**分 类 号 学号 M201672904**

**学校代码 10487 密级**



**硕士学位论文**

**基于Kinect的手势识别系统的设计和实现**

|  |  |  |
| --- | --- | --- |
| **学位申请人** | **：** | **蔡碧海** |
| **学科专业** | **：** | **计算机技术** |
| **指导教师** | **：** | **张杰** |
| **答辩日期** | **：** | **2018年5月21日** |

**A Thesis Submitted in Partial Fulfillment of the Requirements**

**for the Degree for the Master of Engineering**

**An Gesture Recognition Tool on Kinect: Design and Implementation**

|  |  |  |
| --- | --- | --- |
| **Candidate** | **：** | **Bihai Cai** |
| **Major** | **：** | **Computer teconology** |
| **Supervisor** | **：** | **Jie Zhang** |

**Huazhong University of Science & Technology**

**Wuhan 430074, P.R.China**

**May, 2018**

**独创性声明**

本人声明所呈交的学位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，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□， 在 年解密后适用本授权书。

本论文属于

不保密□。

（请在以上方框内打“√”）

学位论文作者签名： 指导教师签名：

日期： 年 月 日 日期： 年 月 日

摘 要

在人工智能和大数据技术飞速发展的今天，基于图像的识别技术也必将发挥极其重大的作用。与此同时，新的人机交互方式在也在迅猛的发展着，比如语音识别技术、手势识别技术、人脸识别技术等，这些技术使得人不必受限于遥控器，手柄，鼠标键盘等控制设备，以更自然的方式和计算机进行交互着。手势识别技术就是其中比较热门的研究方向，也必然影响着未来。一方面，新型的人机交互模式体现了人们对于人机交互中人扮演角色的思考，人往往是人机交互的主体，新的人机交互必然会朝着人性化、便捷化、自由化的方向发展。另一方面，在准确识别人的手势的基础上，也必定会影响机器人研究过程中对于人机交互的思考，准确的手势识别使得机器人能够真正看懂操纵者的“语言”。在花大精力研究遥控系统的同时，简单的手势会显得更加地便捷，并且能用简单的手势可以解决的问题，往往使得遥控器的交互过程会显得比较复杂。基于机器视觉的人体动作和手势识别是实现新一代人机交互系统所不可缺少的一项关键技术。作为一个革命性的产品，Kinect体感设备能够获取彩色图像、深度图像以及人体骨骼图像，并提供了一种全新的人机交互的方式。它能够捕捉、跟踪人体的动作、手势和声音。

在此基础之上，本课题旨在研究一种利用Kinect获取手势信息，并基于动态时间规整算法（DTW）的手势识别的交互系统，并将这种交互系统运用于实践，代替鼠标和键盘的一些基本的操作。

**关键词：**图像处理，手势识别，人机交互，动态时间规整，计算机视觉

Abstract

英文摘要，应用格式“宋体小四”

**Key words：**HUST, Thesis, Word, Template

目 录

[摘 要 I](#_Toc512245437)

[Abstract II](#_Toc512245438)

[目 录 III](#_Toc512245439)

[1 绪论 1](#_Toc512245440)

[1.1 研究背景 1](#_Toc512245441)

[1.2 国内外研究现状 2](#_Toc512245442)

[1.3 论文的内容和意义 5](#_Toc512245443)

[2 关键技术概述 7](#_Toc512245444)

[2.1 Kinect简介（3000字） 7](#_Toc512245445)

[2.2 OpenCV概述 8](#_Toc512245446)

[2.3 动态手势识别（2000字） 8](#_Toc512245447)

[2.4 本章小结 8](#_Toc512245448)

[3 手势图像数据的获取和处理 9](#_Toc512245449)

[3.1 基于Kinect的需求分析 9](#_Toc512245450)

[3.2 图像数据的获取 9](#_Toc512245451)

[3.3 图像获取的模块设计 12](#_Toc512245452)

[3.4 图像数据的处理 12](#_Toc512245453)

[3.5 本章小结 17](#_Toc512245454)

[4 手势识别模块的实现 18](#_Toc512245455)

[4.1 手势轨迹的获取 18](#_Toc512245456)

[4.2 基于数值特征的手势识别 20](#_Toc512245457)

[4.3 DTW算法核心技术 22](#_Toc512245458)

[4.4 基于DTW算法的手势识别 27](#_Toc512245459)

[4.5 手势识别的模块设计 35](#_Toc512245460)

[4.6 本章小结 35](#_Toc512245461)

[5 系统框架搭建和测试 36](#_Toc512245462)

[5.1 软件系统的实现框架 36](#_Toc512245463)

[5.2 几种手势的应用场景 36](#_Toc512245464)

[5.3 手势识别过程分析 36](#_Toc512245465)

[5.4 本章小结 36](#_Toc512245466)

[6 总结与展望 37](#_Toc512245467)

[6.1 总结 37](#_Toc512245468)

[6.2 展望和改进 37](#_Toc512245469)

[致谢 38](#_Toc512245470)

[所有题录的数据正常 40](#_Toc512245471)

# 绪论

## 研究背景

如今，计算机技术正在高速地发展着，在人机交互的技术方面，人和计算机的交互也越来越多样化，然而新的人机交互方式也朝着更加自然和更加便捷的方向发展着。人工智能技术的发展使得基于图像和语音的人机交互方式更加稳定和准确地能表达人与机器之间的“交流”。目前基于视觉的手势识别方法是智能化、鲁棒性的人机交互领域中学者研究的热门方向[1]。

随着计算机互联网技术在我国的迅速普及，人们发现传统的鼠标键盘等机械输入方式，在三维空间自由度方面越来越体现其所固有的局限性，这种交互方式可能并不是对所有人都能很快上手，有一个适应学习的过程，因此更加随心所欲的人机交互方式呼之欲出。随着一些科学技术的发展，越来越多的研究更多的在关注人脸识别、语音识别、人体识别等方面对于人机交互方面的应用。

一方面，新型的人机交互模式体现了人们对于人机交互中人扮演角色的思考，人往往是人机交互的主体，新的人机交互必然会朝着人性化便捷化自由化的方向发展，与相较于计算机的鼠标键盘交互模式、游戏机手柄的交互模式、智能家电遥控器的交互模式、平板电脑和智能手机基于触摸屏的交互模式想比较而言，基于空间的手势识别的交互模式将显示出其巨大的发展潜能，这种技术体现了对于人而言相对自由便捷的交互方式。

另一方面，在准确识别人的手势的基础上，也必定会影响机器人研究过程中对于人机交互的思考，准确的手势识别使得机器人能够真正看懂操纵者的“语言”。在花大精力研究遥控系统的同时，简单的手势会显得更加地便捷，并且能用简单的手势可以解决的问题，往往使得遥控器的交互过程会显得比较复杂。

其次，深度摄像头设备的出现，使得手势识别这一种技术的实现变得更加方便。相较于传统二维图像的基于图形学的轮廓检测和跟踪，三维摄像头由于提供了z轴方向上的深度数据，因此类似于手掌按压这种二维图像这种几乎很难实现检测的动作，在三维摄像头中得到了很好地检测和跟踪。作为一个革命性的产品，Kinect体感设备能够获取彩色图像、深度图像以及人体骨骼图像，并提供了一种全新的人机交互的方式。它能够捕捉、跟踪人体的动作、手势和声音。因而基于Kinect手势识别的相关研究也层出不穷，这一点也使得新型的基于深度的人机交互方式有了很好地实现背景和应用前景。

在此基础之上，本课题旨在研究一种利用Kinect获取手势信息，并基于动态时间规整算法（DTW）的手势识别的交互系统，并将这种交互系统运用于实践，代替鼠标和键盘的一些基本的操作。

## 国内外研究现状

手势是指手指手掌或者手掌连同手臂产生的各种动作或姿势，其所能表现的含义是十分丰富的，并且手势表达有着非常简单自然的特点，因此手势作为人机交互的一种输入方法是比较理想的。一般而言，手势识别有两种类别：静态手势识别和动态手势识别，这两种手势识别的方向都有着比较悠久的历史和相对成熟的研究理论，从不同特征而言，静态手势识别技术着重点在于研究手部的形状变化，对于二维图像中的手势识别技术而言，理论是趋于成熟的，而动态手势识别技术会较为复杂，它更着重于研究手掌在三维空间中位置变化，针对这些某段连续的位置变化信息产生手势语义，对其进行实时分类。

手势识别所具有其他一些交互方式不存在的优势：其一、手势是比较容易别用户理解的，比如用手比一个“OK”的动作就可以表示确认；其二、使用手势与计算机进行交互可以更加自由地表达用户的想法；其三、准确的手势识别可以提高用户和机器之间交互的效率。因此国内外的研究者在手势识别方面进行了大量的工作，也有着丰厚的研究成果。对于手势识别的方法，大致可以分为三种：基于数据手套的手势识别，基于双目摄像头的手势识别和基于Kinect的手势识别。

### 基于数据手套的手势识别

基于数据手套的手势识别时目前运用比较广泛的研究方法，同时也是比较经典一种研究方法。数据手套能直接收到传感器反馈给系统的用户手指尖在三维空间中的坐标位置和手指运动的相关信息，因此用于手势识别时，数据精度是比较高的，而且可识别的手势种类也比较多，但也存在一些缺陷：例如由于这种设备本身昂贵的造价，使得这方面的应用产品并不常见，同时这种设备本身对于用户来说体验度不是很好，长时间的佩戴可能会造成手掌心出汗等。

最早进行手势研究的是AT&T的Grimes，他在1983年发明了“数据手套”[2]，其前身是手套式传感器系统，早期的产品是Sayre Clove[3]。此后，基于这种设备的研究层出不穷，Kim等人利用一种叫做KHU-1的数据手套开发了一个三维手掌移动跟踪和手势识别系统，该设备包含一个三轴加速传感器、一个控制器和一套蓝牙机制，能够通过无线蓝牙给PC设备发送手掌移动信号，并利用获取到的三维数据手掌模型进行手势识别[4]；J.Weissman和R.Salomon利用数据手套对手指关节的角度提取了18个测量值，实现了一套面向虚拟现实应用程序的手势识别系统[5]；D.L.Quam做了一项使用可以准确获取到手指位置的数据手套进行手势识别实验，对22类手势进行了稳定的识别[6]；韩国成均馆大学的In-Kwon Park等人利用数据手套基于FPGA（现场可编程逻辑门阵列）电路，实现了一套手势识别系统，解决了无线手势识别系统中用户和设备在空间和移动上的限制以及光照等环境因素造成的识别不稳定的问题，其平均识别率达到了94%以上[7]。

### 基于双目视觉的手势识别

基于数据手套的手势识别方法在早期的手势识别的研究过程中起着中重要的作用，这种方法虽然对于识别率和准确率的稳定性上有着足够的保证，但是这种也存在着一些不足的地方，比如手势识别不够便利，不够自由等。近30年来，人们越来越关注更加自然便捷地手势识别，这也是目前国内外手势识别的趋势。

2013年，Leap公司发布了面向PC和苹果电脑Mac的体感控制器Leap Motion，它是一个基于双目视觉的手势识别设备。双目视觉就是有两个摄像头，利用立体视觉成像的原理可以提取手掌在三维空间中的位置，然后对手部建立立体模型，根据获取到的手掌数据实现手势识别。

国外在双目视觉方面做了比较多的研究，其中，A.A.Argyros等人[8]实现了一种利用两种彩色摄像头提供的视频流数据确定三维手势位置的手势识别方法，使用一种视角校准算法计算两种信息之间形成的三维数据；日本的A.Utsumi等人[9]使用多摄像头系统检测手部的骨架图片实现了基于COG（手掌中心点）检测算法的一种可行的手势识别系统，解决了普通摄像头下获取到的手掌和手臂区域连接的情况会造成手势识别的结果非常不稳定的情况；在90年代初，Krueger[10]基于一个双摄像机的Video Desk系统，实现了对双手捏去和托转等手势的匹配；J.Segen[11]实现了一个可以用双摄像头捕获手势的系统，该系统对点、点击和伸张这三种手势进行了准确的识别，并且利用这三种手势实现了对控制虚拟物体的功能；多伦多大学的Mailik等人[12]实现了一个叫做Visual Touchpad的系统，该系统在双摄像头和一块黑白的矩形纸板基础上，实现了对手指进行检测和跟踪，能比较精确地获取手指的位置和移动方向，对PC机实现了人机交互的功能。

同时，国内在基于双目视觉方面研究也有一些研究成果：浙江大学的郭康德[13]通过计算机双目视觉技术实现了一个三维指尖检测系统，利用算法基础以及其他关键技术实现了一种框架，并利用该框架实现了三维鼠标和若干基于指尖点击、指尖移动和手掌翻转等多种手势的三维手势交互游戏；郑州大学的谭同德等人[14]根据双目视觉定位数学模型计算目标位置信息的算法，提出了一种一人手的质心为特征点提取的方法，实现了一个队虚拟物体进行抓取、移动和释放操作的系统。

### 基于Kinect的手势识别

针对双目摄像头进行的手势识别的研究，要获取比较精准地手势识别所需要的手势位置以及轨迹信息，可能需要时效性更高的双目摄像头设备，这是因为双目摄像头获取到图像必须是同步的，否则无法准确计算出场景中运动部分的深度信息[15]，同时，由于双目摄像头图像获取需要特定的光照环境，过明或者过暗的环境会影响手势识别的精度，这是目前利用普通双目摄像头实现手势识别时存在的不足之处，而深度摄像头技术在这个方面也显示出极大的优势。

目前相对成熟的带有深度摄像头的产品之一是Kinect，近年来，国内外学者在Kinect手势识别方面做了很多相关的研究，其中，新加坡的Zhou Ren等人[16]基于Kinect传感器实现了一种鲁棒的手势识别系统，该系统提出了一种通过计算手指EMD（FEMD，Finger-Earth Mover's Distance）距离的计量方法，来测量手势形状之间的不同点，同时消除了从Kinect摄像头获取到手势形状带来的噪声，使得手势识别率达到了93.2%；印度的K.K.Biswas等人[17]使用了支持向量机（SVM，Support Vector Machine）的方法，构建了一个基于Kinect设备的手势识别系统，并实现了对8种典型的手势的准确识别；美国的Arun Kulshreshth等人[18]实现了一种基于Kinect设备的手指检测和追踪技术，并将实验结果和传统的基于K-曲率的手指尖识别和检测的技术进行了对比，初步的结果证明了Kinect设备获取到的稳定的手指尖技术能达到传统的K-曲率方法获取手指尖技术的效果；F.Pedersoli等人[19]搭建了一个叫做XKin的开源动态手势识别的框架，该框架从Kinect设备中获取动态手势轨迹，并基于隐马可夫（HMM）分类模型的方法，对16中美国手语进行了自然且直观的识别；基于Kinect的3d视觉和语音识别功能，S.Fakhteh等人[20]实现了一个游戏软件，主要用于识别聋哑人的手势，解析手势命令，帮助他们得到实时的游戏体验。

此外，国内有很多将Kinect作为一种输入设备，获取人体骨骼的坐标信息，并进行手势识别，实现了一些人机交互的应用。上海交通大学的钱鹤庆[21]在利用Kinect传感器获取手部轮廓的基础上，使用人工智能、计算机视觉和多媒体技术，开了一种结合手势识别和增强现实技术的教育辅助系统，该系统选取DTW算法进行动态手势识别，并提出一种动静态结合的手势识别方法；北京工业大学的李小龙[22]使用结合Unity3D的三维场景渲染系统和脚本控制系统，使用Kinect获取控制人体解剖学中的人体模型的的输入数据，定义了一些常用的手势接口，实现了实时识别手掌握拳、张开、向左或向右滑动、两手张开或并拢等手势，用于控制相应模型的抓取、放开、向左或向右移动、放大或缩小等操作。通过以上方法实现了具有较高的实用性、交互性与趣味性的虚拟解剖教学系统。香港大学的Chong Wang等人[23]提出了一种基于超像素的EMD（SP-EMD, superpixel earth mover's distance）的测量方法，对从Kinect传感器获取出来的深度信息进行处理，实现快速和高校的手势识别系统，并将这种系统应用于实践。

由上可知，手势识别的一直是计算机视觉和人工智能领域中比较热门的研究方法，而由于近年来深度摄像技术的快速发展，一些基于深度图像的手势识别方法被提出或者被改进，稳定的鲁棒的手势识别方法一直是计算机领域研究的趋势，一些成熟的理论和技术已经应用于实践。

## 论文的内容和意义

### 课题的内容和意义

前文对手势识别领域的三个研究方面做了目前国内外研究现状描述，相较于基于数据手套这种接触性设备的手势识别和基于双目视觉这种基于视觉性设备的手势识别，Kinect为研究者提供更加便利的图像数据，尤其是它具备可以同时获取深度图数据和人体骨骼数据，使得它是用作手势识别系统研究的一种比较好的选择，因此本文是基于Kinect获取的图像数据，实现的一种手势识别系统。本课题主要研究的内容概括为:

1. 在微软提供的Kinect for Windos SDK v2的基础上，设计一个基于数值特征的动态手势识别接口系统，对单手左右滑动，手掌向前点击、双手张开和双手角度变化进行识别。
2. 研究动态手势识别的先关算法，并实现基于模板匹配的DTW动态手势识别算法，对具有特定含义的手势进行稳定的识别。
3. 实现一个基于手势识别接口系统的图片浏览器，并利用识别到的手势实现左右翻页、图片选择、图片缩放和旋转的功能。

本课题研究的主要意义在于：

1. 可以应用于虚拟教学领域：手势识别系统的特点之一就是它能够简单快捷自然地表达用户的想法，并将它传递给机器，这使得在虚拟教学领域能更加方便地对机器进行控制，比如在教学中可以直接用双手来演示教学ppt；
2. 可以帮助语言残疾者：研究动态手势识别可以将手语动作翻译自然语言，实现这种手势识别的框架对于帮助语言功能残疾者是十分有意义的；
3. 可以虚拟现实辅助功能：目前虚拟现实技术还在发展中，而再一些娱乐领域正起着十分重要的作用，一些技术已经商业化，比如索尼的PSVR就是实现目前最新的VR技术。那么手势识别作为一种更加自然的人机交互方式必定会在未来的娱乐领域大放异彩。

### 论文组织结构

本论文由六个章节组成，其组织结构如下所述：

第一章介绍的是本文的的研究背景和意义，介绍有关手势识别的相关含义，之后从基于数据手套、基于双目视觉、基于Kinect这三个方面上逐一介绍了目前国内外手势识别在人机交互和计算机视觉中的相关研究和应用现状，并介绍论文的主要内容和意义，最后给出了论文组织结构。

第二章介绍的是手势识别技术的相关步骤和算法，从手势检测和手势识别两个方面介绍，为之后的章节做铺垫。

第三章首先提出的基于Kinect手势识别的需求分析和系统框架，然后依次谈到图像数据获取的流程和各个图像数据处理模块的设计工作，详细说明每个模块的意义、具体要解决的问题以及具体的执行流程。

第四章首先介绍了手势轨迹获取的相关流程设计和实现步骤，之后说明基于数值特征的手势识别的具体实现。从总体程序设计的角度说明实现的具体内容，然后重点阐述了基于DTW算法的手势识别模块的具体实现思路，然后对于手势库建立给出具体实现过程。

第五章是对本文所设计和实现的手势识别系统进行测试，首先分别对图像获取和处理模块单独测试，之后再对手势识别模块进行测试，并对每一个测试的结果进行结果展示和分析，对最终的测试结果进行评估。

第六章对本文进行了总结和展望，首先总结了本文的工作成果，最后客观地分析了本文所实现的手势识别系统的不足，同时给出了未来可以改进的方向。

# 关键技术概述

## Kinect技术概要

本文使用了Kinect for windows v2来获取手势轨迹信息，Kinect2设备能够获取深度图像数据，并且支持实时的全身和骨骼跟踪[24]，Kinect2具备红外感应器，因此即使在黑暗环境下也能稳定地获取深度图像数据，Kinect 2的优势在于它能够基本不受外界光照环境影响，稳定实时的获取深度数据，相较于比普通二维摄像头，它能够获取物体在三维空间中的具体位置。

### Kinect简介

在2010年，微软发布了一款给其游戏机产品Xbox360提供的体感控制设备Kinect，其全称为Kinect for Xbox360，该设备包含了一个RGB彩色摄像头、一对深度摄像头一组麦克风阵列局别即时动态捕捉、影像识别、语音识别等功能。该设备能对玩家全身上下的动作进行捕捉，使得游戏者可以脱离手柄等游戏控制器的束缚，使用语音指令或者手势动作及其他人体肢体动作更加直观地对游戏进行控制。

2012年，微软把这一先进的技术推广到了Windows平台，即Kinect for Windows使得其不仅仅局限于娱乐游戏这一行业。微软鼓励更多的开发者来设计基于Kinect体感技术的应用，在各个行业去改变人们的工作、生活和娱乐方式。

2014年，微软发布了Kinect第二代的产品Kinect for windows v2，相较于一代有着明显的提升。Kinect 的外观如图 2‑1所示。

|  |  |
| --- | --- |
| https://img-blog.csdn.net/20171107091402236?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcXFfNDAyNTgxOTk=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast | 这里写图片描述 |
| 1. Kinect 1 | 1. Kinect 2 |

图 2‑1 Kinect一代和二代外观图

Kinect的最大创新之处在于，就是它对于传统的人机交互方式的变革：传统的人机交互方式是图形用户界面（GUI），其比较明显的特点就是用户必须要学习软件开发者预先设定好的操作，在传统的显示设备上进行交互，而Kinet提供了一种自然用户界面（NUI）的人机交互方式，使得用户能以最自然的方式和机器互动。

### Kinect原理

从系统架构的角度来看，Kinect包括四层，从下到上依次是：硬件层、驱动层、接口层和应用层。如图 2‑2所示，

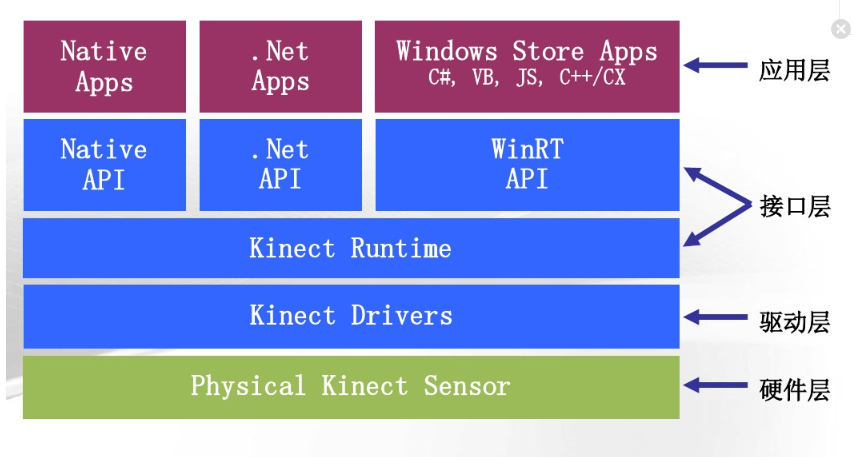


图 2‑2 Kinect系统架构示意图

在硬件层，第二代的Kinect配备了彩色RGB摄像头、红外摄像头、红外线投影机，同时，第二代Kinect还具备四元线性麦克风阵列。彩色摄像头用于拍摄视角范围内的彩色视频图像，红外摄像头和红外投影机用于获取可视范围内人体物体的深度图像，在一代的Kinect中，获取深度图像的原理是采用了以色列PrimeSensor公司的Light Coding技术，即将红外光投影到一维或者二维的图像中，根据图像的形变情形可以判断被测物体的表面形状，当物体离Kinect设备距离不同时，图像的形变参数就会不同，根据存储在内部的模板进行匹配，就能获取每个像素的深度值，与一代的Kincet所不同的是，第二代的Kincet采用的深度测距原理是TOF（Time of Flight，光线飞行时间）技术，它是一种主动式深度感应技术，是使用激光探测目标物，并在反射出接收目标物的反射光，根据测量时间差算出目标物的距离[25]；Light Coding技术和TOF技术获取深度数据如图 2‑3所示。

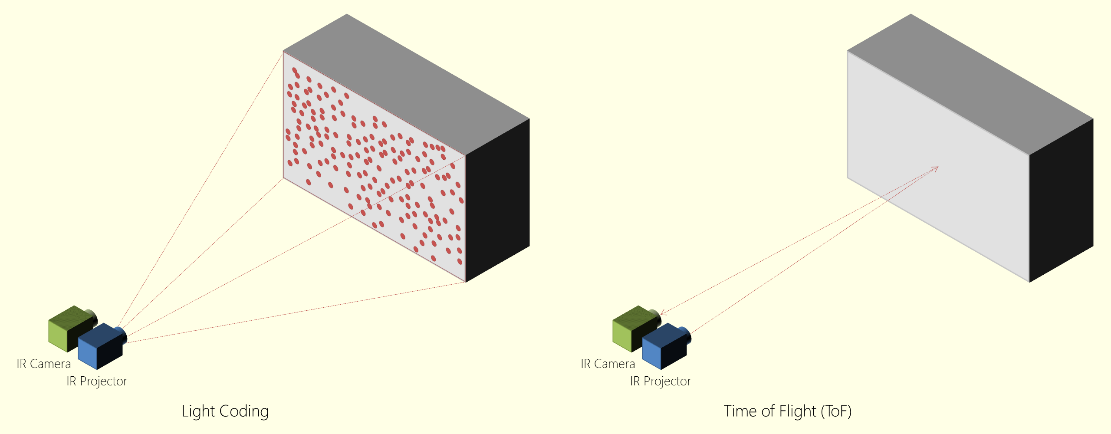


图 2‑3 LigthCoding和TOF获取深度数据的示意图

与第一代的Kincet相比较，第二代的Kincet传感器在性能上都有着大幅度的提升，具体表现如表 2‑1所示，

表 2‑1 Kinect v1和Kincet v2参数比较

|  |  |  |
| --- | --- | --- |
| 项目名称 | Kinect v1 | Kincet v2 |
| 颜色图分辨率 | 640×480 | 1920×1080 |
| 深度图分辨率 | 320×240 | 512×424 |
| 可检测人物数目 | 6 | 6 |
| 可追踪人物骨骼数目 | 2 | 6 |
| 可检测骨骼关节点数目 | 20关节点/人 | 25关节点/人 |
| 每秒帧数 | 30fps | 30fps |
| 摄像头侦测距离范围 | 0.8~4.0m | 0.5~4.5m |
| 摄像头视野水平角度 | 57° | 70° |
| 摄像头视野垂直角度 | 43° | 60° |

在Kinect 2的接口层，微软提供了六种数据源，以供开发者使用：

1. 彩色帧数据（ColorFrameSource）：提供30fps的1920\*1080分辨率的彩色图像，支持RGBA、YUV等格式；
2. 红外帧数据（InfraredFrameSource）：提供30fps的512\*524的红外帧数据，且每个像素占2个字节的存储空间；
3. 深度帧数据（DepthFrameSource）：提供测量范围为0.5~4.5米的物体深度图帧数据，每一帧中的深度表示红外摄像头到该物体的距离，单位是毫米；
4. 人体索引帧数据（BodyIndexFrameSource）：提供追踪到的最多6个人体的索引值，用一个字节来表示，其中0~5表示被追踪的人体索引编号，-1（0xFF）表示为发现人体；
5. 人体骨骼帧数据（BodyFrameSource）：提供最多6个人体的25个关节点的集合，每个帧中包含了关节点的三维坐标点和方向信息，每秒30fps的刷新率；
6. 音频帧数据（AudioFrameSource）：提供Kincet的四个麦克风阵列的声音接收覆盖角度为100度的声音信息。

在接口层提供者六种数据源的目的就是为了能够很方便地在应用层获取最后需要的数据，在Kincet2里这个获取数据的步骤中，获取数据会依次调用Sensor->Source->Reader->Frame->Data这些接口。获取手势图像就是通过这种方法得到的，具体步骤见第三章。

### Kinect局限

虽然目前Kinect的技术比较成熟，但由于目前深度摄像头依旧处于发展的阶段，Kinect也存在着一些局限，主要体现在以下三点：

1. Kinect的获取深度数据范围比较有限：对于Kinect 2，只有人体在0.5~4.5米的范围之内，才能比较准确的获取到深度图像数据[2]。
2. Kinect的骨骼数据必须在全身出现在摄像头之内的情况下才能准确无误的获取到，否则会出现一些偏差。
3. 虽然kinect2有自带的手势识别技术，但是对于动态的手势识别并没有比较完善的实现接口。

## 手势识别技术（3000字）

手势识别的过程的第一步就是手势检测，在这一步中，主要完成的工作就是对计算机视觉中手掌位置的描述和定义，同时在时间序列中对手掌的位置进行实时的跟踪；那么手势识别过程的第二步就是对获取到的手掌信息进行分类处理，给获取到手掌位置特征点定义特定的手势含义，根据既定的手势含义对手掌位置信息进行实时分类。

### 手部特征值提取

在基于数据手套的手势识别中，由于数据手套这种传感器本身可以比较便捷地获取到手部区域，所以获取手部特征点不是一个难点，而在基于自然视觉的手势识别的方法中，由于获取手部区域的视频图像往往不是单独存在的，因此无法十分方便获取到手部区域的一些新，所以在这种方法中对于手部区域的分割是比较重要的一点，也是影响到手势识别率的关键性因素。提取手部特征值进行手势识别的一般流程如图 2‑4所示。



图 2‑4 提取手部特征点手势识别流程

在这个过程中第一步就是手势分割，对于基于计算机视觉的手势分割可以有两种可行的方法：

1. 肤色模型：在现实生活中，人种之间的差异导致不同的人种的肤色不尽相同，其具体表现为黄种人、白种人和黑种人，但是尽管存在着差别，在对人类肤色排除了环境造成的一些影响后，色调是基本一致的。这种方法的基本原理是在YCbCr（或者HSV）颜色空间中构造一个肤色检测模型，然后对足够多的肤色样本进行训练，使用机器学习的方法对待识别的有肤色的图像数据进行分类识别[26]。肤色检测模型中有单高斯模型、混合高斯模型、贝叶斯模型和椭圆模型等，通过前人的研究可知，皮肤像素点在CbCr构成的坐标系中可以近似看做一个椭圆分布，因此肤色模型可以归结为判定坐标点(Cb, Cr)是否在该椭圆和其边界上[27]。

基于肤色模型的手势提取主要局限在于：第一、肤色模型对于外界环境要求比较高，光照等条件对于肤色检测影响比较大，强光或者黑暗条件会是肤色检测的准确率急剧下降；第二、对于人体皮肤其他的部分往往不能绝对区分开来，对于需要准确识别到指尖点或者提取轮廓点这些步骤影响会比较大。

2. 深度阈值：对于配备有深度摄像头的设备能够获取具有人和物远近距离的深度图像，同时在能够准确追踪到骨骼点坐标信息的情况下，可以通过深度阈值的方法对人体手部进行分割。在获取到的手掌骨骼点附近，规定一个深度阈值d，手掌平面的宽W和高H，然后在深度图空间中规划一个长为L，宽为W，高为H的立方体空间区域，其中长度L = 2×d，这样可以将立体空间中的手部区域提取出来。这个过程如图 2‑5所示。



图 2‑5 深度阈值提取手部示意图

这种方法主要利用到深度图数据中的深度信息能表示物体在三维空间中距离摄像头的远近，那么它的特点很明显：首先、摄像头必须能够十分准确的获取并追踪人体骨骼数据；其次、摄像头必须能够十分稳定地获取到深度图数据，以便在人体手掌骨骼点附近对深度数据进行划分。

Kincet就是兼具了以上两种特性，所以在本文中是用Kincet对手掌区域进行分割提取，后续的具体步骤间第三章。

### 动态手势识别的方法

动态手势识别这个概念是相对静态手势识别而言的，它指的是对于一系列具有时间顺序的手势序列进行识别，准确表示用户要表达的意思。动态的手势包括手的位置运动轨迹，手的旋转等其他变化。其一般的具体的识别流程如图 2‑6所示。



图 2‑6 动态手势识别一般流程图

首先，获取动态手势的序列，然后对于手势序列进行手部分割，将手部区域提取出来，根据提取出来的手部区域计算特征值，一般是手势轨迹的坐标点，也可能是手对于某个方向上的夹角余弦值或者手旋转的某个角度值，对于这一序列的数据进行组合，形成一个特征值序列。最后在已经定义好的手势库中去匹配手势模板，并对匹配结果进行评估，得到最终的识别结果。动态手势识别的方法主要有以下几种方法[2]：

1.基于模板匹配的方法：这种主要是将每个动作的特征序列先用模板的方式保存起来，然后对于实际中获取到的手势序列进行与模板的匹配，计算与每个模板中手势序列之间的相似程度，选取相似程度最高的那个模板手势序列所表示的手势结果作为动态手势识别的结果[28]。这种方法可以分为两类：直接匹配法和基于动态DTW（动态时间规整）算法的匹配算法。DTW算法经过实践的检测，该方法能够达到比较高的识别率，但是由于计算量比较大，所以识别所耗的时间比较长。

2.基于统计的方法：这种算法是目前大多数手势识别技术使用的，其识别精度比较高。其识别原理是将每一种手势定义一种手势归类，然后根据归类对训练样本的特征向量进行统计和分类，在实际中提取手势的特征向量，并根据训练产生的分类结果得出最后手势识别的结果。目前这种方法中使用较多的是HMM（隐马可夫）模型，训练时对每一种手势建立HMM模型，识别的时候从大量训练的样本中选取概率最大的那个作为最后的识别结果。例如美国虚拟计算实验室的J.Schlenzing等人[29]在实验室中使用了基于HMM模型的手势识别，对于每个手势，他们分别定义了一个HMM，该研究表明HMM对于动态手势识别有着比较好的性能和比较高的识别率，但同时由于在这种方法中，对于训练的依赖性比较大，因此如果要加入新的手势，都要设计与之对应的HMM模型，并需要从大量的训练数据中得出参数，其可扩展性比较差。

3.基于数据分类的方法：这种方法是基于某种学习准则，并循环地进行学习，具备自组织和自学习的能力，能有效的避免一些识别错误的情况。其中，比较典型的方法是基于BP神经网络（Error Back Propagation Neural Network）的技术，该技术是一种能向着满足给定的输入输出关系方向进行自组织的神经网络[30]，在输出层上的实际输出与给定的输入不一致的时候，用下降法修正各层之间的结合强度，直到最终满足给定的输入输出关系为止。这种方法最大的优势就是容错性能强，不易受噪声干扰，缺点是计算量比较大，识别过程比较耗时。

4.基于语法的方法：语法是编译原理中提出的概念， 它是一种高层含义，建立在运动特征的基础之上，从抽象的角度对运动的属性进行描述[31]。基于语法的手势识别方法之一就是基于有限状态机（FSM）的方法，例如林永强等人提出了一种基于姿势序列有限状态机的动作识别框架，对人体肢体点序列进行采样分析，采用正则表达式来表示肢体动作的轨迹，并够造一种有限状态机，实现对17种预定动作的识别，识别准确率在94%以上[32]。基于FSM的手势识别方法也存在一些局限：第一、状态定义的方法决定了这种手势识别方法没有用到样本来进行训练，因此这种手势识别方法在准确率上可信度不高；第二、基于有限状态机的方法是一种相对于状态转移的方法，它对于状态要求比较高，即在特定的状态才会对手势进行识别，如果状态不正确就会导致错误的识别结果，这导致了这种方法的鲁棒性不高。

本文研究了基于FSM的手势识别方法和基于DTW算法的手势识别方法，第4章详细讲解。

## 本章小结

本章主要说明了本文中用到的一些关键技术，首先简单介绍了Kinect设备、其中获取深度图像的原理以及Kinect设备自身的一些局限；然后介绍了手势识别的相关技术，从手部特征提取和动态手势识别的方法来进行论述，通过对比，选择适合本文的一些方法，为后续章节做铺垫。

# 手势图像数据的获取和处理

## 基于Kinect的需求分析

Kinect设备很方便的提供了深度图像、彩色图像、红外图像以及人体骨骼数据，这些数据使得开发人员对于人体骨架的25个关节点的坐标信息在摄像机坐标系中的获取和追踪变得十分方便。在微软的Kinect for Windows SDK v2中提供的接口函数中，人的手掌信息是由人体骨骼在手掌区域的3个关节点（手掌心点、手指尖点、手腕点）来确定的，然而这些信息并不能准确的提供手掌和手指的轮廓信息以及指尖点信息，因此本文中对于手势轨迹获取，是使用Kinect获取骨骼坐标点以及深度图像的相关数据，然后进行了一些必要的处理，用以更加方便和准确的获取手掌心点的坐标信息。

本文基于Kinect的骨骼图像和深度图像获取手势信息的流程如图 3‑1所示：首先通过骨骼信息中掌心点在Kinect摄像头三维空间坐标系中的位置，确定其在深度图像中的位置，基于这些信息利用深度阈值法获取手掌区域，然后对获取到的数据再一次结合深度图像进行手掌轮廓的提取，最后基于获取到的轮廓信息进一步分析手掌中心点和手指尖点。

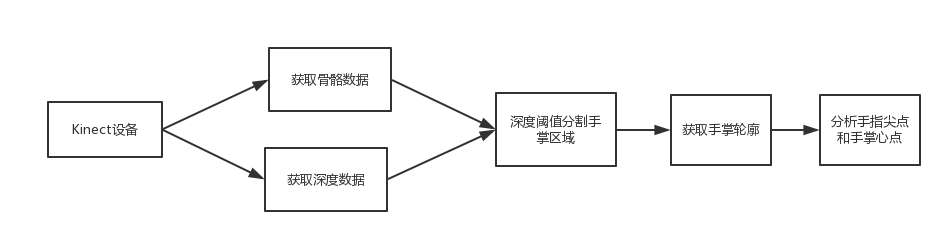


图 3‑1 Kinet图像模块的流程图

## 图像数据的获取

### 利用Kinect获取人体骨骼数据

与2011年微软推出的Kinect 1代相比，Kinect2在数据稳定性上有着非常显著的提升，其中一点就是Kinect对于骨骼数据的获取。Kinect可以在三维空间中实时地追踪人体骨骼的25个关节点，并获取这25个关节点的三维坐标值，如图 3‑2所示。骨骼坐空间坐标系是右手坐标系，X 正方向朝右，Y 轴正方向朝上，X轴数据范围为－2.2~2.2 m，总共范围为4.2 m，Y 轴范围为－1.6~1.6 m，Z 轴范围为0~4 m。

：

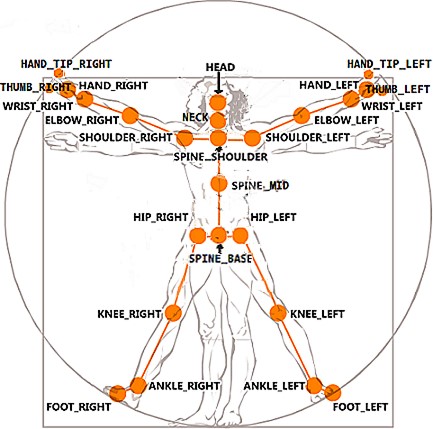


图 3‑2 Kinect2 中获取并追踪的25个关节点信息

并对这些人体的关节点进行追踪。在微软提供的Kinect for Windows SDK v2中，关节点信息是存储在一个叫Jonit的结构体中的，如图 3‑2所示：

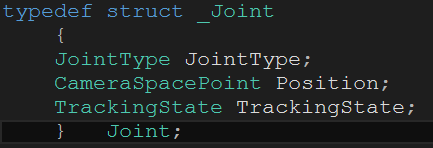


图 3‑3 Kinect SDK中定义的关节点结构体

在这个结构体中存储了三个信息：关节点类型（JointType）、关节点的摄像头空间坐标点（Position）和追踪状态（TrackingState）；另外，Kinect可以最多一次性跟踪6个人体的数据，存放在一个叫IBody的类对象的数组中，从每一个IBody对象出发可以获取到与每个人体相关的骨骼关节点。

在论文中，提取人体骨骼关节点的主要步骤是：

第一步：初始化Kinect传感器（IkinectSensor）结构体指针，打开Kinect设备，做好一系列的准备工作；

第二步：通过获取到的IkinectSensor结构体指针循环获取身体帧数据，提取出身体帧数据源（IBodyFrameSource），并打开身体帧读取器（IBodyReader）；

第三步：根据获取的身体帧读取器获取到各个关节点信息，存放到一个Joint结构体数组中，获取人体的25个关节点；

第四步：读取帧数据并重复第二步和第三步，直到最后程序关闭，关闭Kinect传感器，并释放相关指针所占据的内存空间。

### 利用Kinect获取深度图像数据

Kinect设备的另外一个特性就是可以很方便的物体在三维空间中的深度图像：相较于Kinect1代提供的320\*240的深度图像而言，Kinect2能提供512 \* 424范围的深度图像。由于深度图像的坐标系是一个x轴正方向向右，y轴正方向向下的坐标系，和OpenCV的坐标系统能够十分契合，可以很便捷地在OpenCV将获取到的深度图像数据用窗口显示出来，以便下一步的研究，因此本论文中的设计是将Kinect获取的骨骼数据和三维空间中的深度图数据结合起来进行处理的。

在Kinect for Windwos SDK v2提供的API函数中，获取Kinect中获取深度图像数据和获取人体骨骼数据是相似的，所不同的是，深度图像获取的数据是存储在一个大小为512\*424的一维2字节（16位）数组中，在这个数组中，每一个16位数据代表深度图中对应位置与摄像头之间的距离，那么这个数组中下标参数与对应下标的16位数据值关系是：

设获取的深度图数据用一个数组UINT16[512\*424] depthArray来表示，x表示深度图中的x坐标值，y表示深度图中的y坐标值，那么数组的下标参数为index = y\*424 + x，且depthArray[index]中存储的就是深度图中坐标(x, y)处与Kinect摄像头之间的距离。

获取深度图数据的步骤如下：

第一步：初始化Kinect传感器（IkinectSensor）结构体指针，打开Kinect设备，做好一系列的准备工作；

第二步：通过获取到的IkinectSensor结构体指针循环获取深度帧数据，提取出深度帧数据源（IDepthFrameSource），并打开深度帧读取器（IDepthReader）；

第三步：根据获取的身体帧读取器获取到各个关节点信息，存放到一个512\*424大小的一维2字节（16位）数组中，获取当前时间内的深度数据帧；

第四步：读取帧数据并重复第二步和第三步，直到最后程序关闭，关闭Kinect传感器，并释放相关指针所占据的内存空间。

## 图像获取的模块设计

本论文研究的是基于视觉图像的手势识别系统，图像获取是基于Kinect的摄像头，在搭配Kinect for Windows SDK v2的基础上，对Kinect摄像头中人体骨骼帧数据和深度帧数据进行了获取，将获取到的两种数据结合起来，计算出稳定的手掌信息，并对获取的手掌信息进行实时的跟踪。因此比较重要的一点就是图像获取的模块设计了。在本文中，获取图像的流程是基于Kinect SDK的提供的接口函数，对于接口函数中提供的多种数据源进行了整合，主要获取关键的数据，对数据进行了封装。

## 图像数据的处理

### 数据封装

通过微软提供的Kinect SDK能十分方便的获取到人体骨骼数据和深度图数据，在此基础上，本文对获取到的数据进行了封装，以方便后面进一步的研究；首先，将获取到的深度图数据存储到一个二维数组中去，再获取到人体骨骼流数据，提取手掌心点的坐标值，然后将掌心点坐标转换到深度图的坐标系中去，将两者结合起来考虑，这样做的目的是使得z方向上的数据值更加准确。

从程序设计的角度来讲，本文中设计了两个主要的类来存储从Kinect获取到的三维坐标点：HandPoint类和Hand类，HandPoint类；Hand类保存了与手掌相关的信息：（1）手区域中一些关键点的坐标值，这些点有包含手区域矩形的左上角点和右下角点信息、手掌心点、手腕点、指尖点；（2）手掌状态：从Kinect可以获取到手的五种状态，其中包括手掌张开（HandState\_Open）、手掌闭合（HandState\_Close，即握拳）、手指套索状态（HandState\_Lasso，即伸直食指和中指的状态）、未知状态（HandState\_Unknown）、未跟踪（HandState\_NotTracked）（3）手掌轮廓点集合：保存了从手掌区域提取出来的手掌轮廓上的点。

### 手掌模块的提取

对于Kinect获取到的深度数据和骨骼数据，要将手掌区域提取出来才能进一步进行手势识别，因此第一步就是提取手掌区域：由于Kinect对二维图像中手掌区域有优势的地方就是它可以提供二维图像无法提供的深度信息，且二维摄像头往往需要双目摄像头才能提供三维空间上的深度信息，而Kinect的摄像头本身就能获取红外信息，所以Kinect的摄像头能够非常便利的获取骨骼关节点的三维坐标信息，并且基于此获取到的数据是不受光照条件影响的，亦即在黑暗条件下也能准确的获取这些骨骼关节的深度信息，因而在第一步中获取手掌区域要利用Kinect提供的深度图数据，使用深度数据阈值方法提取出手掌区域，但是它也存在局限，其中比较重要的一点就是这样获取手掌区域只能利用Kinect提供的SDK来获取，而且目前Kinect在Linux和Mac OS平台上没有提供有效的驱动程序，所以平台会被限制在Windows操作系统上。

那么对于Kinect中获取手掌模块的步骤要分为三部分：获取手掌区域、获取手掌轮廓、获取手掌心点。

#### 获取手掌区域

获取手掌区域的方法主要是深度阈值法：根据骨骼点能准确确定掌心点所在位置的三维坐标点，假设该点的位置是Pcenter(x, y, z)，那么根据x和y的值可以在Kinect获取的424\*512分辨率的深度图数据depth[424\*512]数组中找到相应的点，这个转换公式(3.1)如下：

(3‑1)

假设设定的阈值为k，那么选取深度图数据中深度值大小在范围[HandDepth – k, HandDepth + k]之间，则认为深度图数据中数值在这个范围内的深度值是属于手掌区域的，这是因为手掌区域的数据往往在深度值上和掌心点比较接近，而手掌本身往往是处于身体其他部位的前方，和手掌不在同一深度范围内，因此可以用这个深度平面内的数值来反馈手掌区域。其算法描述如下：

（1）定义一个布尔类型的二维数组HandArea[424][512]，初始化并全部赋值为FALSE；

（2） 获取Kinect中骨骼帧数据中的手掌心点HandCenter点，转换到深度图坐标系中去，获取在深度图坐标系中HandCenter点的深度值CenterDepth，即令CenterDepth = depth[]；

（3）设置两个下标值i和j，遍历整个深度图数据的数组，取出depth[j \* 424 + i]的值，赋值给nDepth, 即令nDepth = depth[j \* 424 + i], 判断nDepth是否在

集合内，若nDepth的值在该集合中，则令HandArea[i][j] = TRUE, 否则令HandArea[i][j]为FALSE。

这样二维数组HandArea中值为TRUE与深度图对应的位置就是提取出来的手的区域。

#### 获取手掌轮廓

从前一步中提取出来手掌区域的二维数组后，下一步就是要根据提取到的手掌区域数据进行手掌轮廓提取，提取手掌的思路是根据轮廓点附近的8个点的特征值变化，判别在手掌区域中哪一些点事属于手掌中的轮廓点，哪些是属于手掌中的非轮廓点。

根据上面获取手掌区域的二维布尔矩阵HandArea可知，矩阵中的点，可以分为三大类：第一类就是手掌区域内的点，第二类是在手掌轮廓上的点， 第三类是在手掌区域外的点。这三类点分别有一下特征如下：

1. 若HandArea[i][j] = FALSE，则深度数据图中坐标点(i, j)对应的点在手掌区域外部；
2. 若HandArea[i][j] = TRUE, 且在坐标点(i, j)周围的8个点(i - 1, j - 1), (i, j - 1), (i + 1, j - 1), (i - 1, j), (i + 1, j), (i - 1, j + 1), (i, j + 1), (i + 1, j + 1)中，存在一个点(x, y ) ∈{(x,y)|, 且(x,y)≠(i,j)}，使得HandArea[x][y] = FALSE，则深度数据图中坐标点(i, j)对应的点在手掌轮廓上；
3. 若HandArea[i][j] = TRUE, 且在坐标点(i, j)周围的8个点(i - 1, j - 1), (i, j - 1), (i + 1, j - 1), (i - 1, j), (i + 1, j), (i - 1, j + 1), (i, j + 1), (i + 1, j + 1)中，对于所有的点(x,y) ∈{(x,y)|, 且(x,y)≠(i,j)}，使得HandArea[x][y] = TRUE，则深度数据图中坐标点(i, j)对应的点在手掌区域内；

算法实现步骤如下：

1. 初始化一个布尔矩阵HandOutline, 其大小为424\*512，将HandArea矩阵中的所有数据都复制到HandOutline中去；
2. 设定遍历下标i和j，i的值从1到424变化，j的值从1到512变化，获取HandArea[i][j]的值；
3. 声明一个字节的变量value，并初始化为0x00(二进制数值为00000000)，若HandArea[i][j] = FALSE，保持value的值为0x00不变；
4. 若HandArea[i][j] = TRUE，依次获取HandArea[i-1][j-1]、HandArea[i-1][j]、HandArea[i-1][j+1]、HandArea[i][j-1]、HandArea[i][j+1]、HandArea[i+1][j-1]、HandArea[i+1][j]、HandArea[i+1][j+1]得值，若获取的值为TRUE，则计算位运算 value = value << 1 | 0x01，否则令计算位运算 value = value << 1 | 0x00；如此经过8次运算，就可以将坐标点为(i,j)位置的左上（↖）、上（↑）、右上（↗）、左（←）、右（→）、左下（↙）、下（↓）、右下（↘）这8个方向的临近点所存储的布尔值转换到value的第7~0位上，并且该位置上位1表示对应方向的临近点在HandArea中为TRUE，0表示对应方向上的临近点在HandArea中为FALSE；
5. 根据第四步计算的这八个方向上临近点在HandArea中的布尔值，结合上文介绍的算法，比较容易知道，当value的值为0xFF（对应二进制数值为11111111）的时候，表示坐标(i, j)在深度图数据中对应的点是在手掌内部区域，则将HandOutline[i][j]设定为FALSE，而当value的第7~0位上存在任意一位位1的情况的时候，表示坐标(i, j)在深度图数据中对应的点是在手掌的轮廓上，则将HandOutline[i][j]设定为TRUE。

根据以上算法，可以获取到手掌轮廓上的点信息，并将这些信息存储到一个二维布尔矩阵，同时用一个数组points来记录这些轮廓上的点信息。

#### 获取手掌心点

在获取了手掌轮廓之后，接下来比较关键的一步就是获取手掌心点，。本文采取的手掌心点的检测方法是根据手掌的几何特征，计算轮廓上的点到手中心点的中心点，根据几何知识可以知道，假设平面上有n个点：P­1(x1, y1), P2(x2, y2), … , Pn(xn, yn)，这些点的中心坐标设为C(xc, yc)，则中心点计算算法如公式(3‑2)所示

(‑2)

在上一步中，将获取到的轮廓点的数据都放在一个数组中了，用points来表示，即points[i].x和points[i].y分别表示公式(3‑2)中xi和yi，但是在处理中心点之前应该剔除四只手指对于获取中心点的干扰，以保障获取到的手掌心点的坐标点不会偏上，因此对于手掌轮廓分析，可以大致剔除四只手指的部分，其步骤如下：

1. 设置遍历下标i和j，在轮廓点集合中计算x方向和y方向上的最大值和最小值，设获取到的结果分别是xmin , ymin , xmax , ymax 那么对于手掌轮廓所在的区域可以用一个矩形区域包含之，这个矩形的左上角坐标是(xmin, ymin)，右下角坐标是(xmax, ymax)；
2. 双重循环：在下标i从ymin遍历到ymax的情况下，将下标j从xmin遍历到xmax，j遍历的一次称为一趟，在这一趟的遍历过程中，记录取得HandArea[i][j]中布尔值从TRUE突变到FALSE的次数，那么按照这种遍历方式取得的变化次数依次会是2、6、8、10、4、2，如图 3‑4 (a) 所示:

|  |  |
| --- | --- |
|  |  |
| (**错误!文档中没有指定样式的文字。**‑a) 布尔值变化次数 | (**错误!文档中没有指定样式的文字。**‑b) 获取中心点计算区域 |

图 3‑4 从上到下遍历获取手掌轮廓点剔除四只手指示意图

那么取得第一次出现变化次数为4的情况，就可以确定一个y轴方向上的约束值Ystrict，使得Ystrict之上的图像大略是四只手指所在的位置，Ystrict之下的图像就是需要考虑的，可以用来计算中心点的区域，如图 3‑4(b)中蓝色框所标识的区域

进行上面的处理之后，将蓝色区域中表示的位置在HandOutline矩阵中找到对应的坐标数据，根据公式(3‑2)就可以计算出手掌的中心点，从而进行下一步的研究工作。

## 本章小结

本章首先给出了手势识别系统的需求分析，详细说明了Kinect提供的几种数据，以及课题需要的内容。其次介绍了从图像数据的获取这个方面说明了获取整个手掌图像模块数据的实现过程，并介绍了人体骨骼数据和深度数据结合的模块设计。接下来介绍了本文中图像数据的处理。最后对图像模块的显示进行详细说明分析。

首先详细叙述了图像模块的程序设计过程，该模块作为本系统的核心功能之一，能够获取用户输入的手势信息，本章介绍图像模块方法的设计思想，为后续实现章节提供指导。

# 手势识别模块的实现

在获取手掌数据后，至关重要的部分就是手势识别模块的实现了。本文在基于Kinect获取的图像基础之上，对动态手势识别进行了实现，搭建了手势库。手势识别模块主要分为手势轨迹处理、DTW算法实现、手势库搭建三个步骤。本章主要从算法实现角度来介绍手势识别模块的实现。

## 手势轨迹的获取

手势轨迹是在一段时间内手掌坐标信息的集合，它有两个明显的特征：一、时效性，手势的轨迹是在一定时间内确定的，所以手势轨迹必定有个开始时段和结束时段；二、关联性，手势的轨迹是坐标点的集合，集合中点不是孤立的，往往点与点之间有着关联。

### 坐标转换

从获取到的帧数据手掌心点信息中，可以比较精准地获取到掌心点在Kinect深度图像这个二维平面上的坐标信息。考虑到坐标系的本身的约束，获取到的坐标点往往只是在当前坐标系中才有实际意义，因此获取手掌心点坐标点后还要对这些数据进行坐标系转化，以提供给手势系统的坐标系使用。对于坐标系转换，有两种可行的方法：

第一种方法是等比例映射法：对于平面上的两个坐标系Oxy和Ox’y’，用点(x, y)以及点(x’,y’)分别来表示这个两个坐标系中两个可以互相转化的点，设由x转换到x’的比例系数是k1，由y转换到y’的比例系数是k2，那么这两个点之间的转换算法如公式(4‑1)所示：

(‑1)

对于一般的可视化的坐标，都是以屏幕向右的方向为x轴的正方向，屏幕向下的放下为y轴的正方向，坐标值都为正值，可视的范围是整个屏幕或者整个窗口的大小，所以可视点的坐标值会规范到一个矩形之中。假设待转换的两个可视化的坐标系的矩形分别为frame1和frame2，那么如果由frame1表示的坐标系映射到frame2表示的坐标系，k1 = frame2.width / frame1.width，k2 = frame2.height / frame1.height；

第二种方法是变化量转化法：对于有时效性的坐标系Oxy和Ox’y’，设某一时刻t，两个坐标系中坐标点分别为(x(t), y(t))，(x’(t), y’(t))；在t+1时刻，(x(t), y(t))变化为(x(t+1), y(t+1))，t ~ t+1时间内，Oxy坐标系中在x方向上的变化量为dx = x(t+1)-x(t)，dy = y(t+1)-y(t)，对于Oxy坐标系中的变化量，指定一个坐标值变化的范围，设该范围是一个矩形区域frame1，而假设Ox’y’坐标系中坐标值变化的矩形区域范围是frame2，那么可以将Oxy坐标系中坐标值变化量等比例映射到Ox’y’坐标系中去，即:

(‑2)

如此计算出在Ox’y’坐标系中x方向和y方向上的变化量，从而计算t+1时刻在Ox’y’坐标系中，由在坐标系Oxy上的点(x(t+1),y(t+1))转换而来的坐标值是(x’(t+1), y’(t+1))，其中x’(t+1) = x’(t) + dx，y’(t+1) = y’(t) + dy；

其中，第一种方法转换的特点是坐标点转换都是按等比例转换的，所以每一个时刻，两个坐标系中获取到的轨迹点是相似的，而且只要原坐标系中坐标点的数值没有发生突变，转换后的坐标点数值就不会发生突变，因而只要元坐标点足够精确，经过转换后的坐标点也是精确的；第二种方法中坐标点数值是基于坐标点变化量而言的，本身转换后的坐标点对于原坐标点的依赖性并不是太强，对于获取手势轨迹而言，往往会更加稳定，本文采用第二种方法对坐标点进行转换。

### 数据封装

从Kinect中获取的帧数据，是按照时序一帧一帧获取的，所以获取到的数据和时间是有关系的。在微软提供的Win32API函数中，调用GetTickCount这个函数可以获取系统时间，并且精确到毫秒。本文中，申明了一个结构体HandArgs，用来存当前时刻获取到手掌信息的各种参数：

typedef struct \_HandArgs

{

int x, y, z;

int vx,vy,vz;

int dx, dy,dz;

…

}HandArges

其中x,y,z的值是保存的当前时刻手掌心点的在各个方向上的坐标值，vx，xy，vz用来计算当前时刻和上一时刻掌心点在各个方向上的变化率，dx，dy，dz用来保存当前时刻和上一时刻掌心点在各个方向上的变化量。这些数据是通过以下步骤获取到的：

1. 定义两个时间参数currentTime和lastTime，分别记录当前时刻和上一时刻的毫秒数，并均初始化为0；
2. 从Kinect传感器中获取帧数据，对帧数据进行处理后，获取到手掌心点的坐标信息，将坐标点信息保存到HandArgs的x、y、z变量中并将当前系统时间的毫秒计数保存在currentTime中，。
3. 在判断lastTime非零的情况下，计算获取到两帧数据之间的时间差timeInterval = currentTime – lastTime，并获取HandArgs中保存的前一帧的x，y，z值，计算两者之间的差值，保存到dx，dy，dz，同时计算dx / timeInterval，dy / timeInterval，dz / timeInterval分别保存到vx，vy，vz中。

## 基于FSM的手势识别

不同的手势有不同的数值特征，包括手掌在某一特定时间内的坐标点的变化量和变化速率，双手之间的距离和双手关于y轴方向上形成的角度值等，根据这些基本特征可以利用数值算法对手势轨迹进行分类。

### 手掌的状态定义

在某一单独的特定时间、特定条件下，手势会呈现出不同的状态，本文中，定义了17种手势识别的状态：

enum GestureType

{

TYPE\_OPEN,

TYPE\_OPEN\_RETAIN,

TYPE\_OPEN\_RETAIN\_MOVE,

TYPE\_OPEN\_CLICK,

TYPE\_OPEN\_MOVE,

TYPE\_OPEN\_SHIFT,

TYPE\_HOLD,

TYPE\_HOLD\_MOVE,

TYPE\_UNKNOWN,

TYPE\_GRAB,

TYPE\_RELEASE,

TYPE\_DRAG,

TYPE\_ZOOM,

TYPE\_ZOOM\_IN,

TYPE\_ZOOM\_OUT,

TYPE\_ROTATE\_CW,

TYPE\_ROTATE\_CCW

};

这17中状态分别是：

1. TYPE\_OPEN：从Kinect2中能够获取到的手掌张开的状态；
2. TYPE\_OPEN\_RETAIN：手掌张开后，停留在屏幕上一段时间不动的状态；
3. TYPE\_OPEN\_RETAIN\_MOVE：手掌在屏幕上停留后在一段时间内移动的状态；
4. TYPE\_OPEN\_CLICK：手掌在屏幕上停留后在z方向上快速向前移动的状态，触发点击手势；
5. TYPE\_OPEN\_MOVE：手掌张开状态后移动的状态；
6. TYPE\_OPEN\_SHIFT：手掌张开后，在平面坐标系上快速移动的状态；
7. TYPE\_HOLD：手掌握拳状态，可以从Kinect2中读取到的状态；
8. TYPE\_HOLD\_MOVE：手掌握拳后在三维空间中移动的状态；
9. TYPE\_UNKNOWN：手掌未知状态；
10. TYPE\_GRAB：手掌从张开手掌到握拳变换的中间状态；
11. TYPE\_RELEASE：手掌从握拳到张开手掌变换的中间状态；
12. TYPE\_DRAG：手掌握拳后在平面坐标系上快速移动的状态；
13. TYPE\_ZOOM：手掌开始识别缩放旋转手势的状态；
14. TYPE\_ZOOM\_IN：双手放大手势状态
15. TYPE\_ZOOM\_OUT：双手缩小手势状态
16. TYPE\_ROTATE\_CW：双手顺时针旋转状态
17. TYPE\_ROTATE\_CCW：双手逆时针旋转状态

### 手势状态转换

有限状态机用于对系统的动态行为的建模，一般用状态图来可视化表示。状态机是一个可以确定状态规则以及在特定条件下实现状态转移的设计模式[33]，可以确定的是，手掌数据一定是处于一种既定的状态之下，在给定的外在条件下，会发生状态转移。在本文中，实现了手势状态转换的状态机，这样手势状态构成了一个有向图，如图 4‑1所示，手势状态通过条件的变化就会发生改变。在状态发生改变的一些时机就会触发手势事件。



图 4‑1 手势状态转换图

### 数值特征

对于单手的移动可以确定存在着三种类型：移动、快速移动、停留。根据获取到手掌参数，可以确定手掌在坐标系中的移动速度，移动变化量，再根据前后两帧之间的数值变化，可以对这三种单手移动类型进行判别。假设对于手势的前后两帧f1和f2，获取到的参数是三维空间上的变化量dx、dy、dz和前后两帧的速度值vx、vy、vz，这几种手势移动的类型数值特征如下：

手掌移动：手掌心点坐标在三维空间中发生变化，具体表现为|dx| > 1单位或者|dy| > 1单位或者|dz| > 1单位；

手掌快速移动：手掌心点坐标在三维空间中速度出现快速变化，具体表现为|vx| > 60 单位/ms或者|vx| > 60 单位/ms或者|vy| > 60 单位/ms；

手掌停留：手掌心点坐标在三维空间中在一个数值范围内保持不变，具体表现为|dx| ≤ 1单位或者|dy| ≤1单位或者|dz| ≤ 1单位；

## DTW算法核心技术

一段具有时序的数据往往是指将同一统计指标的数值按其发生的时间先后顺序排列而成的数列，对于大部分的学科而言，时间序列是数据处理中一种十分常见的表达方式，比如一段视频数据可以看成是图片数据的时间序列、一段语音数据可以看成音节数据的时间序列等。在时间序列中，需要比较相似性的两段时间序列的长度可能并不相等，为了解决时间序列长度不同的模板匹配问题，DTW算法在上世纪70年代左右被提出来，并最早用于处理语音方面的识别分类问题。

DTW（Dynamic Time Wrapping，动态时间规整）算法是语音识别中的一种经典的算法[34]，该算法是基于动态规划（Dynamic programming，DP）的算法思想实现的。它能够简单灵活地实现模板匹配，能解决很多离散时间序列匹配问题。本文中获取到手势轨迹就是基于时间序列，和语音识别的过程是十分相似的，因此可以利用该算法实现对既定手势的匹配，达到比较精准地识别结果。

### DTW算法原理

假设有两个时间序列M和N，他们对应的长度（即帧数长度）分别为m和n，其中M是参考模板，它对应的序列设为{M1, M2, … ,Mi , … , Mm}，N是待匹配的模板，它对应的序列设为{N1, N2, … ,Nj, … , Nn}，规定一个距离函数Dis(M, N)，计算时间序列M和时间序列N之间的距离，由此可知，如果两个时间序列之间的距离越小，它们之间的相似度就越高。为了获取两个时间序列的距离，假设i和j分别表示M和N中任意选择的两个对应帧的帧号，则规定一个计算法则d，用d(Mi , Nj)表示这两帧之间的距离，在一般的DTW算法中，距离计算通常采用的距离是欧式距离。那么对于不同的m值和n值，有以下两种情况：

(1) 若m = n，说明序列M和序列N中帧数是一致的，这样可以直接计算两个序列一一对应的距离，即将M1与N1帧、M2和N2帧、… 、Mm和Nm帧对应起来，依次计算d(M1 , N1)，d(M2 , N2)，… ，d(Mm , Nn)的值，然后再求和，得到两个序列的距离值，如公式(4‑3)所示

(‑3)

(2) 若m≠n，说明序列M和序列N中的帧数是不一致的，对于这种情况，有两种可行的办法进行处理：

1. 采用线性缩放对齐的方法：将短的序列线性放大到和长的序列长度相同，然后再进行(1)中的距离计算，或者将长的序列线性缩短到和短序列一样的长度再进行(1)中的距离计算。

下面对此方法进行说明：假设m > n，也即模板序列M中的帧数长度大于待匹配序列N的帧数长度，这种情况下，可以将待匹配序列N用线性放大的方式，扩大成一个帧数长度为m的序列N’{N’(1), … , N’(n) }，然后再和M模板序列进行匹配，计算得到两者的距离。反之，m < n 的情况也可以通过这种方法获取到最终结果。

这种方法，虽然可以解决不同长度的时间序列之间的匹配问题，但是对于不同的时间序列，在各段的不同情况下会产生或长或短的变化，即各个序列中元素的伸缩程度是不同的，如果采用这种方法，就忽略了这种不同伸缩程度带来的变化，而统一地使用线性收缩的方式使得两个时间序列的长度变得相同，势必会造成识别的结果精度不准，为了更佳的识别效果，一般会采用动态规划的方法；

2．采用动态规划的方法：为了将这两个序列对齐，动态规划算法会构造一个m×n的矩阵网格，矩阵中元素(i, j)表示序列M中第i帧和序列N中第j帧的距离，用d(Mi, Nj)来表示，这个距离越小，表示这两个序列中对应的这两帧之间的相似度越高，也可以理解为失真度越低。

由于这两个序列是有时间顺序的，也就是说整个匹配的过程，必定是要寻找一条路径，经过矩阵网格的(1,1)到达矩阵网格的(m,n)。直观来看，如果把序列M的帧号1~m放到一个二维坐标系的横轴，做为横轴的标号，把N的帧号1~n放到这个坐标系的纵轴，作为纵轴的标号，那么动态规划的方法归结为从这个坐标系中找到一条路径，使得这条路径能够从从左下角到达右上角，并且这条路径经过的元素值之和最小。

为了方便描述这条路径，假设这条路径经过的点依次是(X1,Y1)，… ，(Xi,Yj)，… (Xm, Yn)，其中(X1,Y1) = (1,1)，(Xn, Yn) = (m, n)，定义一个二维函数φ(i, j)表示到达点(Xi, Yj)时，与邻近点元素积累的距离之和的最小值，那么满足公式(4‑4)

(‑4)

这个公式体现了局部最优路径的选择，那么从匹配起点就能沿着局部最优的路径反向找到最初的匹配点，获的全局最优路径。这种逐点求取帧匹配距离的算法就是DTW算法。

### DTW算法的约束条件

根据上面介绍的DTW算法基本原理，对于满足可以用DTW算法匹配的两段时间序列M(M1, M2, … , Mi, ... , Mm)和N(N1, N2, … , Nj, ... , Nn)，设最后找到的满足两者之间距离和最小的路径L，这条路径一共经过s个点且Li表示这条路径上面的第i个点（1≤i≤s），那么它将满足三个约束条件：

1. 边界条件唯一性：这条路径的起点必定是(1, 1)，终点必定是(m, n)，即L1=(1,1)，L2=(m, n)；
2. 连续性：对于路径L，如果第k（1≤k＜s）个点Lk为(i, j)，那么在路径L上第k+1个点必定是(i+1, j)、(i, j+1)、(i+1,j+1)之中的一个；
3. 单调性：根据条件(2)，路径L上第k（1≤k＜s）个点(ik, jk)和第k+1个点(ik+1,jk+1)，必定满足：ik≤ik+1且jk≤jk+1

这三个约束条件保证了在上面构造的m×n的矩阵网格中每一个点的路径只有三个方向，如图 4‑2所示

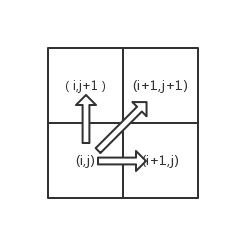


图 4‑2 DTW矩阵网格最优路径的三个方向

也就是说，若最优路径通过了网格中的点(i, j)，那么在最优路径的下一个网格点必定只有三种情况：(i, j+1),(i+1, j),(i+1, j+1)

### DTW算法实例

下面用一个实例来说明DTW算法的计算过程，假设有模板时间序列M{3,5,2,9,10}，有两个待匹配的时间序列N1{4, 13,12,12,6,4}，N2{1,6,3,12}，它们的帧长度分别为m=5、n1=6、n2=4，它们的时序图如图 4‑3所示，

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a) M序列时序图 | (b) N1序列时序图 | (c) N2序列时序图 |

图 4‑3 样例时序图

那么计算M序列和N1序列的距离值（相似度）的DTW算法过程如下：

第一步、构造距离矩阵Dis[m][n1]，为方便理解，距离矩阵的右方向是i值的递增方向，上方向是j值的递增方向，Dis[i][j]中计算M中第i帧和N1中第j帧的欧式距离，即两个帧数据值之间的差值的绝对值，那么构造的距离矩阵如图 4‑4所示

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 2 | 5 | 6 |
| 3 | 1 | 4 | 3 | 4 |
| 9 | 7 | 10 | 3 | 2 |
| 9 | 7 | 10 | 3 | 2 |
| 10 | 8 | 11 | 4 | 3 |
| 1 | 1 | 2 | 5 | 6 |

图 4‑4 序列M和序列N1的距离矩阵

第二步、构造累加求和距离矩阵D[m][n1]，D[i][j]表示经过(i, j)是累加的最小距离值，根据公式(4‑4)，计算所得的结果如图 4‑5所示

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 33 | 25 | 26 | 22 | 22 |
| 32 | 24 | 27 | 17 | 16 |
| 29 | 23 | 26 | 14 | 12 |
| 20 | 16 | 19 | 11 | 10 |
| 11 | 9 | 13 | 8 | 11 |
| 1 | 2 | 4 | 9 | 15 |

图 4‑5 序列M和序列N1的累加求和距离矩阵

右上角的数值D[m][n1]=22就是最后所求的序列M和序列N1的之间的距离，根据箭头指示的方向，由DTW算法求得的这条路径是(1,1)→(2,1)→(3,1)→(4,2)→(5,3)→(5,4)→(5,5)→(5,6)。

计算M序列和N2序列的距离值过程和上述算法相同，第一步构造的距离矩阵如图 4‑6所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | 7 | 10 | 3 | 2 |
| 0 | 2 | 1 | 6 | 7 |
| 3 | 1 | 4 | 3 | 4 |
| 2 | 4 | 1 | 8 | 9 |

图 4‑6 序列M和序列N2的距离矩阵

第二步中构造的累加求和距离矩阵如图 4‑7所示，求得两个序列之间的距离为D[m][n2]=9,再根据箭头的方向，算得距离最短的路径是(1,1)→(2,2)→(3,3)→(4,4)→(5,4)。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14 | 12 | 14 | 7 | 9 |
| 5 | 5 | 4 | 10 | 17 |
| 5 | 3 | 7 | 10 | 14 |
| 2 | 6 | 7 | 15 | 24 |

图 4‑7 序列M和序列N2的累加求和距离矩阵

由于Dis{M，N1} > Dis{M，N2}，说明N2与M的距离比N1与M的距离更小，即N2与M更相似。

## 基于DTW算法的手势识别

前文提到，手势轨迹是一个时间序列，它必定是一个有起点和终点，一段时间内有序的手掌坐标点数据的集合。因此，借鉴语音识别的案例，DTW算法也可以用来进行手势识别。

由于DTW算法是一种基于模板匹配的算法，所以需要提供样本模板数据，建立模板手势库。在已规定好的轨迹特征模板手势库的情况，再用DTW算法对手势和手势库中的模板数据进行匹配，才能有效地识别出特定的手势。

### 建立模板手势库

对于手势库的建立，首先第一点就是要设计几种已经可以实际使用的手势，并把手势的特征轨迹点记录下来。考虑到易于理解和易于扩展手势库的使用场景，手势库数据使用xml文件来记录，xml的层次结构图 4‑8所示，即xml的根节点手势库（gestureLib）保存了多个手势(gesture)，手势又是由多条轨迹（stroke）组成，一个轨迹是有多个轨迹坐标点（point）组成，如此就构建了一个保存样本模板的xml数据库文件。

图 4‑8 手势库xml文件的层次结构图

构建xml文件的一个显著的优点就是文件易于理解，易于编辑，可读性高。但是对于大量数据的解析会比较缓慢，且文件格式复杂，如果读取到内存中会占据大量不必要的格式标签信息，因此对于读取到内存中的模板文件信息，定义了一个协议规范，将xml文件转换成二进制的model文件，用以缩减内存存储消耗。

model文件的协议规范：首先定义了一个ModelHeader，用来保存模板文件的基本信息，结构体定义如下：

struct ModelHeader

{

char sign[4];

short ver;

short type;

short width;

short height;

short pointDist;

int count;

};

其中，各个变量的意义是：

sign：模板文件的标志，是一个字符串，用来标识模板文件，默认为“SGR”，是简单手势识别（simple gesture recognizer）的意思；

ver：模板文件的版本号，用来标识模板文件的版本；

type：模板文件的类型；

width：手势轨迹的默认宽度；

height：手势轨迹的默认高度；

pointDist：手势轨迹之间默认的距离差值，用以方便把轨迹稀疏化或者补全离散轨迹的临界阈值；

count：模板文件中已经定义了的手势个数。

然后对于各个手势，定义了一个UTF-8字符来标识手势轨迹的信息，称为手势标识码，比如，用○（UTF-8编码为&#x25CB;）来标识圆圈手势。在程序设计中定义FeatureHeader和Feature结构体与各个手势对应，FeatureHeader结构体中存储了各个手势的基本信息，而Feature结构体中定义了手势的内容，包括手势坐标点的信息。他们的结构体定义如下：

struct FeatureHeader{

char word[4];

int trackCount;

int count;

};

struct Feature{

int trackCount;

string word;

vector<int> feature;

};

其中word表示手势轨迹的标识码；trackCount标识一个手势中包含的轨迹数目，一般而言手势轨迹数目只有一条，但是例如对于与“×”对应的手势轨迹，可以看成需要两条轨迹来确定手势信息，所以需要定义一个trackCount变量，以保证程序可扩展性；FeatureHeader中的count和Feature中的feature.size()是一致的，Feature.feature中存储的是各个手势轨迹的特征点坐标信息，具体内容是，将各个手势轨迹的特征点坐标的x值和y值拆分开来，依次按序保存到Feature.feature这个动态数组中。

定义的model文件格式协议如图 4‑9所示：

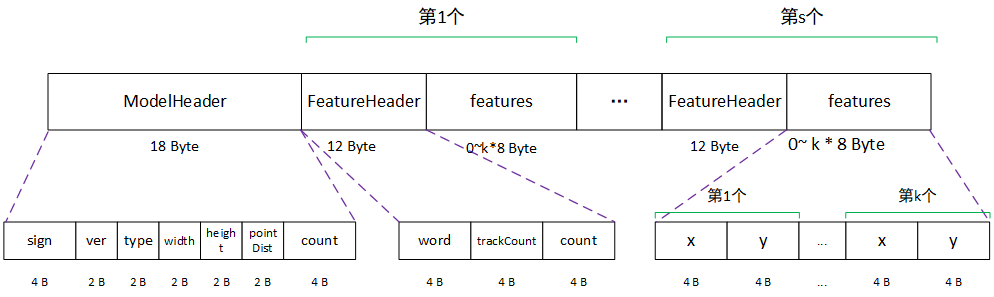


图 4‑9 model文件的数据格式

一个model文件在头部会插入一个ModelHeader，占据18个字节，然后根据手势库xml文件中手势的个数s，依次插入s个FeatureHeader和s个features字段，其中一个FeatureHeader占据12个字节，features所占据的空间由手势确定的特征点坐标个数决定，如果特征点坐标个数是k，那么features所占据的空间就是k \* 8 字节（x数值和y数值会分开来，因此是4B+4B=8B）。

### 模板手势库轨迹处理

xml模板手势库文件中会对手势提取合适的特征坐标点然后记录下来，在组建手势库的过程中，由于xml文件中定义的手势轨迹点只有离散的几个特征点，手势数据往往需要通过轨迹补全算法是离散的点连续化，然后在获取连续轨迹点基础上做后续处理。例如假定存在一个三角形（△）手势，那么它的手势特征坐标点就是三角形的三个顶点。根据这三个顶点就可以补全完整的三角形轨迹，使得轨迹连续化。

#### 手势轨迹连续化

那么手势轨迹特征点的连续化这个问题归结为在相邻两个离散的坐标点之间插入连续线性变化的坐标点，其算法执行过程如下：

1. 对于两个离散点(x1, y1)和(x2, y2)，定义两个变量Δx和Δy，分别计算两个点x方向和y方向上的变化值绝对值，即Δx=|x2-x1|，Δy=|y2-y1|，比较Δx和Δy的大小，取两者中的最大者并减一后的数值，设这个值为Δp，即Δp= max{Δx, Δy} - 1，那么在(x1, y1)和(x2, y2)之间插入的坐标点个数是Δp个，Δp确定了插入点是默认以平面上哪个方向进行推进。
2. 设插入的Δp个坐标点是P{p1, p2, …, pΔp}，对于第i个坐标点pi(xi,yi)，有一个斜率值ki与之对应，且ki的计算如公式(4‑5)所示，

(‑5)

1. ki表示pi(xi, yi)与(x2, y2)之间的斜率，这个值的变化会影响第（i+1）个点pi+1(xi+1, yi+1)的计算。根据计算出的pi，设第i+1个点是p，设p0=(x1, y1)，若Δp=Δx，则这个坐标点迭代的计算如公式(4‑6)所示

(‑6)

而若Δp=Δy，则这个坐标点迭代的计算如公式(4‑7)所示

(‑7)

如此按照上述迭代计算过程，就可以将轨迹上每相邻的两个离散点之间的数据补全了，下面用一个示例来解释这种算法的正确性：

假设有两个离散点A(1,1)和B（10,6），则上式中Δx=9，Δy=5，取Δp=8，即需要在点A和点B之间插入8个连续的点，根据上述计算过程，算得计算所得的这个8个点是(2, 2)、(3, 2)、(4, 3)、(5, 3)、(6, 4)、(7, 4)、(8, 5)、(9, 5)，在坐标系中绘制得到的散点图如图 4‑10所示，看上去似乎并不是一条连续的直线，

图 4‑10 示例数据插入连续点后的散点图

但是在系统屏幕的坐标系中，坐标系变化往往是以整数值变化的，一般不会存在小数的情况，也就是说屏幕坐标系中0到1,1到2的变化是看做连续的，因此上述计算的结果在屏幕坐标系统中是可以看成一条连续的直线的。

#### 手势轨迹稀疏化

对模板数据中离散的轨迹点进行连续化补全，这样取得数据量虽然整体而言比较精确，但是实际上在模式匹配的过程中，并不一定需要连续的坐标点数据量，由于基于DTW算法的模板识别是一种模糊识别算法，只需要计算实际时间序列与模板时间序列之间的相似度就可以了，因此一种比较可行的优化方法就是对获取到的连续轨迹点进行稀疏化，在连续的轨迹点中每隔一段距离后取值，取连续轨迹点中的部分轨迹点。对连续化的轨迹点进行稀疏化，然后存储在模板数据库的有二个好处：

第一、可以保证准确识别手势的同时又能节省模板数据库的存储空间：由于xml模板数据中定义的手势轨迹是过于“稀疏”（或者说过于离散）的，而对这种手势轨迹点进行连续化后的手势轨迹点又是过于“密集”（或者说过于连续）的，因此实际上对连续化后的手势轨迹点再次进行稀疏化，是一个对前两种情况一种折中的选择，在保证可以比较准确识别到手势的同时，又可以在存储这些模板数据时可以节省存储空间。

第二、可以节省手势识别所用的时间：由于DTW算法是一个计算量比较大的算法，因此对于连续化的轨迹坐标点信息，大量的模板数据的计算会占据比较长的算法执行时间，为了提高手势识别的效率，将带有大量模板数据的手势轨迹坐标点进行稀疏化后，带来的手势识别效率的提高还是比较可观的。

对于手势轨迹稀疏化的处理过程如下：

1. 对于某一手势的某一条轨迹P{p1, p2, … , pn}，初始化两个变量p和q，以及一条新的轨迹Q{}，令p=q=p1，将Q清空；
2. 设定一个距离阈值d，用p和q遍历这条轨迹，q向前遍历，每遍历一次则比较一下点p和点q之间的距离，用dis(p, q)保存计算的结果；
3. 若dis(p, q)≥d，则将点p加入到轨迹点集合Q中，然后令p=q，重复步骤(2)，否则q继续遍历，这样直到p=pn为止。

上述处理过程中，稀疏化手势轨迹比较关键的一点就是设置距离阈值d，这个数值的设置会对算法识别效果有影响。有上述过程可知，如果d越大，手势轨迹的稀疏程度就越高，那么会对手势识别的精确度有影响，而如果d越小，手势轨迹的密集度程度就越高，那么手势识别算法执行过程所占据的时间就越长。经测试，在本手势识别系统中d值设置为50时，手势识别取得的效果会最好。

下面是对于手势库中三角形（用“△”标识该手势）手势的进行手势轨迹处理后的结果。整个过程如图图 4‑11所示，在xml模板文件中定义的三角形手势只给了四个点（起点和终点算两个点，它们重合了）如图 4‑11-(a)所示，经过手势轨迹连续化处理后如图 4‑11-(b)所示，最后根据连续化处理后的轨迹点再次进行距离阈值为50的稀疏化后，得到的手势轨迹图如图图 4‑11-(c)所示。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1. xml文件中模板手势轨迹 | 1. 手势轨迹连续化处理 | （c）手势轨迹稀疏化处理 |

图 4‑11 三角形手势轨迹处理过程示意图

### 手势识别的过程

经过建立手势库模板文件，并对手势库中手势轨迹进行处理之后，接下来就是使用通过Kinect获取到手势轨迹，和手势库中的模板数据进行匹配了。由于获取的手势轨迹数据是相对于深度图的坐标系而言的，因此在进行手势匹配之前，首先要对获取到的手势轨迹进行归一化处理。具体步骤如下：

1. 坐标系转换，这一点在前文获取手势轨迹这一小节中已经说明，采取获取到的手势轨迹坐标点在x和y方向上的变化量参数进行坐标转换的方法；
2. 捕获手势轨迹点所在的矩形区域，将矩形区域尽量映射到和模板手势库中的手势矩形区域相近，以便于获得更加精确的识别结果；
3. 由于获取到的手势轨迹可能并不是连续的，因此需要对获取的手势轨迹进行连续化处理，为了节省识别耗时，需要对连续化处理后的手势轨迹进行稀疏化处理，这个处理过程和对模板手势库中的手势轨迹点的处理相同。

接下来就是利用DTW算法进行模板匹配了，手势识别的流程图如图 4‑12所示，



图 4‑12 手势识别的流程图

这个流程中，只要手掌输入被Kinect检测和跟踪到，就会一直获取到手势数据，因此比较重要的就是对于获取手势轨迹的开始时机和结束时机的把握，也就是说必须要确定什么时候开始捕获手势轨迹以及什么时候结束捕获手势轨迹。在前文提到的，系统会根据手掌的几何参数特征对Kinect获取的手掌数据进行状态划分，由于这些状态是比较容易确定的，因此一旦手掌处于移动（HandMove）状态，就开始捕获手势轨迹点，一旦手掌处于握拳（Grab）状态，就结束捕获手势轨迹点。这样就能对比较准确地获取到一段时间内的手势轨迹数据，然后对其进行处理，最后利用DTW算法进行手势匹配和相应的手势识别。

## 手势识别的模块设计

## 本章小结

本章说明了整个手势识别系统的实现过程，首先阐述获取手势轨迹点的过程：说明对于获取到的手掌轨迹坐标点信息的转换和数据封装，接下来从数值特征方面对手势信息进行手势的状态划分，并设计了一套手势状态机，根据状态转换实现一些比较简单的手势识别过程，然后说明了DTW算法的核心技术，并介绍系统在DTW算法上的实现，从模板手势库的建立、模板轨迹处理、手势匹配这几个方面阐述了整个动态手势识别的核心部分。

# 系统框架搭建和测试

第3章和第4章主要从手势图像数据获取和手势识别模块介绍了本文对于手势识别系统的实现，从算法理论和软件开发的角度进行了详细说明，本章主要阐述系统框架搭建搭建和总体设计模式，以及对于手势识别系统的测试和部分实践应用到的场景。

## 系统的实现框架

本文基于Kincet

### 开发环境

### 系统模块设计

## 手势捕获模块测试

## 手势识别模块测试

## 手势识别应用场景

## 本章小结

# 总结与展望

## 总结

## 展望和改进

致谢

**References:**

[1] 冯志全, 蒋彦. 手势识别研究综述. 济南大学学报（自然科学版）, 2013, 4: 336-341

[2] 王松林. 基于 Kinect 的手势识别与机器人控制技术研究:[ 万方数据资源系统, 2014.

[3] Defanti T, Sandin D. Final Report to the National Endowment of the Arts. US NEA R60-34-163, University of Illinois at Chicago Circle, Chicago, Illinois, 1977,

[4] Kim J H, Thang N D, Kim T S. 3-D Hand Motion Tracking and Gesture Recognition Using a Data Glove. In, ed. IEEE International Symposium on Industrial Electronics. 2009. 1013-1018

[5] Weissmann J, Salomon R. Gesture Recognition for Virtual Reality Applications Using Data Gloves and Neural Networks. In, ed. Neural Networks, 1999. IJCNN'99. International Joint Conference on. IEEE, 1999. 2043-2046

[6] Quam D L. Gesture Recognition with a Dataglove. In, ed. Aerospace and Electronics Conference, 1990. NAECON 1990., Proceedings of the IEEE 1990 National. IEEE, 1990. 755-760

[7] Park I, Kim J, Hong K. An Implementation of an FPGA-based Embedded Gesture Recognizer Using a Data Glove. In, ed. Proceedings of the 2nd international conference on Ubiquitous information management and communication. ACM, 2008. 496-500

[8] Argyros A A, Lourakis M I. Binocular Hand Tracking and Reconstruction Based On 2D Shape Matching. In, ed. Pattern Recognition, 2006. ICPR 2006. 18th International Conference on. IEEE, 2006. 207-210

[9] Utsumi A, Miyasato T, Kishino F. Multi-Camera Hand Pose Recognition System Using Skeleton Image. In, ed. Robot and Human Communication, 1995. RO-MAN'95 TOKYO, Proceedings., 4th IEEE International Workshop on. IEEE, 1995. 219-224

[10] 邬铎. 基于双目视觉的手势识别算法研究与实现:[ 硕士学位论文]. 东北大学, 2011.

[11] Segen J, Kumar S. Video-Based Gesture Interface to Interactive Movies. In, ed. Proceedings of the sixth ACM international conference on Multimedia: Technologies for interactive movies. ACM, 1998. 39-42

[12] Malik S, Laszlo J. Visual Touchpad: A Two-Handed Gestural Input Device. In, ed. Proceedings of the 6th international conference on Multimodal interfaces. ACM, 2004. 289-296

[13] 郭康德. 基于视觉的三维指尖检测算法和应用:[ 硕士学位论文]. 浙江大学计算机学院 浙江大学, 2010.

[14] 谭同德, 郭志敏. 基于双目视觉的人手定位与手势识别系统研究. 计算机工程与设计, 2012, 1: 259-264

[15] 何云龙. 基于双目立体视觉的手势识别技术:[ 硕士学位论文]. 华中科技大学, 2013.

[16] Ren Z, Yuan J, Meng J et al. Robust Part-Based Hand Gesture Recognition Using Kinect Sensor. Ieee T Multimedia, 2013, 5: 1110-1120

[17] Biswas K K, Basu S K. Gesture Recognition Using Microsoft Kinect®. In, ed. Automation, Robotics and Applications (ICARA), 2011 5th International Conference on. IEEE, 2011. 100-103

[18] Kulshreshth A, Zorn C, Laviola J J. Poster: Real-Time Markerless Kinect Based Finger Tracking and Hand Gesture Recognition for HCI. In, ed. 3D User Interfaces (3DUI), 2013 IEEE Symposium on. IEEE, 2013. 187-188

[19] Pedersoli F, Benini S, Adami N et al. XKin: An Open Source Framework for Hand Pose and Gesture Recognition Using Kinect. The Visual Computer, 2014, 10: 1107-1122

[20] Soltani F, Eskandari F, Golestan S. Developing a Gesture-Based Game for Deaf/Mute People Using Microsoft Kinect. In, ed. Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on. IEEE, 2012. 491-495

[21] 钱鹤庆. 应用 Kinect 与手势识别的增强现实教育辅助系统:[ 上海交通大学, 2011.

[22] 李小龙. 基于Kinect手势识别的虚拟人体解剖教学系统的设计与实现:[ 硕士学位论文]. 北京工业大学, 2014.

[23] Wang C, Liu Z, Chan S. Superpixel-Based Hand Gesture Recognition with Kinect Depth Camera. Ieee T Multimedia, 2015, 1: 29-39

[24] 张毅, 张烁, 罗元 et al. 基于Kinect深度图像信息的手势轨迹识别及应用. 计算机应用研究, 2012, 9: 3547-3550

[25] Wiley W C, Mclaren I H. Time‐of‐Flight Mass Spectrometer with Improved Resolution. Rev Sci Instrum, 1955, 12: 1150-1157

[26] Hsu R, Abdel-Mottaleb M, Jain A K. Face Detection in Color Images. Ieee T Pattern Anal, 2002, 5: 696-706

[27] 余旭. 基于Kinect传感器的动态手势识别:[ 硕士学位论文]. 西南大学, 2014.

[28] 孙丽娟, 张立材, 郭彩龙. 基于视觉的手势识别技术. 计算机技术与发展, 2008, 10: 214-216, 221

[29] Schlenzig J, Hunter E, Jain R. Recursive Identification of Gesture Inputs Using Hidden Markov Models. In, ed. Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on. IEEE, 1994. 187-194

[30] 江立, 阮秋琦. 基于神经网络的手势识别技术研究. 北京交通大学学报, 2006, 5: 32-36

[31] 陈静. 基于 Kinect 的手势识别技术及其在教学中的应用:[ 硕士学位论文]. 万方数据资源系统, 2013.

[32] 林水强, 吴亚东, 余芳 et al. 姿势序列有限状态机动作识别方法. 计算机辅助设计与图形学学报, 2014, 9: 1403-1411

[33] 徐小良, 汪乐宇, 周泓. 有限状态机的一种实现框架. 工程设计学报, 2003, 05: 251-255

[34] 童红. 孤立词语音识别系统的技术研究:[ 硕士学位论文]. 江苏大学, 2009.

**校对报告**

当前使用的样式是 [硕士论文]

当前文档包含的题录共36条

有0条题录存在必填字段内容缺失的问题

所有题录的数据正常