代　号	10701	学　号	1101120458
分类号	TP391	密　级	公开
U D C		编　号	

题（中、英文）目　人机交互中的动态手势识别及应用研究

Research on Dynamic Gesture Recognition and its Application in Human-Computer Interaction

作者姓名	郭小爽	学校指导教师姓名职称	卢朝阳 教授
工程领域	电子与通信工程	企业指导教师姓名职称	杨鹏斌 高工
论文类型	技术论文	提交论文日期	二〇一三年十二月

西安电子科技大学
学位论文创新性(或独创性)声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的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中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技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做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期_____

西安电子科技大学
关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律署名单位为西安电子科技大学。(保密的论文在解密后遵守此规定)

本人签名：_____ 日期_____

导师签名：_____ 日期_____

摘 要

人机交互是目前国内外研究的一个热点问题，也是未来计算机技术发展的必然趋势。手势识别是人机交互中的一项关键技术，在很多领域都得到了广泛的应用，如：聋哑人手语识别、机器人控制以及智能家居等，但由于受到环境因素、手势多变性的影响，目前手势识别还面临很多困难和挑战，处于研究初级阶段，需要进行深入、广泛地研究。

本文主要研究人机交互中动态手势识别及应用，并设计了一个可以实现动态手势识别和手势控制 PPT 两大功能的完整系统。主要包括以下内容：

1．自定义 8 种动态手势，并利用 Kinect 进行实时采集，建立了手势数据库；在分割手势时，首先用两次静止法确定动态手势的起止帧，然后将颜色与深度信息融合并结合帧间差分来进行分割。经过测试，这种分割算法受光照、背景等环境的影响较小，可以得到完整的动态手势。

2．在提取特征时，根据动态手势的运动轨迹模型，提取相邻轨迹点之间的角度值作为手势特征，并采用 12 方向链码进行量化编码，得到最终的手势特征序列。

3．在手势识别时，首先介绍了基于位置的识别算法，但是只能对四种简单手势进行识别，并且这种算法受环境影响较大，识别率较低；然后重点介绍了使用动态时间规整(DTW)算法识别动态手势具体过程，包括模板训练和识别。实验结果表明：DTW 算法对 8 种动态手势的平均识别率为 94%以上。

4．搭建了完整的动态手势识别及应用系统，可以在实时条件下识别动态手势进而用手势控制 PPT，实现简单人机交互。

关键词：人机交互　手势识别　　手势分割　运动轨迹　　动态时间规整

Abstract

Human-Computer Interaction(HCI) is a hot reaearch problem both at home and abroad, and it is the inevitable trend that the computer technology develops in the future, as well. As a kind of key technology in HCI, hand gesture recognition can be applied in lots of fields, such as sign language recognition for the deaf, robotic control systems, smart homes and so on. However, the environment and hands have great effect on gesture recognition, so that it faces lots of difficults and challenges and is in a primary stage of research. Owing to that, further and wide study is required.

This paper mainly studies the dynamic gesture recognition and its application in HCI. And at last, a whole system is founded, which has two kinds of function. One is dynamic gesture recognition, and the other is ppt control using gestures. This paper mainly talks about the following:

1. Firstly, eight kinds of dynamic gestures are defined and collected using Kinect in real-time, what's more, a gesture database is founded. In order to segment real-time dynamic gestures, the method of detecting twice static is employed to detect the beginning and end frames. Furthermore, besides the frame difference, the color and depth information fusion is employed. Experiments show that the environment, such as light and background, has little effect on the segment algorithm, and hence full gestures can be obtained.

2. To extract the features of gestures, a gesture model of motion trajectory is founded. According to that, the angle between adjacent track points is extracted. Moreover, twelve-directional chain code is employed to quantify and encode the angle value. As a result, gesture feature series are obtained.

3. For the recognition step, the paper introduces the recognition algorithm based on the position firstly. However, this algorithm is adapted to four kinds of simply gestures. What's worse, it is affected by the environment deeply, so that the rate of recognition is not high. After that, the paper mainly talks about the process of dynamic gesture recognition using Dynamic Time Warping(DTW), including the template trianing and gesture recognition. Experiments show that the rate of recognition for the DTW algorithm can reach 94%.

4. On the whole, a complete system for dynamic gesture recognition and its application is founded, which can recognize dynamic gestures and handle ppt in real-time, that is, human-computer interaction is realized.

Keywords: **Human-Computer Interaction** **Hand gesture recognition** **Gesture segment** **Motion trajectory** **Dynamic Time Warping**

目 录

第一章 绪论

1.1 课题研究背景及意义

人机交互[1](Human-Computer Interaction, 简称 HCI)是指人与计算机之间使用某种对话语言，以一定的交互方式，为完成确定任务的人与计算机之间的信息交换过程。它发展至今，共经历了五个阶段：早期的手工作业阶段、作业控制语言及交互命令语言阶段、图形用户界面(Graphical User Interface, 简称 GUI)阶段、网络用户界面的出现和多通道、多媒体的智能人机交互阶段[2]。

随着计算机的快速发展和科技水平的逐步提高，人机交互技术(Human-Computer Interaction Techniques)越来越受到人们的关注，传统的键盘和鼠标交互方式逐渐显示出它们的局限性[3]，迫切需要有新的交互方式来适应时代的变化，人机交互逐渐从“以机器为中心”向“以人为中心”转移。目前已经有许多新的交互方式，如：手势识别、语音识别、表情识别等，市场上也出现了许多新的交互设备，如：Leap Motion、Oculus Rift、Kinect、Google Glass 等，它们为人机交互带来了新的挑战和机遇。

作为日常生活中一种常见的交流方式，手势不仅具有直观、简单、形象的特点，而且蕴含着丰富的信息。因此，手势识别作为一种新兴的人机交互方式，以其自然性、直接性、简单丰富性的特点，逐渐成为人们研究的热点[4]。同时，手势识别也具有广泛的应用价值：

1．聋哑人手语识别领域

手语识别就是通过采集聋哑人的手语数据，利用模式识别算法，将具有一定含义的手语翻译成语音，让不懂手语的正常人能够明白手语的含义[5]；同时结合手语合成技术，即通过类似的翻译系统将语音翻译成手语，从而可以促进聋哑人与正常人的交流，为聋哑人的生活带来极大便利。另外，手势识别也可以用于计算机辅助双语教学、电视节目双语播放等。

2．用于机器人控制领域

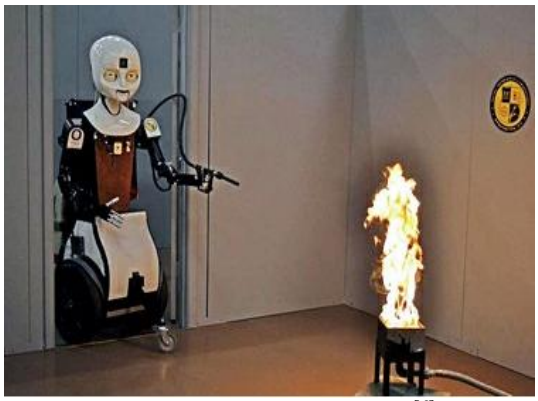
在一些特殊的场合或不便于人直接操作的场合，可以用手势控制机器人来完成一些操作。比如：在抗震救灾中时，在一些比较危险的区域或人不便于到达的区域，可以让机器人代替人来完成搜救工作；在太空中，可以让机器人辅助宇航员完成一些太空作业。另外，还可以实时远程监控家中情况，控制家用电器的提前打开等。如图 1.1(a)所示，是美国研究员研发的可以代替消防人员进入大楼灭火和救人的机器人。

3．用于日常生活和娱乐领域

利用手势识别技术，可以用手势来播放电影、控制图片、放映 ppt 等，给人们的生活带来了极大的便利；还可以通过体感设备控制屏幕上的人进行打球、跳舞、玩游戏等。如图 1.1(b)所示，是通过体感设备手势控制屏幕玩游戏。

4．用于未来的智能家居及智能交通领域

可以想象，在不久的将来，通过在家中安装手势识别系统，只需要挥动手势就可以开启电视、播放音乐等，也可以在汽车中安装手势控制系统，进行无人驾驶等。



(a) 消防机器人发现火源[6]

(b) 通过体感设备玩游戏[7]

图 1.1 手势识别的应用

这些只是目前手势识别已经实现和能预知的一些应用，也许在未来，随着技术的发展，会有更多更大范围内的应用。作为一种新兴的人机交互方式，手势识别有着广泛的应用前景，随着 2010 年 Kinect 的发布及今年 7 月份新版 Leap Motion 的发布，必将为手势识别带来新的研究前景。同时，目前手势识别的研究还处于实验阶段，特别是实时条件下的动态手势识别，理论还不够成熟，还不能适应复杂多变的环境，这也正是本课题研究的意义所在[8]。

1.2 手势识别技术的国内外研究现状

根据不同手势输入设备，手势识别技术通常分为两大类。一类是利用专门的硬件设备来进行输入，如数据手套。这方面的主要研究成果有：1983 年，Bell 实验室的 Grimes[9]最早获得“数据手套”的专利；随后，美国卡耐基 •梅隆大学(CMU)的 C. Lee 和 Y. Xu[10]利用 CyberGlove 数据手套作为输入设备，开发了一个控制机器人的手语识别系统；Mohammed 等[11]利用 PowerGloves 数据手套，识别由 95 个孤立词构成的词汇集，正确率为 80%；台湾大学的 Liang 等人[12~13]采用单个 VPL 数据手套对台湾手语课本中 250 个基本手语词条进行识别，识别率超过 90%；在我国的国家 863 计划中，吴江琴、高文等[14]采用 CyberGlove 数据手套，提出神经网络和学习判定树混合算法，在识别 30 个汉语手指字母时，识别率达到 98%以上。这种识别技术虽然能准确提取到手势相关信息，系统识别率较高，但是需要用到

特殊的硬件设备，成本较高，也不方便。

另一类是基于视觉的手势识别技术，它利用单个或多个摄像头采集手势视频或图象，再处理采集到的视频或图象，通过手势建模、手势分析，最终识别出手势。目前的研究成果主要有：1996 年，Starner 等[15]利用隐马尔可夫模型实现了由 40 个词语随机组成短句的美国手语识别，识别率达到 92%；1997 年，Grobel 和 Assam[16]利用隐马尔可夫模型(Hidden Markov Model, 简称 HMM)，对视频中 262 个孤立词的识别率为 91.3%。 2000 年，清华大学的祝远新、徐光祐等对基于视觉的动态手势识别做了深入的研究，提出了基于表观的动态孤立手势识别技术，将时空表示和总体图像运动表示两种表观变化模型作为手势的表观特征，对 120 个手势样本的识别率都在 88%以上[17]；在复杂背景下连续动态手势识别方面，提出了动态时空表观模型、多模式信息分层融合策略和动态时空规整算法，识别 12 种手势时的平均正确率达到 97%[18]。虽然这种技术没有前一种识别率高，容易受光照、运动、背景等环境因素的影响，但不需要复杂的硬件设备，成本较低，也更方便和自然。

另外，C. Vogler 和 D. Metaxas [19]将这两种识别方法结合在一起，采用一个位置跟踪器及三个互相垂直的摄像机作为手势输入设备，用自适应的 HMM 进行美国手语识别，对 53 个孤立词的识别率达到 89.9%。在指尖检测方面，中国科学技术大学的梅萍华[20]，提出了基于改进的径向对称变换(Radial Symmetry Transform)和肤色模型的指尖检测算法，对光照变化具有较好的鲁棒性，并能达到实时效果。

2010 年，随着微软 Kinect 的出现，使手势识别的研究更加方便。如：路易斯维尔大学的 Yi Li[21]利用 Kinect 实现了一个手势识别系统，能够实时检测到存在的手势并区分出每个手指，同时能够识别出 9 种预定义的手势；斯坦福大学的 Matthew Tang[22]利用 Kinect 提供的彩色和深度数据，提出了一种新的手势识别算法，对“grasp”和“drop”两种手势的识别率超过 90%；罗元，张毅等[23~24]利用 Kinect 传感器设计了智能轮椅手势控制系统，能够利用 Hu 矩向量识别出 5 中静态手势来控制轮椅的前进、后退、左转、右转和停止，随后又利用 HMM 识别出 5 种动态手势运动轨迹来控制轮椅运动；大连理工的王莹[25]利用 Kinect 传感器采集手势，并提出基于三维视觉的手势跟踪算法，提出了 tri-tracking 跟踪算法，在处理运动模糊、复杂背景、不同光照、遮挡等情况具有一定的鲁棒性；在本实验室中，高静雅[26]分别在普通摄像头和 Kinect 摄像头两种平台下识别静态手势，并提出了凹凸点检测和基于轮廓序列两种数字手势识别方法以及基于支持向量机(Support Vector Machine，简称 SVM)的手势识别算法，对八种自定义手势的识别率分别达到 90%和 85%以上。本文正是以静态手势识别为基础在 Kinect 平台下识别动态手势。

1.3 手势识别研究内容及关键技术

手势(gesture)[27]可定义为人手或者手和手臂相结合所产生的各种姿态和动作，它分为静态手势（指姿态，单个手形）和动态手势（指动作，由一系列姿态组成），前者对应模型空间里的一个点，后者对应一条轨迹。相应地，可以将手势识别分为静态手势识别和动态手势识别。本文研究的对象就是动态手势，下面首先介绍基于视觉的手势识别系统及其关键技术。

1.3.1 基于视觉的手势识别系统

基于视觉的手势识别系统主要包括：手势采集、手势分割、手势建模、手势分析及手势识别五部分。通常，为了提高手势图像精确度，根据实际需要，在输入端还要加上一些预处理，包括图像的去噪、平滑等；而在系统输出端，根据不同应用场景可以连接不同的应用系统，本文连接的是控制 PPT 操作。一个完整的手势识别与系统构成如图 1.2 所示：

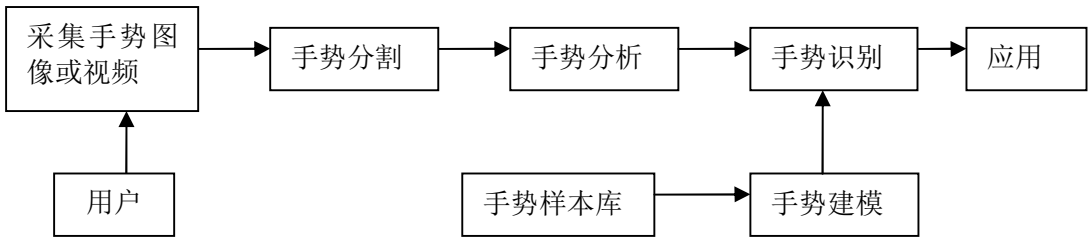


图 1.2 基于视觉的手势识别系统

1．采集手势图像或视频

要进行手势识别，首先要采集图像或视频作为系统输入，采集设备可以是 PC 机自带的摄像头，也可以是其他摄像头，本文使用微软的 Kinect 传感器采集图像，因为它不仅能采集彩色图像，还能提供深度图像，利用深度信息进行手势分割时，不受光照影响。

2．手势分割

要对手势图像或序列进行处理，首先要将手势从复杂背景中分割出来，手势分割的好坏直接影响手势识别结果，在整个手势识别系统中起着关键作用，也是手势识别技术的一个难点。

对静态手势而言，只是将单个手形分割出来，进行一些预处理，供后续使用。常用手势分割方法有：基于颜色空间的分割方法[28~29]、背景差分法[30~31]、基于边缘的方法[32]、基于深度的分割方法[33~34]等。

对动态手势而言，首先需要将不同手势动作分割开，即确定手势序列的开始与结束，然后将整个手势图像序列从背景中分割出来。在分割动态手势时，常使

用运动检测与跟踪的方法，在确定手势的开始帧后，对运动手势进行跟踪，直到手势结束。本文在确定手势开始帧后，利用帧间差分不断检测出每帧运动手势图像，以检测代替跟踪，直到手势结束，并利用 Kinect 提供的深度信息分割出运动手势，再以肤色信息作为辅助手段，得到完整的手势图像序列。

3．手势建模

手势建模就是选择一个合适的模型来描述手势库中的手势，这个模型包含了系统要研究的所有手势属性。在基于视觉手势识别系统中，手势模型起着至关重要的作用，是一个手势识别系统的基础。

对不同应用场景，模型选择也不同。在静态手势识别中，主要有两类模型[35]，一类是 3D 模型，包括体模型、几何模型、网络模型和骨架模型，其中最常用的是骨架模型；另一类是表观模型，主要是根据手势的形状、结构等表观信息来建立手势模型。

对动态手势，常用模型主要有三类[36]：(1)基于图像本身的模型，与静态类似，也可以将一段时间内的图像累加形成一幅 2D 图像；(2)基于运动轨迹的模型，在该模型中，不考虑手形变化，将手势的运动轨迹作为手势主要特征进行建模，这也是本文主要采用的模型；(3)将手形和运动轨迹结合起来总体考虑的模型，比较常用的是光流法，但计算较复杂，不适合实时条件下的手势识别。

选定手势模型后，建模就是利用手势库中的手势图像，提取相应特征，确定模型参数的过程。

4．手势分析

所谓手势分析，就是根据选取的手势模型，对待识别手势提取相应的特征，并找到一种确定模型参数的方法，用于模板匹配与识别。针对不同的手势模型，需要提取不同的手势特征，并采用不同的方法识别手势。

对于静态手势，如果采用 3D 模型，常提取手掌手指结构特征；如果采用表观模型，常提取 Hu 矩、Zernike 矩特征、傅里叶描述子、方向直方图等特征。

对于动态手势，如果是图像本身的模型，可以提取模型图像序列作为参数；如果是运动轨迹模型，常常提取手势轨迹点组成的运动轨迹，并将相邻手势轨迹点间的运动方向、运动距离作为特征；对于将手形和运动轨迹结合起来看做一个整体的模型，常采用光流法将光流矢量直方图作为手势特征。本文忽略手形的变化，采用运动轨迹模型，并将相邻轨迹点之间的角度作为手势特征。

5．手势识别

静态手势识别算法主要有模板匹配和统计学习，前者主要是基于距离比较的，特点是计算简单、速度快，但不适用于大规模的手势识别；后者主要有神经网络、Boosting 学习方法[37]、SVM 算法[38]等，适用于处理大规模的手势识别，缺点是计算较复杂。

目前常用的动态手势识别算法有： HMM、动态时间规整(Dynamic Time Warping, 简称 DTW)[39]和基于压缩时间轴的算法[40]等。压缩时间轴是利用具有某种属性的函数，把动态手势的运动轨迹压缩为一个点，然后利用静态手势识别的算法实现动态手势识别的功能，但这种算法计算量较大，并且确定时间压缩时的函数也较困难。HMM 是一种统计分析模型，具有一般性的拓扑结构，广泛应用于语音识别、行为识别、手势识别等领域，但由于它的一般性，在得到训练模板和进行手势识别时，都需要很大的计算量，因此本文采用 DTW 算法来识别手势，它与 HMM 算法相比，训练的计算量较小，并且能够解决不同手势的时间差异性问题，第四章将进行详细介绍。

1.3.2 动态手势识别研究难点

虽然越来越多的科研工作者开始研究动态手势识别，目前也取得了一些成绩，但整体来说，目前的识别系统只能做一些简单的应用，在做手势时还受到很多限制，不能完全适应各种复杂环境。主要原因是，在实时动态手势识别领域还存在很多的困难需要克服，下面具体说明。

手势分割和手势识别是整个识别系统最关键的技术，也是最难实现的技术。手势分割要求检测到完整的手势序列，并将它们从复杂的背景中分割出来；而手势识别要求根据手的不同特征来区分出不同的手势，但在实现过程中它们要受到复杂多变的环境因素影响和用户的手的影响。一方面，在不同的环境中，光照、图像背景、摄像头参数、计算机处理能力等都不相同，且随着时间和空间而不断改变；另一方面，人手是一个非常灵活的结构，做手势时具有不确定性，不同的人在做不同的手势时存在很大的差异性，即使是同一个人做同一个手势，前后两次也不一定完全相同，特别是动态手势，还具有时间和空间的可变性，同时手的运动速度和保持的时间也不相同。

这两方面的原因，给手势分割、特征提取及手势识别都带来极大的困难，另外，对实时的动态手势识别，分割时还需要确定手势的开始帧和结束帧，比静态手势识别更难完成。

因此，作为一种多学科交叉技术，动态手势识别仍然是一个极富挑战性的课题，需要进行深入研究。

1.4 论文组织结构

本文利用 Kinect 传感器，主要研究人机交互中的动态手势识别技术，目标是实现一个完整的系统，可以在实时条件下识别动态手势及手势控制 PPT。主要章节安排如下：

第一章是绪论。主要介绍本文研究的背景及意义，分析国内外手势识别的研究现状，并针对本文的研究对象，介绍基于视觉的手势识别的关键技术及动态手势识别研究的难点。

第二章是动态手势的采集及分割算法研究。主要介绍本文采集图像的方法及用到的数据库，并详细介绍本文采用的手势分割算法。

第三章是手势特征提取。主要介绍本文所用的手势模型，并针对该模型所提取的手势特征。

第四章是动态手势识别算法研究。主要介绍本文所采用的动态手势识别算法，并通过实验测试其性能。

第五章是动态手势识别及应用系统的实现。主要介绍本文所实现的手势识别及应用系统，包括系统的流程、操作方法及功能实现等，并对系统性能进行测试。

第六章是总结与展望。主要总结本文的研究内容，并对未来的工作内容进行展望。

第二章 动态手势的采集及分割算法研究

2.1 引言

手势采集是整个识别系统的基础，提供研究的对象；而手势分割是识别中的关键技术，分割的好坏直接影响手势识别的结果。对静态手势而言，识别对象是单个手形，因此只需要获取单个手势图像，将其从背景中分割出来；而动态手势包含一系列手势动作，需要将一系列手势图像序列从复杂背景中提取出来，比静态手势的分割更加困难。对实时的动态手势识别而言，还要首先确定手势动作的起止点，再提取手势序列，进一步增加了手势分割的难度。

本文研究的对象正是实时的动态手势，因此分割手势时，首先要确定动态手势的起止帧。目前使用的确定手势起止帧方法有：设定计时器[41]、计算手势运动速度、两次静止法[42]等。设定计数器的方法虽然能够确定何时开始采集手势和何时停止采集手势，但是每个动态手势的帧数和持续的时间都不同，需要经过反复的实验测试才能得到设定计时器的阈值，但是手是灵活多变的，对一些手势可能并不能得到好的效果；计算手势速度的方法与设定计时器类似，也需要经过大量实验反复测试手势准备、手势执行和手势结束的速度阈值，对于多变的手势和复杂的环境，确定手势速度的阈值也较困难。因此文中采用两次静止法来确定手势起止帧，后面将详细介绍。

本章主要介绍手势图像序列的采集过程及手势分割的相关算法，为后面的手势建模、手势分析及手势识别打下基础。

2.2 动态手势的采集

2.2.1 手势序列图像采集

本文采集手势的设备是微软的 Kinect 传感器，如图 2.1 所示。Kinect 一共有 3 个摄像头：中间的是 RGB 摄像头，可以获取 640×480 的彩色图像，最大帧率为 30 帧/秒；左右两侧是红外线发射器和红外 CMOS 摄像头组成的 3D 深度传感器，通过以色列 PrimeSense 公司的光编码(Light Coding)技术获取深度图像，分辨率为 640×480。两侧的麦克风阵列可以获取声音信息，进行语音识别，底座还有一个马达，可以根据目标位置调整 Kinect 的角度。

利用 Kinect 进行开发有两种方式，一种是使用微软(Microsoft)的软件开发包(Software Developmet Kit, 简称 SDK)，称为 Kinect for Windows SDK，它支持全身骨骼追踪，支持音频和马达，安装简单，编程语言为 C#，但是只能在 Windows 平台上开发，不支持其他特殊部位追踪，同时消耗更多的 CPU，所以常用于骨骼追

踪识别；另一种是使用 PrimeSense 公司提供的 OpenNI(Open Natural Interaction)，它定义了撰写自然操作程序所需的 API(Application Programming Interface)，提供一个多语言（主要是 C/C++）、跨平台的框架，能使软件开发人员更加方便地使用视觉、声音相关感应器以及相关数据、分析的中介软件(middleware)等，同时还可以在非 Windows 平台上做开发，非常适合做手势识别。本文就是使用的后者。

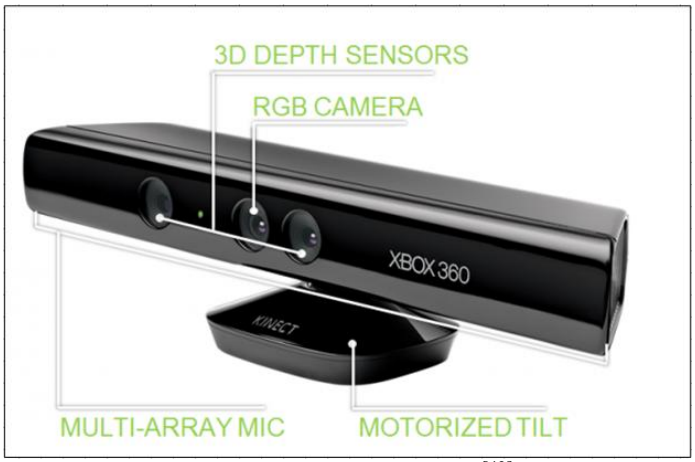


图 2.1 Kinect 结构组成图[43]

为了容易实现手势分割，采集手势时要求：1. 手部立起，正对 Kinect 摄像头挥动手势，手要在身体之前，并与身体保持一段距离；2. 作手势时，手与 Kinect 摄像头之间不能有其它物体，即手是 Kinect 前面的第一个物体。如图 2.2 所示，分别为 Kinect 采集的彩色和深度图像。

(a)彩色图像

(b)深度图像

图 2.2 Kinect 获取的彩色和深度图像

由前面介绍已经知道，动态手势是由一系列手势图像序列组成，因此在采集手势时，按照要求挥动手势，按每秒 30 帧进行采集。本文要研究的是实时条件下的动态手势识别，因此首先要确定手势的起止帧，即确定开始帧与结束帧。这里，按照两次静止法的思路，把手首次静止定义为手势开始，第二次静止定义为手势结束。手势静止的判断条件为：连续 3 帧手势的轨迹点都在阈值范围内，则认为静止。具体来说，就是：

1．设定阈值，打开 Kinect，举起手部正对 Kinect 并保持静止；

2．计算当前手势图像的轨迹点（具体求法会在后面介绍），如果连续 3 帧的

轨迹点都在阈值范围内，则认为手处于静止状态，把当前帧作为开始帧，继续执行步骤 3；否则返回步骤 2 继续执行；

3．在手势动作运行过程中，如果再次检测到手静止，则认为手势结束，并把当前帧之前的第二帧作为结束帧。

经过测试，这种方法可以将不同动态手势分割开，并且对环境的适应性较强，因此可以作为确定手势起始点的方法。

另外，为了使手势有意义，本文规定：每个动态手势的帧数在 10 到 40 之间。这样开始帧与结束帧之间的图像序列就组成了一个完整的动态手势。

2.2.2 手势数据库的建立

为了建立手势模型，使手势系统的研究具有一般性，需要基于一个手势数据库。以下将介绍本文用到的数据库。

要实现手势识别的研究，必须事先约定一个有意义的手势集合，针对本文的研究内容，自定义 8 种动态手势，分别为“right”、“left”、“up”、“down”、“V”、“vdown”、“triangle”和“circle”，如图 2.3 所示：

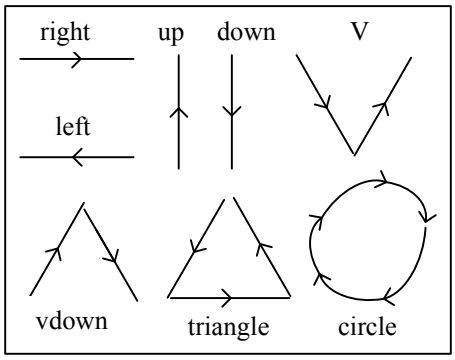


图 2.3 自定义的 8 种动态手势

建立数据库时，让 10 位实验者在实验条件下正对 Kinect 摄像头做手势，每人每个手势做 10 次，每个手势对应一个手势视频及一系列手势序列，共得到 10×10×8=800 个不同的手势，组成手势数据库。手势数据库中的部分手势序列图像如图 2.4 所示。

(a)“right”手势

(b)“left”手势

(c)“up”手势

(d)“down”手势

(e)“V”手势

(f)“vdown”手势

(g)“triangle”手势

(h)“circle”手势

图 2.4 手势库中 8 种手势的部分手势图像序列

2.3 动态手势分割算法

由动态手势的定义可知：动态手势的分割就是把动态手势包含的所有图像序列都从背景中提取出来。在实时条件下，为了获得完整的手势图像序列，首先要检测开始帧，然后对其进行跟踪，直到手势结束。为了进行后续的手势分析和处理，在检测到每帧图像后，将其从背景中分割出来，组成一个完整的动态手势。本节首先介绍目前常用的动态手势分割方法，分析其优缺点，然后详细说明本文使用的分割方法，并分析其性能。

2.3.1 常用动态手势分割算法

动态手势分割可以分解为一系列手势图像的分割，因此可以采用静态手势图像的分割方法，如：肤色分割法、边缘分割法及多信息融合等，由于是对手势序列进行处理，如果每个图像都采用静态的方法，速度较慢，达不到实时的要求，且受环境影响较大。考虑到动态手势与静态手势的区别，常采用运动检测与跟踪的方法来进行分割，即检测并分割出开始帧后，对其进行跟踪，直到手势结束。

以下是几种常用的检测算法：

1．光流法

所谓光流(optical flow)，就是当人的眼睛观察运动物体时，三维的物体会在人眼的视网膜上形成一系列连续变化的二维图像序列，这些连续变化的图像不断从视网膜“流过”，就像是光的“流动”。实际上，光流是空间中的运动物体投影在二维平面上时每个像素运动的瞬时速度，能够反映出物体的运动状态。

最早研究光流的是 20 世纪 50 年代的 Gibson 和 Wallach 等人，他们提出了 SFM(Structure From Motion)假设，但直到 70 年代末该假设才得到验证。1981 年，Horn 和 Schunck[44]首先将二维速度场与灰度联系起来，推导出灰度图像光流场基本计算方法，是经典光流法。目前常用的是 Bruce D. Lucas 和 Takeo Kanade 于 1981 年提出的 Lucas-Kanade（简称 LK）光流法[45]。基于光流的算法能够广泛地应用在交通、气象及医学等领域，比如在交通场景中检测运动车辆、在气象中分析云图、在医学上分析并诊断异常器官细胞等。

用光流法进行运动检测的基本原理[46]是：将图像中的像素点与速度联系起来，把一个速度矢量赋予给每一个像素点，就可以形成一个反映物体运动状态的光流场。如果图像中物体全部静止，那么在图像全部区域内，光流矢量都连续变化；当图像中有运动物体时，运动物体形成的速度矢量和周围背景的速度矢量必定不同，运动目标和图像背景间就会有相对运动，从而可以检测出运动物体并确定其位置。

由于光流能够反映物体的运动状态，并带有三维物体结构等信息，因此光流法检测运动物体的效果很好，但是计算量较大，很难在实时条件下使用。

2．背景去除法

背景去除法是将当前图像与背景图像相减来得到运动的前景图像的方法，算法简单且较易实现。该算法要求摄像头是固定的，首先根据采集的图像序列信息得到初始化背景模型，然后将当前帧与背景图像进行对比，将背景从当前帧中去除，结果就是运动的前景图像。在计算时，将图像中差别较大的像素区域作为运动区域，其他基本不变的区域就认为是背景。需要注意的是，在计算过程中，背景可能是随时变化的，所以要不断的更新背景，再利用新的背景模型来得到运动

目标。

该算法适用于图像背景已知的情况，在获取静态背景时，最简单的是将时间平均图像或加权平均图像作为背景模型。然而在实际中，背景是动态变化的，它会随着光照、运动、物体的移动等随时发生变化，有时甚至是不可预料的。如何快速高效地建立背景模型并根据这些变化实时更新背景模型，是背景去除法中要研究的关键问题[47]。

3．帧间差分法

帧间差分法与背景去除法相似，但背景去除是在建立并更新背景模型后，将当前图像与背景进行差运算，而帧间差分是将连续图像序列中几个相邻图像帧按像素值相减来检测运动物体，得到的是运动目标的轮廓。最简单的是相邻两帧进行差分，并阈值化。

$$\mu_t(x,y)=\left|f_t(x,y)-f_{t-1}(x,y)\right| \qquad \text{式(2-1)}$$

其中，$f_t(x,y)$ 表示第 t 帧图像，$\mu_t(x,y)$ 表示第 t 帧与第 t-1 帧图像差分结果，差分后的二值图像为：

$$f(x,y)=\begin{cases}0，\text{当}\mu_t(x,y)<\varepsilon\text{时}\\1，\text{其他}\end{cases} \qquad \text{式(2-2)}$$

其中，ε 表示差分时的阈值。

相邻两帧图像差分时，算法简单高效，容易实现，受光线影响也较小。但是，当目标运动较快时，可能导致相邻两帧图像没有公共区域，无法检测到目标；反之，若目标运动太慢时，相邻两帧图像差别较小，也无法检测到运动目标。所以使用帧间差分检测运动物体时要选择合适的阈值。

目标检测与跟踪一直都是热点研究问题，在手势识别中也应用较多。前面已经介绍了目标检测算法，下面介绍两种常用的跟踪算法：

1．粒子滤波算法

所谓粒子滤波就是指：在状态空间中寻找一组传播的随机样本，然后用这些样本来近似的表示概率密度函数，并用样本均值代替积分运算，从而获得系统状态的最小方差估计，可以形象地将这些样本称为“粒子”，故而叫粒子滤波。它源于 Montecarlo 的思想，即以某事件出现的频率来指代该事件的概率，主要用于状态空间模型[8]。

粒子滤波的应用范围很广，在很多领域都有涉及，比如跟踪交通场景中的行人和车辆以及跟踪经济领域中数据的变化等，还有其他一些非线性的随机系统问题，都可以使用粒子滤波来解决，因此，它也是目前目标跟踪的研究重点。。

由于粒子滤波是使用若干个离散采样点的加权和来代替状态空间模型的后验概率，因此在使用粒子滤波进行目标跟踪时，可以在同一时间内进行多个假设来提高跟踪的稳定性。但是这种算法计算量很大，很难实现实时条件下的动态手势

跟踪，因此在实际应用中可以根据需要进行一定的简化。另外，实验表明要保证跟踪的正确性和实时性，常需要把摄像头前的手缩小到一定大小，而且相对 Camshift 算法而言，它的计算量还是比较大的[8]，因此本文并未使用这种方法。

2．Camshift 算法

Camshift 算法，即连续自适应数学期望移动算法，核心是 Mean Shift 算法[48]。它最早是用于人脸的动态跟踪[49]，并根据颜色概率分布图来搜索目标。事实上，任何颜色信息比较明确的物体都可以使用 Camshift 来进行跟踪，当然也包括手势图像的跟踪。

Camshift 算法的实现可以分为三部分：Back Project（反向投影）计算、Mean Shift（均值漂移）算法和 Camshift 算法[50]。

Back Project 算法用于计算目标的颜色概率分布图，由于 RGB 颜色空间对光照亮度变化较为敏感，而 HSV 颜色空间的 H 单通道灰度图能够反映目标的颜色信息且不受光线影响，所以首先将图像从 RGB 颜色空间转换到 HSV 空间，然后基于 H 通道的颜色概率分布图来计算原始图象的颜色概率分布图。

Mean Shift 最先是 Fukunaga 等人在一篇关于概率密度梯度函数的文章中提出的，是指偏移的均值向量。而我们经常说的 Mean Shift 算法，通常是指一个迭代的步骤，就是首先计算出当前点的偏移均值，将该点移动到其偏移均值，然后将该点作为新的起始点继续移动，直到满足一定的条件为止，是为了计算新的搜索窗口的位置和大小，并使之收敛。Camshift 算法是指连续的 Mean Shift 算法，就是对视频中每一帧图像都按照 Mean Shift 算法进行运算，运算结果作为下一帧 Mean Shift 算法的搜索窗口的初始值，使视频中每一帧成为一个连续的迭代过程，实现目标跟踪。

Camshift 算法的收敛速度很快，抗干扰能力也较强，但它不能自动跟踪，每次运行之前必须手动添加待跟踪的目标，不能直接用于实时动态手势的跟踪。

2.3.2 多信息融合与帧间差分相结合的分割算法

文献[26]中，作者在分割静态手势时，将基于颜色的手势分割与基于深度与颜色相结合的手势分割算法做了比较，实验结果表明：深度与肤色信息结合的分割算法对光照、背景颜色及运动物体都有较强的适应性，且分割速率更快，更能满足实际应用的要求[26]。但动态手势分割与静态手势分割有明显不同，不能直接采用这种算法来分割动态手势。而作为运动检测的一种常用方法，帧间差分运算简单，速度较快，适合实时条件下的手势检测。

本文将深度和颜色信息相融合的分割算法与帧间差分结合起来，作为动态手势的分割算法。具体思路是：根据手势静止条件确定开始帧后，利用帧间差分检测每帧运动的手势，并且用不断检测来代替跟踪，直到手势结束；检测到运动的

手势后，利用 Kinect 提供的深度信息分割出手势，并用肤色信息来判断分割出的是否为人手。具体流程图如图 2.5 所示。

根据手势状态的不同，可以将用户做手势的过程分为三个阶段：手势开始、手势进行和手势结束。系统正常启动后，举起手部开始做手势，这时进入手势开始阶段，首先判断手是否处于静止状态，若是，则将此帧图像作为手势的开始帧，进入手势进行阶段，同时手开始运动；若不是，则手继续保持静止，直到确定手静止为止。在手势进行阶段，手保持在视野范围内做运动，若手势动作做完，则将手保持静止，将进入手势结束阶段。在手势结束阶段，判断出手再次处于静止状态时，将当前帧作为手势结束帧，认为本次手势动作完成，立即放下手，准备做下一个手势动作。

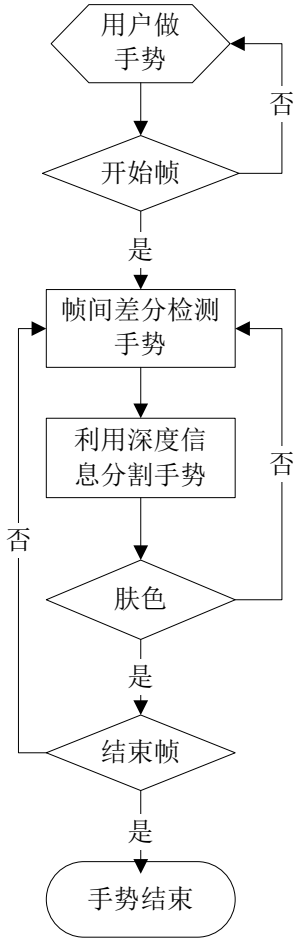


图 2.5 动态手势分割流程图

实际中，可以通过设置手势的状态标识符来区分不同阶段。若用 is_gesture 表示获取到完整的手势，is_static 表示手处于静止，is_process 表示手处于运行状态，那么在初始状态，is_gesture、is_static 和 is_process 均设为 0，手势开始后手静止时 is_static=1，手势动作进行时 is_process=1，is_static=0，手再次处于静止时，

is_static 又变为 1，is_process=0，手势完成后 is_gesture=1。若要开始下一手势，所有运动标示符又回到初始状态。

该算法的具体步骤是：

1．用户举起手部，正对 Kinect 摄像头做手势，用深度与肤色相融合的方法分割出当前静态手势，并找到其手心点，如果连续三帧的手心点都在阈值范围(0,segTH)内，则认为手处于静止状态，令 is_static=1，并将当前分割出的手势图像作为开始帧；

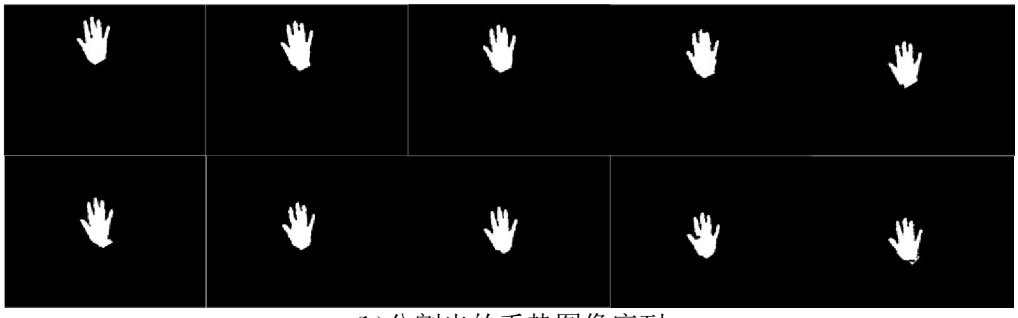
2．进入手势进行阶段，此时 is_process=1，is_static=0，对彩色视频做帧差，分析得到的差分二值图像和深度图像，若在深度阈值范围有运动物体，则将该运动物体分割出来；

3．首先根据肤色模型公式判断分割出的图像是否为手势图像，若是，则将其加入到手势图像序列中，若不是，则将其舍弃。然后反复执行步骤 2 和 3，直到再次判断出手势静止即 is_static=1，认为手势结束，得到了一个包含开始帧、中间手势序列及结束帧的完整动态手势，令 is_gesture=1，is_static=0，is_process=0。

2.3.3 实验结果与分析

为了测试该分割算法的性能，分别在不同场景下进行了实验，如图 2.6 和 2.7 所示。图 2.6 是在光线较暗时分割动态图像序列的效果，图 2.7 是在光线正常时分割动态手势的效果，其中 2.6(a)和 2.7(a)均表示原始彩色图像序列，而 2.6(b)和 2.7(b)为对应的分割出的手势图像序列。

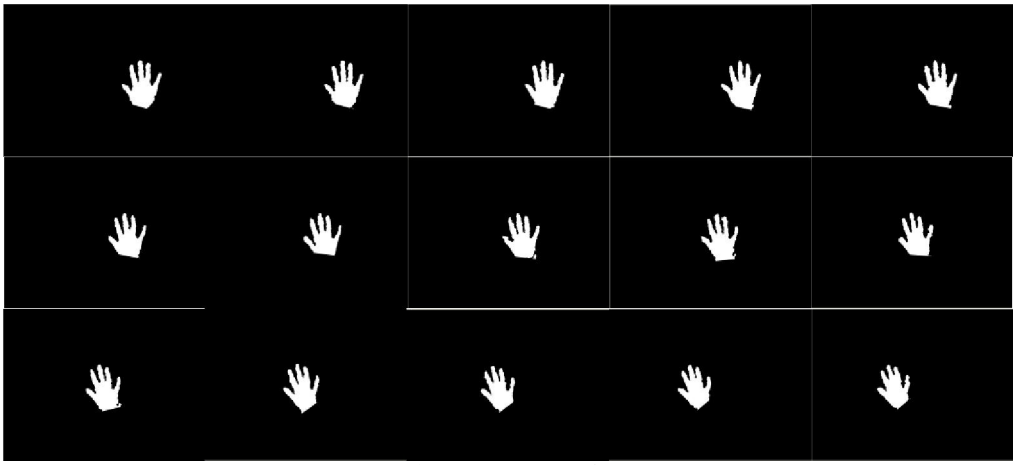
(a)原始彩色图像序列



(b)分割出的手势图像序列

图 2.6 光线较暗时动态手势分割效果

(a)原始彩色图像序列



(b)分割出的手势图像序列

图 2.7 正常光线时动态手势分割效果

由实验结果可以看出：

1．当光线正常时，利用该分割算法能够完整的将整个手势图像序列都分割出来，且效果较好；

2．当光线较暗时，分割出的动态手势与光线正常时几乎没有差别，这是因为检测运动手势时使用帧间差分，受光线影响较小，在分割出手势图像时是利用 Kinect 红外摄像头返回的深度信息，不受光照的影响，因此仍然能够获得较好的分割效果。

通过以上对这种动态手势分割方法的分析，我们可以看出它具有以下特点：

1．用运动差分进行运动手势的检测，速度较快，能够满足实时性要求，也减少了光照的影响，并且以不断检测代替跟踪，减少了跟踪的算法开销，；

2．先检测到运动手势再进行分割，而不是将深度范围内的手势都分割出来，减少了冗余信息，降低了计算量和时间开销；

3．对序列中每个手势的分割采用深度信息与颜色信息相融合的分割算法，首先利用 Kinect 传感器提供的深度信息分割出手势，再将肤色信息作为辅助手段判

断分割出的是否为人手，既降低了光照的影响，又确保了分割的正确性；

4．利用了帧间的运动和深度信息，将运动手势限定在阈值范围内，使其不受其他环境因素的影响，增加了对复杂环境的适应性。

2.4 改进的手臂去除算法

与静态手势一样，由于分割得到的很多手势图像都带有手臂，并且每个手势图像中手臂长度不一样，如果直接作为分割结果，在识别手势时将会造成错误。因此，必须首先切除手臂。在文献[26]中，作者采用基于宽度的手臂去除算法来切除手臂，切除成功率达到 95%以上。

与静态手势不同的是，静态手势只是对单个手形进行处理，因此在做手势时，手一般位于视野的中心，分割出的手势图像（可能包含手臂）大多都是在竖直方向；但在做动态手势时，手的位置是时刻变化的，因此得到的带有手臂的手势图像可能是倾斜的，如图 2.8 所示，如果直接采用基于宽度的手臂去除算法进行手臂切除，必将造成错误。因此，这里采用改进的手臂去除算法，即首先将倾斜的手势图像旋转到竖直方向（默认为手指朝上），再切除手臂，最后将得到的手势反向旋转得到最终的手势图像。

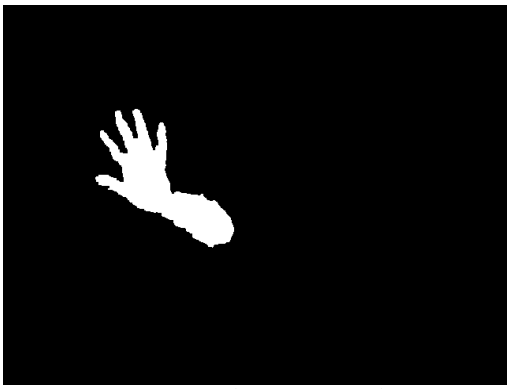


图 2.8 倾斜的手势图像

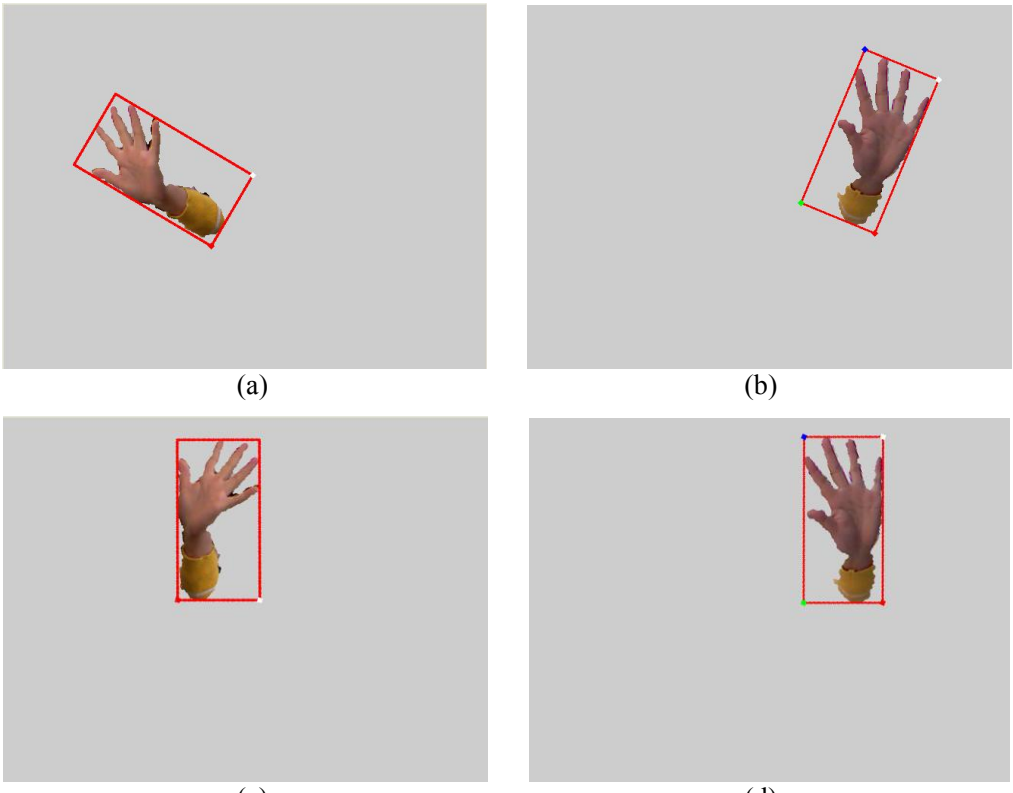
该算法步骤是：

1．首先得到手势图的可倾斜最小外接矩形，定义为一个二维手势盒子 handbox。手势盒子的角度 handbox.angle 定义为：水平轴逆时针旋转，第一次与盒子的边重合所划过的角度，为负值；手势盒子最下方的顶点为第一个顶点，沿逆时针方向，依次为第二、三、四个顶点，其中，第一、二顶点间的距离为手势盒子的高 handbox.size.height，第一、四顶点之间的距离为盒子的宽 handbox.size.width，高和宽中较大者 max_handbox 为盒子的长，即手和手臂的总长，另一个 min_handbox 为盒子的宽。

2．计算手势盒子的长宽比是否在阈值范围内，如果是，则表明没有包含手臂，不需要切除，否则继续执行步骤 3；

3．判断 handbox.angle 是否在阈值范围内，若是，则认为手势倾斜较小，不需要旋转，旋转标志 isroat=0，直接执行步骤 5，否则 isroat=1，先按步骤 4 旋转，再执行步骤 5；

4．计算要旋转的角度 angle，若 $handbox.size.hight > handbox.size.width$，则 $angle = handbox.angle$，为负值，顺时针旋转；否则，$angle = 90 + handbox.angle$，为正值，逆时针旋转；如图 2.9 所示。

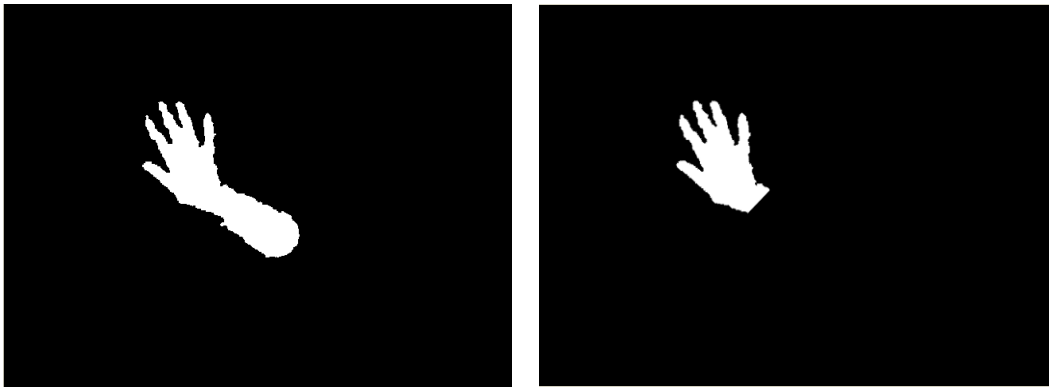


(a) (b)

(c) (d)

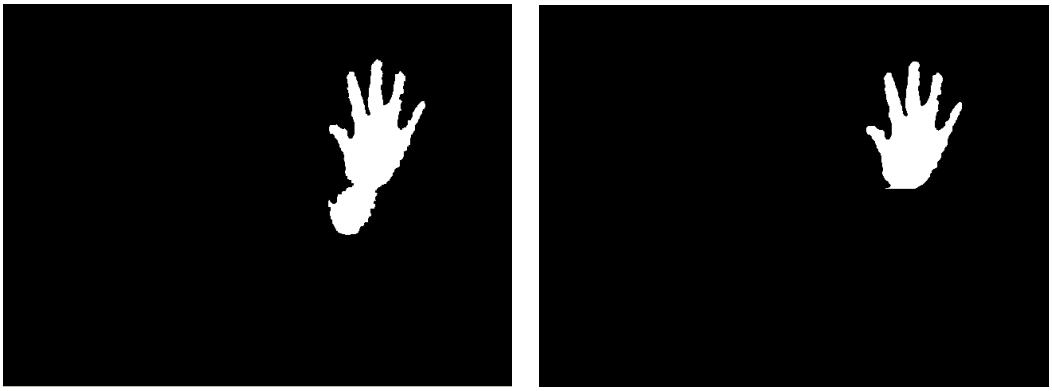
图 2.9 手势图像旋转

5．按基于宽度的手臂去除算法进行切除手臂，此时将原来算法中手势的长更新为手势盒子的长 max_handbox，手势的宽更新为手势盒子的宽 min_handbox，手势盒子的长宽比即为手势的长宽比，其他不做更改，具体算法这里不再赘述；



(a) 原始分割图像 (b) 切除手臂后的图像

图 2.10 手向左偏时去除手臂效果



(a) 原始分割图像 (b) 去除手臂后图像

图 2.11 手向右偏时去除手臂效果

6．若手势图像进行过旋转，即 isroat=1，则切除手臂后，再将手势图像反向旋转，即旋转的角度为-angle。

去除手臂效果如图 2.10 和 2.11 所示，其中 2.10(a)和 2.11(a)分别为手向左偏和手向右偏时直接分割后的原始图像，2.10(b)和 2.11(b)为相应切除手臂后的图像。

切除手臂时需要注意的问题是：由于手势需要旋转，可能使旋转后的手势超出图像边界。为防止这种情况发生，做手势时尽量使手位于视野中央区域范围。

2.5 本章小结

本章主要介绍动态手势的采集及分割算法。采集手势时使用微软的 Kinect 传感器，它不但能提供彩色图像还能提供深度图像，文中简单介绍了 Kinect 的组成及各部分功能，采集手势时的注意事项。在动态手势分割算法方面，首先说明了动态手势与静态手势的不同，对实时动态手势采用两次静止法来分割手势的起止点，在对手势序列分割时，分析了目前常用的动态手势分割算法，提出本文的分割算法——将深度信息和颜色信息相融合，并结合帧间差分的分割方法，不但对光照、背景等环境因素有很好的鲁棒性，还能满足实时性的要求，并在最后针对分割图像出现带倾斜手臂的现象提出了基于宽度的改进算法来去除手臂，为下一章的特征提取提供处理的对象。

第三章 手势特征提取

3.1 引言

在上一章中，我们已经分割出了动态手势图像序列，但它们包含较多冗余信息，不能直接用于手势的识别，必须首先针对选取的手势模型提取相应的特征。所谓特征提取，就是从原始图像序列的全部信息中选出一定数目的、能有效表征动态手势的信息，作为计算手势模型空间参数和手势识别时分类的依据，如果特征选取不正确，将会直接导致分类的错误。提取出的特征不但要能区分不同的手势，还要能够适应环境的变化，对不同用户和不同手势都有较强的适应性。对实时的动态手势而言，选取的特征还要尽量计算简单、速度快，以便能够满足实时性的要求。

直观看来，动态手势可以分为手形的改变和运动轨迹的变化。对手形而言，可以静态手势的方法提取特征，比如几何特征、不变矩特征和傅里叶描述子等；对运动轨迹而言，可以通过获取手势的轨迹点，得到轨迹特征来进行识别；如果将二者综合考虑，可以直接将两者相结合来提取特征，也可以通过光流算法，将手形和轨迹看作一个整体来提取特征。

在本文中，忽略手形的变化，选取动态手势的运动轨迹作为手势模型，因此本章主要介绍动态手势运动轨迹的特征提取。

3.2 运动轨迹的特征提取

只考虑运动轨迹时，动态手势的识别可以转化为手势轨迹点的匹配，即首先获取手势轨迹点，然后提取特征，进行匹配和识别。手势轨迹点的获取可以采用不同的方法，如 Camshift 算法跟踪手势轨迹点、将手势序列图像质心作为手势轨迹点等；在得到运动轨迹时，也可以提取不同的特征来识别手势，如方向、距离等。下面将介绍本文中获取手势轨迹点的方法，以及得到运动轨迹后特征提取的思路和方法。

3.2.1 手势轨迹点

所谓手势轨迹点，就是忽略手势的大小和形状，用来表示手势图像的一个点。如果要提取动态手势运动轨迹的特征，必须首先获取每帧图像的手势轨迹点。

对于用 Camshift 算法跟踪运动手势的，可以使用收敛的搜索窗口的质心作为手势轨迹点。而在本文中，前面已经分割出手势序列，并且序列中的每个手势图像都包含完整的手形，因此可以直接使用手势图像的质心作为手势轨迹点，如图

3.1 所示。

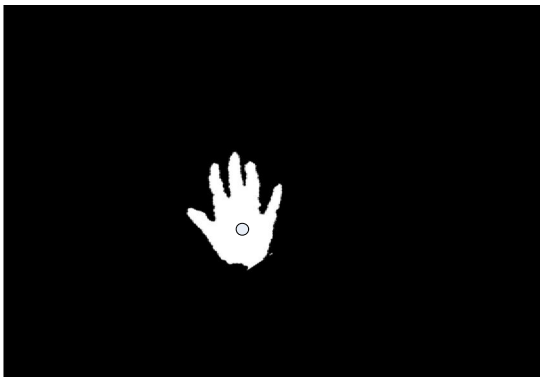


图 3.1 手势图像的质心

根据质心的定义，对于一幅大小为 M×N 的灰度图像 f(x,y)，其质心点坐标(cx,cy)的计算公式为：

$$cx = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} xf(x,y)}{\sum_{x=1}^{M}\sum_{y=1}^{N} f(x,y)} \qquad \text{式(3-1)}$$

$$cy = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} yf(x,y)}{\sum_{x=1}^{M}\sum_{y=1}^{N} f(x,y)} \qquad \text{式(3-2)}$$

由于本文分割得到的手势序列图像都是二值图像，其质量分布是均匀的，计算出的质心也为手的中心点。因此计算质心的公式可简化为：

$$cx = \frac{1}{M}\sum_{x=1}^{M} x, \quad cy = \frac{1}{N}\sum_{y=1}^{N} y \qquad \text{式(3-3)}$$

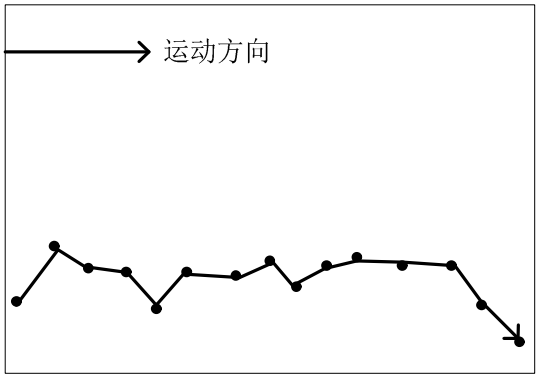


图 3.2 手势轨迹点组成的运动轨迹

实际中，手心点的计算可以采用这样的思路：将得到的二值手势图像进行距离变换，距离手势边缘最远的点即为手心点。具体步骤如下：

1．对分割得到的二值手势图像进行距离变换，计算手势图像上每个像素点到

它最近邻零像素点的距离，得到一幅新的图像，该图像上手势边缘的点全部变为零，其他手势点都变换为该点距离最近边缘的距离；

2．找出变换后的图像上的最大值点，即距离原始二值手势图像边缘最远的点，即为手心点 handcenter(x,y)。这样，就可以将手心点作为该手势图像的手势轨迹点。该计算方法可以分别使用 OpenCV 中的 cvDistTransform 函数和 cvMinMaxLoc 函数实现。

对于一个包含 N 个手势图像的序列，反复执行步骤 1 和 2，每幅手势图像都可以得到一个手势轨迹点，这样就可以得到 N 个轨迹点，共同组成手势的运动轨迹，如图 3.2 所示。本文中就是采用这种思路来计算轨迹点并得到手势运动轨迹的。

3.2.2 特征提取

在得到手势的运动轨迹后，我们并不能直接用来进行模板匹配，进而识别出手势，必须从运动轨迹中提取出能够表示手势运动状态的一般特征，并能够对环境和噪声有较强的适应性。对于实时的手势，还必须保证提取的特征计算简单、容易实现，能够满足实际的需要。

我们知道，由手势轨迹点组成的运动轨迹包含三个信息[41]：轨迹的起始点、轨迹的运动方向和轨迹运动的距离。

1．轨迹的起始点

轨迹的起始点只表明手势开始的位置，任何一个手势从视野范围内的任一位置开始都是有意义的，对手势的类别是没有影响的，因此轨迹的起始点并不能反映手势的运动状态，更没有必要单独提取出来作为手势轨迹的特征；

2．轨迹的运动方向

每个轨迹点的运动方向都表明了该轨迹点运动的趋势，如果将运动轨迹上每个轨迹点的运动方向联系在一起，将能够显著地表示手势的运动状态，因此可以将轨迹的运动方向作为手势运动轨迹的特征；

3．轨迹运动的距离

由于个体的差异性，不同用户在做不同手势时的速度是随机变化的，每次运动的时间也会改变，即使是同一个人，在前后两次做同一个手势时速度和时间也可能不完全相同，所以每个手势运动的距离存在很大的不同，并不能直接将轨迹运动的距离作为手势分类的依据；但是，如果将距离和轨迹的方向结合起来，组成运动矢量，通过归一化，也可以作为动态手势的特征。

由以上分析可知，轨迹点的运动方向是动态手势运动轨迹的显著特征，只需要提取运动轨迹的方向特征就可以识别出动态手势，并且文中对手势运动的速度和时间都没有严格的限制，也没有必要提取手势运动距离，因此本文只考虑运动轨迹的方向特征。

数学上，轨迹点的运动方向用相邻轨迹点之间的方向矢量(r,θ)来表示，如图3.3所示：

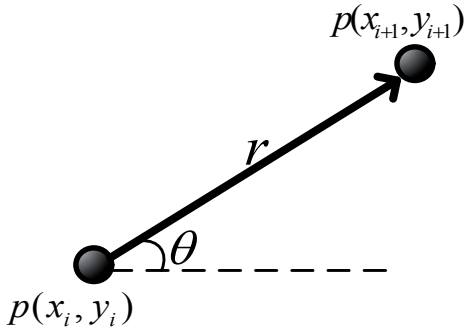


图 3.3 相邻轨迹点的方向矢量

其中，r 表示方向矢量的大小，即相邻轨迹点之间的距离，θ表示相邻轨迹点之间的角度，即从 x 轴正方向运动到方向矢量所划过的角度。

对于较简单的运动轨迹，仅用角度就可以作为运动轨迹的特征，而复杂的动态手势则可以再结合相邻轨迹点之间的距离，用方向矢量来作为运动轨迹的特征；而本文中虽然定义了圆圈和三角形手势轨迹，但相互之间的区分度较大，为了减小计算量，提高运行效率，也可以只选择角度作为动态手势的特征。

角度的计算公式为：

$$\theta = \arctan(\frac{y_{i+1}-y_i}{x_{i+1}-x_i}) \qquad \text{式(3-4)}$$

其中，x_i和y_i分别为当前轨迹点的横坐标和纵坐标，x_{i+1}和y_{i+1}表示相邻下一轨迹点的横坐标和纵坐标，角度值θ的取值范围为$(-180^\circ,180^\circ]$。

对于一个包含 N 个图像的手势序列，在得到运动轨迹后，通过计算相邻轨迹点之间的角度特征，就可以得到一组包含 N-1 个角度值的特征序列。

3.3 角度量化

对运动轨迹提取特征后，得到的角度都是连续值，并且包含很多冗余信息，不能直接进行训练和识别，必须首先进行数据压缩，将角度离散化。本文采用矢量量化编码的思想，对得到的角度值进行量化编码。下面首先对矢量量化编码进行简单介绍。

矢量量化(Vector Quantization, 简称 VQ)是上世纪 70 年代后期发展起来的一种数据压缩技术。1978 年，旅美墨西哥学者 Buzo 在他自己的博士论文中提出第一个实际的矢量量化器，并在 1980 年出现了以 Linde、Buzo 和 Gray 三个人的名字命名的 LBG 算法（矢量量化器的设计算法），随后矢量量化技术广泛应用于图像压缩和语音信号编码中。

所谓矢量量化编码，就是把大量的输入数据平均组成若干组，每组由其中的

几个组成，然后每一组分别量化编码，也就是说，将几个标量数据组成一个矢量，每个矢量的维数相同，然后以矢量为单位逐个进行量化这样既可以达到数据压缩的目的又不损失太多信息。

VQ 技术中最关键的就是码书的设计，即量化编码后输出的矢量空间的选取，码书设计地越优越，矢量量化器的性能就越好，最经典的码书设计算法就是 LBP 算法。

由于角度只有一维信息，因此可以看做最简单的矢量量化。对角度量化编码时，首先要确定量化的标准和编码方法，也即量化编码后特征值的范围，本文是基于方向链码的思想。

链码（又称为 freeman 码）是一种边界的编码表示法，在计算机图形学、图像处理和模式识别中很常用。按照不同斜率，经常使用 4 方向链码和 8 方向链码。根据实际的需要，也有人根据方向链码的思路，使用 12 方向链码、7 方向链码[51]和 20 方向链码。分的方向数越多，量化结果越精确，但计算量也随之增大。考虑到手势特征的准确性和计算效率，本文采用 12 方向链码进行角度量化。

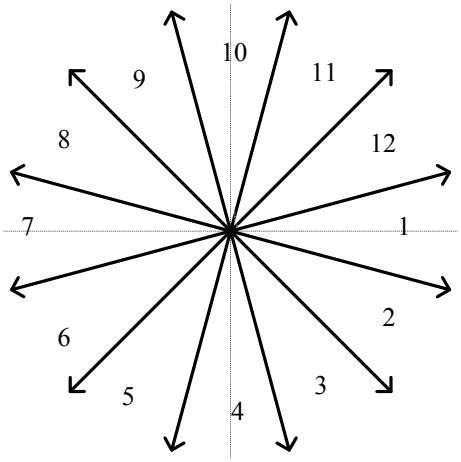


图 3.4 采用 12 方向链码量化编码

如图 3.4 所示，将整个平面平均分成 12 份，每份 30 度，从 x 轴正方向所在的区域开始，按照顺时针，每个区域依次标记为 1~12。

确定编码方法后，将得到的角度值序列中的每个角度，依次映射到相应的区域，进行量化编码。计算公式为：

当$\theta >=15^{\circ}$时，

$$k=\frac{\theta}{30}+0.5 \qquad feature=13\text{-}k \tag{式(3-5)}$$

当$\theta <15^{\circ}$时，

$$k=\frac{\theta+180}{30}+0.5 \qquad feature=(k+7)\%12 \tag{3-6}$$

其中，θ为待量化的角度值，k 是量化计算时的中间变量，feature 是角度值量

化后的特征值，范围为 1~12 的整数。

这样，将运动轨迹中提取出的每个角度值都按公式(3-5)和(3-6)进行计算，就可以将提取出的 N-1 个角度特征量化为 N-1 个 1 到 12 之间的数字,用于手势识别。对于数据库中的手势，每个动态手势都能提取出一个量化后的数字序列，保存为 txt 文件。

3.4 本章小结

本章主要介绍了在运动轨迹模型下的特征提取，即首先获取手势轨迹点，组成运动轨迹，然后提取特征。文中首先给出了轨迹点的计算方法，然后通过分析比较，提取相邻轨迹点的角度值作为轨迹特征，并使用 12 方向链码进行量化编码得到最终的特征序列，作为下一章手势识别中模板匹配的标准。

第四章 动态手势识别算法研究

4.1 引言

手势识别是识别系统的最后环节，也是识别系统的关键技术。由于复杂环境的影响和时间的限制，以及人手的灵活多变性，手势识别技术还面临很多困难，需要进行深入的研究。

目前动态手势识别最常用的算法是 HMM 和 DTW 算法。HMM 是语音识别、信号处理及手势识别领域常用的一种方法，但它要求在训练阶段提供大量训练数据，并且在计算模型参数时需要经过反复的迭代运算；另外，考虑到不同手势可能持续时间不同的问题，HMM 对此显得无能为力，而 DTW 算法却表现出明显的优势。除此之外，也有人采用曲线拟合的动态手势识别算法，即得到手势运动轨迹后，利用三次 B 样条拟合曲线来描述该运动轨迹，并选择合适的曲线矩来识别手势，这种算法识别率较高，但是计算非常复杂，不能满足实时性的要求[52]，应用时必须进行改进。

本章主要介绍本文所用的动态手势识别算法，首先简单介绍了基于位置的识别方法，然后重点研究基于 DTW 算法的动态手势识别，并通过实验分析不同识别算法的性能。

4.2 基于位置的简单动态手势识别

在对静态手势识别时，可以使用数字手势识别算法，不需要设计复杂的分类器、也不需要进行模板训练，直接根据手势的结构或几何特征就能够识别出手势。这种方法简单直观、计算量小，很适合实时条件下的手势识别。因此识别动态手势时也可以考虑类似的算法，本文就利用动态手势的运动轨迹上各轨迹点的位置来进行手势识别，以下详细介绍。

在前面的章节中，我们已经分割出了动态手势，并且获取了手势的轨迹点坐标，因此可以考虑根据不同轨迹点之间的位置关系来识别动态手势，但它只能识别“right” 、“left”、“up”和“down”这 4 种位置关系较明显的简单手势。

下面以“right”手势为例来说明这种算法的思路。如图 4.1 所示：用 lengthx 表示在水平方向上运动的总距离，lengthy 表示在竖直方向的距离，x 表示任一轨迹点到下一轨迹点在 x 轴方向上的距离，可正可负，方向与 x 轴正方向一致，即下一轨迹点的 x 坐标与当前轨迹点 x 坐标的差值。识别时规定：若 lengthx>50，lengthy<30 且 x>0，则认为该手势为“right”手势。

在进行 4 种手势识别时的具体步骤为：

1．计算 lengthx 和 lengthy，若 lengthx>50 且 lengthy<30，则说明该手势可能是“right”或“left”手势中的一种，转向步骤 2 进行判断；反之，若 lengthx<30 且 lengthy>50，则说明该手势可能是“up”或“down”中的一种，转向步骤 3 进行判断；否则返回“错误”的识别结果；

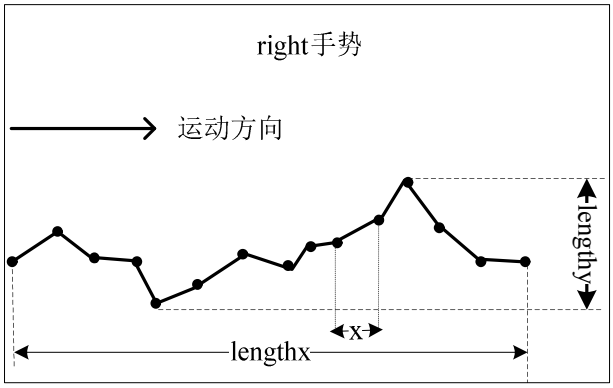


图 4.1 基于位置的算法识别 right 手势

2．计算在整个运动轨迹中 x>0 和 x<0 的次数，若前者占总次数的 4/5 以上，则认为该手势为“right”手势，若后者占总次数的 4/5 以上，则认为该手势为“left”手势，否则认为是错误的手势；

3．与步骤 2 类似，用 y 表示任一轨迹点到下一轨迹点在 y 轴方向上的距离，计算整个运动轨迹中 y>0 和 y<0 在总次数中占的比例，若前者在 4/5 以上，则识别结果为“down”手势，若后者在 4/5 以上，则识别结果为“up”手势，否则返回“错误”的识别结果。

为了测试这种算法的性能，让 10 个人每种手势做 10 个进行测试，对四种手势的识别结果如表 4.1 所示：

表 4.1 基于位置的 4 种动态手势识别结果

手势类别	right	left	up	down
正确个数	85	80	68	74
识别率	85%	80%	68%	74%
平均识别率	76.75%			

由表 4.1 可以看出：4 种手势中，“right”手势的识别率最高，“left”次之，“up”最差，但平均识别率并不高。可能原因是：手在水平方向上移动时比较自然，特别是向右移动，因此手势做的比较标准，识别率也相对较高；而在竖直方向上移动时常常由于手发生倾斜而导致识别错误，特别是向上移动时最不好把握，所以“up”手势的识别率最低；同时由于手在运动过程中的抖动或噪声的影响也会影响识别的结果，所以总体识别率并不高。

由以上分析可以看出，这种基于位置的识别方法不需要通过复杂的训练，减小了计算量；算法简单，容易实现。但是它只能识别“right” 、“left”、“up”和

“down”这四种简单的手势，对其他动态手势并不适用，因此不能进行推广，实用性差；同时由于手的灵活多变和随机性，这种算法的适应性较差，不能应对复杂多变的环境影响，因此必须寻找其他更加稳定和实用的动态手势识别算法。

4.3 DTW 识别算法研究

DTW（动态时间规整）是语音识别中的一种经典算法[53]，最早用于孤立词语音识别，解决了发音长短不同的模板匹配问题，后来逐步用在计算机的行为识别、信息检索和手势识别等领域。它能够解决动态手势持续时间不同的问题，与 HMM 算法相比，在训练中几乎不需要额外的计算，从这方面来看，它比 HMM 算法具有明显的优势。这也是本文使用 DTW 识别手势的一个重要原因。下面将详细介绍 DTW 算法。

4.3.1 DTW 算法实现原理

DTW 是一种基于动态规划(Dynamic Programming, 简称 DP)思想，对非线性时间进行归一化再模式匹配的算法[8]。基本思想是：对于两个时间范围不同的序列，将其中一个作为参考序列，另一个作为待匹配序列，寻找一个非线性时间规整函数，来调整待匹配序列的时间范围，使之尽可能与参考序列一致。

假设存在两个特征时间序列

$$A=\{a_1,a_2,\cdots,a_i,\cdots,a_I\} \text{ 和 } B=\{b_1,b_2,\cdots,b_j,\cdots,b_J\}$$

A 为手势模板的特征序列，B 为待识别手势的特征序列，其中 I 与 J 不一定相等，但 a_i 和 b_j 的维数一定相同。当 A 和 B 的时间长度不同时，按照一般的思路，首先需要将时间轴归一化，使得 A 和 B 的时间长度一致再进行匹配和识别。利用 DTW 算法计算时，事先不需要改变时间轴的长度，而是在匹配过程中动态调整时间范围。DTW 算法的思路就是寻找一个时间规整函数 C，能够将序列 B 的时间轴非线性地映射到序列 A 的时间轴上，并且失真尽可能的小。也就是说，我们要寻找一条最优路径，即一个最佳时间规整函数

$$C=\{c_1,c_2,\cdots,c_N\},(\text{N为路径长度}) \qquad \text{式(4-1)}$$

使得这条路径上所有匹配点的特征矢量之间的加权距离总和最小。

其中，$c_n=(i_n,j_n)$ 表示第 n 个匹配点，是由序列 A 的第 i 个特征矢量与序列 B 的第 j 个特征矢量组成，a_i 和 b_j 之间的距离称为局部匹配距离，记为 $d(i_n,j_n)$。

DTW 算法的加权距离和 $D(A,B)$ 和最优路径 C^* 可用公式表示为：

$$D(A,B)=\min_C\{\sum_{n=1}^{N}\omega_n d(i_n,j_n)\} \qquad \text{式(4-2)}$$

$$C^*=\arg\min_C\{\sum_{n=1}^{N}\omega_n d(i_n,j_n)\} \qquad \text{式(4-3)}$$

目标函数可表示为：

$$DTW(A,B)=\frac{\min\limits_{C}\{\sum\limits_{n=1}^{N}\omega_n d(i_n,j_n)\}}{\sum\limits_{n=1}^{N}\omega_n} \quad \text{式(4-4)}$$

其中，ω_n 是加权因子，对不同的局部匹配距离，加权因子的大小不同，从而可以区分不同局部距离的重要性。当所有局部距离的加权因子都相同时，DTW 算法的匹配距离就可以简化为：

$$DTW(A,B)=\frac{\min\limits_{C}\sum\limits_{n=1}^{N}d(i_n,j_n)}{N} \quad \text{式(4-5)}$$

即最佳路径上所有匹配点的局部距离和的平均值。

所以说，DTW 算法就是按照动态规划的思想，将一个复杂的全局优化问题转化为许多简单的局部优化问题，并按照先局部后整体的顺序进行计算。

4.3.2 路径约束条件

由 DTW 算法的实现过程可以看出：计算过程是在规整路径上进行的，因此路径的选择至关重要。在 DTW 算法中，路径的选取并不是任意的，需要增加一些约束条件，否则可能导致错误的识别结果。

DTW 对规整路径的约束分为局部约束条件和全局约束条件。局部约束包括以下 3 种类型：

1．连续性约束

由于特征序列中某些特征值可能在识别中起到关键作用，为了保证识别的正确性，必须要求时间规整函数不跳过模板序列中的任何匹配点。

即：如果时间规整函数 C 中的一个匹配点为 $c_n=(i_n,j_n)$，下一个匹配点为 $c_{n+1}=(i'_{n+1},j'_{n+1})$，那么一定满足：

$$i'=i+1 \quad \text{式(4-6)}$$

2．单调性约束

虽然不同手势序列的时间可能不同，但在时间上都是有序的，所以必须要求选取的路径不违背时间顺序，即对时间规整函数 C 中的两点 $c_n=(i_n,j_n)$ 和 $c_{n'}=(i_{n'},j_{n'})(n'>n)$，满足：

$$c_{n'}\geq c_n \quad \text{式(4-7)}$$

也即：

$$i_{n'}\geq i_n, j_{n'}\geq j_n \quad \text{式(4-8)}$$

3．端点约束

一般情况下，要求两个特征序列的起点和终点都必须为各自的第一个点和最

后一个点，即序列 A 的a_1必须和序列 B 的b_1匹配，序列 A 的a_I必须和序列 B 的b_J匹配，规整函数表示为：

$$c_1 = (1,1), c_n = (I,J) \qquad \text{式(4-9)}$$

实际中，也可以根据具体需要放宽对端点约束的限制，使起始点和终止点分别对应地在序列起点和终点的某个窗口范围内，称为端点松弛。

在搜索路径的时候，首先要满足上述 3 个约束条件，然后再根据实际需要增加其他约束条件。图 4.2 给出了一种局部路径约束。

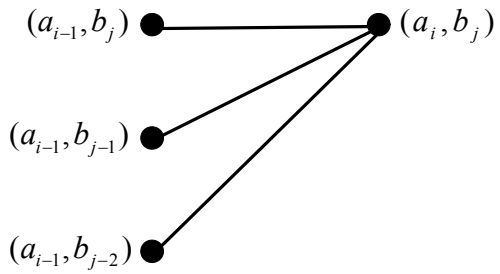


图 4.2 一种局部路径约束

在该约束条件下，利用 DTW 算法搜索最佳路径时，在路径 C*上，点(a_i,b_j)的前一格点只能是(a_{i-1},b_j)、(a_{i-1},b_{j-1})和(a_{i-1},b_{j-2})三个中的一个，然后将它们到点(a_i,b_j)的加权距离和 D 最小的点作为其前一格点。则计算过程中匹配点(a_i,b_j)的累积距离为：

$$D(a_i,b_j) = \min\begin{Bmatrix} \omega_0 d(a_i,b_j) + D(a_{i-1},b_j), \\ \omega_1 d(a_i,b_j) + D(a_{i-1},b_{j-1}), \\ \omega_2 d(a_i,b_j) + D(a_{i-1},b_{j-2}) \end{Bmatrix} \qquad \text{式(4-10)}$$

其中，ω_0、ω_1和ω_2分别为(a_{i-1},b_j)、(a_{i-1},b_{j-1})和(a_{i-1},b_{j-2})三个点对应权值，通常情况下，也可以取$\omega_0 = \omega_1 = \omega_2 = 1$，这时：

$$D(a_i,b_j) = d(a_i,b_j) + \min\{D(a_{i-1},b_j), D(a_{i-1},b_{j-1}), D(a_{i-1},b_{j-2})\} \qquad \text{式(4-11)}$$

在实际应用中，约束条件不同就可以得到不同的迭代关系，同时，在进行迭代运算时，必须给定初始条件，即开始几个匹配点的累积距离，如：

$$D(1,1) = d(1,1), \quad D(2,2) = D(1,1) + d(2,2) \qquad \text{式(4-12)}$$

按照这种计算方法，就可以将（1,1）作为起始点，(I,J)作为结束点，通过反复的递推和迭代运算直到获得最优的路径，并得到最终的累积距离$D(a_I,b_J)$。如图 4.3 所示，给出了一种搜索路径。

前面已经说过，对路径的约束出了局部约束外，还有全局约束。全局约束主要是在搜索路径时，将规整的路径限制在一定区域内，即事先约定一个子集作为最佳路径的搜索空间，常用的有带状区域、平行四边形区域等，如图 4.4 所示。

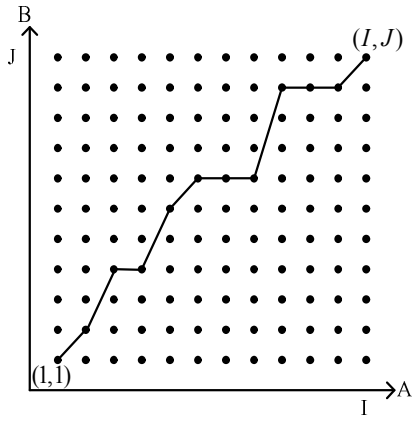
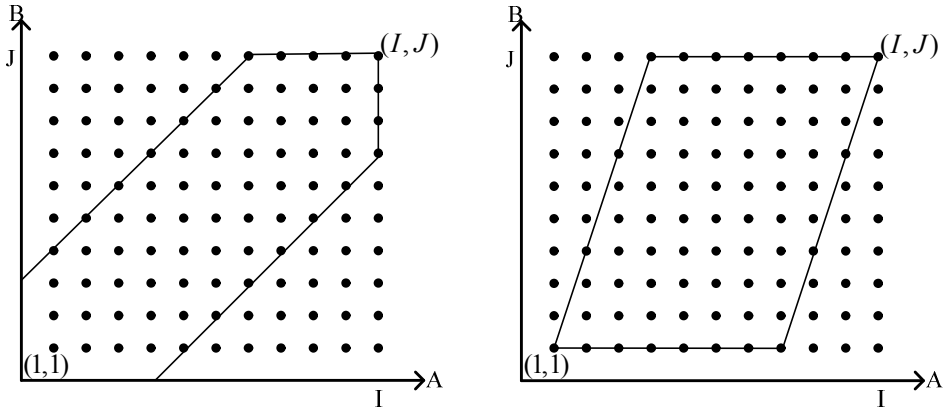


图 4.3 一种搜索路径



(a) 带状区域 (b) 平行四边形区域

图 4.4 常用全局路径约束

由于 DTW 算法需要通过反复的递推和迭代，如果搜索空间过大，搜索路径时的分支路径就会很多，将会导致运算量增大，而其中的很多分支路径在实际中是根本不可能出现的，我们却人为地增加了很多计算量。为了解决这个问题，就可以通过对搜索路径进行全局约束，减少不必要的运算量，从而也可以提高运算效率。比如在文献[8]中，作者将搜索路径限制在斜率为 1/2~2 的菱形内，既可以保证路径不过分倾斜，也可以减少手势的误匹配，同时也大大的减少计算量，提高运行效率。另外，对搜索路径上的匹配点，不需要保存每个点的匹配距离和累加距离，只需要保存路径上各点的局部匹配距离并在计算过程中不断更新即可。

除此之外，根据 DTW 算法的计算过程可知，在 X 轴正方向计算某一特征点的累积距离时，只需要知道当前点的局部距离和可能是前一匹配点的几个特征点的累积距离，因此，在实现 DTW 算法时，并不需要将搜索空间中所有点的累积距离都保存下来，而只需要保存所需要的数据，这样就可以很大程度的减少存储空间和计算量，从而提高识别效率。

4.4 基于 DTW 算法的实时动态手势识别

DTW 算法实质上也是一种模板匹配算法，因此在识别动态手势时，首先要得到手势模板序列，然后将待识别手势与模板匹配，进而识别出手势。上节已经介绍了 DTW 算法的原理及实现过程，本节将详细介绍基于 DTW 算法识别动态手势的过程。

4.4.1 模板训练

在进行模板训练时，使用手势数据库中的手势模板，并用前面介绍的手势分割和特征提取方法得到每个动态手势的特征序列，这些特征序列值均为量化后的角度特征。

由于手的灵活多变性，手势库中可能包含一些特别不标准的手势，为了减小这些手势带来的干扰，在进行 DTW 匹配计算时，首先设置一个距离匹配阈值 D_t，如果两个序列匹配计算结果不在阈值范围内，则认为待匹配的手势与模板相差较远，并将其从待匹配模板库中去除。

用 DTW 算法得到训练模板的计算过程如下：

1. 将两个特征序列按 DTW 算法计算其匹配距离 D，并与阈值比较，若 D> D_t，则认为这两个手势差别较大，将待匹配序列舍弃，返回步骤 1，继续将下一个待匹配特征序列与模板比较；若 $D \le D_t$，则按步骤 2 搜索匹配路径；

2．从最后一个匹配点开始，根据路径约束条件，在搜索空间内逐点向前搜索，将每一列上待匹配的几个点中累积距离最小的点作为最佳路径上的匹配点，直到第一个匹配点为止，这样就可以得到这两个序列的最佳匹配路径；

3．将这两个序列按照最佳路径上，逐点进行加权平均，得到一个新的手势模板序列；

4．再将下一个待匹配手势特征序列与更新后的模板比较，循环执行步骤 1~3，直到把待匹配的序列全部计算完，得到最终的特征序列模板，作为手势识别时的分类依据。

在计算过程有两个需要注意的问题：一是局部匹配距离的计算，二是路径约束。下面分别进行说明：

1．局部匹配距离

由于角度量化时采用图 3.4 所示的 12 方向链码进行量化，在计算局部匹配距离时，我们明显可以看出序列值 1 与 12 和 2 都是相邻的，但如果直接根据欧式距离来计算，得到的结果分别为 11 和 1，显然与实际情况不符，所以这里重新对距离进行定义。

假设相邻区域的距离为 1，则距离的公式可以表示为：

$$d(a_i,b_j)=\min(\left|a_i-b_j\right|,12-\left|a_i-b_j\right|) \qquad \text{式(4-13)}$$

这样，就可以计算出搜索路径上每个匹配点的局部匹配距离。

2．路径约束

在进行 DTW 模板训练时，按照图 4.2 的局部路径约束，显然该路径约束满足连续性和单调性，对于端点的约束，我尝试用两种约束条件进行计算。

(1)端点固定

仍按照前面假定的模板序列 A 和待匹配序列 B，由于 A 和 B 的长度分别为 I 和 J，则从点(1,1)开始搜索路径。

根据图 4.2 的路径约束，容易知道在最优路径不可能出现在分别过点(1,1)和点(I,J)且斜率为 2 的两条直线之外，所以搜索空间可以限制在平行四边形区域内，如图 4.5 所示（若 I 的大小不足 J 的一半，即(I,J)点不在搜索空间内，则可以调换顺序，将 B 作为匹配模板即可）：

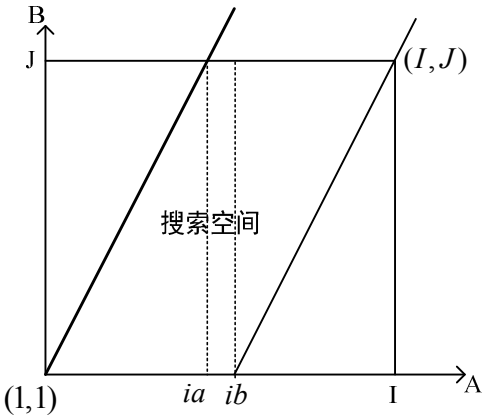


图 4.5 端点固定的全局约束

任一特征序列在与模板匹配时，首先计算 A 和 B 之间的最小加权距离和，搜索空间内每个点的累积距离可以根据公式 4.10 求出。为了突出每个待选择特征点的重要性，我将权数设为 ω_0=1、 ω_1=2 和 ω_2=3 。这样，公式变为：

$$D(a_i,b_j)=\min\begin{cases}d(a_i,b_j)+D(a_{i-1},b_j)\\2d(a_i,b_j)+D(a_{i-1},b_{j-1})\\3d(a_i,b_j)+D(a_{i-1},b_{j-2})\end{cases} \qquad \text{式(4-14)}$$

另外，初始条件为：

$$D(1,1)=\omega_0 d(1,1)$$

$$D(2,2)=D(1,1)+\omega_1 d(2,2) \qquad \text{式(4-15)}$$

由于进行了全局约束，所以在进行匹配计算时要注意边界条件，当 ia=ib 时，可以将空间分两段进行计算

$$j\in\begin{cases}[1,2i+1] & i\le ia\\[2(i-ib)+1,J] & i>ia\end{cases} \qquad \text{式(4-16)}$$

当 ia<ib 时，分三段进行计算

$$j \in \begin{cases} [1, 2i+1] & i \le ia \\ [1, J] & ia < i \le ib \\ [2(i-ib)+1, J] & i > ib \end{cases} \quad \text{式(4-17)}$$

当 ia>ib 时，也分三段进行计算，与 ia<ib 的情况类似。

这样，在计算出累积距离后，就可以从最后一个匹配点(I,J)向前逆向搜索得到最优路径。

(2)端点松弛

在手势动作开始和结束时，由于抖动或其他因素可能导致手势不稳定，将端点固定可能会使匹配不准确，所以我尝试用松弛端点，根据局部的路径约束，将第一个匹配点和最后一个匹配点都限定在包含 3 个点的窗口内，考虑到路径不可能出现在分别过点(1,3)和点(I,J-2)且斜率为 2 的两条直线之外，因此全局约束为带状区域，如图 4.6 所示。

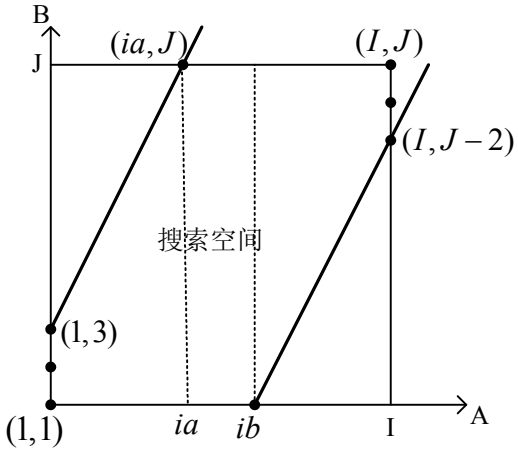


图 4.6 端点松弛的全局约束

在进行 DTW 计算时，初始条件为：

$$\begin{aligned} D(1,1) &= \omega_0 d(1,1) \\ D(1,2) &= \omega_1 d(1,2) \\ D(1,3) &= \omega_2 d(1,3) \end{aligned} \quad \text{式(4-18)}$$

按照公式(4-10)计算每个匹配点的累积距离，其中，参数设置仍为 ω_0=1、ω_1=2 和 ω_2=3，然后逆向搜索最优路径。另外在计算时，仍然根据 ia 与 ib 的大小关系将搜索空间分为三段或两端进行计算，过程与端点固定的情况类似。

4.4.2 手势识别

得到手势模板后，就可以使用 DTW 算法进行手势识别，由于是实时条件下的手势识别，动态手势的采集、分割和特征提取都按照前面章节介绍的方法进行，这里只说明对待识别手势进行特征提取后识别手势的方法。

用 DTW 算法识别动态手势的思路是：将待识别手势的特征序列分别与得到的 8 个手势模板进行 DTW 匹配计算，并将其中匹配距离最小的模板所代表的的手势作为识别结果。为了减小计算量，与模板训练一样，首先设定一个匹配距离阈值 D_t，在计算过程中，若某一个匹配距离大于阈值，则认为待识别手势与该模板代表的手势类别不同，继续计算下一个。

具体的识别过程如下：

1．按照 DTW 算法，将待识别手势的特征序列与 8 个手势模板中的某一个进行匹配计算，若匹配距离 D> D_t，则认为待识别手势与该模板所表示的手势差别较大，必定不能作为识别结果，是无效的，舍弃该匹配距离，返回步骤 1，继续与下一个模板比较；若 $D \le D_t$，则保存该匹配距离，并将该匹配距离作为新的距离阈值 D_t，按步骤 2 进行；

2．反复执行步骤 1，将待识别手势与 8 个手势模板序列依次进行匹配，并与更新后的距离阈值进行比较，直到与所有的模板都匹配结束，那么，最终阈值即为最小匹配距离，也就是说，记录与待识别手势的匹配距离为最终阈值的模板，该模板所代表的的手势为识别结果。

需要注意的是：DTW 计算过程中的初始条件、参数的设置等都与模板训练时相同，这里不再具体说明；如果所有的匹配距离都比初始阈值大，那么可能是因为初始阈值设置的太小，应当修改初始阈值，如果初始阈值的设置没有问题，则说明待识别的手势与所有的模板都不匹配，该手势不是模板库中的手势。

从模板训练和手势识别的过程可以看出：在计算过程中，如果将待匹配特征序列与模板序列依次运算，每次匹配都要进行一次完整 DTW 计算，运算量是很大的。为了提高运算效率，事先设置了一个距离阈值，当两个特征序列在进行 DTW 计算时，首先将匹配距离与阈值比较，如果比阈值大，则继续执行下一个。这样可以通过设置阈值来省去一些无效的计算，从而提高计算的效率。

4.4.3 识别结果与分析

为了测试 DTW 算法对 8 种自定义手势的识别性能，首先用手势数据库中的手势进行测试，并根据端点固定和端点松弛两种约束条件分别进行。

1. 测试端点固定的情况

在手势数据库每种手势选取 20 个作为训练数据，得到手势模板，然后分别用这 20 个和其余 80 个作测试数据，识别结果如表 4.2 和表 4.3 所示。表 4.2 是用训练数据的 20 个手势序列作为测试数据的识别结果，表 4.3 是其余 80 个手势序列作测试数据的识别结果。

表 4.2 端点固定时测试结果一

手势类别	right	left	up	down	V	vdown	triangle	circle
正确个数	20	20	19	20	20	20	20	19
正确识别率	100%	100%	95%	100%	100%	100%	100%	95%
平均识别率	98.75%							

表 4.3 端点固定时测试结果二

手势类别	right	left	up	down	V	vdown	triangle	circle
正确个数	80	79	78	79	80	79	80	76
正确识别率	100%	98.75%	97.5%	98.75%	100%	98.75%	100%	94%
平均识别率	98.43%							

2. 测试端点松弛的情况

与端点固定时的测试方法一样，仍在数据库中每种手势选 20 个作为训练数据，再分两组进行测试。表 4.4 是把作训练数据的 20 个手势序列作为测试数据的识别结果，表 4.5 是其余 80 个手势序列作测试数据的识别结果。

表 4.4 端点松弛时测试结果一

手势类别	right	left	up	down	V	vdown	triangle	circle
正确个数	15	20	8	20	17	20	1	20
正确识别率	75%	100%	40%	100%	85%	100%	5%	100%
平均识别率	75.63%							

表 4.5 端点松弛时测试结果二

手势类别	right	left	up	down	V	vdown	triangle	circle
正确个数	32	4	33	80	27	3	0	78
正确识别率	40%	5%	41.25%	100%	33.75%	3.75%	0%	97.5%
平均识别率	40.16%							

由测试结果可以看出：

(1)端点固定时：两组测试的平均识别率都在 98%以上，说明这些手势模板能够作为手势分类的依据，用于动态手势识别。

(2)端点松弛时：测试结果一的识别率比测试结果二的识别率高的多，但与端点固定时相比，识别结果严重下降，甚至是错误的结果；出现这种情况的原因可能是：这些动态手势序列在进行匹配时，由于匹配的端点值位于一个滑动窗口内，可能使相似的手势相互混淆，再加上动态手势起始点的不稳定性，导致出现错误的识别结果，另一个原因可能是数据库较小，得到的手势模板存在较大误差。所以端点松弛得到的手势模板不能作为手势识别的依据，在实时识别动态手势时仍按端点固定得到的模板来计算。

以下用端点固定时得到的 8 个手势模板来识别实时条件下的动态手势，一共 10 个人，每人每个手势做 10 次，实验结果如表 4.6 所示：

表 4.6 基于 DTW 算法的动态手势识别结果

手势类别		right	left	up	down	V	vdown	triangle	circle
识别结果	right	98							
	left		96						
	up			92		2	2		3
	down				96	2	3	2	
	V					96			
	vdown						95		10
	triangle							98	
	circle								85
	error	2	4	8	4				2
正确识别率		98%	96	92%	96%	96%	95%	98%	85%
平均识别率		94.5%							

由实验结果可以看出：

(1)8 种手势的正确识别率都比用手势数据库中特征序列测试时下降了，这是因为手势库中的特征序列是提前计算出的，而实时的手势还需要经过分割、特征提取等步骤，受环境因素影响较大；

(2)在“right”、“left”、“up”和“down”四种手势中，“right”手势的识别率最高，这是由于人的习惯，“right”手势比其它三种手势做起来更自然，做出的手势就会更标准，识别率最高；

(3)这 8 种手势中，“circle”手势的识别率最低，可能是因为该手势包含曲线，而提取特征时只考虑了相邻轨迹点的角度特征，没有考虑距离，导致识别不准确；

(4)平均识别率为 94.5% ，每个手势的识别率都在 85%以上，说明该算法能够实时识别出动态手势，满足实际的需要。

4.5 本章小结

本章主要研究动态手势的识别算法，首先介绍了基于位置的简单动态手势识别算法，这种算法不需要复杂的模板训练和匹配，计算简单直观，但只能用于四种简单手势的识别，并且经过实验发现识别率并不高，不能应对复杂多变的环境变化；然后详细介绍了 DTW 算法识别手势的原理及模板训练和识别手势的过程，这种算法能够很好地解决不同手势的时间长度不同的问题。实验表明：DTW 算法识别文中 8 种自定义手势时平均识别率达到 94%以上，可以满足实际的需要。

第五章 动态手势识别及应用系统的实现

5.1 引言

研究手势识别的最终目的是为了实际应用，以便能够更好的为人类服务。由于手势具有简单、直接、自然的特点，因此更容易被人们所接受。作为人机交互中一种关键技术，目前手势识别能够广泛应用于手语识别、机器人控制和生活娱乐等领域，未来还会有更多更大范围内的应用。随着各种交互设备的出现，也为手势识别的研究带来了前所未有的发展前景。

本文将动态手势识别技术用于控制 PPT 操作，设计了一个完整的系统，可以实现动态手势识别和手势控制 PPT 两大功能，在给予用户一些极少限制的条件下，能够实时识别出自定义的 8 种动态手势，并控制 PPT 的打开、运行、翻页和关闭等，在一定程度上摆脱了鼠标和键盘的束缚，达到了人机交互的目的。

本章将详细介绍该系统的开发环境、实现流程及操作方法等，并对系统性能进行分析。

5.2 系统环境及开发流程

本系统是在 PC 机上实现的，电脑型号为 Lenovo Z475，Win7（32 位）操作系统，处理器为 AMD A6-3400M，4G 内存，编译环境为 Visual Studio 2010，编程语言为 C/C++。采集手势图像序列时使用微软的 Kinect 传感器，并使用 PrimeSense 公司的 OpenNI 软件开发包及 OpenCV 库。

该系统主要有动态手势识别和手势控制 PPT 两大功能，它能够实时地识别出自定义的 8 种动态手势：“right”、“left”、“up”、“down”、“V”、“vdown”、“triangle”和“circle”（已在第二章介绍过），并将这些手势转化为控制指令实时控制 PPT 操作。在对用户的手势动作给予一些限制的条件下，可以实现友好、自然的人机交互。

在控制 PPT 操作时，这 8 种动态手势对应的功能如表 5.1 所示：

表 5.1 自定义 8 种动态手势对应的功能

手势类别	right	left	up	down	V	vdown	circle	triangle
控制 ppt 的功能	向下翻动 ppt	向上翻动 ppt	翻到 ppt 第一页	翻到最后一页	运行 ppt	停止运行 ppt	关闭 ppt	打开 ppt

如图 5.1 所示，给出了该系统的完整流程图，系统实现可以分为 3 个阶段：手

势准备阶段、手势执行阶段和手势应用阶段。

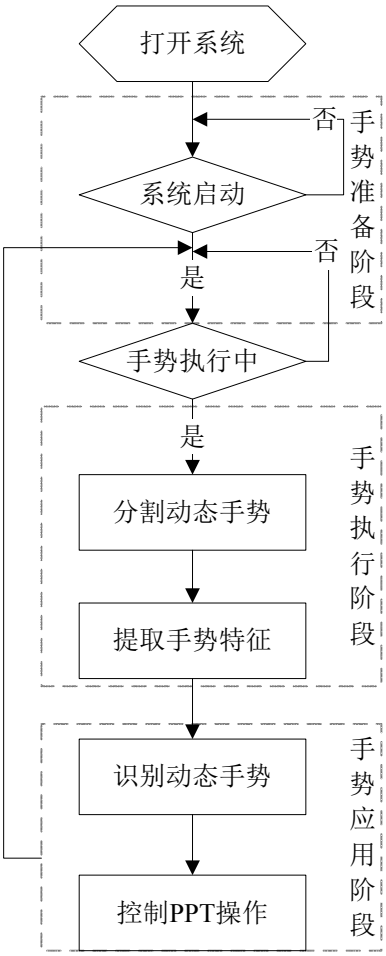


图 5.1 手势识别及应用系统流程图

1. 手势准备阶段

做手势之前，需要判断系统是否已开启，如果已开启，则可以开始做手势，进入手势执行阶段；如果未开启，则首先要开启系统，系统开启的方式有多种，本系统中使用一种常用的开启方式，即正对摄像头做挥手动作，若在 50 帧内往返次数达到三次，则认为系统开启成功，可以开始做动态手势，否则重新开启系统。

2. 手势执行阶段

进入手势执行阶段后，首先确定动态手势的起始点，按照第二章介绍的动态手势分割算法分割出动态手势序列，直到手势结束，然后提取手势序列的轨迹特征，得到特征序列。此时手势执行阶段结束，将进入手势的识别。

3. 手势应用阶段

手势动作完成后，将按照 DTW 算法与手势模板进行比较，得到识别结果。如果要实现的是手势识别功能，则直接将识别结果输出；如果要实现的是手势控制 PPT 功能，则再将识别出的手势转化为控制指令，控制 PPT 进行相应的操作。

一个手势动作的功能实现后，可以再次开始准备做下一个手势，并按照上述过程继续执行，直到停止使用该系统，关闭即可。

5.3 系统操作过程及功能实现

上节对该系统的开发环境和整体框架进行了简单介绍，本节将对使用该系统时的具体操作进行详细说明。文中首先介绍系统界面及操作注意事项，然后详细说明操作步骤的具体过程。

利用该系统实现实时动态手势识别和手势控制 PPT 两大功能时，需要首先打开系统，这时，将进入系统的初始界面，如图 5.2 所示：

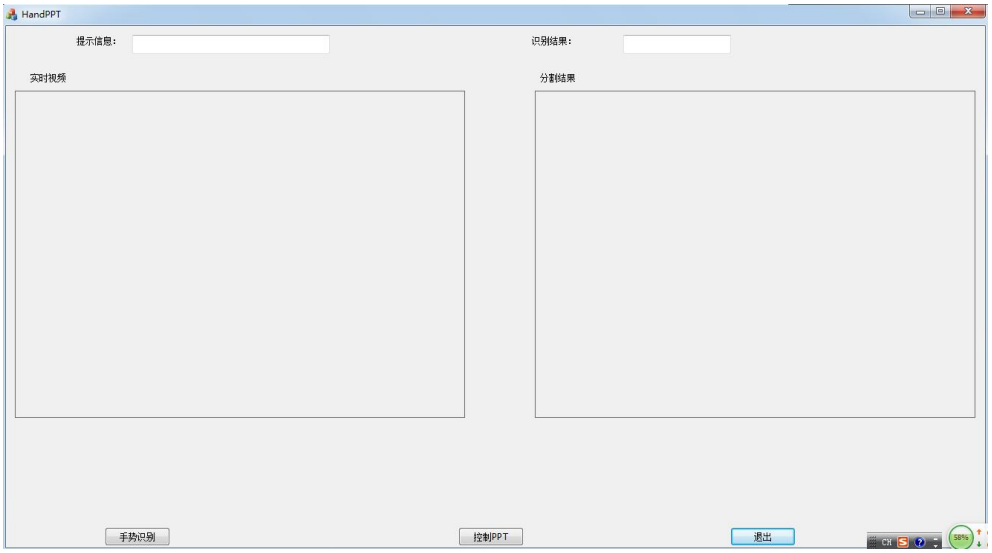


图 5.2 系统初始界面

界面上各按键的功能如下：

最上方的“提示信息”和“识别结果”栏中将显示手势操作过程中手势状态、识别状态及 PPT 状态；“实时视频”和“分割结果”图片框中分别显示手势识别过程中的实时彩色视频流和分割出的手势序列；“手势识别”按钮控制手势识别系统的开启，“控制 PPT”按钮控制 PPT 操作系统的开启，“退出”按钮结束软件的运行。

需要说明的是，完成识别手势或控制 PPT 中一个功能后，实现另一个功能时，需要退出软件，重新启动。

另外，为了使该软件的性能达到最佳，对做手势的用户作如下限制：

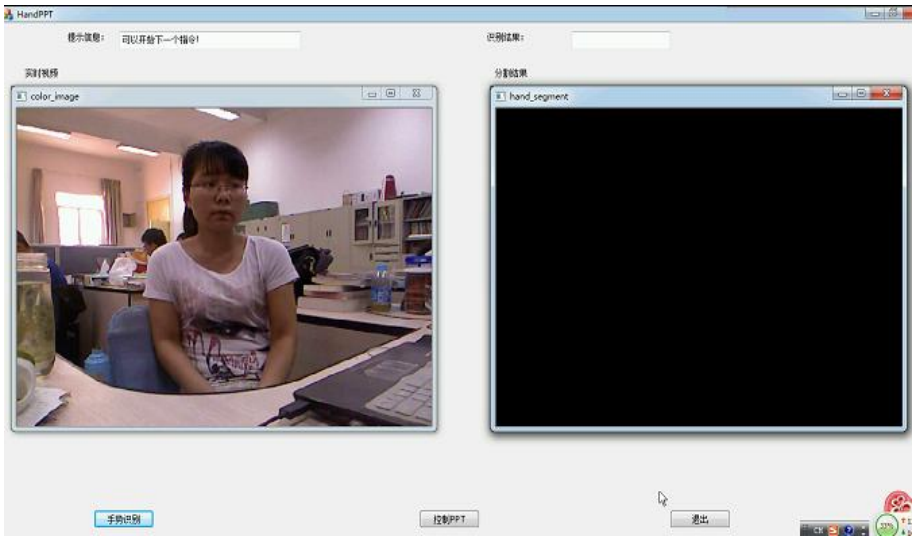
1．手与传感器之间不能有遮挡物，手要位于身体的前端，并与身体保持尽可能大的间隔；

2．用户的手必须要正对 kinect 传感器的摄像头，作手势时尽量立起手部，，做手势时手部与 kinect 摄像头平面的垂直距离为 50cm 到 80cm 之间；

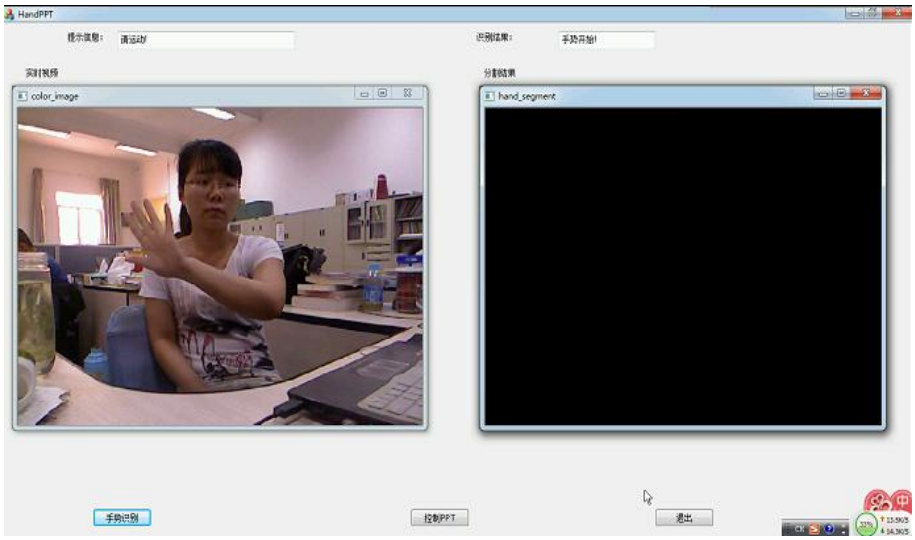
3．某一时刻只能有一只手操作，做完一个手势后立即放下手部，然后重新抬起手部至视野中心做下一个手势。

在了解了系统界面和相关注意事项后，下面分别对手势识别和控制 PPT 两大功能的具体操作步骤进行说明：

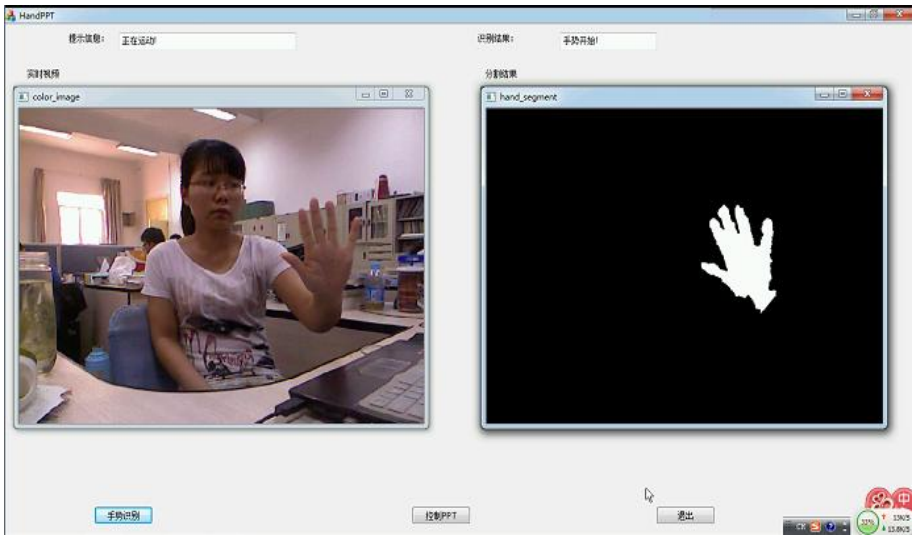
1.“手势识别”系统



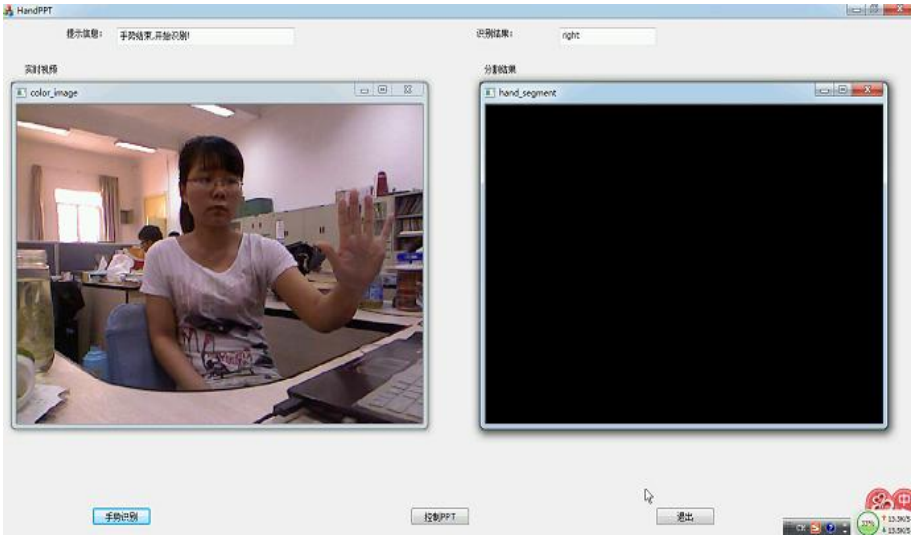
(a)准备做手势



(b)手势动作开始



(c)手势运动过程中



(d)手势结束

图 5.3 动态手势运行过程

打开系统界面后，点击“手势识别”按钮，将打开手势识别的系统，具体操作如下：

(1)打开系统后，“提示信息”栏中将出现“请做挥手动作，开启系统!”，这时，举起手部正对 Kinect 摄像头做挥手动作，在这个过程中，可能会出现“距离摄像头太近，请远一点!”或“距离摄像头太远，请近一点!”的提示信息，只需要根据提示信息相应的调整手部距离即可，当出现“开启系统成功，可以开始做手势!”时，放下手部，准备开始做动态手势，如果出现“开启系统失败，请重新做挥手动作!”，再次举起手部重新挥手，直到开启系统成功为止；

(2)开启系统后，“提示信息”栏中出现“请开始做手势”或“可以开始下一个指令”，这时举起手部准备做手势。手部立起，正对 Kinect 摄像头，并保持静止，当“识别结果”栏显示“开始运动”或“提示信息”栏显示“请运动”时，手部开始运动，做出定义的 8 种动态手势的任一种，“提示信息”栏将会显示“正在运动”，手势动作做完后，使手部再次保持静止，“提示信息”栏会显示“手势结束正在识别”或“手势失败，请重新开始!”，若手势成功，“识别结果”栏将会显示手势的识别结果，然后立即放下手部，准备开始下一个手势，整个过程如图 5.3 所示。

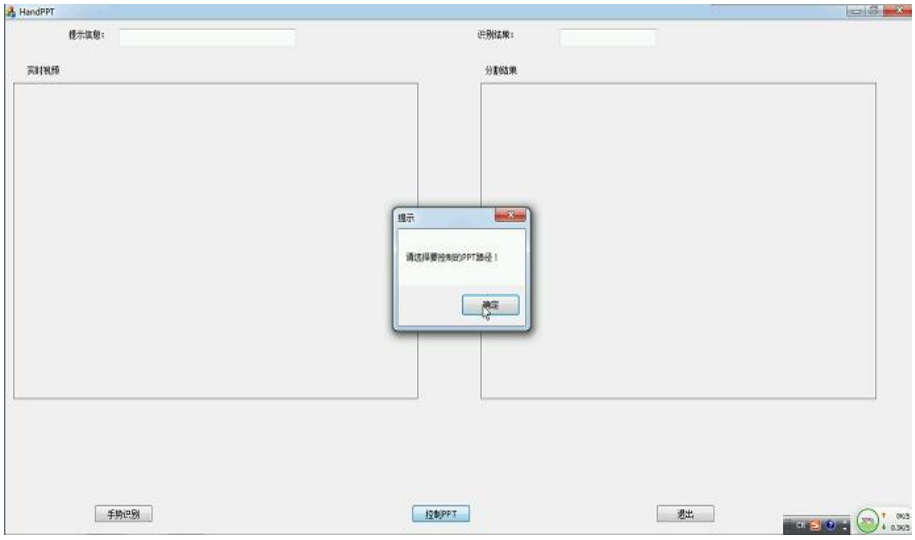
(3)手势做完后，按“退出”按钮退出系统。

2.“手势控制 PPT”系统

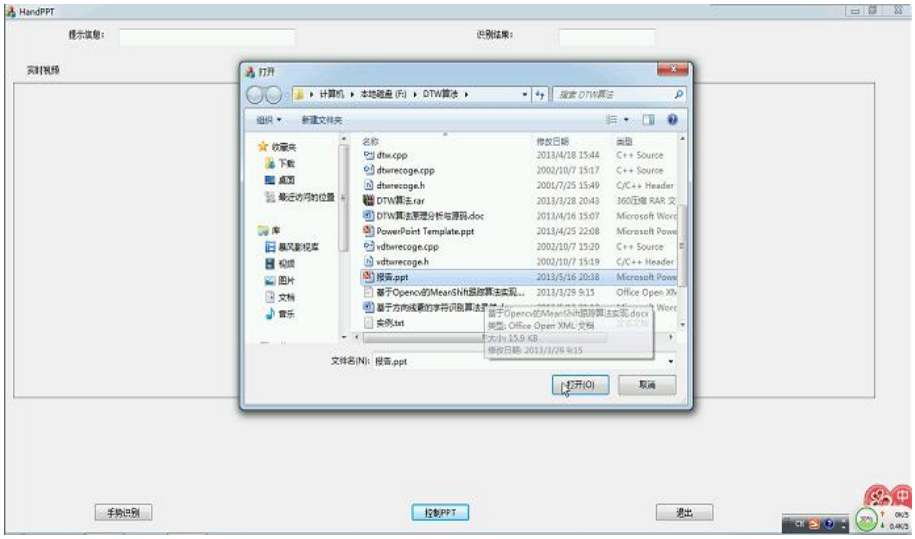
打开系统界面后，点击“控制 PPT”按钮，将开启手势控制 PPT 的应用系统，具体步骤如下：

(1)打开系统后，将弹出选择 PPT 的对话框，点击“确定”，然后选择要控制的 PPT（注意这里使用的是 03 版 PPT）；

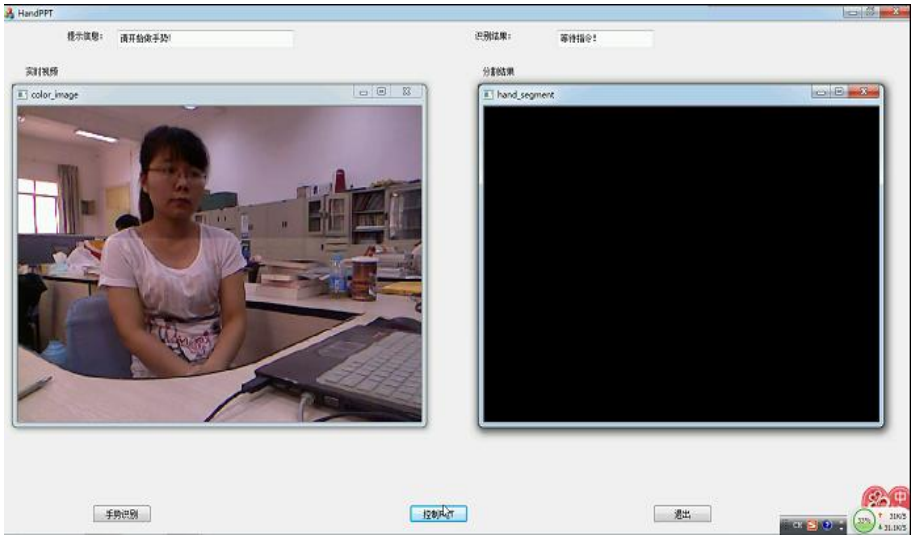
(2)选择好要控制的 PPT 后，就可以做手势来控制 PPT，开启系统和做手势的方法都与“手势识别”系统相同，“提示信息”栏和“识别结果”栏会显示一些信息，用户做出相应的手势控制 PPT 即可。具体操作为：开启系统后，首先做手势“triangle”打开 ppt，这时彩色视频和分割出的手势框会消失，只留下 PPT 界面和提示信息栏，然后做手势“V”运行 PPT，这时，可以用手势“right”向下翻页，用手势“left”向上翻页，如果想直接翻到第一页或最后一页，可以直接做手势“up”或“down”，若暂时不需要有操作时，可以做手势“vdown”使 ppt 停止运行，需要使用时再运行 ppt 即可，若不再使用 ppt，可以做手势“circle”将其关闭。如果在对 ppt 操作过程中，手势做的不合适，系统也会给出提示信息，比如：ppt 没有运行时，做翻页的手势，就会给出提示信息“请先运行 ppt”；如果已经是在最后一页，又做出“down”手势，系统就会给出提示信息“这已经是最后一页”。整个过程如图 5.4 所示。



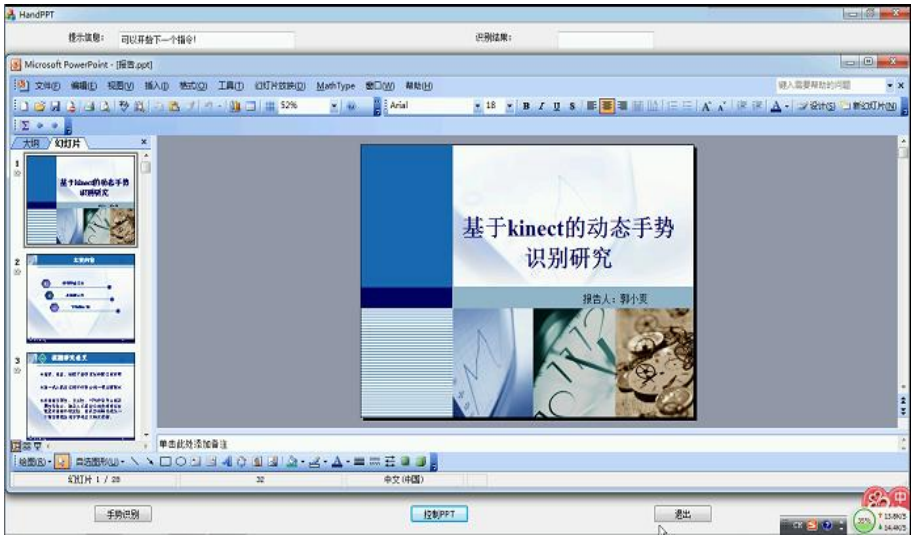
(a)ppt 路径选择对话框



(b)选择要控制的 ppt



(c)准备做手势来控制 ppt

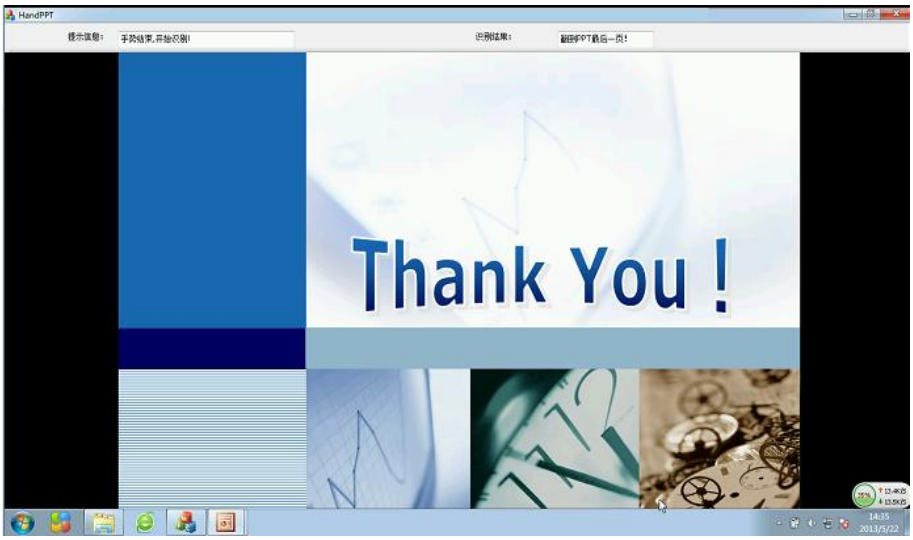


(d)打开 ppt（手势 triangle）

(e)ppt 翻页前要先运行

(f)运行 ppt（手势 V）

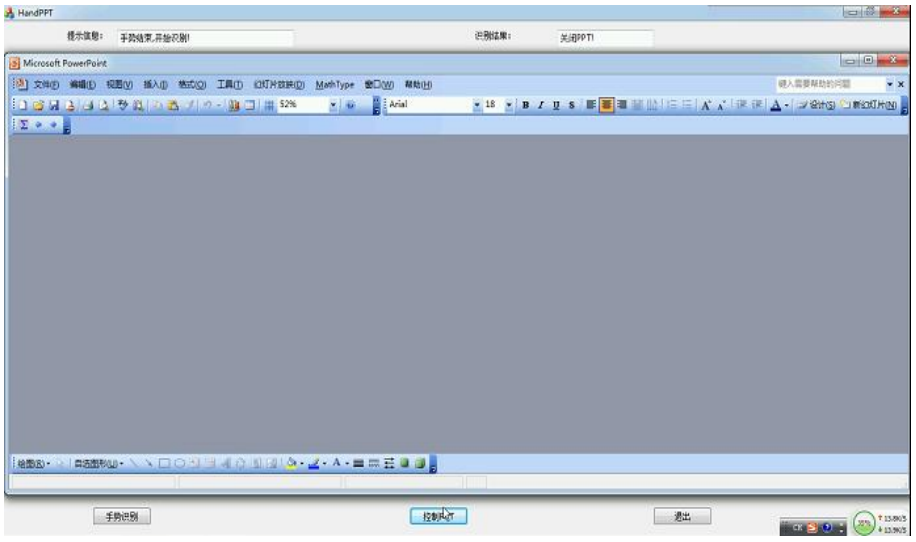
(g)向下翻页（手势 right）



(h)翻到最后一页（手势 down）

(i)向上翻页（手势 left）

(j)翻到首页（手势 up）



(k)关闭 ppt（手势 circle）

图 5.4 手势控制 ppt 的过程

(3)手势做完后，按“退出”按钮退出系统。

这样，根据以上操作要求及操作过程，就可以利用该系统实现实时条件动态手势识别和手势控制 PPT 两大功能。

5.4 系统性能测试

为了测试该系统的性能，针对系统的两种功能，请实验室 10 位同学分别进行测试。

在测试手势识别功能时，对光线和背景条件均无限制，每个同学每种手势做 10 次，实验时一些图像序列如图 5.5 所示。可以看出，这些图像序列是在不同时间、不同光线及不同背景下进行采集的。

(a)“right”手势

(b)“left”手势

(c)“up”手势

(d)“down”手势

(e)“V”手势

(f)“vdown”手势

(g) “triangle”手势

(h)“circle”手势

图 5.5 系统性能测试的手势图像序列

识别结果如表 5.2 所示：

表 5.2 系统识别手势的实验结果

手势类别		right	left	up	down	V	vdown	triangle	circle
识别结果	right	98							
	left		96						
	up			91		3	2		4
	down				93	1	3	3	
	V					96			
	vdown						95		10
	triangle								
	circle							97	83
	error	2	4	9	7				3
正确识别率		98%	96%	91%	93%	96%	95%	97%	83%
平均识别率		93.63%							

实验结果表明：在对光线、背景等都无限制时，8 种手势的平均识别率在 93.63%，系统平均响应时间也较短，说明该系统可以快速准确地识别动态手势。

另外，在识别出动态手势后，对手势控制 PPT 的功能也进行了测试，在控制 PPT 时用户等待时间较短，可以满足实际需要。

5.5 本章小结

本章主要介绍一个完整的动态手势识别及应用系统，该系统能够完成实时条件下的动态手势识别和手势控制 PPT 两大功能。在控制 PPT 时，通过约定自定义的 8 种动态手势的含义，可以实现 PPT 的打开、运行、翻页和关闭等功能。文中对该系统的开发环境、系统流程及操作步骤都进行了说明，并根据实验结果对系统性能进行分析，可以看出，该系统可以达到我们预期的效果，满足实际系统的需要。

第六章 总结与展望

6.1 本文总结

手势识别是一种新兴的人机交互方式，根据研究对象不同可以分为静态手势识别和动态手势识别。本文在前人研究的基础上，主要研究人机交互中的动态手势识别及应用，所做工作总结如下：

1．在手势采集方面

本文自定义了 8 种动态手势，使用微软的 Kinect 传感器，并结合 PrimeSense 公司的 OpenNI 来采集手势，建立了手势数据库，可以为手势识别提供研究数据。

2．在手势分割方面

通过对目前常用动态手势分割算法比较，本文在静态手势分割基础上，提出了多信息融合与帧间差分相结合的手势分割算法。首先通过两次静止法来确定手势起止帧，在确定手势的开始帧后，用帧间差分来检测每帧运动手势，直到手势结束；然后利用 Kinect 提供的深度信息将每帧运动手势图像分割出来；最后再用肤色判断分割出的是否为手。实验表明：这种算法在实时分割动态手势时能够得到完整的动态手势序列，并且每一帧都能得到完整的手形；它受光照、肤色和背景等影响较小，对环境有较强的适应性。但是这种分割算法得到的手势可能包含手臂，本文考虑到动态手势与静态手势的区别，在原来基于宽度的手臂去除方法上进行改进，能够快速、有效地去除手臂，最终得到完整的动态手势序列。

3．在特征提取方面

本文基于动态手势运动轨迹模型进行建模，主要提取了运动轨迹特征。经过分析，相邻轨迹点的方向特征可以反映动态手势的运动状态，但考虑到计算的效率和本文自定义手势的特点，只提取了相邻轨迹点之间的角度特征，并以此作为计算模型参数的依据。由于这些角度值都是连续的，并且包含很多冗余信息，不能直接用于模板训练和匹配，因此采用 12 方向链码进行量化编码，将角度特征序列变换为离散数字序列，用于模板匹配与识别。

4．在手势识别方面

本文首先探索了基于位置的简单动态手势识别算法，该算法不需要复杂的模板匹配和训练，只根据手势运动轨迹上轨迹点之间的位置关系来识别手势，但是只能识别“right”、“left”、“up”和“down”四种手势，并且受环境和噪声影响较大，平均识别率只有 76.75%；为了得到稳定、高效的识别效果，本文重点研究了 DTW 算法，它与 HMM 相比，在训练期间不需要太多数据，并且能够解决不同手势之间的时间差异性问题。实验表明，DTW 算法在识别自定义的 8 种动态手势时

平均识别率为 94.5%，手势识别效果较好。

5. 在系统实现方面

本文搭建了一个完整的动态手势识别及应用系统，可以实现动态手势识别和手势控制 PPT 两大功能，文中对该系统的环境、工作流程及操作过程等都做了详细介绍，经过测试表明，该系统能过达到预期的效果，满足实际需要。

6.2 工作展望

手势识别是一种多学科、多领域交叉的技术，需要研究的问题很多，应用领域也很广泛，针对不同场景也可以有不同的研究方法。本文只是针对动态手势识别的一个方面进行了研究，也取得了一定成果，但是目前本文工作已接近尾声，还有很多不足和需要改进的地方，主要表现在以下几个方面：

1．本文定义的手势种类较少并且数据库较小，以后可以考虑定义更多种类的手势进行识别，并且采集更多手势扩充数据库。

2．本文设计的系统虽然可以在实时条件下识别动态手势识别和控制 PPT，但有时运行不太流畅或分割手势时较敏感，以后可以对分割算法进行改进，提高系统性能；同时该系统界面也较简单，以后可以对界面进行改进。

3．本文研究的主要是动态手势识别，而在实际场景中往往需要动静结合，所以以后可以考虑将动态手势和静态手势结合在一起进行研究，从而更加自然友好地进行人机交互。

4．手势识别可以应用于很多领域，本文只是用于控制 PPT，以后可以考虑其他应用领域的研究或将手势识别与其他技术结合起来，实现更多价值。比如：用于智能家居领域，用户不需要移动位置，只挥动手势就可以控制家电的操作、音乐的播放以及电脑的开启等；手势识别技术与增强现实相结合，实现虚拟控制等。

致 谢

时光飞逝，转瞬之间我的研究生生活就要结束了，在这两年多的时间里，我学到了很多知识，也成长了很多。在此论文完成之际，感谢所有陪伴我、支持我和在论文完成过程中给我帮助的人。

首先，衷心感谢我的导师卢朝阳教授。感谢他两年多以来给予我的指导和关心，在论文写作中给我提供的帮助和意见。卢老师知识渊博、阅历丰富，对待学术认真严谨、不容许有一点瑕疵，在工作上认真负责、公正无私，这些都给我留下深刻印象，使我终生受益。他不仅是我学术上的导师，更是我人生的导师。

其次，感谢李静老师。我们在科研中遇到什么问题时，李老师总是认真帮我们分析，耐心给我们讲解；在论文完成过程，李老师也悉心指导我们写作；李老师和蔼可亲，平易近人，是我们的良师益友，感谢她给我的所有帮助和指导。

感谢实验室所有的博士师兄和师姐们，在科研过程中，他们起到了良好地带头作用，使我能够迅速投身到学术研究中；也感谢实验室其他所有同学，大家在科研中一起学习、一起讨论，在生活中，一起聊天、一起游玩，是他们给我带来了很多快乐，陪伴我渡过研究生生活；也感谢其他所有关心、鼓励和支持我的老师、同学和朋友，他们是我一辈子的财富。

由衷地感谢我的父母和亲人，感谢他们一直以来对我的关心和支持，是他们给了我强大的精神动力，没有他们，我难以取得今天的成绩。

感谢本文所涉及的到的所有学者。在此论文完成过程中，我参考了很多资料，引用了很多文献，如果没有他们的研究成果和启发，我很难完成本文的写作。

最后，向审阅本文和参加论文答辩的各位专家老师们表示深深的谢意。

参考文献

[1] V. Pavlovic, R. Sharma,T. Huang. Visual Interpretation of Hand Gestures for Human-Computer Interaction[J]. IEEE Transition on Pattern Analysis and Machine Intelligence. 1997, 1, 9(7): 677-695

[2] http://baike.baidu.com/link?url=acX2P7aPeaqIEyZOUuFOHfyYFMiptnLRMk2ftV4CLNv_2c03ksFjlS88DolmAlXCd_3IvpEJV-KqPh7TM_6v87QmO9PjzeLnNpSth3JE0LRuEMp6eDw2N7Ne3zzdLa-T. [2013 available]

[3] Y. Wu, T. Huan. Vision-based Gesture Recognition[A]. France:Proceedings of the International Gesture Recognition Workshop. l999: 103-115

[4] 王云飞. 动态手势识别中关键技术的研究[D]. 四川师范大学. 2011, 5

[5] 汤益军, 徐洁, 方志刚. 嵌入式便携手语识别手套的设计[A]. 第九届全国信息获取与处理学术会议论文集. 2011, 8

[6] http://xmwb.news365.com.cn/xz/201204/t20120413_360661.html. 消防机器人, 灭火当先锋. [2013 available]

[7] http://www.qhnews.com/newscenter/system/2013/04/05/011056179.shtml. Kinect: 体感技术的全新应用. [2013 available]

[8] 邹洪. 实时动态手势识别关键技术研究[D]. 华南理工大学. 2011, 5

[9] G. J. Grimes. Digital Data Entry Glove Interface Device [P]. Technical Report US Patent 4-414-537. 1983, 11

[10] C. Lee, Y. Xu. Online Interactive Learning of Gestures Interfaces [A]. Proceeding of IEEE International Conference on Robotics and Automation. 1996, 3(1): 30-42

[11] Mohammed, Waleed, Kadous. Machine Recognition of Auslan Signs Using PowerGloves: Towards Large-lexicon Recognition of Sign Language [A]. Proceedings of the Workshop on the Integration of Gesture in Language and Speech, October 1996: 165-174

[12] Liang R-H, Ouhyoung M. A Real-time Countinus Alphabetic Sign Language to Speech Conversion VR System. Computer GRAPHICS Forum. 1995, 14(3): 67-77

[13] Liang R-H. Ouhyoung M. A Sign Language Recognition System Using Hidden Markov Model and Context Sensitive Search. Proceedings of the ACM Symposium on Virtual Reality Software and Technology. Hong Kong. 1996: 59-66

[14] 吴江琴, 高文, 陈熙霖. 基于数据手套输入的汉语手指字母的识别[J]. 模式识

别与人工智能. 1999, 3, 12(1): 74-78

[15] Starner, Pentland. A Real-time American Sign Language Recognition from Video UsingHiddenMarkovModels.URL:ftp://whitechapel.media.mit.edu/pub/tech-reports/TR-375.ps.Z. 1996, 8

[16] Kirsti Grobel, Marcell Assam. Isolated Sign Language Recognition Using Hidden Markov Models [A]. Proceedings of the IEEE International Conference on Man and Cybernetics, Orlando. 1997: 162-167

[17] 祝远新, 徐光祐, 黄浴. 基于表观的动态孤立手势识别[J]. 软件学报. 2000, 11(1): 54-61

[18] 任海兵, 祝远新, 徐光祐. 等. 连续动态手势的时空表观建模及识别[J]. 计算机学报. 2000, 8, 23(8): 824-828

[19] C.Vogler, D.Metaxas. Adapting Hidden Markov Models for ASL Recognition by Using Three-dimensional Computer Vision Methods[A]. SMC'97: 156-121

[20] 梅萍华. 复杂背景下的指尖检测方法研究[D]. 中国科学技术大学. 2010, 5

[21] Yi Li. Hand Gesture Recognition Using Kinect[J]. Proceedings of 2012 IEEE Third International Conference on Software Engineering and Service Science(ICSESS 2012). 391(41): 196-199

[22] Matthew Tang. Recognizing Hand Gestures with Microsoft's Kinect. IEEE Computer Society Press. 2011, 3

[23] 罗元, 谢彧, 张毅. 基于 Kinect 传感器的智能轮椅手势控制系统的设计与实现[J]. 机器人. 2012, 1, 34(1): 110-113

[24] Zhang Yi, Zhang Shuo, Luo Yuan, etal. Gesture Track Recognition Based on Kinect Depth Image Information and its Appliactions[J]. Application Research of Computers. 2012, 9, 29(9): 3547-3550

[25] 王莹. 基于 Kinect 的 Tri-tracking 视频跟踪算法研究[D]. 大连理工大学.2012,6

[26] 高静雅. 人机交互中的手势识别技术研究[D]. 西安电子科技大学. 2013, 1

[27] 任海兵, 祝远新, 徐光浴, 等. 基于视觉手势识别的研究综述[J]. 电子学报. 2000, 2, 28(2): 118-121

[28] M. J. Jones, J. M. Ring . Statistical Color Models with Application to Skin Detection[J] . International Journal of Computer Vision . 2002. 46(1): 8l-96

[29] Y. Fang, K. Wang, J. Cheng eta1 . A Real-time Hand Gesture Recognition Method . IEEE International Conference on Multimedia and Expo. 2007: 995-998

[30] W. T. Freeman, C. Weissman. Television Control by Hand Gesture . IEEE International WorkShop on Automatic Face and Gesture Recognition. Zurich,Switzerland . 1995: 179-183

[31] J. Letessier, F. Berard . Visual Tracking of Bare Fingers for Interactive Surfaces . In UIST l7th Annual ACM Symposium on User Interface Software and Technology New York. NY,USA:ACM Press. 2004: 119-122

[32] J.Canny. A Computational Approach to Edge Detection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, 11, 8(6): 179-185

[33] J.Canny. A Computational Approach to Edge Detection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, 11, 8(6): 179-185

[34] Wu Xiaoyu, Yang Cheng, Wang Youwen, eatl. An Intelligent Interactive System Based on Hand Gesture Recognition Algorithm and Kinect[A]. 2012 Fifth International Symposium on Computational Intelligence and Design. 2012: 294-298

[35] 刘云, 孙玉, 刘继超. 基于 Camshift 和时序模板轨迹的动态手势跟踪与识别[C]. 中国电子学会第十五届信息论学术年会暨第一届全国网络编码学术年会. 2011, 7: 110-113

[36] 张博洋. 复杂背景下的手势分割与轨迹识别研究[D]. 山东大学. 2003, 5

[37] J. L. Hernandez-Rebollar. A New Instrumented Approach for Translating American Sign Language into Sound and Text[C]. IEEE International Conference on Automatic Face and Gesture Recognition, Korean. 2004: 547-552

[38] Luca Zanni, Thomas Serafini. Parallel Software for Training Large Seale Support Vector Machines[J] . Journal of Machine Learning Research. 2006, 7: 1467-1492

[39] 徐波, 余劲松, 李行善. 基于路径约束的动态时间规整方法研究[J]. 系统工程与电子技术. 2004, 1, 26(1): 103-105

[40] 贺顾一. 复杂背景下基于视觉的动态手势识别研究[D]. 华东师范大学.2008,4

[41] 江超, 艾娇艳. 基于 OpenCV 的摄像头动态手势轨迹识别及其应用[J]. 计算机应用. 2012, 7, 32(1): 131

[42] 张毅, 张烁, 罗元, 等. 基于 Kinect 深度图像信息的手势轨迹识别及应用[J]. 计算机应用研究. 2012, 9, 29(9): 3548

[43] http://tech.xinmin.cn/3c/2010/12/02/8060508.html. [2013 available]

[44] B. K. P. Horn, B. G. Schunck. Determining Optical Flow[J]. Artifical Intelligence Laboratory. 1981: 185-203

[45] B. Lucas, T. Kanade. An Iterative Image Registration Technique with An Application to Stereo Vision[J]. In Processdings of the International Joint Conference on Artificial Intelligence. 1981: 674-679

[46] Sun Chengzhi, Xiong Tianzhong, Ji Shunping. Application of Optical Flow Algorithma Based on Difference in Target Detection and Tracking[J]. Machine

Tool and Hydraulics. 2010, 7, 38(14): 59-62

[47] 万缨, 韩毅, 卢汉清. 运动目标检测算法的探讨[J]. 计算机仿真. 2006, 10, 23(10): 221-226

[48] 张秋余, 胡建强, 张墨逸. 基于区域生长的 Mean shift 动态变形手势跟踪算法[J]. 模式识别与人工智能. 2010, 8, 23(4): 580-585

[49] G. R. Bradski. Computer Video Face Tracking for use in a Perceptual User Interface. http://developer.intel.com/technology/itj/ q21998/pdf/camshift.pdf

[50] O. D. Nouar, G. Ali, C. Raphael. Improved Object Tracking with Camshift Algorithm[A]. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing[C]. Piscataway, NJ, USA:IEEE. 2006: 635-660

[51] 王燕妮, 樊养余. 七方向差分链码的视频对象形状编码[J]. 电讯技术. 2010, 6, 50(6): 32-36

[52] 王云飞. 动态手势识别中关键技术的研究[D]. 四川师范大学. 2011, 5

[53] 刘长明, 任一峰. 语音识别中 DTW 特征匹配的改进算法研究[J]. 中山大学学报. 2006, 27(1): 37-40