples/cardboard-unity

[10] Carl Zeiss Inc. VR ONE Unity 3D SDK 1.3 [EB/OL], 2015. https://bitbucket.org/vrone/unity3d

[11] devsnd. Low-Latency Live Streaming your Desktop using ffmpeg [EB/OL], August 2015. http://fomori.org/blog/?p=1213

[12] Fabrice Bellard, Martin Bohme. An ffmpeg and SDL Tutorial [EB/OL], 2003. http://dranger.com/ffmpeg/

[13] Andrew Birkett. Reading data from the MPU-6050 on the Raspberry Pi [EB/OL], November 2013. http://blog.bitify.co.uk/2013/11/reading-data-from-mpu-6050-on-raspberry.html

[14] Andrew Birkett. 3D OpenGL visualisation of the data from an MPU-6050 connected to a Raspberry Pi [EB/OL], November 2013. http://blog.bitify.co.uk/2013/11/3d-opengl-visualisation-of-data-from.html

[15] Andrew Birkett. Using a complementary filter to combine Accelerometer and Gyroscopic data [EB/OL], November 2013. http://blog.bitify.co.uk/2013/11/using-complementary-filter-to-combine.html

[16] He Jin, He Jia-ming. Research on the Synchronized Transmission Algorithm of Embedded FFmpeg Multimedia Data [C]. IEEE International Conference on Internet Technology and Applications(Itap),2011,Page(s):1-3

[17] 树莓派实验室.树莓派 40Pin 引脚对照表 [EB/OL]. http://shumeipai.nxez.com/raspberry-pi-pins-version-40.

[18] S. Colton. The Balance Filter, A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform [D]. Massachusetts Institute of Technology, Cambridge, Massachusetts, 2007.

[19] 刘杰. H.264编解码算法在网络视频传输中的应用 [D].西安.西安电子科技大学.2010:40-45.

[20] 段学东,何九周.基于敏捷思想的iOS平台软件的设计与实现 [J].计算机技术与发展.2012:40-43.

[21] 张龙龙.基于ios平台的视频监控系统的设计与实现 [D]. 首都经济贸易大学 ,2012.

[22] 韩海宏.基于iOS平台移动视频监控客户端的设计与实现 [D]. 电子科技大学 2013.

[23] Boyd Wang. Adafruit的树莓派教程第十课：步进电机 [EB/OL]. 2014. http://www.geekfan.net/9926.

[24] M. R. Zakerinasab, M. Wang. A Cloud-Assisted Energy-Efficient Video Streaming System for Smartphones [C]. IEEE/ACM IWQoS '13, Jun. 2013:1-10.

# 致 谢

在这里我要首先感谢我的毕业设计指导教师冯林教授，冯老师在我完成毕业设计工作以及撰写毕业设计论文期间提供了许多帮助，同时对我在设计方向上存在的偏差也及时指出，并指导我从软件硬件方面不断完善。正是由于冯老师的悉心教导和无私帮助，我才能够顺利的完成毕业设计论文的写作。从最早的毕业设计选题到中期答辩到完成毕业设计论文，冯老师每周都会定期对我上一周完成的工作进行检查，听取我在设计实现以及论文写作上的一些想法，对我设计实现以及论文写作上存在的错误以及不足提出修改建议。在这里，再次感谢冯老师对我的支持以及帮助。

其次还要感谢我的辅导员于聪聪老师，在我完成毕业设计期间，于老师对我学习生活工作上都给予了无私的关怀，对我心理上出现的烦恼苦闷，于老师会耐心地帮助我解决。

我还要感谢与我朝夕相处的各专业的同学们，感谢大家在我完成毕业设计期间对我的帮助以及对我软件硬件设计上存在的瑕疵提出修改意见，以及在我完成硬件时为我提供的各种资源以及信息使我能够顺利完成硬件平台的设计和搭建。

我还要感谢电子信息与电气工程学部各位老师，感谢各位老师四年来的谆谆教导，让我学到了这么多的知识，感谢各位老师在中期答辩中对我的毕业设计提出的宝贵意见和建议，让我能够顺利完成毕业设计。

最后我还要特别感谢我的家人，是你们为了提供了完成学业的机会，也是你们在我遇到困难时为我提供最为坚实的后盾，正是因为有了你们的支持，我才能够无所畏惧的完成学业中以及毕业设计期间面对的所有挑战。