北京工商大学

硕士学位论文

机器人网络控制平台系统设计和实验研究

姓名：刘振宝

申请学位级别：硕士

专业：机械设计及理论

指导教师：辛洪兵

20070501

摘　要

基于 Internet 的远程控制机器人系统结合了传统的机器人控制技术与先进的网络通讯技术。它可以被应用于危险环境下的远程作业、远程医疗、远程教学、远程监护以及传统生产模式的改造等众多方面，具有广阔的应用前景。

本文首先阐述了国内外在基于 Internet 的远程控制机器人系统方面的研究状况，介绍了各研究阶段典型的机器人系统，以及最新的研究状况。对国内外现有的基于 Internet 的远程控制机器人系统软件框架、控制方法和网络通信协议等方面进行了详细的分析和研究。

在此基础上，本文研究了网控机器人网络通讯的数据传输特点，提出了一种新的基于 TCP 的机器人数据控制协议，来实现客户端和服务器端的数据通信。分析和研究了多机器人接入系统的问题，基于面向对象编程思想，采用了多态和模板的设计方法，实现了一种多机器人动态接入的方法。

本系统以首钢莫托曼机器人 MOTOMAN－HP3 为实验对象，设计了基于客户端/服务器端（C/S）的多层分布式网络控制系统，构建了以视频服务器、应用程序服务器的分布式架构，并根据网络机器人控制系统的特点，实现了监督控制和自主任务控制两种策略，实现了多机器人的载入问题，提出并实现了自己的基于网络的机器人控制系统。

关键字 ： 机器人，网络，控制系统

ABSTRACT

The remote control robot system based on Internet integrates traditional robot controlling technology and advanced network communication technology systematically. It can be applied in long range projects, medicine, teaching, monitoring in dangerous environment, and in reconstruction of traditional produce mode and so on. It will be a vast applying prospect.

This paper first elaborates the research condition of remote control robot system based on the internet domestic and abroad, introduces the typical model robot system for each stage and the latest research condition of this aspect. Then it analyzes the framework of the remote control robot system based on internet, controlling method, research categorization and network communication protocol internationally.

On this foundation, this paper studies data transmission characteristics of network communication of net controlled robot, proposing a new data control protocol on the base of TCP, which enable the data communication between client and server. It analyzes the problem of multi-robots' connecting system, and performed a method that enable multi-robots' dynamic connecting according to the face to object programming thought and adopting design method of polymorphism and template.

This system adopts Shougang MOTOMAN's robot MOTOMAN-HP3, and designs a multilayer distributed network control system on account of Client/Server, sets up the distributed structure with video frequency server and utility program server, carries out strategies of Supervisory control and independence mission control according to the characteristics of the network robot control system, and puts forward my own robot control system based on network.

Keyword: Robot, Network, Control system

北京工商大学学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师指导下进行的研究工作所取得的研究成果。除了文中已经注明引用的内容外，论文中不包含其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律后果完全由本人承担。

学位论文作者签名：刘振宝 日期：2007年 5月27日

北京工商大学学位论文授权使用声明

本人完全了解北京工商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北京工商大学。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，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

学位论文电子版同意提交后，可于 □当年 □一年 □二年后在学校图书馆网站上发布，供校内师生浏览。

学位论文作者签名：刘振宝 导师签名：辛洪兵 日期2007年5月27日

第1章 绪　论

Internet 的迅速发展不仅使社会经济结构和人们的生活方式发生了巨大的变化，同时也给新世纪的机器人研究和开发开辟了新方向。把机器人接入网络，使网络拥有能够对环境进行操作的终端执行能力，可以极大地拓展网络和机器人的应用范围。

Internet 与机器人相连，推动了信息领域与机器人领域的融合。网控机器人系统结合了传统的机器人控制技术与先进的网络通讯技术。它可以被应用于危险环境下的远程作业、远程医疗、远程学、远程监护以及传统生产模式的改造等众多方面，具有广阔的应用前景。因此，基于 Internet 的远程控制机器人系统的研究成为当今世界的一个前沿课题和热点问题。

1.1 机器人网络控制平台系统设计和实验研究的意义和特点

1.1.1 机器人的简介

机器人是传统的机构学与近代电子技术相结合的产物，也是当代高新技术发展的一个重要内容。人们制造机器人是为了让机器人来代替人的工作，所以希望机器人能够具有人的劳动机能。早期的机器人只是单纯从技术上模仿人的某些功能，由于人工智能和其它智能技术落后于人们对它的期望，致使绝大部分研究成果始终走不出实验室。因此 20 世纪 80 年代末，各国把发展的目标调整到更现实的基础上，即把以多传感器为基础的计算辅助遥控加上局部自治作为发展非结构环境下工作的机器人的主要方向，而把智能自治式机器人则作为一个更长远的科学问题去探索。这就是新一代机器人化的机器。

到了 20 世纪 90 年代，随着计算机技术、微电子技术、网络技术等的快速发展，机器人技术也得到了飞速的发展。智能机器人的研究是目前机器人研究中的热门课题。

1.1.2 把 Internet 引入机器人领域的意义和特点

将机器人技术与 Internet 技术结合可以说是机器人技术发展的一个重要方向，尽

管目前的网络机器人系统还处于实验阶段，但其广阔的应用前景和巨大的开发价值正吸引着越来越多的研究人员投身其中，推动着机器人在信息时代中的发展。以 Internet 为媒介的网络机器人正日益成为研究的热点，人类希望能够不受距离的限制获得直接操作的真实感觉，达到操作的临场感，并能够控制机器人完成更为复杂的作业。但要达到这种水平还有许多难题需要解决，其中控制系统的稳定、机器人的智能化和更好的实时性都是网络控制系统中值得研究的。

基于 Internet 的网络机器人的研究是一个综合了机器人技术、智能控制技术、虚拟现实技术、计算机仿真技术、计算机网络技术等科学的前沿研究领域，对于机器人技术的理论和实际应用具有重大的理论与应用价值。

网控机器人的研究思想是把网络和机器人技术融合起来，从系统的角度来进行研究开发。它把网络的开发和机器人的开发作为一个大的系统的两个方面结合起来进行研究开发，通过计算机网络的发展来扩展机器人控制系统的多样性，使机器人的控制系统与网络中的其他成员（通过网络互连的其他智能设备）进行交互和协作，从而使信息和数据的获取与共享以及与人的交互界面都将随之改变。机器人将可以越来越多的享受网络资源，利用网络的方便，使机器人成为网络的一员，这就可以突破距离的限制，真正实现跨区域、跨空间的远程控制。网络也可以借以机器人的特点来扩展它的外在延展性.

将实体机器人与计算机网络相连，通过网络通讯实现在 Internet 上对机器人进行控制和监视，这样的系统可称为网络机器人系统，它与传统的机器人系统相比有如下特点[1]：

（1）跨空间性

网控机器人控制系统完全克服了空间限制,只要有 Internet 接入点,将客户的控制终端连接到互联网并安装相应的控制软件或硬件系统，即可构成网路控制系统，从而实现对网络机器人的控制。

（2）交互的多样性

客户端可以通过多种方式与机器人终端交流,获得现场信息进行控制,如现场图像实时反馈、对现实环境的虚拟重构以及遥操作和力反馈等。

（3）分布性

可以通过 Internet 将多个机器人连接起来，实现分布式任务及协作。各地的专家们通过 Internet 控制多个机器人，完成远端的高精度和相对复杂的作业，这大大提高

了人类在恶劣环境下安全完成复杂作业的能力。

(4) 成本低，容易维护

不需要为网路控制系统建立特殊的操作站，铺设专用线路，而可以直接利用廉价的 Internet 线路。网络技术的成熟增强了系统可靠性和可测试性，易于维护并能够长期稳定地工作。

1.2 国内外的研究现状

1.2.1 国外的研究现状

基于 Internet 的远程控制机器人系统的思想是由 Ken Goldberg[2] 于 1994 年春首先提出的。其最初的构想是给公众提供可通过 Internet 访问的遥控机器人，并支持用户对其实施远程操作。这一构想极大扩展了于 50 年前提出的遥操作概念，它使得全球的网络用户都可通过 HTTP 提供的低成本且能被广泛应用的接口来访问并共享远程资源。Ken Goldberg 很快就将这一构想应用到 Mercury Project 中。因此，Mercury Project 是第一个基于 Internet 的远程机器人控制系统。在这个站点上，用户通过远程控制一台配备了 CCD 摄像头和气囊系统的 IBM SCARA 型机器人，在装有沙子并埋有人造物的半圆形工作空间中实施发掘工作。1995 年 Mercury Project 被 Telegarden Project 所取代，新计划中仍是采用装有 CCD 摄像头的机械手复合其它机构通过 Internet 给网络用户提供远程培植操作。

几乎同期，西澳大利亚大学 Kenneth Taylor[3]等人研发的安装有固定摄像头并具有六自由度的机械手臂 ASEA IRb-6 的机器人 Telerobot，它支持网络用户在工作空间中实施用积木建造复杂建筑的操作。Telerobot 是早期基于 Internet 的远程机器人控制系统的典范。用户可以填写 HTML 表单，在反馈的机器人工作区图像上直接点击，高级用户还可以输入系列连续的移动指令，以向机器人发送请求指令。服务器端的 CGI 脚本处理用户的请求，分别与机器人控制服务器和图像服务器通信，控制机器人的移动，并在机器人完成任务时，返回当前位置的不同角度的图像。

在 Ken Goldberg 和 Kenneth Taylor 的开创性工作之后，越来越多的学者投入到基于 Internet 的机器人技术的研发工作中来，更多有创意的网络机器人站点也相继出现在 Internet 上。例如：英国 Bradford 大学的远程机器人望远镜系统；南加州大学可播

种和浇水的远程花园 Telegarden；澳大利亚 Wollogong 大学拾取木块的 Roboty；德国以“Hanoi 塔”方法搬运木块的 Net-Robot；加州大学 Berkeley 分校的 PRoP（Personal Roving Presence）等等。NASA（美国航空航天局）的一些专家开始研究通过 Internet 对空间飞行器实施遥操作的可行性[4,5]。

基于 Internet 的遥操作机器人系统在初期只能提供机器人工作环境的静止画面，有些系统如 Telegegerden 也尝使用 CAD 技术来回馈被控机械臂状态动画。当支持通过网络传输流式图像数据的网络摄像头技术出现后，现场环境的图像反馈变得易于实现。要使用户能远程控制机器人并完成一系列复杂动作，就需要使用更先进的技术来实现复杂且友好的用户界面。在这些工作中，Matthew R. Stein 的网络交互绘画机器人 PumaPaint Project 十分引人注目。它允许任意 Internet 用户通过网络控制 PUMA760 机械臂在远程实验室画布上完成绘画操作。用户界面提供了用 Java 设计的虚拟画布，并且系统通过不断的图像更新给用户提供及时的视觉反馈，使得没有任何专业知识的网络用户也可轻松实施操作。

Internet 上还活跃着另一类远程控制机器人系统：基于 Internet 的远程控制自主式移动机器人。和上述的远程控制系统相比，它的自主性和移动性的特点将在更高程度上满足人们对远程空间探索的要求，并为最终与非结构化的、未知的远程环境的交互提供了研究平台。

基于 Internet 的远程自主式移动机器人系统各有特色。瑞士联邦工学院洛桑科技研究所的 Khep On The Web 是第一个具有代表性的基于 Internet 的远程自主式移动机器人系统。用户不但可以控制机器人 Khepera 的位置和速度，还可以通过对摄像机旋转角度及镜头伸缩的控制得到需要的图像反馈，所以它可以在人造迷宫中进行运动并能同时经摄像头的图像反馈进行观察。

美国 Carnegie Mellon 大学研发的 Xavier 是第一个可通过网络控制并运行于复杂办公环境的自主移动机器人。它可以在线或离线接收请求命令并在穿行于实验室和教室之间的运行时段内进行处理，如果用户给出本人的 E-mail 地址，它会在到达目的地之后通知用户，并包含图片或最拿手的碰碰笑话。Xaiver 的网络界面使用了 client-pull 和 server-push 技术来获取图像，用户界面还提供办公环境的地图并显示机器人在其上的位置。德国 Bonn 大学开发的用于博物馆导航的 Rhino 和 Minerva 机器人以及 NASA 的火星极地登陆者 Mars Polar Lander 则在基于 Internet 的远程自主式移动机器人的两大主要应用领域进行了积极的尝试。这些机器人的特点是自身具有较高自主性，而网

络技术又给异地操作者提供了进行远程控制的手段[4,5]。

目前，国外已经着手向市场推出基于 Internet 远程控制技术的机器人产品。美、日作为机器人技术大国，在这方面的开发和研究非常活跃。日本富士通研究所开发了一种能够用手机在外进行遥控的机器人 MARON－1[6]，该机器人能够发挥看门人的作用，可以监视不法者的入侵。新开发的机器人高和宽均为 32 厘米，前后距离为 36 厘米，重 5 公斤，带有两个驱动轮，可以越过 2.5 厘米高的台阶。它能够借助两个镜头和距离传感器，避开障碍物到达人所指定的目的地。通过操作机器人的镜头，家中的状况就可以在手机中显示出来。借助组装在机器人身上的遥控器，还可以对电视和录像机进行遥控。一旦发现不法者入侵，机器人就会拉响警报，

在美国处于领先地位的机器人技术研发企业—iRobot[7,8]公司，正在研制一类可以代替人类在世界各地漫游的新型远程遥控机器人。它们充分利用了互联网庞大的信息输送能力，装备了一个视频摄像机，一个麦克风，以及一个可以与互联网接口的无线发射器。如果一位远处的人类控制者登录正确的网页，他将可以看到和听到机器人看到和听到的事情。而且，人类控制者还可以通过点击鼠标，来命令机器人从一个地方移动到另外一个地方。在人工智能软件和各种传感器的帮助下，远程遥控机器人可以在走廊中移动，而不会碰到墙壁，甚至可以攀越楼梯。

这类机器人还能解决目前商界视频会议等技术中的缺陷。它们不仅可以解决摄像机对准会议参与者的问题，而且还解决了所用设备的随处移动问题。远程遥控机器人几乎可以移动到任何地点，把它的摄像机调整到控制者所希望的角度。采用远程控制机器人，用户可以在办公室里观察企业仓库中的活动，或者在码头上检查送来的货物。

近几年，德国汉堡大学提出了一种基于移动机器人的新的控制思想，他们把机器人的移动特性提取出来，进行专门的编程，形成分门别类的单元，综合这些单元形成了一个框架（framework）。使其他非专用人士也可以在这个框架（framework）上进行机器人的远程编程，发展形成了一种面向网络的任务级编程思想。这就是 Roblet 技术[10]。

这个框架可以用来在简单机器人中为特殊的任务编写分布控制或监控程序。它也可以允许程序员发送一个运行在机器人上的 Roblet 程序给 Roblet 服务器。相对于其他的分布式系统架构来说，Roblets 是由数据和代码组成的。Roblet 服务器用定义好的动作行为来执行 Roblets 程序，即使出现故障也会执行。此框架对程序员隐藏了所有的网络细节，以至于在本地计算机上写程序就像直接在远程机器人工作一样。这个减少

了机器人控制程序的开发时间。

这种方法是基于 java 平台的一种控制方式，包括 Roblet 和 Roblet-Server，它借鉴了 applet 的思想，提出自己的 Roblet 想法。Applet 是利用 JAVA 语言生成的可运行代码，Applet 可以嵌入 Web 页面中，在支持 JAVA 虚拟机的浏览器上运行。Applet 被布署在 Web 服务器（如 IIS）上，当客户端请求 Web 页时，浏览器从 Web 服务器上将其下载到本地客户端浏览器创建该 Applet 类的实例并执行[11,12]。Applet 是从服务器端发送到客户端运行，而 Roblet 是在客户端发送到服务器端执行。它在 java 的基本环境中，提供一个 framework 给用户。它把监督控制、预测/预演显示控制、事件智能控制三种方式结合起来，抽象出它们的控制特点，把共性统一起来，并兼具它们的特点。这个提出的框架已经被证明是很有用的。它的执行很简单，在 java 平台下利用了很多不同技术如 Jini、RMI 和 JNI 技术。通过定义新的特殊函数可以扩展框架提供的功能。用户利用框架提供的函数就可以编写出自己的基于网络的机器人控制程序，要求机器人执行新的任务只需简单的更换相应的函数即可，它也提供像监督控制一样的方法[11]。Roblet 在远程系统上的运行可以降低传输的数据，如果像无线网络一样只有有限的带宽可以使用，这就是很重要的一个方面。但是它有个缺点，就是跟机器人的关联性太大，对同一个机器人来说可以方便的实施存在的功能，当换成其他机器人时，又要重新设计框架。就是说机器人的控制系统和机器人关联性太强。

而 Roblet－Server 方法是现在比较先进的控制方式，不管在处理延时和任务的更改性都有比较显著的优势。

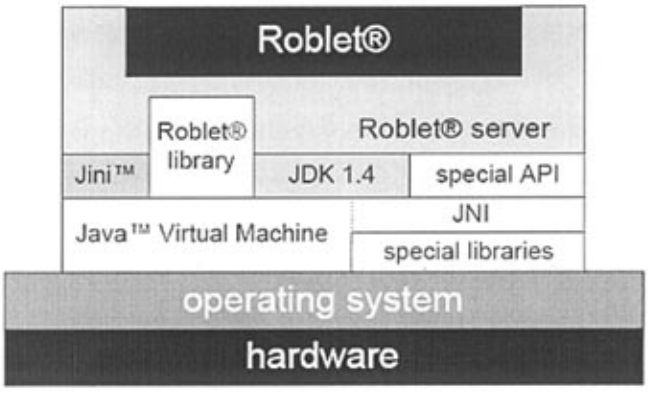


图 1.1　Roblet 服务的系统架构[10]

1.2.2 国内的研究现状

在过去的几年中我国 863 高科技发展计划和国家自然基金会重点资助了与网控机器人方面有关的项目，并且已取得了相关的研究成果。我国如中科院沈阳自动化研究所机器人实验室、哈尔滨工业大学、清华大学、同济大学等国内高校和科学院所进年来都已开展了基于 Internet 的远程控制机器人系统的研究[9]。

哈尔滨工业大学研究的基于 Internet 的遥操作机器人系统：Telerobot 采用由 Internet 服务器和机器人服务器构成两层服务器结构，该系统可以使得用户通过 Web 浏览器对一台 PUMA562 机器人进行控制，完成抓取、搬运、堆放等操作。其 Web 服务器采用 Java 语言编写的通用网关接口 CGI，Web 服务器与机器人服务器之间采用 Winsock 通信。该系统没有提供动态的实时图像，而是以静态的图片反馈给客户端。

上海交通大学也开发了基于网络的机器人遥操作系统，被控机器人是一台 Adept604-S 型工业机械手。机器人服务器、图像服务器和 CGI 程序用 VB 升发，他们之间的通信通过 DDE（动态数据交互）进行。由于此系统的人机交互界面也是基于 HTML 技术的，交互性差，控制方式单一。此后，他们对这个工业机械手，又开发了一个基于 Web 的机器人遥操作系统，使用 Java Applet 开发客户端用户界面，但对视频图像传输等未涉及。

2000 年华南理工大学吴国钊学者报告了他们基于 Internet 的机器人实时跟踪系统，系统采用了客户/服务器模式，通过图像采集，图像传输，客户请求应答三个线程实现机械臂 Zebro 状态控制。在时延控制、人机界面等各方面的工作未深入探讨。

沈阳自动化研究所的遥操作机器人系统，采用 Java 语言进行开发，实现了图像以及指令信息的传送。

总之，国内对于基于 Internet 的机器人控制正处于由实验室阶段向实际应用得过渡时期。2004 年 4 月 26 日，内地首例远程遥控机器人成功操刀切除患者胆囊，也在证实国内远程控制机器人开始逐步应用在各个领域。2005 年 12 月 23 日，由北京航空航天大学机器人研究所和海军总医院神经外科共同研制的可通过互联网异地遥控指挥的机器人为一患病老人做开颅，手术成功仅耗时 40 分钟，这是国内首例通过普通互联网控制机器人成功手术。

1.3 本文的主要工作

针对机器人网络控制的网络时延影响和实现机器人的可分离性，进行监督控制和用户的自主编程，本文提出了一种新的机器人网络控制平台系统，并开发了实验研究。

本论文研究的着眼点主要在于：开发机器人的网络控制平台系统，实现基于网络的机器人控制。根据不同的应用和研究可以更换服务器终端的机器人，即可以为研究人员提供一个实践平台，也可以让非专业研究人员可以在此平台上进行更深层的理论研究和开发工作，还具有一定的实际应用能力。该系统的创新点在于：

（1）除包含监督控制模式外，还提供用户自主性编程功能，使高级研究人员可以在系统提供的类库基础上实现自己的控制程序。

（2）根据网控机器在 Internet 上的数据传输特点，建立一个以 TCP 协议为基础的网控机器人数据传输协议，进行对机器人远程控制的数据交换。

（3）网控系统相对独立，实现上位机控制系统与具体机器人的分离性、独立性。系统接入不同机器人的便捷性。

本论文分为 5 章，各章节主要内容如下：

第 1 章 绪论。对网络远程机器人控制系统的概念、研究意义、国内外的发展状况与趋势、目前的一些成功实例、以及存在的问题和解决途径等问题作了较为全面的分析和介绍，同时，对本论文的研究目的、研究的主要内容作了概要性的介绍。

第 2 章 机器人网络控制系统相关的控制策略的分析。本章归纳了目前几种网络远程控制系统体系结构的区别和特点，分析了不同控制系统在不同条件下的优点和不足。也对机器人控制模式进行了详细的分析，最后介绍基于 Internet 的网控机器人采用的 C/S、B/S 模式以及对现场信息进行反馈的几种方法。

第 3 章 机器人网络控制平台系统的详细设计。本章详细的介绍了本系统实现的方法和原理。介绍了系统采用的 Motoman Hp3 机器人。讨论了网控机器人中的数据传输机制。详细的介绍了系统各个功能模块和编程模块的实现。提出了一个基于 TCP 协议的机器人数据控制协议。介绍了多机器人接入系统所采用的方法。介绍了为保障机器人网络控制平台系统安全运行的一些保障措施。

第 4 章 机器人网络控制平台系统的实验研究。本章介绍了系统的实际运行的效果，介绍了系统的运行方式，探讨了在整个系统中延时的存在环节。

第 5 章 总结和展望。对本系统的优点和不足加以总结，提出了进一步改进措施。

第 2 章 机器人网络控制系统相关的控制策略的研究

2.1 机器人网络控制系统的几种策略的分析和研究

基于 Internet 的远程控制机器人系统是一个覆盖面很广的复杂系统。它不仅拓展了传统的机器人遥操作的应用，而且随着互联网的迅猛发展，它在许多新兴领域中，如自主机器人系统、远程制造业、远程培训、分布式制造系统以及家用机器人和娱乐机器人等，也有良好的应用前景。以下按照网络技术在机器人学中不同的应用方向及不同的技术侧重点，对国外研究机构基于 Internet 的遥操作技术、基于 Internet 的自主移动机器人控制技术、基于中间件技术的分布式机器人系统和分布式机器人软件库几个方面进行控制系统的分析和研究。

2.1.1 基于网络的遥操作技术

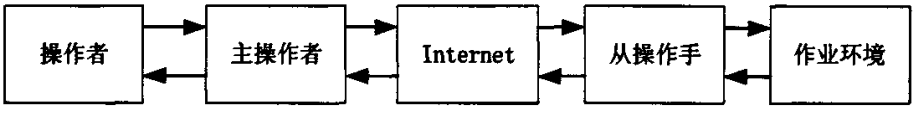
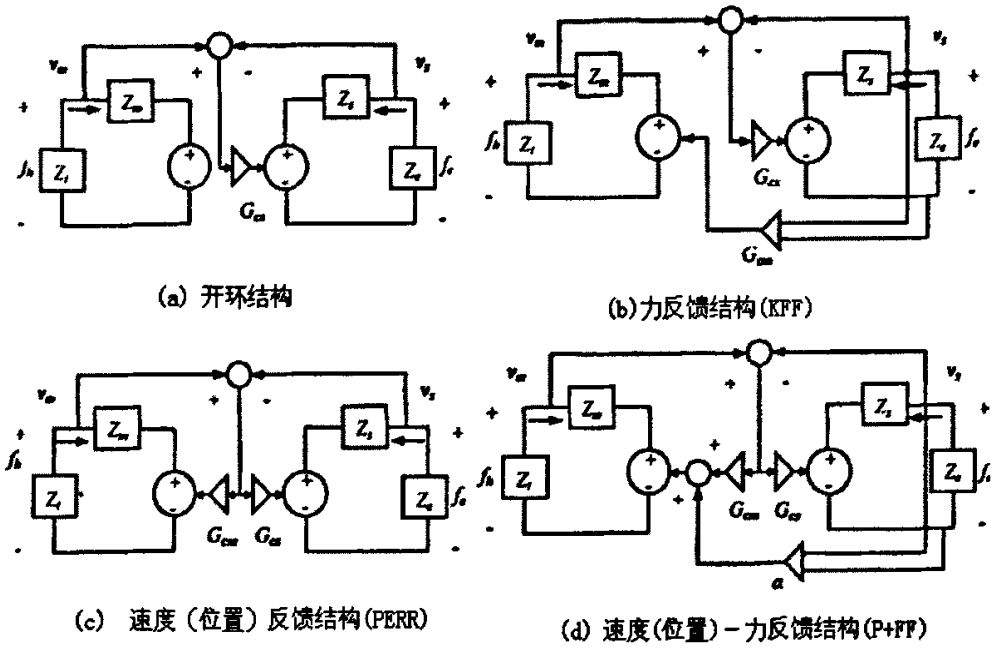


图 2.1 基于 Internet 的远程控制遥操作系统结构

基于网络的远程控制机器人技术继承和发展了遥操作技术，是给用户提供通过 Internet 对远程设备实施遥操作控制。传统的遥操作系统允许操作者通过主从机器人来实现对远程设备的控制。它主要应用于空间技术、核废料的处理、显微外科、微电子装配、水下操作、采矿业及消防救援等方面。基于网络的远程控制遥操作系统的一般结构如图 2.1 所示。按系统结构的不同，还可以将其分为两大类：基于 Internet 的双向力反馈遥操作机器人系统和基于 Web 浏览器的遥操作机器人系统。前者只是在通讯手段上与传统遥操作不同，从而在结构上和技术上继承了传统遥操作系统采用的方法和技术。后者完全以网络为主体，系统中只存在从操作手，而主操作手被 Web 浏览器代替。

现有的遥操作系统多采用以下四种控制结构，如图 2.2 所示：



(a) 开环结构　　(b)力反馈结构(KFF)

(c) 速度（位置）反馈结构(PERR)　　(d) 速度(位置)－力反馈结构(P+FF)

图 2.2 遥操作系统的控制结构[13]

Z_t、Z_e、Z_m、Z_s 分别为操作者阻抗、环境阻抗、主机械手阻抗和从机械手阻抗。这四种结构都在从端构成速度的本地闭环控制，G_{cs} 为控制器。开环结构不含有任何信息的反馈，速度（位置）反馈结构返回两机械手速度或位置的差值，这两种结构都不具有力觉临场感，多用在自由运动任务中，依靠视频信息来获得临场感。在受限运动即从机械手和环境有相互作用的任务中，多采用力反馈或速度（位置）－力反馈结构。力反馈结构仅返回从机械手与环境的相互作用力，G_{cm} 为控制器，速度（位置）－力反馈结构反馈的是从端作用力和两机械手速度（位置）差的线性组合，它是速度（位置）反馈结构和力反馈结构的合体。后三种控制方式都属于双边控制，Sherman 对这三种控制方式的性能进行了分析[14]，并指出速度（位置）－力反馈结构的遥操作系统具有最好的控制性能。但是这种方式比较复杂，对速度或位置的差比较敏感。

遥操作系统的临场感可以分为视觉临场感和力觉临场感，遥操作系统的透明性是针对这两种临场感的一个遥控机器人学专有名词。不言而喻，透明就是操作者借助主从机械手和信息传输通道与远端环境相互作用，感觉就像在直接对环境进行操作一样。首先操作者看到的要是能反映真实从端环境的图像信息，其次操作者手上感觉到的力是真实的环境作用力。因此透明是遥操作系统实现操作者临场感的根本。力觉临场感即“力觉”上的透明性，以下的分析中简称为透明性。

一般认为，操作者感觉到的阻抗定义为：

$$Z_t = \frac{f_h(s)}{v_m(s)} \tag{2.1}$$

环境阻抗：

$$Z_e = G_e = \frac{f_e(s)}{v_s(s)} \tag{2.2}$$

对于力觉临场感系统，如果操作者感觉到的阻抗与从机械手和环境作用的阻抗相等，即 $Z_t = Z_e$，操作者感觉像在直接对环境进行操作，则说这个系统是透明的。文献[15]指出系统要达到透明，除了阻抗匹配以外，还要实现主从手之间行动和力的跟踪，即：$v_s(s) = v_m(s)$ ，$f_e(s) = f_h(s)$。显然，由于时延的存在，速度与力的信息滞后，系统完全透明是无法实现的。透明性是比静态跟踪性能更强的操作性能，它更强调了系统的动态跟踪，而且要求跟踪比率相等。临场感遥操作系统各性能之间的关系如图 2.3 所示：

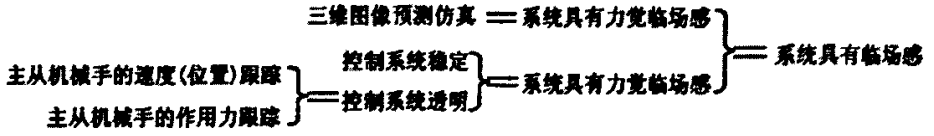


图 2.3 临场感遥操作系统各性能之间的关系[15]

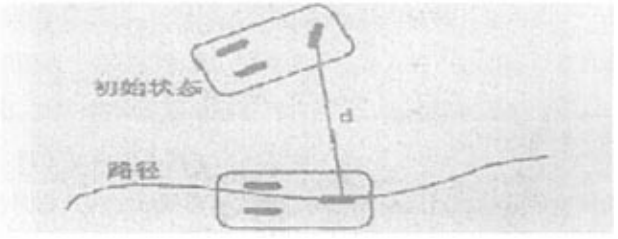
由于遥操作机器人上的双工控制器负责主从设备间的双向信息流传输，所以双工控制器是否被设计成稳定且透明的系统是非常重要的。但是，由于通讯中的延时和滞后和系统中自身存在的不稳定因素，要实现稳定且透明的双工系统操作仍有很大的困难。由于网络系统存在显著的通讯延迟，对于基于 Internet 的遥操作机器人系统而言，如何设计具有自适应性的双工控制器，提高它对网络时滞的鲁棒性并对环境阻抗能进行在线辨识，是远程控制机器人在遥操作方面的研究重点。基于 Internet 的机器人遥操作技术的提出开创了一个崭新的研究领域，使遥操作技术的应用走向网络化和全球化。

2.1.2 基于 Internet 的远程控制自主式移动机器人技术

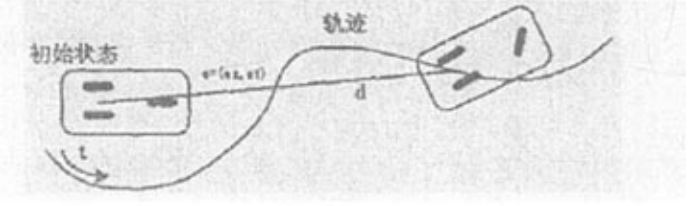
国外在基于 Internet 的远程控制自主式移动机器人系统方面已经开展了深入研究，如前面所提到的 Carnegie Mellon 大学研发的办公室自主移动机器人 Xavier 和可以远程控制的博物馆导游机器人 Minerva。

最初基于网络的移动机器人的自主性能有限，而且同一时间内只能向单一用户提供网络服务。瑞士联邦洛桑科技研究所的 KheP On The web 就是其中典型代表。用户可通过网络控制小型移动机器人在人造迷宫中进行运动并能同时经摄像头的图像反馈进行观察。

移动机器人根据控制目标的不同，可以分为三类典型的控制问题，分别是：路径跟踪问题、轨迹跟踪问题和点镇定问题。如下图 2.4 所示：



(a) 路径跟踪问题



(b) 轨迹跟踪问题

(c) 点镇定问题

图 2.4 移动机器人的三类典型控制问题[16]

（1）路径跟踪问题

所谓路径跟踪问题，就是在惯性坐标系里，机器人必须从一个给定的初始状态出发，到达和跟踪一条理想的几何路径，而机器人的初始点可以在这条路径上，也可以

不在路径上。如图 2.4（a）所示。

在移动机器人的路径跟踪问题上，一般地，我们会选取两个输入量中的前进量为任意的常量或时变量，而另一个输入量用作控制量。所以路径跟踪问题可以转换成关于路径跟踪误差的一个标量函数的零点稳定问题。控制的目标是使到机器人和给定路径之间的距离为零。

（2）轨迹跟踪问题

所谓轨迹跟踪问题，就是在惯性坐标系里，机器人必须从一个给定的初始状态出发，到达和跟随一条理想轨迹，同样的，机器人的初始点可以在这条轨迹上，也可以不在轨迹上。而与路径跟随问题不同的是，这条理想轨迹是一条和时间成一定关系的几何路径。如图 2.4（b）所示。

在移动机器人的轨迹跟踪问题里，机器人必须跟踪满足特定时间规律的笛卡尔路径，也即机器人必须跟踪一个移动的参考机器人。通常的，我们将移动机器人的轨迹跟踪问题转化为这样的描述：要求机器人参考点的位置跟踪一个随时间的理想变化曲线。

其数学描述如下，给定一参考机器人[17]

$$\begin{cases} \dot{x}_c = v_r \cos\theta_r \\ \dot{y}_r = v_r \sin\theta_r \\ \dot{\theta}_r = \omega_r \end{cases} \qquad (2.3)$$

其中，参考机器人的状态向量 $q_r = [x_r \quad Y_r \quad \theta_r]^T$ 参考速度向量 $V_r = [v_r, \ w_r]^T$.

（3）点镇定问题

所谓点镇定问题，指的是机器人从一个给定的初始状态到达一个理想的目标状态，并稳定在给定目标点，该问题也即是在机器人状态空间的（平衡）点的稳定问题。如图 2.4.（c）所示。

其数学描述为：给定一任意的参考机器人的状态 q_r，先设计一个平滑的速度控制 $v_c = f_c(e_p, v_r, K, t)$，使 $\lim_{t\to\infty}(q_r - q) = 0$。然后再计算加到机器人上的力矩 τ，使到 $t \to \infty$ 时，$v \to v_c$。

我们注意到，在移动机器人的这三类控制问题中，路径跟踪问题由于只需要通过改变机器人的角速度（线速度可以设定为一常值）来减小机器人和参考路径之间的距离，所以它是相对简单的。而点镇定问题相对复杂一些，它的速度控制 v 是时变的。

相对于路径跟踪和点镇定问题，轨迹跟踪问题更具代表性，因此本章选择轨迹跟踪问题进行讨论。

对于跟踪控制，前人已经作了大量的工作，许多先进的控制算法被用与这一问题的解决，这些方法大体可以分类如下：

（1）基于滑模控制的方法

（2）基于反馈线性化的方法

（3）回退法

（4）自适应控制方法

（5）智能控制方法（包括神经网络，模糊控制，预测控制，小波算法，鲁棒控制，遗传算法等）

以上控制方法在移动机器人的运动控制上都取得了不同程度的成功，但同时也具有一定的局限性。从现有文献来看，所采用的智能控制方法，往往是一种或几种算法的结合，它使系统设计不再依赖于数学模型，摆脱了线性局限，并且抗干扰和自适应性强，同时也为解决非完整移动机器人运动控制问题提供了新的手段，具有巨大的理论价值和应用前景。

由于不确定网络时滞和复杂的现场环境，提高被控机器人的自主性对实现实时远程控制具有重要意义。被控机器人本身具有的避障功能、对路径的自主规划和对多任务的决策能力，都有利于减少网络通讯中不可预测时滞对机器人控制实时性的影响。这一思想的实质是将属于远程控制的部分功能下放到控制系统本地来实现，远程的操作者只需对预先定义好的操作指令实施控制操作，或直接通过对运行于客户端的辅助软件的图形界面及命令菜单的点击来实施远程操作控制。这一方法降低了系统的复杂度，将各种简单的基本操作集成为高层次的复合动作，从而提高了系统操作的集成度，加快了系统的控制响应速度，保证了控制的实时性。

有专家指出基于网络的室内自主机器人控制采用事件驱动和任务控制的系统体系结构时，其自主性和基于网络的远程控制会得到更完美的结合。

2.1.3 分布式机器人系统

Internet 技术使人能够与网络上任何地点的设备相连接、交互。随着接入控制系统的机器人数量不断增多，以及需要同时访问系统的用户数量的增加，原有的远程控制

机器人系统服务器已不能满足网络负荷。因此，需要将服务器上的控制功能模块分解到其他计算处理设备上。比如，将机器人本地控制器或图像处理控制器等管理由其他的计算机完成。这些设备在物理空间上可以处于不同的地点。这样就形成了基于Internet 的分布式机器人控制系统。分布式机器人系统最大的特点就是低价、灵活、可扩展。对于一对多结构、多对一结构还有多对多结构的远程控制机器人系统，分布式结构是其理想的选择。美国航空航天管理局的寻路者计划（Pathfinder mission）就是通过 Internet 实施分布式控制的成功实例。

对于分布式机器人控制系统来说，需要解决网络上分布式对象之间的数据交换问题，即构建高性能的通讯协议体系以实现多个用户和代理间协作控制。目前不同的分布式机器人系统使用了不同的通讯协议和技术，这些机制包括：RMI（emote Method Invocation，远程方法调用）、CORBA（Common object Request Broker Architecture）和 MOM（Message-Oriented Middleware，面向报文的中间件）[18,38,39]。

（1）RMI（Remote Method Invocation，远程方法调用）是 Java 虚拟机之间对象互相调用对方函数、启用对方进程的一种机制。RMI 采用序列化（serialization）机制通过网络发送对象，实现了 Serializable 接口的类的信息可以被复制，而文件格式、TCPIIP， Socket 和 I/O 等细节问题被隐藏。

（2）CORBA（Common Object Request Broker Architecture）规范是由 OMG（国际对象管理组织）发布并制定的标准。它定义了一种互相交换数据和发现服务的机制。其基本思路是以 ORB（对象请求代理）作为中间件来建立两个对象间的客户/服务器关系。

（3）MOM（Message-Oriented Middleware ，面向报文的中间件）与 CORBA 和 RMI 不同，它不是工业标准，而是在分布式应用环境中支持特定种类通用目标报文交换的中间件的集成术语。MOM 数据的交换是通过报文传输或报文队列，并支持分布式计算进程间的同步或异步交互。MOM 系统使用可靠队列来确保报文传送，并对所支持的报文提供目录检索、安全及管理等服务。队列特别适用于那些逐步递进的进程。MOM 模型中的报文是基于事件驱动的系统，而不仅仅是简单程序调用。客户可应用具有优先级机制的队列，这使得高优先级的报文可超越不重要的报文，这对分布式机器人的多任务控制十分重要。

通过比较可发现上述三种机制的通讯机理和应用范围不尽相同。RMI 和 CORBA 把客户和服务器间的通信建立在更高的对象机制上。对象间互相使用对方所提供的方

法，而不必知道它们是远程方法，数据编发的协议被隐藏。RMI 的最大优点是实现了整个对象在客户和服务器之间的传输，因此对于本系统采用纯 Java 编写的客户和服务器间的分布式对象间的通信，无疑是最佳方案。CORBA 与 RMI 的最大不同之处在于 CORBA 实现了不同语言写成的对象之间的调用。尽管 CORBA 避免了 RMI+JNI 的间接实现，但 CORBA 机制的实现，依赖于第三方提供的软件，如 Inprise 公司的 VisiBroke ，IONA 公司的 Orbix 等。相比之上，CORBA 适合于更加庞大、复杂的分布式机器人系统。由于 MOM 支持的是报文格式，因此它主要适用于有延迟的异步通讯；RMI 和 CORBA 均基于 RPC（远程过程调用）语义，是设计用以支持同步通讯的。此外基于报文的通讯允许向多个同位体广播信息。虽然 RMI 受到所有基于其上的应用必须是用 Java 编写的限制，但 RMI 也和 CORBA 一样是通用标准，所以具有巨大的互联潜力。与之比较 MOM 可提供嵌入于应用程序中的轻型客户端应用编程接口，而且由于 MOM 不存在同步通讯也不需要预装任何先决软件，所以它在简单环境中更具有应用潜力。

2.1.4 分布式机器人系统软件库

网路技术的发展已经能使各地的、各种各样的机器人终端连接到互联网上，并通过网络对机器人进行实时有效控制和及时的技术更新。这项技术包括网络接口装置、众多信息组的压缩与解压方法及先进的通讯传输技术。科研人员对能使大家共享各种最新机器人技术的机器人软件库（分布式机器人软件库 Distributed robotic software library）的需求非常迫切。当 CORBA 技术被采用时，在 Internet 上构建分布式机器人软件库就变得可行了。研发分布式机器人软件库主要基于以下原因[18]：

（1）机器人系统本身是多种应用技术的综合，其基本技术就包括有机器人视觉、力的感知、机器人运动规划及各种相关控制等。

（2）当机器人系统较复杂时，对各技术细节无需做出具体要求，只要能达到目标性能即可。

（3）随着机器人的复杂度不断提高，单个科研机构对各种技术都进行独立研发比较困难。

（4）机器人控制软件的编写、安装和测试的工作量巨大。

（5）科研人员和研发机构需要分布式机器人软件库来不断进行技术更新。

要实现基于网络的分布式机器人软件库首先要制定安全且高性能的通讯协议。通过对系统的资源管理可提高系统的容错性，并能在服务器间执行任务的调度和实现负载的均衡。由于 Internet 的不可预测时滞及有限的通讯带宽，导致基于其上的分布式软件库传输的数据类型不能太大、计算精度不能太高及进行实时处理的能力有限。

2.2 机器人控制模式的几种策略的分析和研究

对利用互联网的机器人远程控制系统的设计，一般分别在两个前提条件下进行。一个前提条件是，假定网络时延是某一个固定值或时延存在一个变化的上界。在控制系统的设计中是以时延的这个固定值或时延的上界作为纯滞后系统的时延参数。以这种方式设计的控制系统存在着一个最大弱点，当时延偏离固定值较大或在零和时延上界之间变动时，系统的稳定性很难保证。另一个前提条件则充分考虑了网络时延的变化性和不确定性特点，认为时延是时时变化的，也不考虑时延的大小。在此基础上建立起来的控制系统，使系统的稳定性得到保证，却往往在一定程度上牺牲了系统的实时性。然而，系统的实时性也是控制系统设计中需要考虑的一个重要因素。控制系统的实时性强，才能充分发挥人的能动性，及时做出判断和决策，提供效率。

换个角度考虑基于互联网的控制系统中的时延问题。在整个控制系统中机器人工作地点是定，虽然不同的网络用户端到机器人端的通讯时延不同，且时延大小差别非常大，如果遥操作的远端用户确定后，就形成了类似于通过专用线路的机器人远程控制过程。从前面对网络的测试结果的分析中可以知道，互联网上任意两个接入点之间的时延是相对固定的。因此，在某一网络用户对机器人实施控制前对其通讯时延进行测试，以该时延作为控制系统的时延参数，并结合与相适应的控制方式，将会最大程度上满足远程控制系统的稳定性和实时性要求。

基于时延的远程控制方式有很多种，如直接控制方式、预测显示方式、基于事件的控制方式、监督控制方式和学习控制方式等。每种控制方式都有其各自的优势，但又都有其局限性。由于网络时延变化存在不确定性，只采用一种控制方式很难适应网络上小同终端用户进行控制，同时控制模型对时延的补偿精度也难以保证[19, 20]。

2.2.1 直接控制模式

直接控制方式具有直观性特点，在机器人控制过程中现场感强，并能充分发挥操

作者的判断能力和决策能力。但这一控制方式对网络性能的要求比较高，要求网络传输时延小波动小大。在网络存在明显的通讯时延情况不，控制过程将会形成“运动一等待”控制结果，降低控制的效率，也会对控制过程的稳定性造成影响。如果网络通讯状况好，也就是时延小且随机波动相对缓和，采用直接控制的方式将是最简单有效的方式。从前而对时延测试结果的分析中可以发现，对于局域网用户和教育网条件下的大多数用户，网络的传输时延小且在每一段时间内相对稳定，如果在用户提出控制请求时传输时延小于 10 oms （RIT），用户进入直接控制方式，直接向机器人发出控制指令，如前进、转弯和停止等。用户端直接显示摄像机采集的现场图像信息外需要补偿，用户借助现场的实时环境的图像信息操作机器人。而移动机器人的自主能力在这一控制方式不体现不出来。

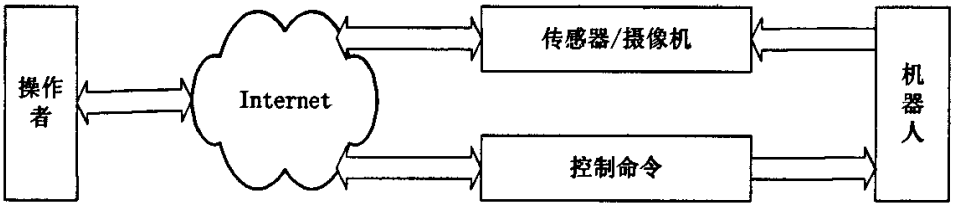


图 2.5 遥操作系统直接控制方式结构示意图

2.2.2 监督控制模式

这种控制方式首先是由 W.R.Ferrel 和 TB.Sheridan 于 1967 年提出的，其最初是用以研究空间探险机器人的。在传统的双向力反应遥控机器人系统中，当主从两端存在明显的通讯时延时，直接主从操作不得不采取“走一走，等一等”的方式来保证操作过程的稳定性，这就增大了任务完成的时间。也加大了任务的难度，因此提出了监督控制。按照 Ferrell 与 Sheridan[21]的定义，Supervisory 控制实质上是一个或多个操作者通过间断的程序命令控制以计算机为核心的独立闭环控制系统进行作业，并连续地接收由传感器采集的被控对象或作业环境的信息[22].Ferrell 与 Sheridan 等人[23]在监督控制的理论与应用上作了大量的工作，使之成为人在回路的主要控制模式，并在许多领域得到了广泛的应用。其基本思想就是将远程操作人员置于控制结构之外，从而努力减小传输时延对整个系统的影响。远端的机器人系统自己形成具有一定自主能力的独立闭环控制系统，操作者通过任务一级的命令来控制机器人系统，同时得到作业现场的反馈。时延环节并不存在于机器人系统自身的闭环回路中，因此时延对系统的稳定性不造成影响。机器人接到命令后，根据它进行自己的任务规划，并自主地执行任务。

作为解决通讯实验技术的一部分早在60年代就在空间作业中担任重要角色，现在的空间与深海遥操作系统更是广泛采用监督控制来解决问题。目前，监督控制仍然是解决时延问题的主要手段，特别是在空间与深海作业中存在长达几秒或几十秒时延的情况下，其它方法都不能胜任，结合预测显示等方法，Supervisory控制可以取得非常好的效果，ROTEX计划等空间遥操作机器人项目都采用了Supervisory控制方法。基本的Supervisory控制结构如图2.6所示。

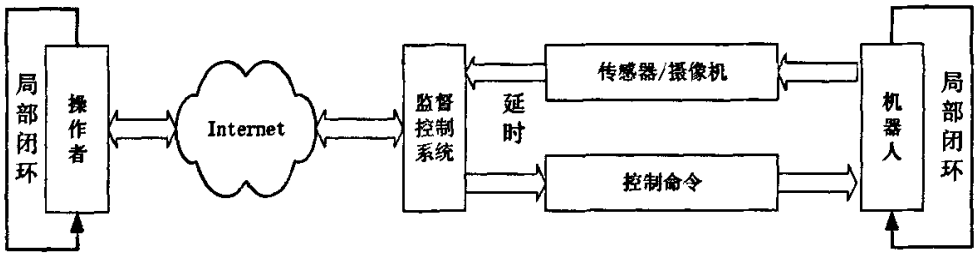


图2.6 遥操作系统监督控制方法结构示意图

目前，监督控制仍是解决问题的主要手段，特别是在空间与深海作业中存在长达几秒或几十秒时延的情况下，其他方法都不能够胜任。

2.2.3 预演预测模式

这种方法是在将控制指令发给远端之前，先在本地模型对象上进行模拟预演指令执行结果，这样就大大提高了动作执行的可靠性和正确性。在时延系统中，预测显示经常被用来解决时延对视觉反馈造成的影响。Sheridan[24]， Hirzinger[25]等人通过不同的实验对预测显示技术进行了研究，证明了该方法的有效性。早期的预测显示仅限于对视觉信息的预测，采用的方法主要有两类[24,26]

（1）简单地对状态、时间关系的Taylor展开；

（2）根据操作的输入、输出以及作业对象的模型进行预测。

前者只能够满足时延较小的场合，对于时延长达几秒的时延系统通常采用后者。随着计算机技术与控制理论等相关技术的发展，预测显示系统不仅能够提供视觉上的预测，并且能够建立作业环境的动力学模型来提供对操作过程中的力的预测。近年来，结合虚拟现实技术，这一方向发展成为虚拟环境建模技术，在实际系统操作前可以对虚拟环境进行操作。结合监督控制，通过对逼真的虚拟环境的操作来得到对从操作手一端的控制命令，可以解决操作者的视觉与力觉反馈问题。其基本结构如图2.7所示：

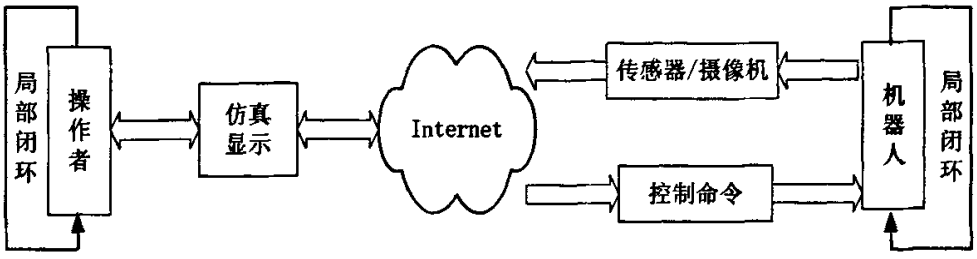


图 2.7 遥操作系统预测/预演控制方法结构示意图

2.2.4 事件驱动模式

事件智能控制方式不再以时间作为参考域，而是以参考事件 s(n)为变化域。由于网络上存在不确定的传输时滞，可能导致一般的基于时间的控制系统的不稳定，所以要消除网络延迟作用就需要采用非时间参考系统。基于事件的远程控制系统原理如图 2.8 所示。行为规划模块根据操作者的事件要求 Z (s(n))与从操作手的状态 y(s(n))来规划伺服系统的参考输入，当一个事件被处理完后再生成下一个事件 s(n+l)。

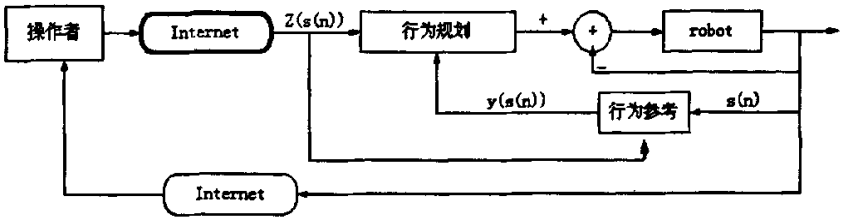


图 2.8 基于事件的智能控制结构[27,28]

以上几种控制方法中，监督控制是比较成熟的远程控制方式，一般应用于存在大时延的远程控制系统。预测/预演显示控制方式能在一定程度上解决时延问题，这对于远程遥操作方面有特别意义，但预测/预演控制系统较为复杂，需要对环境精确建模。基于事件的智能控制是目前比较有前景的控制模式，它能够避免时延的影响，但是它要求服务器端有更强的智能控制能力，技术还很不成熟，理论体系也没有建立。

2.3 采用的不同客户端实现的分析和研究

现有的基于互联网的机器人远程控制系统一般采用两种客户端实现，一种是客户端/服务器（C/S）端应用模式，一种是浏览器/服务器（B/S）应用模式,下面详细的讨论它们的各自的优点和不足，对本系统采用的方式有鉴见意义。

2.3.1 客户机/服务器（C/S）网络应用模式

传统远程控制系统一般采用客户机/服务器模式。其最初的结构为二层应用结构，系统由客户端（Client）应用程序和服务器（Server）组成。客户机是指运行用户服务请求程序，并将这些请求传送到服务器的计算机。服务器是指管理数据资源，响应并受理由客户机发出的请求，并将计算结果传送给客户机的计算机。服务器可接受多个客户机的多个请求，将请求排队或同时处理。服务器用于运行服务器程序，响应并执行来自前端客户的服务请求，最后向前端返回计算结果；而客户机，前端运行客户端程序，向服务器发出请求。二层应用结构的优点在于[29]：

（1） 客户机和服务器基于网络通信机制实现了任务的分工和资源的共享。

（2） 通过面向连接的网络协议，在保证通信带宽的前提下，客户机/服务器结构提高了系统的实时性；

（3） 能够满足用户端和服务器端之间的双向功能触发；

（4） 软件模块的分发授权容易控制。

随着网络技术的发展，在二层应用结构的基础上，将企业逻辑从客户端划分出来形成了客户机/服务器结构的三层应用结构，主要分为：用户界面、企业逻辑、数据库。其具体结构如图 2.9 所示。

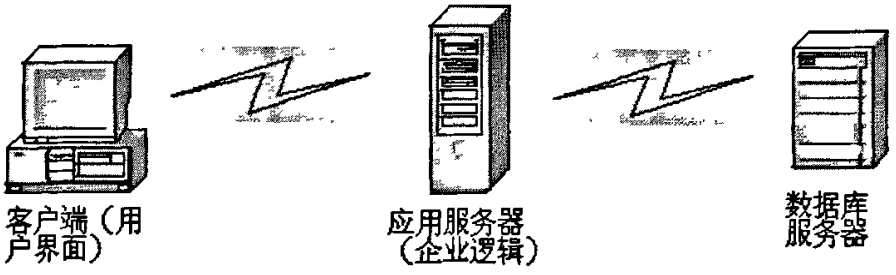


图 2.9 客户机/服务器系统三层结构模型

客户端主要是用户界面，而应用服务器中存放的是企业逻辑，即对数据库的一切操作。客户端通过代理对象向应用服务器提出对数据库操作的请求，由应用服务器执行相应的数据库操作并将结果返回到客户端。三层客户机/服务器结构与传统的一层客户机/服务器结构相比优势主要在：

（1）安全性增强：由于应用服务器把客户端与数据库服务器分开了，客户不能直接访问数据库服务器。增强了系统的安全性。

（2）效率提高：应用服务器减轻了客户端的负担，也降低了数据库服务器的连接

代价。

（3）可伸缩性：三层结构强调逻辑意义而不是物理意义，因此它们的硬件系统构成不受限制，具有很大的灵活性。

（4）易于维护：由于应用逻辑被封装到了应用服务器中。当应用逻辑发生变化，仅需修改应用服务器中的程序，客户端的程序无需更新，维护代价大大降低。

虽然客户机/服务器结构将数据存放在服务器上，应用程序存放在客户端计算机上解决了数据共享等问题，但却具有不易维护、操作风格不一致、处理的数据类型不易扩展等缺点。尽管后来将企业逻辑从客户端划分出来，但随着网络应用系统的发展，传统客户机/服务器模式应用系统的缺点还是不断地暴露出来：

（1）客户机/服务器结构的软件系统中各功能模块都是专用型的，系统功能的表现依赖于各功能模块的组合装配关系，这种装配关系会随着业务空间位置的延伸而显得烦琐和困难，系统的发布方式有很大的局限性；

（2）客户机/服务器系统大多固定于一种运行平台，平台间的可移植性差；

（3）开发周期长，升级维护困难。

2.3.2 浏览器/服务器（B/S）多层分布式系统模式

由于客户机/服务器应用结构的局限性，随着网络技术的发展，促使了浏览器/服务器模式应用系统的产生和发展。浏览器/服务器模式应用系统由浏览器（Browser）和服务器（WebServer，OtherServer，Middle Ware）组成。数据和应用程序分别存放在应用服务器和数据库服务器上，浏览器可以通过 Web 服务器调用应用服务器程序，从而得到动态的结果。

典型的浏览器/服务器结构的应用主要由四大部分组成：浏览器、Web 服务器、应用服务器、数据库服务器及其它企业系统。其具体结构如图 2.10 所示。

随着网络技术的发展和互联网的迅速普及，网络业务信息交互平台越来越多地选择基于浏览器/服务器结构。浏览器/服务器结构具有开放性、灵活性和易用性的特点。以浏览器/服务器模式开发的系统维护上作集中在服务器上，客户端不用维护，操作风格比较一致，只要有浏览器的合法用户都可以十分方便地使用。对于用户来说，面对的是界面统一、易于操作并且与平台无关的浏览器，不需要接受过多的操作培训，也不需要经历烦琐复杂的配置过程，在任何联网的地方都可以处理业务：对于开发者来

说，毋需开发专用的客户端软件，客户端功能模块经过 Web 服务器就可以发布，系统的升级和维护都变得简单化。如果解决了浏览器/服务器结构中的实时通信问题，就可以构造一个以浏览器作为通用客户端的浏览器/服务器结构的远程监测系统，监测人员可以在任何与互联网相连的地方进行实时监测活动，这无疑会以一种低成本的方式拓展监控系统的空间分布范围。

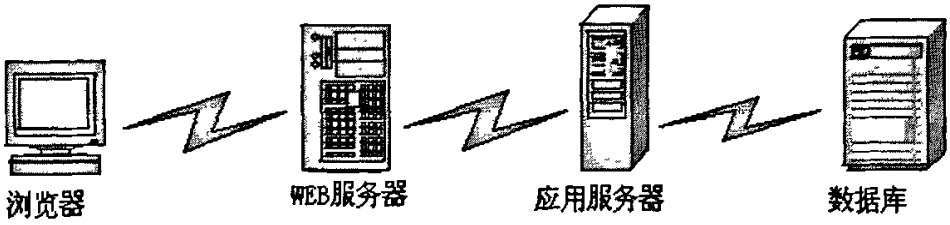


图 2.10 B/S 应用结构

2.4 机器人服务器端信息反馈策略的分析和研究

客户端用户在本地端通过本地端计算机发出控制指令，指令经过互联网传送到机器人服务器，由远程端机器人服务器控制的机器人完成操作者指定的任务后，机器人服务器将指令的执行结果和现场一些状态信息以一定形式传送到客户端，使客户能掌握指令的执行情况和测量现场的环境，从而能够发出进一步的控制命令。为了能让操作者及时得到作业环境以及被操作机器人运行的情况，从而能准确地操作机器人，必须为操作者在本地端建立一定程度的视觉临场感，即现场环境在客户端的模拟或重现。在现有的研究中，主要采取两种信息反馈技术：现场视频信息反馈技术，虚拟现实或仿真技术。

2.4.1 视频反馈

在机器人现场安置视频采集装置，将现场的图像传给客户端，使操作者能掌握机器人作业现场的情。这一方法存在着一定的缺陷：图像质量会受到现场光线情况和能见度等因素的影响；视频图像的信息量很大，由于网络带宽的影响，图像的传输会造成大的延时，使得机器人现场信息反馈的实时性大大降低，造成系统操作效率下降，但是随着网络技术的发展，高速光纤通讯的接入，这些影响将会大大降低。采用视频反馈能够真实的反映机器人现场的环境，给控制者提供最直观的信息。

2.4.2 虚拟现实

采用仿真或虚拟现实技术，在本地端建立基于立体视觉的虚拟作业环境，操作者可以利用生成的虚拟作业环境，观测、监控被操作对象。虚拟环境是依据机器人现场环境的一些特征信息而不是图像并借助于相应的建模技术和设备建立起来的。能使操作者产生一种“临场感”进而对远程端设备进行操作。为了使虚拟环境能及时反映作业现场的实际情况，需要不断地从服务器端反馈回机器人现场的特征信息以更新虚拟环境的显示。采用这种方法，可以为客户端用户提供三维的视觉感受，网络上的数据传输量比采用视频反馈方式时要小得多，整个远程操作系统的操作效率得到很大的提高。但是，一系列的建模过程是很复杂的，并且不能及时反映操作现场的动态变化。当机器人现场发生的改变后，必须重新建模[18]。

2.5 本章小结

本章归纳了目前几种网络远程控制系统体系结构的区别和特点，分析了不同控制系统在不同条件下的优点和不足。也对机器人控制模式进行了详细的分析，最后介绍基于 Internet 的网控机器人采用的 C/S、B/S 模式以及对现场信息信息进行反馈的几种方法。

第 3 章 机器人网络控制平台系统的详细设计

3.1 系统总体方案设计

3.1.1 系统采用的设计路线

根据第二章中对各个不同机器人网络控制平台的研究，总结了优缺点，提出了本系统平台。此机器人网络控制平台系统的设计采用的是分布式机器人网络控制系统，把单一服务器上的功能模块分解到其它的服务器上，这样既方便了机器人功能的扩展，也减少了数据库的负担，并且提高了整体的运行效率。本机器人网络系统控制平台按功能设计为如下几个服务器：应用程序服务器、视频服务器、数据库服务器、用户管理服务器和机器人控制服务器等。各个服务器主要功能如下：

（1）用户管理服务器：负责提供用户访问界面，远程注册服务 ，并进行访问权限设置和身份确认。在多用户进行访问时，维持用户队列 ，负责控制权的分配。任何时刻只能有一个用户获得机器人的控制权，某一用户获得机器人的控制权后，其他用户可以登陆观看机器人的运动；

（2）应用程序服务器：负责接受和发送客户端控制命令以及进行应用层协议命令的解析，并直接调用机器人服务器相应的控制模块；

（3）视频服务器：负责采集机器人现场场景图像 ，进行处理后将视频图像实时地传输给远端用户 ；

（4）数据库服务器：负责提供存储、管理系统运行中所要用到的数据 ，主要包括用户数据、系统信息数据和机器人运动状态数据；

（5）机器人控制服务器：主要提供本地的人机交互界面 ，可以使操作人员对MOTOMAN－HP3 机器人进行参数设置、任务指定等。机器人控制服务器通过串口或网线与 MOTOMAN HP3 机器人进行通讯。

考虑到现在开发的网控机器人平台的负载不是很重，主要对系统资源消耗比较大的是视频服务器的运行，而其他的服务器如（用户管理、应用程序、数据库和机器人控制等服务器）占用的系统资源对机器人控制系统的速度和控制上没有显著的影响，

因此在两台 PC 机器上实现此控制平台，一台机器运行视频服务器，一台运行其他的几个服务器。

整个控制系统的原理控制图和客户端、服务器端功能图如图 3.1。

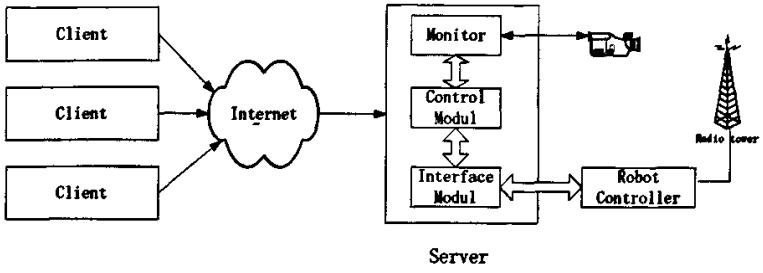


图 3.1 机器人网络控制系统平台

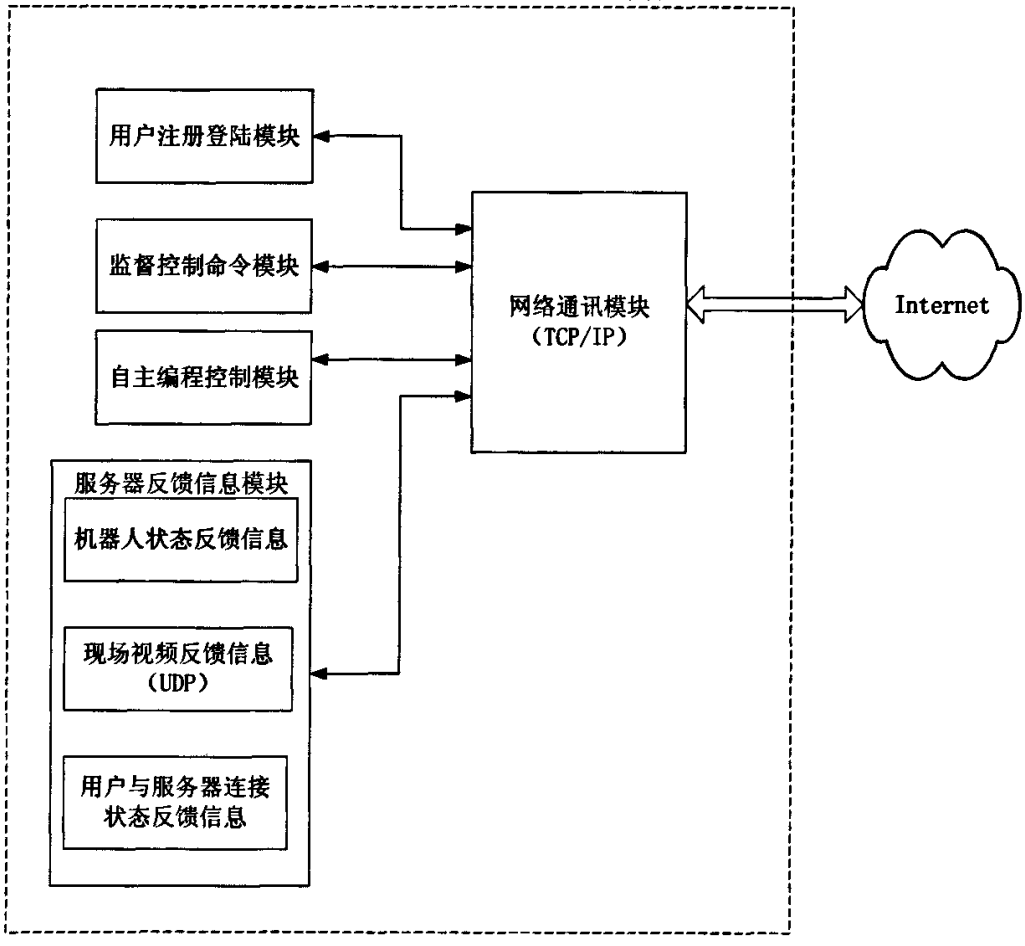


图 3.2 client 端功能图

如图 3.2 所示，本系统包括：用户注册登录模块，用来检验用户是否合法；监督控制命令模块，对机器人进行监督控制；自主编程控制模块，实现用户发送任务级的命令；服务器反馈信息模块，用来向服务器请求现场信息，并反馈给客户端；网络通讯模块，负责上面几个模块和服务器端的信息通讯。

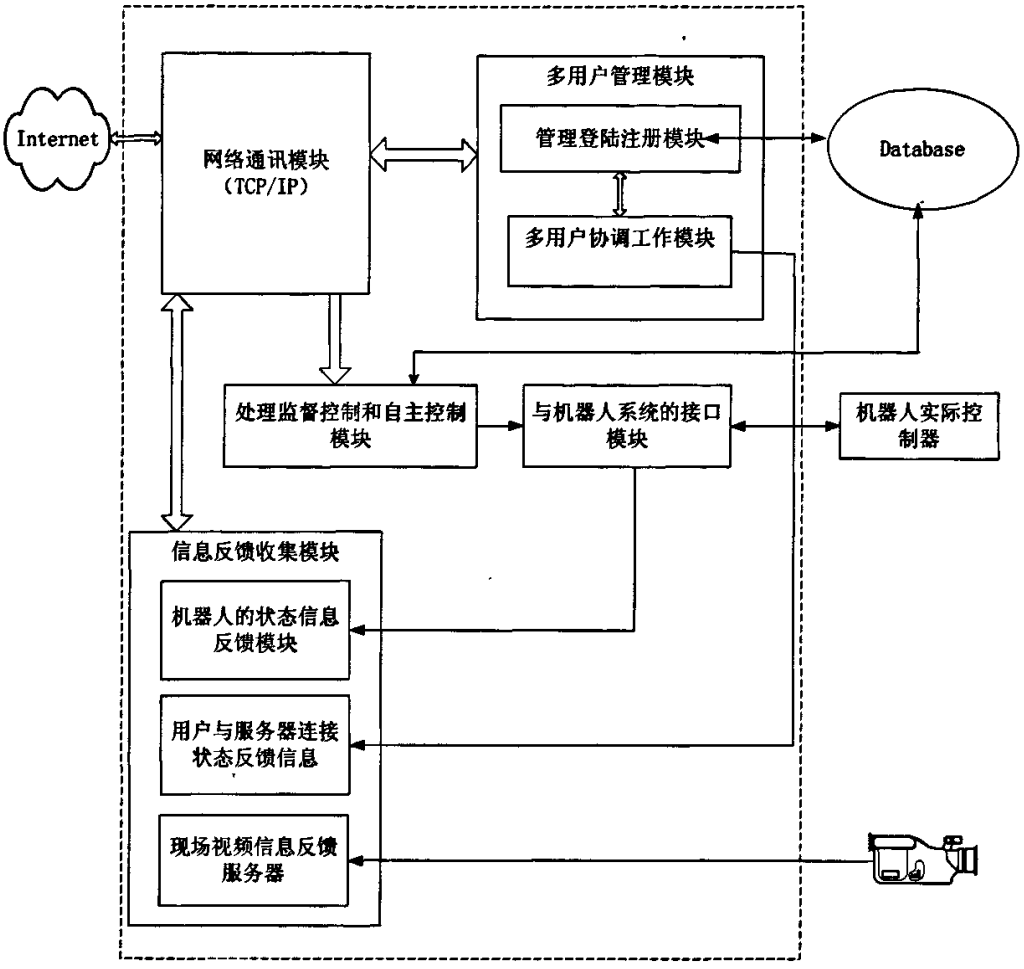


图 3.3 服务器端功能原理图

图 3.3 是服务器端的功能原理图。其中，网络通讯模块是来接受客户端的数据信息，并反馈服务器端的信息；多用户管理模块是用来检验用户合法性，以及用户是否具有机器人的控制权，主要是通过数据库的来进行数据的存储；处理监督和自主控制模块是用来针对不同的控制方式进行控制；机器人系统的接口模块用来发生指令给机器人，使机器人实现预定的动作；信息反馈收集模块用来获取机器人和服务器的信息以及现场视频的信息。

3.1.2 系统的硬件

（1） Motoman HP3 工业机器人[30,31]

本网络控制平台系统使用日本安川工业机器人 MOTOMAN HP3 作为实验研究对象，它的动作速度、精度及可靠性体现了机器人的先进水平。

MOTOMAN-HP 列机器人具有很快的轴动作速度。轻型机体和具备高轨迹精度控制及振动抑制控制的 NX100 控制柜的有机结合，减弱了机器器人启动和停止瞬间的颤动，从而缩短了机器人的运行周期。HP 系列的机器人操作机具备最大的工作半径和最小的干涉半径，工作范围变大，在系统设计上提供了较大的灵活性。机器人后方的工作范围的扩大，HP3 可以提供更广泛的应用方案，使夹具、剪丝机等设备都是更效的安装方式。HP3 机器人操作机采用 IP67 防护等级，手首部可采用防尘防水构造。这种特性可以保证机器人在恶劣环境中工作。

它的控制柜（NX100）运用高精度轨迹控制算法缩短了对命令响应的滞后时间。误差补偿功能使机器人绝对位置精度提高 2—5 倍。机器人的轨迹重复精度可提高 50%。NX100 控制柜可协调控制多达 36 个轴。它也为多机器人协调以及多个机器人接入网络控制平台创造了有利条件。

表 3.1 MOTOMAN-HP3 性能参数[31]

	机械结构	垂直多关节型（6 自由度）
	载荷重量	3kg
	定位精度	±0.03mm
	安装方式	地面安装
	本体重量	45kg
	电源容量*	1kVA
最大动作范围	S 轴（回旋）	±170°
	L 轴（下臂倾动）	+150°～ -45°
	U 轴（上臂倾动）	+210°～ -152°
	R 轴（手臂横摆）	±190°
	B 轴（手腕俯仰）	±125°
	T 轴（手腕回旋）	±360°
最大速度	S 轴	3.66rad/s，210°/s
	L 轴	3.14rad/s，180°/s
	U 轴	3.93rad/s，225°/s
	R 轴	6.54rad/s，375°/s
	B 轴	6.54rad/s，375°/s
	T 轴	8.73rad/s，500°/s
允许力矩	R 轴	7.25N·m
	B 轴	7.25N·m
	T 轴	5.21N·m
允许惯性力矩（$GD^2/4$）	R 轴	0.30kg·m²
	B 轴	0.30kg·m²
	T 轴	0.1kg·m²

MOTOMAN HP3 工业机器人的运动方式在不同的坐标系下会有不同的运动趋势，这些坐标系包括：直角坐标系、关节坐标系、圆柱坐标系、工具坐标系和用户坐标系，如图 3.4 所示。

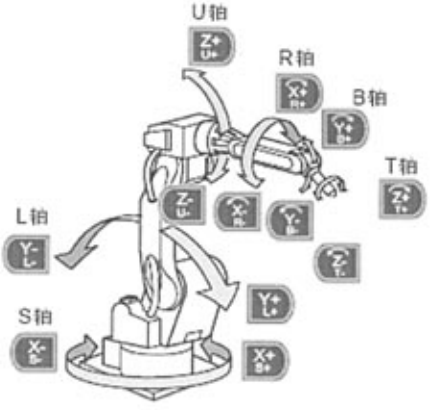
• 关节坐标系机器人各轴进行单独动作，称关节坐标系。

• 直角坐标系不管机器人处于什么位置，均可沿设定的 X 轴、Y 轴、Z 轴平行移动。

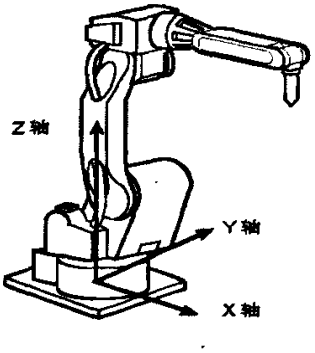
• 圆柱坐标系中，θ 轴绕 S 轴运动，R 轴沿 L 轴臂、U 轴臂轴线的投影方向运动，Z 轴运动方向与直角坐标完全相同。

• 工具坐标系工具坐标系把机器人腕部法兰盘所持工具的有效方向作为 Z 轴，并把坐标定义在工具的尖端点。

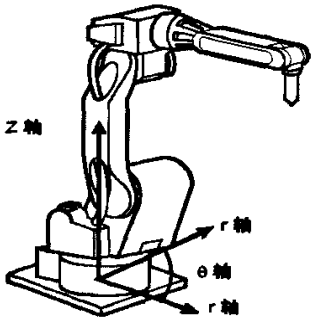
• 用户坐标系机器人沿所指定的用户坐标系各轴平行移动。在关节坐标系以外的其他坐标系中，均可只改变工具姿态而不改变工具尖端点（控制点）位置，



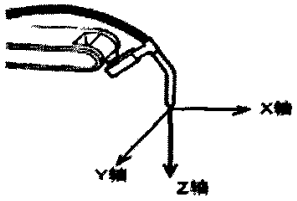
(a) 关节坐标系



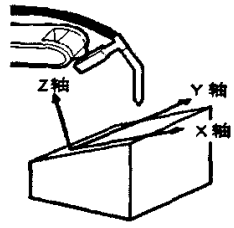
(b)直角坐标系



(c)圆柱坐标系



(d)工具坐标系



(e)用户坐标系

图 3.4 机器人的坐标系[30,31]

（2）视频采集系统

本系统采用的视频采集系统包括 CCD 摄像头和视频采集卡， 把视频采集系统接入视频服务器，通过视频服务器完成对现场信息的采集、编码和压缩，再发送到用户端。视频监控设备包括型号为 FC-903F 的 CCD 数码彩色摄像机和相应的四路视频采集卡，CCD 数码彩色摄像参数如下：

- 水平清晰度：480 线
- 最低照度：0.5LUX
- 电子快门：1/50-1/10000 秒
- 信噪比： 50DB

摄像机的输出通过图像采集卡完成视频图像的采集、暂存和传输的功能，然后把采集的数据在作为视频服务器的计算机上运行 Microsoft Windows Media Encoder 将捕获到的实况图像流转换编码成适合在 Internet 传输的流媒体格式，再由 WMS（Windows Media Server）向客户端发送现场的信息，用户或访问者通过系统的客户端视频显示模块观看到现场图像的反馈。

3.1.3 系统的结构和功能设计

（1）C/S 控制形式

我们从上文中已经分析了 B/S 和 C/S 两种客户端的实现，这两种控制模式都有自己的优点和缺点，根据不同的应用特点可以进行针对性的选择。由于本系统采用的是分布式架构的机器人控制平台系统，每一个功能都有相对的应用服务器来进行控制，而且此网络控制平台系统不仅针对大众的机器人爱好者，为他们提供一个开阔眼界、

提高对机器人科学的兴建，而且可以为专业的机器人研究者提供一个科研和学习平台，因此网络控制系统平台既提供了传统的监督模式控制，也提供了面向高级用户的自主任务编程控制，功能相对丰富。在本系统中采用 C/S 控制模式可以丰富客户端的功能，强化系统的科研能力。

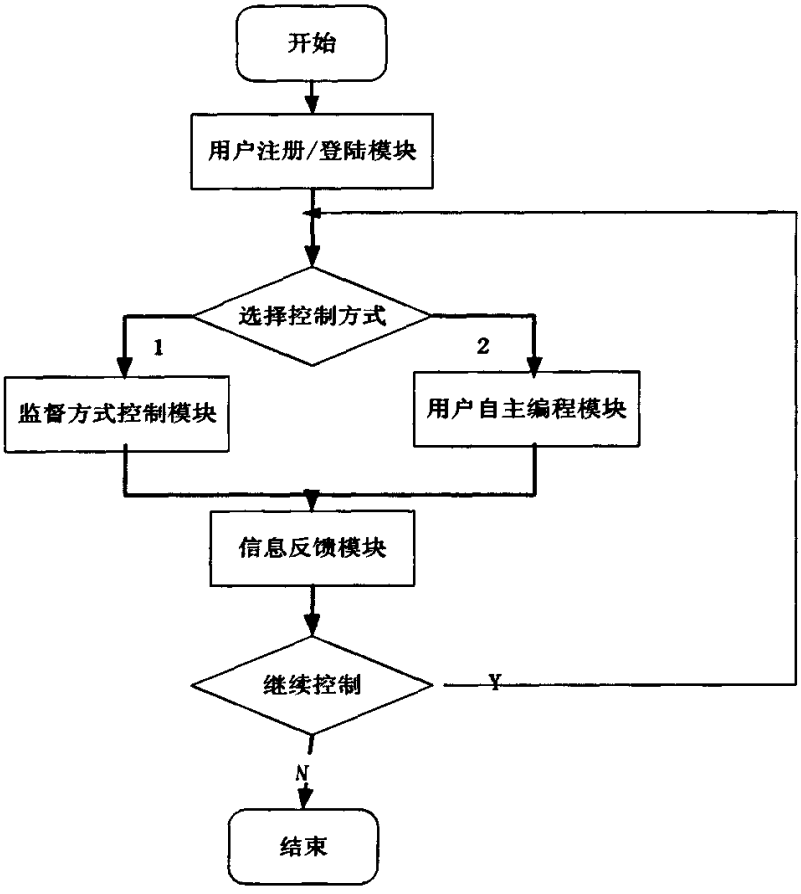


图 3.5　C/S 模式中 Client 流程图

Client 端的控制流程图如图 3.5 所示。用户先要进行身份验证，如果是合法的用户，会根据用户的权限具有相应的控制权限，进入监督控制或用户自主控制。注册用户都可以观看现场的信息和机器人的信息。

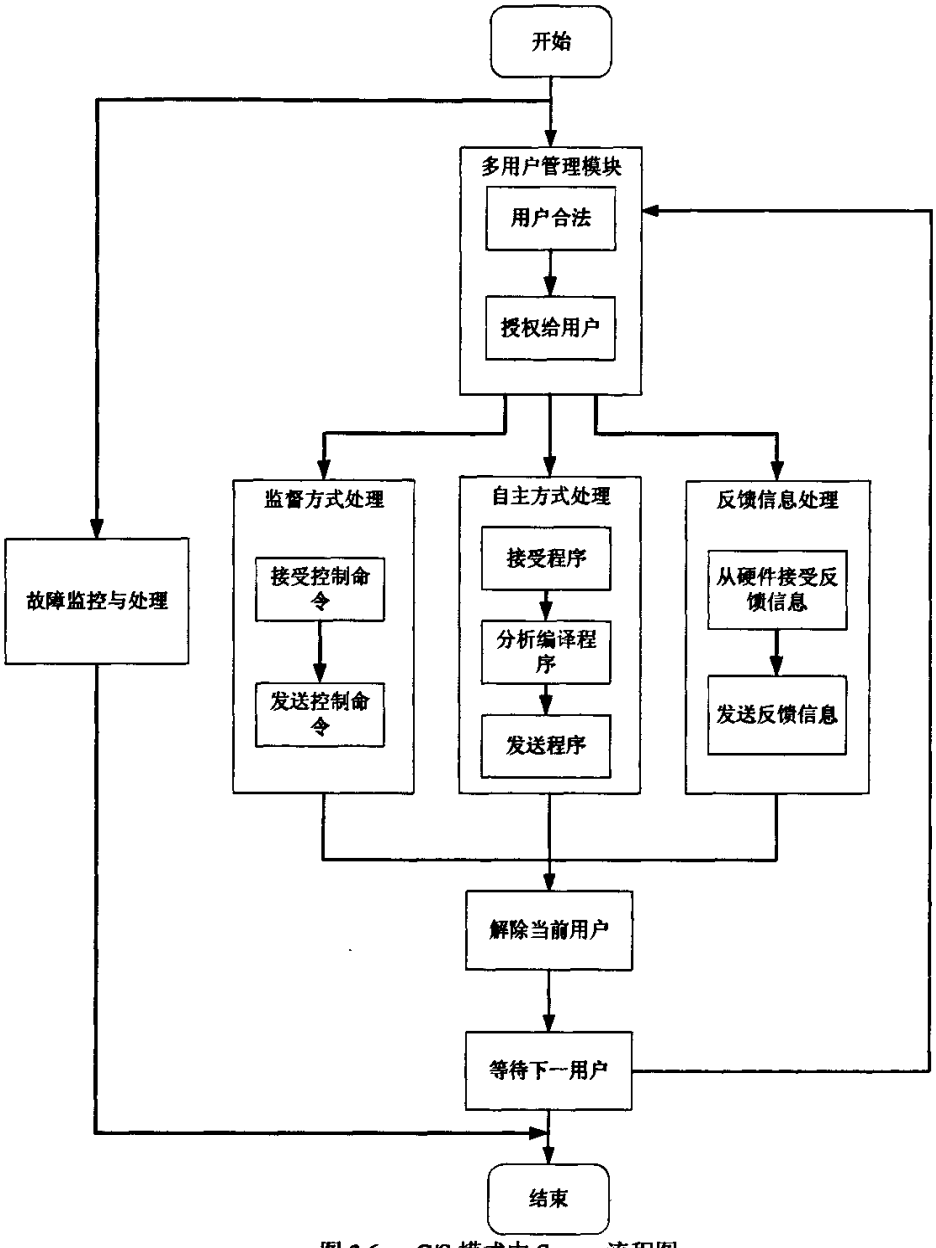


图 3.6　C/S 模式中 Server 流程图

Server 端的流程图如图 3.6 所示，首先根据客户端发送的用户信息进行验证，是合法用户就进入相应的控制模块，进行控制，等待一定的时间，就会释放当前用户的控制权，转交给下一个等待用户，发生异常就调用故障监控处理模块。

（2）监督方式控制

监督方式控制将远程操作人员置于控制结构之外，从而达到减小传输时延对整个系统的影响。机器人系统自己形成具有一定自主能力的独立闭环控制系统，时延环节并不存在于机器人系统自身的闭环回路中，因此时延对系统的稳定性不造成影响。机

器人接到命令后，自主得执行相应得命令。

目前，监督控制仍然是解决时延问题的主要手段，特别是在空间与深海作业中存在长达几秒或几十秒时延的情况下，其它方法都不能胜任。由于监督控制方式在网控机器控制系统的重要地位和它的理论地位，此网络平台也采用了这种方式，并且进行了扩展。

此网络控制平台的监督控制从以下几个方面进行了实现：

（1）采用以直角坐标系作为机器人的空间坐标系，让机器人接受控制指令在 XYZ 三个坐标为基准的空间运行。采用直角坐标系即可以面对移动机器人，又可以针对工业机器人进行控制。

（2）采用关节坐标系为机器人的空间坐标系，让机器人的每一个关节独立的运行，此种方式可以对研究工业机器人的空间运动学起到很大的作用。

（3）针对机器人任务 JBI 文件的特征分析，完成了多个单条指令自动生成任务的功能。

监督控制模块的信息发送和接受流程如图 3.7、3.8 所示。

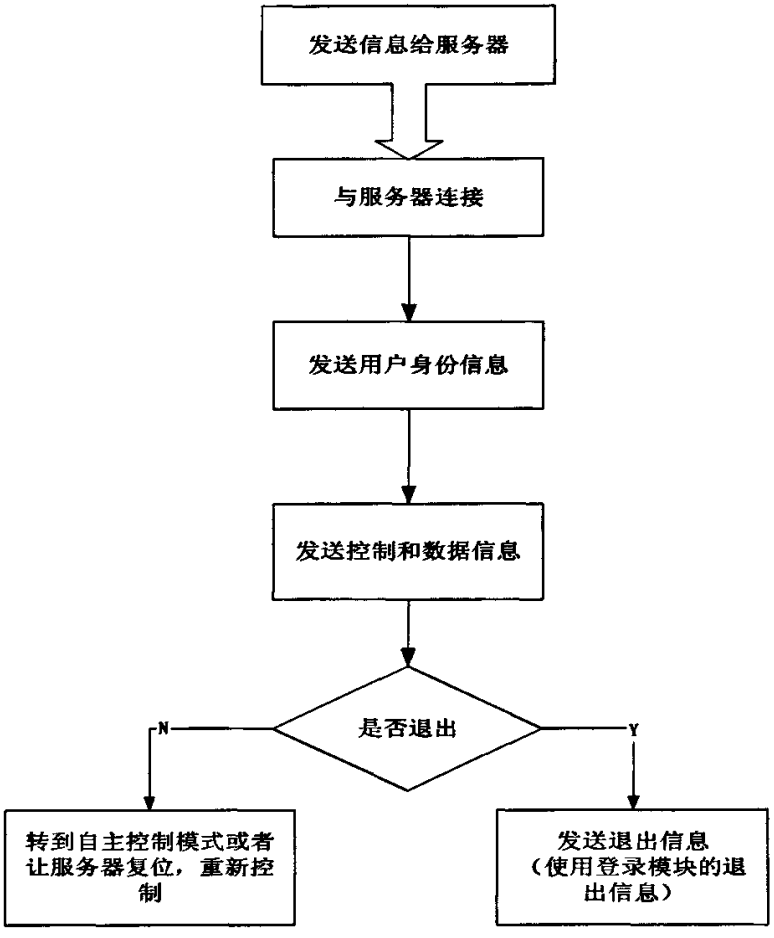


图 3.7　监督控制模块发送信息的流程

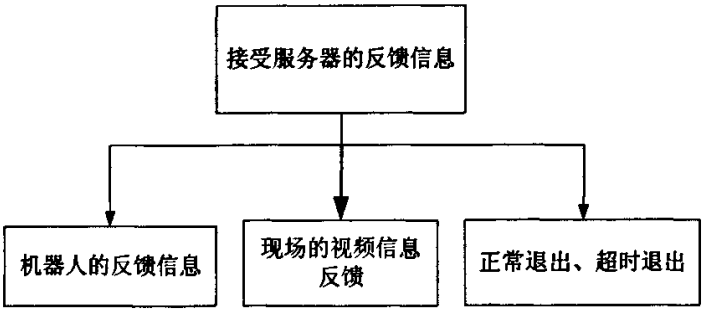


图 3.8 监督控制模块接受信息的流程

（3） 用户自主编程模块

用户自主编程模块主要是面向机器人研究的专业人士，此系统为机器人研究人员提供了一个可以对机器人进行二次开发的平台，可以在此平台上直接发送机器人本体控制程序，对机器人进行任务级别的控制。此种控制方式只是针对特殊优先级的用户开放，需要经过特别授权。

3.2 网控机器人数据传输机制[32,33,37]

网控机器人系统中客户端和服务器端、以及各个应用服务器之间的通讯采用的是基于 TCP/IP 协议的 Windows Socket 通讯。下面将介绍一下 TCP 和 UDP 协议的原理和区别以及 Windows Socket 的通讯机制。

TCP/IP 是一组不同的协议组合在一起构成的协议族。尽管通常称该协议族为 TCP/IP，但 TCP 和 IP 只是其中的两种协议而已。网络协议通常分不同层次进行开发，每一层分别负责不同的通信功能。一个协议族，比如 TCP/IP，是一组不同层次的多个协议的组合。TCP/IP 是一个四层协议系统，每一层负责不同的功能[32]：

（1）链路层，有时也称作数据链路层或网络接口层，通常包括操作系统中的设备驱动程序和计算机中对应的网络接口卡。它们一起处理与电缆（或其他任何传输媒介）的物理接口细节，负责底层 01 码流得数据传送。

（2）网络层，有时也称作互联网层，处理分组在网络中的活动，例如分组的选路。TCP/IP 协议族中，网络层协议包括 IP 协议（网际协议），ICMP 协议（Internet 互联网控制报文协议），以及 IGMP 协议（Internet 组管理协议）。

（3）运输层主要为两台主机上的应用程序提供端到端的通信。包括 TCP 协议和 UDP 协议，TCP 协议提供的是面向连接的可靠的服务，UDP 的实时性较好，一般用于大数据的实时传播，但是不提供数据重传机制。

（4）应用层负责处理特定的应用程序细节。

3.2.1 TCP 协议和 UDP 协议

TCP（传输控制协议）和 UDP（用户数据报协议） 是 TCP/IP 协议族中运输层上两个互不相同的传输协议。TCP 为两台主机提供高可靠性的数据通信。它所做的工作包括把应用程序交给它的数据分成合适的小块交给下面的网络层，确认接收到的分组，设置发送最后确认分组的超时时钟等。由于运输层提供了高可靠性的端到端的通信，因此应用层可以忽略所有这些细节。而另一方面，UDP 则为应用层提供一种非常简单的服务。它只是把称作数据报的分组从一台主机发送到另一台主机，但并不保证数据报能到达另一端。任何必需的可靠性必须由应用层来提供。这两种根据它们的特点分别在不同的应用程序中有不同的用途。

在 TCP/IP 协议族中，网络层 IP 提供的是一种不可靠的服务，它只是尽可能快地

把分组从源结点送到目的结点，但是并不提供任何可靠性保证。TCP 在不可靠的 IP 层上提供了一个可靠的运输层。为了提供这种可靠的服务，TCP 采用了超时重传、发送和接收端到端的确认分组等机制。TCP 和 UDP 是两种最为著名的运输层协议，二者都使用 IP 作为网络层协议。虽然 TCP 使用不可靠的 IP 服务，但它却提供种可靠的运输层服务。UDP 为应用程序发送和接收数据报。一个数据报是指从发送方传输到接收方的一个信息单元（例如，发送方指定的一定字节数的信息）。但是与 TCP 不同的是，UDP 是不可靠的，它不能保证数据报能安全无误地到达最终目的。

由此可见， TCP 协议是面向连接的协议，它比较安全、稳定，而此网络控制平台系统在监督模式和自主任务编程模式对数据的准确性要求较高，因此采用了 TCP 为数据传送的协议。UDP 协议系统开销小、速度快、效率高、占用资源少，本系统中的机器人状态信息和即时消息的发送采用的是 UDP 协议。

3.2.2 Windows Sockets 编程原理

TCP/IP 协议在异网互联中体现出了其强大的生命力，以它为基础组建的 Internet 是目前国际上规模最大的计算机网间网。与计算机网络的普及相呼应的是 Windows 的广泛应用，自 Windows 正式推出以来，现在在世界各地已有千万计的用户在使用不同版本的 Windows。这说明以用户友好的图形界面为基础的 Windows 已得到用户的普遍认可，已经并将继续成为个人机平台上的事实上的操作系统标准。所以研究和开发在 Windows 下的网络编程技术具有普遍的应用价值。

在 Windows 下的各种网络编程接口中，Windows Sockets 脱颖而出，越来越得到大家的重视，这是因为 Windows Sockets 规范是一套开放的、支持多种协议的，已成为 Windows 网络编程的事实上的标准。

通讯的基石是套接字，一个套接字是通讯的一端。在这一端上你可以找到与其对应的一个名字。一个正在被使用的套接字都有它的类型和与其相关的进程。套接字存在于通讯域中。通讯域是为了处理一般的线程通过套接字通讯而引进的一种抽象概念。Windows Sockets 规范支持单一的通讯域，即 Internet 域。各种进程使用这个域互相之间用 Internet 协议族来进行通讯。

用户目前可以使用两种套接字，即流套接字和数据报套接字。流套接字提供了双向的，有序的，无重复并且无记录边界的数据流服务（基于 TCP 协议）。数据报套接

字（基于 UDP 协议）支持双向的数据流，但并不保证是可靠，有序，无重复的。也就是说，一个从数据报套接口接收信息的进程有可能发现信息重复了，或者和发出时的顺序不同。数据报套接字的一个重要特点是它保留了记录边界。对于这一特点，数据报套接字采用了与现在许多包交换网络（例如以太网）非常类似的模型。

套接字有三种类型:流式套接字，数据报套接字及原始套接字[32]。

（1）流式套接字

流式套接字定义了一种可靠的面向连接的服务，实现了无差错无重复的顺序数据传输。面向连接服务器处理的请求往往比较复杂，不是一来一去的请求应答所能解决的，而且往往是并发服务器。使用面向连接的套接字编程，其时序可以通过下图 3.9 来表示:

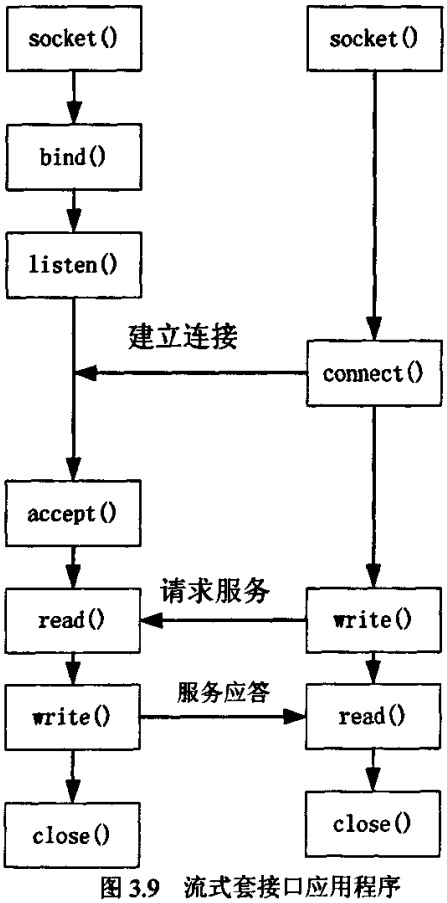


图 3.9 流式套接口应用程序

套接字工作过程如下:服务器首先启动，通过调用 Socket( )建立一个套接字，然后调用 bind( )将该套接字和本地网络地址联系在一起，再调用 listen( )使套接字做好侦听的准备，并规定它的请求队列的长度，之后就调用 accept( )来接收连接。客户在建立套接字后就可调用 connect( )和服务器建立连接。连接一旦建立，客户机和服务器之间

就可以通过调用 read( )和 write( )来发送和接收数据。最后，待数据传送结束后，双方调用 close( )关闭套接字。

（2）数据报套接字

数据报套接字定义了一种无连接的服务，数据通过相互独立的报文进行传输，是无序的，并且不保证可靠，无差错。无连接服务器一般都是面向事务处理的，一个请求一个应答就完成了客户程序与服务程序之间的相互作用。若使用无连接的套接字编程，程序的流程可以用图 3.10 表示。

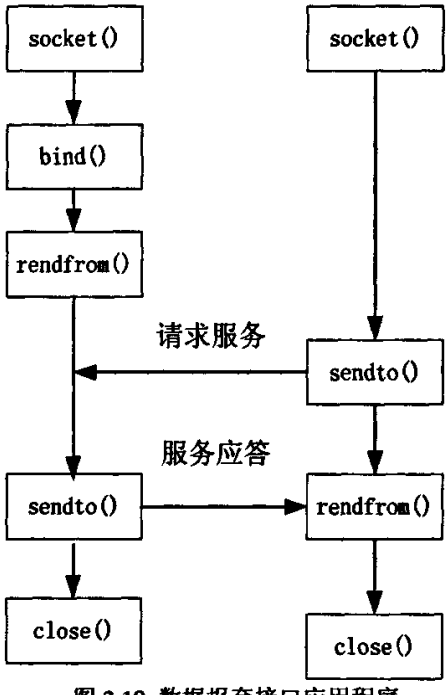


图 3.10 数据报套接口应用程序

（3） 原始套接字

原始套接字允许对低层协议如 IP 或 ICMP 良接访问，主要用于新的网络协议实现的测试等。Windows Sockets 规范并没有规定 Windows Socket 必须支持原始套接字。然而 Windows Sockets 规范鼓励 Windows Sockets DLL 提供原始套接字支持。

3.2.3 Windows Sockets 编程接口

Windows Sockets 规范以 U.C. Berkeley 大学 BSD UNIX 中流行的 Socket 接口为范例定义了一套 Microsoft Windows 下网络编程接口。Windows Sockets 规范本意在于提供给应用程序开发者一套简单的 API，并让各家网络软件供应商共同遵守。此外，在一个特定版本 Windows 的基础上，Windows Socket 也定义了一个二进制接口（ABI），

以此来保证应用 Windows Sockets API 的应用程序能够在任何网络软件供应商的符合 Windows Sockets 协议的实现上工作。因此这份规范定义了应用程序开发者能够使用，并且网络软件供应商能够实现的一套库函数调用和相关语义。

3.2.4 Windows Sockets 编程特点

由于 Window:的基于消息的特点，Windows Sockets 有如下一些新的特点:

（1）异步选择机制

异步选择函数 WSAAsyncSelect( )允许应用程序提名一个或多个感兴趣的网络事件，如 FD_READ. FD_WRITE， FD_ CONNECT， FD_ ACCEPT 等等代表的网络事件。当被提名的网络事件发生时，Windows 应用程序的窗口函数将收到一个消息，这样就可以实现事件驱动了。

（2）异步请求函数

异步请求函数允许应用程序用异步方式获得请求的信息，如 WSAAsyncGetXBYY( )类函数。这些函数是对 BSD 标准函数的扩充，函数 WSACancelAsyncRequest( )允许用户中止一个正在执行的异步请求。

（3）阻塞处理方法

WINSOCK 提供了”钩子函数”负责处理 Windows 消息，使 Windows 的消息循环能够继续。WINSOCK 提供了两个函数（WSASetBlockingHook() 和 WSAUnhookBlockingHook( )）让应用程序设置或取消自己的”钩子函数”。函数 WSAIsBlocking( )可以检测是否阻塞，函数 WSACancelBlockingCall( )可以取消一个阻塞的调用。

（4）错误处理

WINSOCK 提供了两个 WSAGetLastError( )和 WSASetLastError( )来获取和设置最近错误号。

（5）启动和终止

由于 Windows Sockets 的服务是以动态连接库 WINSOCK.DLL 形式实现的，由于以必须要先调用 WSAStartup( )函数对 Windows Sockets DLL 进行初始化，协商 WINSOCK 的版本支持，并分配必要的资源。在应用程序关闭套接口后，还应调用 WSACleanup( )终止对 Windows Sockets DLL 的使用，并释放资源，以备下一次使用。

3.3 客户端和服务器端具体实现

3.3.1 网络通讯模块－设计基于 TCP 协议的网络机器人数据控制协议

机器人网络控制系统平台中，网络传输的信息主要有两类，一是字符串、文本类信息，另一为视频类的实时媒体信息。

（1）字符类信息

机器人的控制指令和位置反馈信息都属此类。这类信息要求数据传输正确、可靠。根据上文中介绍，TCP（Transmission ControlProtocol，传输控制协议）是一个具有重传机制和拥塞控制机制的面向连接的协议，具有高度可靠性，因此系统采用它作为字符类信息的传输协议。

字符流的通信选用 Socket 机制来实现。由于 Soeket 是将用户代码和协议底层实现隔离开的编程对象，并且提供了连接任何流到通信接口的方法，这样就可以通过输入输出流来抽象网络层的具体操作，把应用程序与网络技术细节分开，只需考虑流的读写操作而不用关心网络的具体细节。

（2）媒体类信息

由于多媒体传输对实时性要求高，系统采用基于 UDP（User Data Protocol 用户数据报协议）的 RTP（Realtime Transport Protocol.实时传输协议）传输。RTP 是一个位于 UDP 之上的应用型传输层协议，本身只保证实时数据的传输，并不能为按顺序传送数据包提供可靠的传送机制，也不能提供流量控制或拥塞控制。RTP 虽然没有 TCP 那么可靠，而且需要 RTCP（ Realtime Transport Control Protocol.实时传输控制协议）实时监控数据传输和服务质量。但是由于它传输时延低于 TCP。因此能够与音频和视频更好地配合。而且 RTP 和 RTCP 配合使用能以有效的反馈和最小的开销使传输效率最佳。故特别适合传送网上的实时数据。具体的详细方法会在网络视频传输模块中进行阐述[33]。

基于 Internet 进行控制的机器人控制信息和反馈信息比较复杂，针对不同的机器人有不同的控制命令和控制形式。因此如果需要在系统中接入多种不同的机器人控制系统，很有必要设计建立一个自己的机器人远程控制协议。

本文在此提出一个基于 TCP 协议之上的封装的网络机器人数据控制协议，协议的具体数据结构如表 3.2 所示：

表 3.2 机器人的网络数据传输协议

ControlTypeInformation 控制类别信息	UserName 用户名	Password 密码	LoginType 登录类型	Pivilege 用户权限级别
CurrentlyState 当前用户控制状态	RobotType 机器人类别	ControlModel 控制模式	ControlSign 控制信号	SpeedSign 速度信号
Robot Satus Data 机器人状态数据			Robot Satus Data 机器人状态数据	
Robot Position Data 机器人点位数据			Robot Position Data 机器人点位数据	
Robot Position Data 机器人点位数据			Robot Position Data 机器人点位数据	
Robot Position Data 机器人点位数据			Robot Position Data 机器人点位数据	
ErrorInformation 错误信息码				

本系统的所以的数据流都是以此为格式，在 Internet 上进行传输，下面是每一个字头的详细内容：

3.3.2.1 控制类别信息

控制类别信息是务器端用于识别客户端发送何种类型的信息，根据不同的类别，分别发送给相应的服务器的功能处理模块。可以进行扩展。

数据定义：

```
enum    ControlTypeInformation
        {
          InitialInformation:=0,
          UserGetInformation,
          ControlInformation,
          IndependenceInformation
        };
```

其中：

InitialInformation： 表示初始化信息数据包，此时数据包含有用户名和密码等登录和注册的必要消息。如果服务器端接受到此数据包，将会把此数据包发送给用户管理模块，由用户管理模块进行处理。

UserGetInformation： 表示客户端请求获得服务器状态和机器人状态信息的数据包。服务器端接受到此信息，将会调用服务器状态反馈模块，把相应的机器人的位置信息和系统的信息定时反馈给用户。

ControlInformation：表示用户请求进行监督控制的数据包，数据包中含有具体的位置信息和控制信息。服务器端接受数据包后，将调用机器人监督控制模块进行解析。

IndependenceInformation： 表示用户请求进行自主任务控制的数据包，此数据包中包含了任务的名字和任务大小。服务器端接受此数据包后，会调用任务自主控制模块进行解析。

3.3.2..2 登录类型

表示用户登录的状态，分别是注册新用户，登录，修改用户，退出系统。

```
enum   LoginType
       {
        Register=1，Login，Modify，Exit
        };
```

3.3.2.3 用户权限级别

Pviliege 表示的是用户的级别，不同级别的用户拥有不同的控制权限。

数据定义：

```
enum    Privilege
{
    Idle=0，
    CommandAndEmluator，
    CommandAndVideo，
    IndependenceAndEmluator，
    IndependenceAndVideo
};
```

其中：

Idle： 属于普通用户，不能对系统进行控制，只能通过客户端进行现场视频信息的获取和机器人状态的获取。

CommandAndEmluator： 表示可以进行命令控制模式，当接入虚拟现实系统后使用，反馈信息为仿真图像。对机器人是模拟操作，属于仿真环节。

CommandAndVideo： 表示可以进行命令控制模式，反馈信息为实时视频，对机器人进行实际操作。

IndependenceAndEmluator： 表示可以进行自主控制模式，当接入虚拟现实系统后使用，反馈信息为仿真图像。对机器人是模拟操作，属于仿真环节。

IndependenceAndVideo： 表示可以进行自主控制模式，反馈信息为实时视频，对机器人进行实际操作。

当前用户状态表示的是用户当前操作所处的状态。

3.3.2..4 机器人类别

表示接入机器人网控系统平台的机器人，当有多个机器人的接入时，用户需要选择具体的一个机器人来进行控制。本系统接入了两个 MOTOMAN HP3 机器人。

数据定义：

```
enum  RobotType
{
    motomanHp3_1=1，motomanHp3_2，robot3
};
```

3.3.2.5 控制模式（ControlModel）

指的是在监督控制下，采用的是直角坐标系（Control=0）还是关节坐标系（Control=1）作为机器人的空间坐标系。

3.3.2..6 控制信号

用在监督控制模式下采用直角坐标系是发送给服务器端的控制命令，分别表示直角坐标系下 XYZ 三个方向的位移。Pause、Reset、Continue 分别表示机器人进行暂停、复位和继续运行，在自主控制下发送的命令。此控制信号可以进行扩展。

数据定义：

```
enum ControlSign
{
    Up=1，Down，Right，Left，Fore，Back，Pause，Reset，Continue
};
```

3.3.2.7 速度信号

表示在监督控制模式下采用直角坐标系时，机器人每一次单位位移的大小，即机器人每响应一次命令所移动的位置量。

数据定义：

```
enum  SpeedSign
{   LowSpeed = 10, MidSpeed = 20, HighSpeed = 30};
```

3.3.2.8 机器人状态和位置数据的结构

```
struct STATUSINFORMATION
{
    CString     curJobName;
```

```
    short       curJobLine;
    short       curJobStep;
    WORD        d1;
    WORD        d2;
    short       error;
    double      xp[6];
    double      jp[6];
    WORD        rconf;
};
```

表示机器人系统当前的一些位置信息和任务信息，用来反馈给客户端。

```
struct POSTION
{
    bool        controlMode;
    CString     movtype;
    CString     vtype;
    double      spd;
    CString     framename;
    short       rconf;
    short       toolno;
    double      p[12];
};
```

表示的是机器人接受的位置控制信息和机器人运动状态信息。

3.3.2..9 服务器返回的错误信息数据结构。

```
enum ErrorInformation
{
SUCCESSFUL=0 ， NOTEXISTUSERNAME=1 ， ERRORPOSSWORD ，
N0HIGHPRIVEILEGE, NOTEXISTTTHISROBOT, HAVEANOTHERUSER,
CONTROLTIMEOUT
};
SUCCESSFUL=0            消息正确，没有发生错误
NOTEXISTUSERNAME=1,     不存在此登录用户名
ERRORPOSSWORD,          登录密码错误
N0HIGHPRIVEILEGE,       不具有此控制优先级
```

NOTEXISTTTHISROBOT，　不存在这一款机器人

HAVEANOTHERUSER，　其他的用户正在控制

CONTROLTIMEOUT，　控制机器人时间超时，服务器将取回控制权。

3.3.2..10 机器人网络控制平台系统的服务器和客户端通讯的控制数据流都是基于此传输协议组建的数据包。在客户端和服务器端有相应的协议类进行数据包的组建和解析，对传输的数据包进行辩解码。

```
class InfoFormat
{
public:
        …… 
    void RecoverOriginData (  );
    void AnalyzeString ( CString s );
    void ConveyToString (CString &str );
    void GetRobotStatus   (STATUSINFORMATION &status );
    void GetRobotPosition (POSTION &position );
    void SetRobotStatus (STATUSINFORMATION &status );
    void SetRobotPosition (POSTION &position );
        …… 
};
```

Information 类就是此协议的具体实现。其中，RecoverOriginData ( )方法是用来对协议进行初始化，AnalyzeString (Csring s)是用来分析接受到的数据包，进行解析成需要的控制信息、机器人状态信息和机器人位置信息。CoveyToString (CString &str)方法是用来组建要发送的数据包的。SetRobotStatus (STATUSINFORMATION &status) 和 GetRobotStatus (STATUSINFORMATION &status)两个方法分别是对数据包设置机器人当前状态和解析数据包中的机器人当前状态。SetRobotPosition (POSTION &position)和 GetRobotPosition (POSTION &position)；两个方法分别是对数据包设置机器人当前状态和解析数据包中的机器人当前的位置参数。

3.3.2 数据库通讯模块

本系统中我们采用了 SQL SERVER 2000 数据库来管理和存储用户的信息和用户使用机器人的记录信息以及机器人本体的一些防止冲突的和意外的限制数据。

SQL Server 2000 本身是个关系型数据库，它具有以一下特点[34,35]：

- 集成了数据转换服务，可以与其它数据库资源进行数据转换；
- 支持客户/服务器的数据操作模式，尤其是 SQL Server 2000 能够编程，可以采用本地缓存技术，能够解决数据量大的情况下网络传输的瓶颈问题。这也是现今最流行的操作方式；
- 网络独立性，SQL Server 2000 是独立于网络协议的，它可以和任何操作系统下的客户端通信，只要该操作系统使用符合工业标准的网络协议即可；
- SQL Server 2000 可以很方便地通过 Web 站点共享数据，使用户通过 Web 浏览器就能直接从 SQL Server 2000 数据中访问数据；
- 集中管理，无论企业中有多少个 SQL Server 服务器，也无论它们分布在什么位置，都可以在一个集中的位置来管理。

采用 SQL Server 2000 进行机器人远程控制数据的保存和处理，给解决网络环境下的数据传输问题提供了很好的方案。在机器人远程监控系统中，控制客户数据与机器人实时状态数据是分立的，加上机器人的限制数据，因此需要建立三个数据表，二个数据表中数据的用户是关联的，表 3.3 是“用户信息表”，用来存储用户的状态信息；表 3.4“用户控制点记录表”，用来存储用户在控制中机器人经过了点的位置，采用了什么控制方式。

表 3.3 用户信息表

UserName	char
Password	char
Privilege	Short int
RegisterTime	time

表 3.4 用户控制点记录表

UserName	char
StartTime	time
Count	Short int
ControlMode	bool
MoveType	char
Speed	float
FrameName	char
Rconf	Short int
ToolNo	Short int

P0 / X	Double
P1 / Y	Double
P2 / Z	Double
P3 / TX	Double
P4 / TY	Double
P5 / TZ	Double

表 3.5 机器人极限状态数据表

Speed	Double
P0	Double
P1	Double
P2	Double
P3	Double
P4	Double
P5	Double
Xmin	Double
Xmax	Double
Ymin	Double
Ymax	Double
Zmin	Double
Zmax	Double

表 3.5 “机器人极限状态数据表”存储的是机器人的一些极限点的数据，当用户进行控制时，必须要根据此表进行数据的合法性检查。Speed 表示的是机器人运行速度的最大值。P0~P5 表示的是六个关节的最大脉冲。Xmin、Xmax 指的是 x 方向的最大最小值；Ymin、Ymax 指的是 Y 方向的最大最小值；Zmin、Zmax 指的是 Z 方向的最大最小值。

机器人网络控制系统平台采用的是 VC++下使用 ADO 访问 SQL SERVER 2000。ADO 提供了与任何 ODBC 兼容数据库或 OLE DB 数据源的高性能连接。使用 ADO 的对象可以建立和管理数据库的连接，从数据库服务器要求和获取数据，执行更新、删除、添加数据、获取数据库的错误信息等。ADO 提供了以下三个主要对象[35]：

（1） Connection 对象，表示建立一个数据源的连接；

（2） Command 对象，定义对数据源进行操作的命令；

（3） Recordset 对象，表示由数据库或命令的结果产生的全部记录集。

在这三个对象中 Recordset 最为重要，Connection 和 Command 对象为创建 Recordset 对象服务。Connection 对象主要负责记录有关数据源的一些必要的信息，Recordset 对象将从数据源里产生，ADO 需要通过引用 DSN（data sourcename，简称为 DSN）通知 Windows NT 服务器 ODBC 数据源的存在。Connection 用 Open 方法引用 DSN 的参数并把它记录到 Connection String 属性中。Command 对象提供了第二种创建 Recordset 对象的方法。它产生一个游标，进行参数查询。对指定的值进行参数化查询，能够从结果记录集滤出相应的记录。可以通过在 SQL 语句中增加变量来完成此操作。当用 Recordset 对象创建记录集时，先用 Server 对象的 Create 方法实例化一个 Recordset 对象，然后，用 Recordset 对象 Open 方法在记录集中装满数据，这时就可以通过操纵 Recordset 对象的记录集来实现数据库的应用。

通过 ADO 访问数据库，主要有两个步骤：

首先，创建数据源：

（1）选择一种支持 ODBC 的数据库，如 Access 等，建立数据库和相应数据表。

（2）创建数据库链接，打开数据库。

然后，执行数据库的访问操作。

3.4 服务器与机器人通讯实现

服务器端为了实现针对不同类型的机器人都可以方便的接入系统，不能应用传统的单应用程序控制单个机器人的设计方式，必须采用一种新的设计方式进行系统设计，给不同机器人接入提供便捷的接口。

3.4.1 实现多机器人载入的设计思想

本系统采用的技术是 C++面向对象的编程思想，在程序设计中把机器人作为一个对象，封装了机器人的操作行为和机器人的数据状态。这种实现方式不同于过程性编程，它的重点不是实现任务的过程，而是在概念上对对象的数据和行为设计，对象模型的建立是整个程序设计的关键，是系统稳定运行的前提，采用面向对象的设计方式，可以有效、方便的对系统进行功能扩展。

系统接入不同机器人的设计瓶颈在于：各种机器人的行为具有相似性，但每一种

机器人行为的具体实现都是不同的；机器人控制的状态数据和命令数据的差异性。由此可见，要想实现系统对不同机器人的动态载入，这两个技术必须得到突破。C++的面向对象的编程思想可以实现这些要求[36,37]。

解决多机器人载入的方法有如下几个方面：

（1）以机器人作为基本对象进行建立机器人的数据和行为模型，把机器人的一些共有的行为特征抽象出来，作为基本机器人对象的行为。

（2）基本机器人对象建立完成以后，考虑到不同机器人行为实现的具体差异，采用的是对象的继承和多态的设计思想。不同的机器人对象特征在子类中体现差异。不同机器人的行为在子类中得到具体的实现。

（3）不同机器人的状态数据和命令数据的差异性，采用 C++模板的技术。设计对象模型时，不涉及到具体的数据类型，只有在动态载入的时候才涉及到机器人数据类型。

通过上面的三种设计思想，可以设计出每一种类型的机器人具体行为实现和数据结构的实现，建立每个机器人的类库，系统在设计时，采用基本的机器人类作为下位机进行实现，这样在运行时动态载入需要接入系统的机器人类库，就可以使本系统方便的对机器人进行控制。下面详细的介绍机器人网络控制系统中多机器人动态载入的设计和实现。

3.4.2 机器人基类的实现

机器人基类的设计是实现不同机器人动态载入的基础和关键，它是否具有高度的抽象性和行为的完备性，决定了是否可以设计出有效的代替机器人的基类模型，实现机器人动态载入。

机器人行为抽象可以简单的分为：初始化机器人、得到机器人状态信息、定点行走、执行任务、停止任务、暂停任务、继续执行和删除任务等等。如下所示：

```
class BaseRobot
{
... ... ... ...                                    //一些数据定义
virtual   short BuildConnect( );                    //建立和机器人的连接
virtual   short disConnect( );                      //断开和机器人的连接
virtual   void Init（HWND hwd）;                    //机器人初始化
```

```
//获得机器人的状态信息
virtual   void GetRobotStatus（STATUSINFORMATION &status）;
virtual   bool StartStep（POSTION &pos）;                    //定点行走
virtual   bool StartJob（CString tJobName）;                 //开始一个任务。
virtual   void ContinueRun( );          //任务暂时停止运行后，重新运行任务
virtual   void StopRun( );                                   //暂时运行任务
virtual   void JobDelete（CString jobname）;                 //任务删除。
virtual   bool JobDownLoad（CString jobName）;       //把任务下载到机器人上
  Robot( );
virtual ~Robot( );
... ... ... ...                                              //一些数据定义
};
```

其中，这些定义的都是机器人共有的动作行为，不同机器人具有的特殊的行为可以在继承类里进行实现。

3.4.3 实现不同机器人动态载入

继承技术可以扩展和修改原有基类的行为，所有待接入的机器人类都具有相同的特征，因此可以从机器人基类（BaseRobot）中继承这些公有的行为特征，然后根据本身特征进行相应的扩展和具体的实现，下面是一个具体机器人 Motoman 的类模型，它包括了机器人共有的行为方法，和自己特有的行为。以继承的方式来达到不同机器人类模型的公有和特有的属性。

```
class MotomanRobot    :   public BaseRobot
{
... ... ... ...                                              //一些数据定义
public:
    virtual void GetRobotStatus ( STATUSINFORMATION &status );
    virtual bool StartStep ( POSTION &pos );
    virtual void JobDelete ( CString jobname );
    virtual void Init ( HWND hwd );
    virtual void ContinueRun( );
    virtual void StopRun( );
```

```
    virtual bool StartJob (CString jobName );
    Robot( );
    virtual ~Robot( );
private:
    bool ServoOn( );
     virtual bool JobDownLoad( CString jobName );
    short disConnect( );                                    //中断和机器人连接
    HWND inputHwnd; //inputHwnd 是使用此 robot 类的窗口句柄
    short nCid;                                             //和控制柜连接的通道值
    virtual   short BuildConnect( );                        //建立和机器人的连接
... ... ... ...                                             //一些数据定义

};
```

继承实现的不同机器人类模型的建立，而多态技术实现的是机器人共同的行为特征的具体的不同实现，例如：移动机器人前进一步采用的是空间坐标，而工业机器人前进采用的可以是关节坐标。我们在基类 BaseRobot 和具体机器人 MotomanRobot 中都是用 StartStep( )方法来实现，但是不同机器人实际的实现代码可以是不同的。

3.4.4 解决机器人状态数据差异

机器人种类不同，它本身的状态信息就会不同，状态信息的数据结构就会不同。实现不同的机器人接入必须解决和机器人数据通讯的问题。C++中的模块技术是很好的解决方案。

传统的和机器人用接口函数进行数据交换时，必须定义好数据格式，例如，移动机器人的状态信息可以是：

```
Class MoveRobot
{
virtual void GetRobotStatus(MOVESTATUSINFORMATION &status);
}
Struct MOVESTATUSINFORMATION
{
    ... ... ...                 //一些数据定义
```

```
    int   rightSpeed;          //左轮转速
    int   leftSpeed;           //右轮转速
    POSITION   pos;            //位置坐标
    …… ……                     //一些数据定义
}
```

而对工业机器人的状态信息是：

```
Struct MOTOMANSTATUSINFORMATION
{
    …… ……                                  //一些数据定义
    bool      controlMode;                  //控制模式
    CString   movtype;                      //关节移动方式
    CString   vtype;                        //速度方式
    double    spd;                          //整体移动速度
    short     toolno;                       //机器人采用的工具坐标系
    double    p[12];                        //每一个轴的位置信息
…… ……                                      //一些数据定义
}
```

由此可以，我们在定义机器人具体的方法时，必须正确的使用状态数据结构，使用相应的机器人类模型来调用机器人的方法。

本系统在继承和多态的技术上，使用模板的技术，这样，可以在定义机器人类时采用通用的数据定义格式，而在具体的机器人载入时采把相应的机器人数据结构进行动态载入，实现统一的机器人接口方法。

例如：

```
Template <class T>                      //定义模板，通用数据类型 T
class BaseRobot
{
…… …… ……                              //一些数据定义
    virtual void GetRobotStatus(T &status);
};
Template <class T>                      //定义模板，通用数据类型 T
class MotomanRobot    :   public BaseRobot
```

```
{
…… …… …… ……                                    //一些数据定义
public:
    virtual void GetRobotStatus(T &status);
};
```

如此，各个机器人在方法的接口就实现了统一，方便了不同机器人的调用。在实际调用的时候才实现具体数据类型的载入。

例如：

BaseRobot (MOVESTATUSINFORMATION) *moveRobot = new MoveRobot (MOVESTATUSINFORMATION);

BaseRobot (MOTOMANSTATUSINFORMATION) *motomanRobot = new MotomanRobot (MOTOMANSTATUSINFORMATION);

moveRobot 是机器人基类的指针，它的数据类型是移动机器人的数据类型，它指向了移动机器人（MoveRobot）的具体实现。

motomanRobot 是机器人基类的指针，它的数据类型是 Motoman 工业机器人的数据类型，它指向了 Motoamn 机器人（MotomanRobot）的具体实现。

3.4.5 与机器人接口类库

MotomanRobot 类是本系统与具体的机器人的接口类，在上文中已经提到，它是个独立的类，负责发送具体的控制信息给机器人，并且返回机器人的状态。它包括机器人初始化、定点移动、下载任务、执行任务等 Motoman 机器人所具有的一些行为。

MotomanRobot 类的具体实现是建立在 MOTOCOM32 通讯软件提供了一套动态连接的库函数，在类的设计中调用这些库函数，对机器人进行控制。这些库函数包括了机器人的主要操作功能，例如：状态读取函数用于读取机器人的位置值、当前程序名、运行方式、操作坐标、控制轴组等状态信息以及文件的上传、下载等； 系统控制函数可对机器人进行各种操作，包括伺服电源的通断、程序运行的起停、机器人以各种坐标方式按不同的速度进行目标点或者增量运动。通过这些库函数的组合使用，实现对机器人系统的各种控制与操作， 扩展了机器人的系统集成功能。

(1) 动态连接库的调用

为了使 MotomanRobot 类能够调用动态连接库中的库函数， 首先，机器人类的实

现文件中必须包含："direct . h"、"motocom. h"头文件；其次，复制 Motocom32. DLL、Motolk.DLL、Motolkr.DLL 数据传输动态库到工程文件目录下；最后，把"motocom32.lib"文件加入的类库管理目录中。

(2) 通信的建立

首先利用语句：nCid = BscOpen (jobname, 1) 取得通讯句柄 nCid ，在后续的语句中调用 BscSetCom (nCid ,1 , 9600 , 2 , 8 , 0) 设置通讯口参数，最后进行通讯线路的连接: BscConnect (nCid) 。这样就可以进行文件的上传(函数 BscUpLoad (file- name , nCid))、下载(函数 BscDownLoad (nCid , file- name)) 等操作了。

(3) 通讯线路的关闭

调用关闭通讯口的函数 BscClose (nCid)， 实现和机器人通讯的安全结束。（详细的控制函数见附录 2）

3.4.6 Motoman 机器人任务文件解析

本系统在实现监督控制时，可以把监督控制的每一次单个控制记录下来，最后可以形成一个完整的任务，让机器人一次执行完毕。要实现此功能，必须要把机器人的.JBI 任务文件的格式解析出来，才能进行任务的重建。

通过分析.JBI 的文件，得出了.JBI 文件的格式。一个.JBI 文件包含三个部分：

（1）注释代码段。此代码里面包含了本次任务的任务名、控制方式、控制点的个数、使用的工具系，任务的形成时间以及采用的机器人控制轴组。

（2）数据代码段。此部分是用来存储机器人在任务过程中一共经过的控制点的详细位置，这些数据决定了机器人在运行任务期间的轨迹。

（3）程序代码段。此部分是具体的任务的运行代码，包括了机器人的运行速度等信息。

举例如下

```
/JOB
//NAME NETTEST
//POS
///NPOS 4,0,0,0,0,0                    //注释代码段
///TOOL 0
///POSTYPE PULSE
```

```
///PULSE
C00000=-36336,83865,43468,-1578,-68626,17740
C00001=8570,61489,241,-1336,-50225,-747                    //数据代码段
C00002=2105,36038,30610,-1567,-90869,4812
C00003=0,0,0,0,-82443,0
//INST
///DATE 2007/01/10 15:18                                   //注释代码段
///ATTR SC,RW
///GROUP1 RB1
NOP
MOVJ C00000 VJ=20.00
MOVJ C00001 VJ=20.00                                       //程序代码段
MOVJ C00002 VJ=20.00
MOVJ C00003 VJ=0.78
END
```

本系统在设计网络机器人数据控制协议里已经包含了所有的形成任务的必须消息，在服务器端数据库也把这些数据保存了下来，根据此格式，完成了任务的重建。

3.5 现场视频信息反馈模块具体实现

机器人网络控制系统平台现场视频信息的反馈采用的流媒体技术[33]。流媒体是指在网络中使用流式传输技术的媒体，如音频、视频或多媒体文件。而流式传输技术就是把连续的声音和图像信息经过压缩处理后放到视频服务器上或者进行实时的广播，让用户一边下载一边收听观看，而不需要等待整个文件下载到自己的机器后才可以观看的网络传输技术。流媒体数据流具有连续性、实时性、时序性三大特点，比较适合本系统的现场视频信息的实时传播。

流媒体传输网络是适合是实时传输协议的网络。流媒体在因特网上的传输必然涉及到网络传输协议，这是制约流媒体性能的最重要的因素。为了保证对网络拥塞、时延和抖动极其敏感的流媒体业务在面向无连接的 IP 网络中的服务质量，必须采用合适的协议，其中包括 Internet 本身的多媒体传输协议，以及一些实时流式传输协议等。

目前几种支持流媒体传输的协议主要有用于 Internet 上针对多媒体数据流的实时传输协议 RTP （Real-Time Transport Protocol），与 RTP 一起提供流量控制和拥塞控

制服务的实时传输控制协议 RTCP （Real-Lime Transport ControlProtocol）[33]。

RTP 被定义在一对一或一对多的传输情况下工作，其目的是提供时间信息和实现流同步。RTP 通常使用 UDP 来传送数据，也可在 TCP 或 ATM 等其他协议上工作。RTP 本身并不能为按顺序传送数据包提供可靠的传送机制，也不提供流量控制或拥塞控制，它依靠 RTCP 提供这些服务。

在 RTP 会话期间，各参与者周期性地传送 RTCP 包。RTCP 包中含有已发送的数据包的数量、丢失的数据包的数量等统计资料，因此服务器可以利用这些信息动态地改变传输速率，甚至改变有效载荷类型，以适应网络的带宽。通常采用两个方法来调节:一是窗口法，通过逐渐增大传送的码率，当发现网络上出现了包的碰撞，也就是检测到了丢包时，再减小发送的码率；一是基于速率的方法，先估计网络的带宽资源，再调整编码的口标速率来适应网络的状态。基于窗口的解决方案会引入类似 TCP 的重传，所以经常采用基于速率的解决方案。RTP 和 RTCP 配合使用，能以有效的反馈和最小的开销使传输效率最佳化，因而特别适合传送网上的实时数据。

本系统的现场视频信息反馈模块是基于流媒体系统的框架，包括如下三个方面：

（1）负责针对采集的现场视频信息进行音/视频源的编码压缩。

（2）视频服务器对编码的流式文件进行实时广播。

（3）客户端的视频接受模块进行相应的视频流解码和播放。

3.5.1 现场视频的编码/压缩

现场视频信息经过视频采集卡的捕获后，需要通过压缩和编码才能进行网上实时广播。本系统采用的是 Windows Media Encoder 编码软件，它在保证一定音/视频质量的前提下，媒体流码流速率较低，编码效率很高。

3.5.2 视频服务器

本系统采用的视频服务器是运行在 Windows 2003 Server 上的 Windows Media Services（WMS），它能够很完美的同 Windows Media Encoder 进行接口，实现在 Internet 上的实时传输。WMS 能将多个以不同比特率编码的流文件融合到单一的视频文件中，这样系统就可以通过广播一个流文件而提供给不同连接速率的客户端。

建立服务器的步骤如下：

（1）进入 Windows Media Encoder 后，选择实况广播后，再选择自编码模式，设定编码发送的 IP 地址和相应的端口。

（2）根据不同的网络状况选择具体的编解码率。

（3）设置 WMS ，添加发布点，输入刚才设置的 IP 值和端口号。

进行设置后，就可以在 Internet 上进行现场的信息反馈，通过本系统编写的视频接受模块就可以接受现场的信息了。

3.5.3 客户端现场视频接收模块

此模块的实现是利用 ActiveMovie 控件，用 VC++设计编写的。ActiveMovie 控件是微软公司推出的用于多媒体程序设计的控件，它提供了非常完善的音频和视频媒体文件的回放功能能支持多种文件格式，其中也包括在 Internet 上传输的流媒体文件。

ActiveMovie 控件的特有属性及相关描述见表 3.6。

表 3.6 ActiveMovie 控件特有的属性表

属性	说明
CurrentState	指示控件的状态：stopped，paused，running
DisplayMode	显示模式，即以时间方式还是帧方式
FileName	指定该控件要操作的源文件完整名字
Rate	指示媒体流的回放率
ReadyState	指示控件状态，是否已经装入源文件
SelectionEnd	指示播放媒体流的结束位置
SelectionStart	指示播放媒体流的开始位置
CurrentPostion	指示播放媒体流的当前位置
Volume	设置音量

ActiveMovie 控件常用方法有 Run，Stop，Pause 三个，分别控制媒体流的播放、停止和暂停使用时直接调用它们即可。其它通用方法如 Drag，Move，SetFocus 等，该控件也支持。ActiveMovie 控件的特有事件及相关描述见表 3.7。

表 3.7 ActiveMovie 特有的事件表

事件	说明
DisplayModeChange	当 DisplayMode 属性值发生变化时触发
OpenComplete	当源文件完全载入时触发
PositionChange	当媒体流的当前位置改变时触发
ReadyStateChange	当控件的 ReadyState 属性值改变时触发
StateChange	当播放器状态改变时触发
Error	处理控件的出错事件

3.6 保障机器人网络控制平台系统安全采用的一些措施

由于利用网络作为信息交流载体，基于 Internet 的机器人控制系统在设计中有一些特殊的情况，比如，大的不确定时延。控制命令的传输在网络上经过很多的节点，路由选择不固定，使得时延比较严重，而且不确定，预测和控制都比较困难；数据包的丢失在网络忙碌和连接不稳定时非常多见，普通的网络应用可以采用数据包重发解决，但是重发并不能解决机器人控制的问题。

解决上述问题，除了采用监督控制的控制方式，提高本地服务器自我解决问题的能力，减少网络控制带来的影响外，还应该建立健全的安全机制，当命令数据包发生问题时，及时启动安全防护系统功能，以避免机器人系统的硬件损伤。基本的安全保障机制除了机器人本体系统中的安全控制（采用一些参数的限制）外，还采用了如下措施：

（1）用户准入检查

用户采用注册准入制度，不允许匿名用户登陆，保证用户的可信度，并设置管理员，进行用户授权和剔除工作。在控制进行中采用前述多用户单操作方式，保证实体控制的一对一执行。

（2）控制流程中的监控

确保机器人控制系统的正常工作，更多的是从控制流程上进行多重检验来进行的。这方面包括机器人服务器对于控制命令的格式检验、对控制参数的可行性检验等。在控制流程中的监控，有软件上的检验，如控制参数的运算结果合理性，也有用户（控制人员）的参与，如图像监视等。

3.7 本章小结

机器人网络控制平台系统的详细设计。本章详细的介绍了本系统实现的方法和原理。介绍了系统采用的 MotomanHp3 机器人。讨论了网控机器人中的数据传输机制。详细的介绍了系统各个功能模块和编程模块的实现。提出了一个基于 TCP 协议的机器人数据控制协议。介绍了多机器人接入系统所采用的方法。介绍了为保障机器人网络控制平台系统安全运行的一些保障措施。

第 4 章 机器人网络控制平台系统实验研究

4.1 实验系统组成

在前面的章节中分别对机器人控制模式、网络数据传输协议和机器人网络控制系统平台的具体实现作了较为详细的研究和分析，提出了本系统是基于客户端/服务器端的分布式的机器人控制系统，主要包括了机器人监督控制方式和自主任务方式，采用流媒体的视频广播方式进行现场的视频信息反馈，并定义了一个基于 TCP 协议为底层协议的机器人数据传输协议，本章中对相应的研究给出了实验验证。以下将主要介绍各个模块的具体实现的人机界面和功能运行方式。

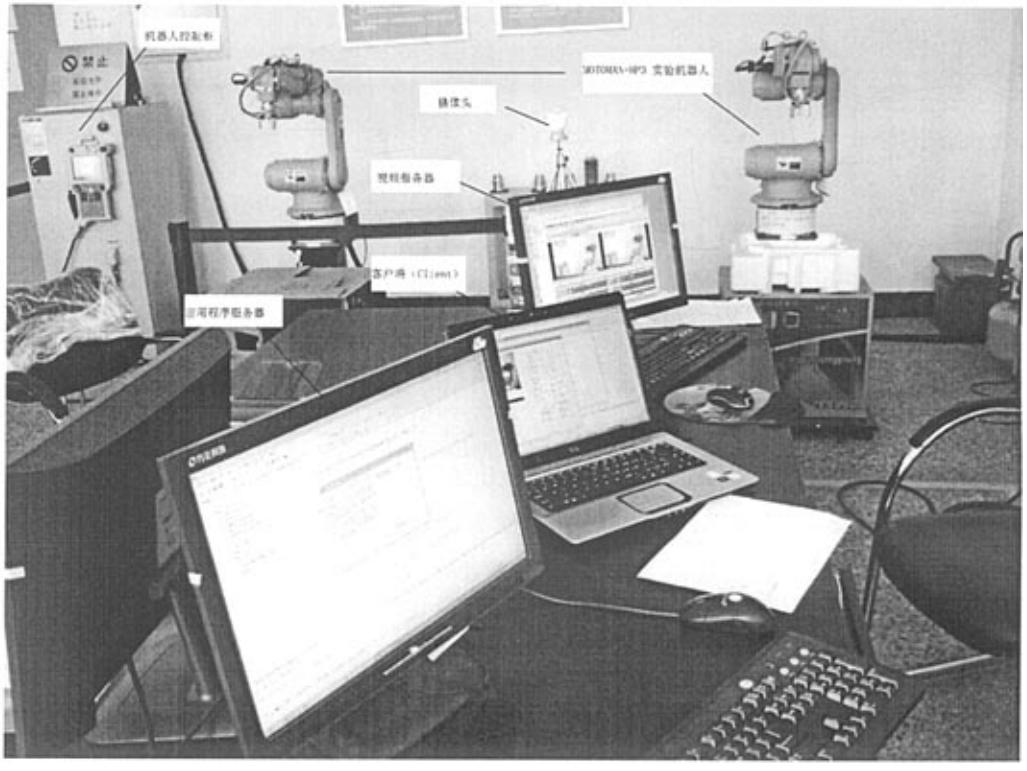


图 4.1　机器人网络控制平台系统的硬件连接图

本系统硬件连接图如图 4.1 所示。其中包括实验对象 MOTOMAN-HP3 机器人，

机器人控制柜 NX100，采集现场信息的摄像头，视频服务器，应用程序服务器，客户端 PC。

4.2 Client 端用户登录程序模块测试

用户如果想要使用机器人网络控制系统平台，必须进行注册和登录，根据用户被分配的权限进行相应的控制选择。所有注册用户都可以通过机器人客户端自由的观看机器人运动的实时图像，查看机器人的状态信息。如果要操作机器人，必须具有更高级的权限。当具有权限的用户申请机器人的控制权时，服务器端把用户加入到后台维护的等待队列。只有该队列中的用户有权操作机器人。按先入先出的原理。每个用户有 5－10 分钟（可以设置改变）的操作权，当用户超时或自动退出时，系统会断开用户的连接，把用户从等待队列中删除。此模块主要使用了网络通讯模块来与服务器端进行通讯，利用机器人数据传输协议把用户信息进行打包，然后发送给服务器端。

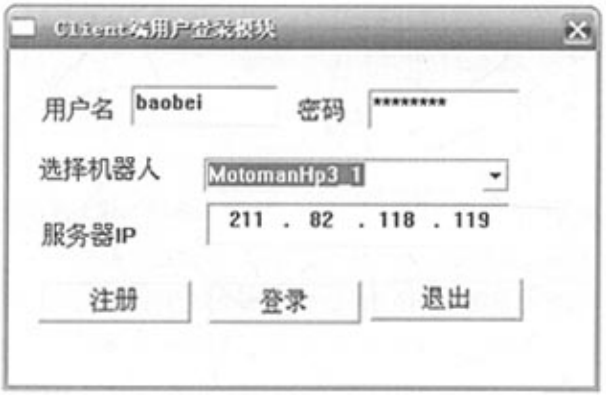


图 4.2　Client 端用户登录界面

登录界面如图 4.2 所示，如果用户属于新用户，必须点击“注册”按钮进行注册，需要经过服务器端系统管理员的授权才能登录本系统。如果是已注册用户，需要输入用户名和密码，“选择机器人”里是本系统的现已接入的机器人，可以进行选择，需要输入服务器的 IP 地址。当点击“登录”按钮后，客户端将会把此信息发给服务器端进行验证，如果信息合法的话就会直接进入 Client 控制界面。

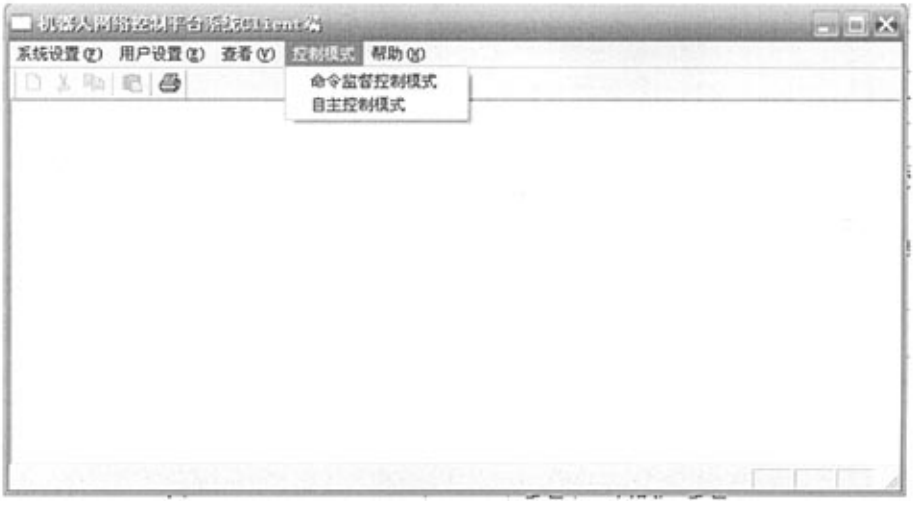


图 4.3 客户端控制界面

客户端控制界面如图 4.3 所示，当用户身份验证成功后，进入此控制界面。用户可以在此界面打开命令监督控制模块和自主控制模块以及机器人状态反馈模块。可以进行系统设置和用户信息的改变。

4.3 Client 端机器人状态反馈信息程序模块测试

此模块是运行在 Client 端的模块，可以提供给所有的已注册用户反馈当前机器人的状态：在关节坐标系下的机器人每一个轴的坐标值，在直角坐标系下的 X、Y、Z 坐标值，相对于 XYZ 三个轴的转角 TX，TY，TZ 的值。系统状态信息包括：机器人是否处于空闲状态、伺服是否通断、机器人系统处于那个控制模式 （PLAY/ TEACH/ REMOTE）、机器人处于单步运行还是循环运行、当前机器人运行的任务名、是否发生错误和当前一共有几个用户在等待。此模块还包括了现场信息的视频反馈。

图 4.4 中显示的是机器人在不同的两个位置时，机器人状态反馈模块返回的信息。图 4.4-1 中视频模块反馈当前机器人所在的位置为：直角坐标系下值为[30132，21564，－25362，－1，－360055，2]；对应的直角坐标系值为[455，184，120，180，－19，22]。图 4.4-2 显示的是机器人不同的一个位置上的反馈值，分别是：直角坐标系下值为[－70011，－4083，2588，25757，－78509，27065]；对应的直角坐标系值为[212，－333，317，154，15，－7]。当前的控制用户为一个用户。系统没有用户进行自主控制或监督控制，每接受一个指令只进行单周期运行，机器人伺服处于关闭状态。

图 4.4-1 Client 端机器人信息反馈界面

图 4.4-2 Client 端机器人状态反馈信息模块

4.4 Client 端监督控制程序模块测试

此模块包括如下四个功能任务：

（1）现场视频的播放。首先根据视频服务器的网络地址，进行连接播放。

（2）直角控制，表示采用直角坐标系对机器人进行监督控制，在此情况下关节控制无效。可以采用高速、中速和低速三种速度进行控制。

（3）关节控制，表示采用关节坐标系进行监督控制，在此情况下直角坐标控制无效。可以指定具体的速度，指定机器人的运动差补方式。

（4）生成任务。系统会记录用户进行监督控制过程中的每一次操作。可以根据用户的要求生成一个完整的任务来进行运行。

图 4.5 直角坐标控制

如图 4.5 所示。首先，需要向系统申请控制权，如果没有用户在进行控制，那么可以得到控制权，否则会加入系统的等待队列。使用直角坐标控制时，相应关控制按钮是无效的。可以选择四种坐标系，每种坐标系机器人的运动方式不同，选择机器人运行速度，然后可以点击 XYZ 轴的六个按钮，操作机器人达到指定的位置。退出时可以释放对机器人的控制权。

图 4.6 关节控制

如图 4.6 所示，使用关节坐标控制时，相应直角坐标的控制按钮是无效的。首先，需要向系统申请控制权，如果没有用户在进行控制，那么可以得到控制权，否则会加入系统的等待队列。其次选择在关节坐标下机器人每一关节运动的方式（自由运行、直线运动、曲线运动或线形运动），需要指定机器人每个轴的坐标点。“机器人复位”按钮是用来让机器人回到初始化位置的。

图 4.7 Client 端监督控制模块界面

图 4.7 显示通过关节控制定点到指定位置[39646,-18514,18803,35298,15918,-27971]后，再有状态信息反馈模块把现场的机器人信息反馈给客户端，数据是一致的。（直角坐标系中反馈的数据经过了从浮点数到整型的转换）

表 4.1 实验中一些极限点的坐标

直角控制						
点	X	Y	Z	TX	TY	TZ
原点	399.443	0	285.002	180	0.35	0
Xmin	177.662	0.002	284.878	180	0.37	0.2
Xmax	536.667	0.009	284.872	180	0.37	-0.1
Ymin	399.437	-375.993	285.001	180	0.36	0
Ymax	399.452	369.007	284.977	180	0.36	-0.1
Zmin	399.437	-0.001	-16.003	180	0.36	-0.1
Zmax	399.437	-0.001	419.991	180	0.36	-0.1
对应关节坐标						
	[S]	[L]	[U]	[R]	[B]	[T]
原点	0	0	0	0	-82243	0
Xmin	1	-76684	-37023	0	-95917	9
Xmax	1	58060	55216	-1	-90039	-7
Ymin	-59020	65602	64167	225	-92133	24618
Ymax	58287	62388	60310	-226	-91168	-24315
Zmin	-1	24575	-60293	-4	-29762	0
Zmax	-1	35583	72519	-3	-112801	-5

表 4.1 中的数据是实验中一些极限点的位置，在系统实验中，这些实验数据将被

采集下来，存储在数据库中，当用户进行操作时，可以对用户发送的控制点进行验证，保障机器人接受到正确、合法的数据，对系统的稳定和机器人的保护都是必要的措施。

4.5 Client 端自主任务编程控制程序模块测试

此模块包括视频接收模块和自主任务模块。用户可以发送具体的任务给机器人执行。发送的任务为机器人可以识别的执行文件。在机器人执行过程中，可以暂停和终止机器人的运行。如图 4.8 所示。

图 4.8　Client 端用户自主任务控制模块界面

4.6 Server 端用户管理程序模块测试

图 4.9 是机器人网络控制平台系统的 Server 端，在应用程序服务器上首先要启动此程序，此界面上显示的是用户每次请求所发送的数据包。“系统设置”是用来对用户进行管理的界面。

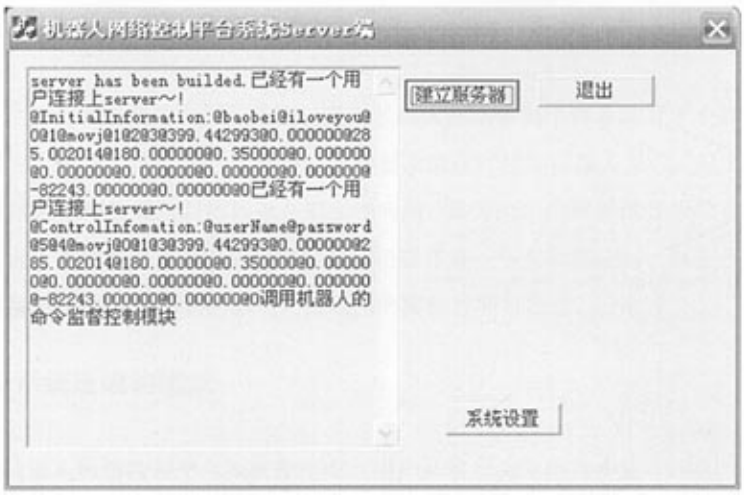


图 4.9 机器人网络恐慌子平台系统 Server 端

用户管理模块是运行在 Server 端的程序，是系统管理员对用户进行动态管理的设置界面。管理员可以在此模块下建立新的用户、修改已有用户的用户名和密码或者删除用户。中间的窗口控件是用来显示当前在线和离线的用户。对在线用户强制的终止用户对机器人的控制权，或者是提高别的用户在用户队列中的顺序，允许他们提前取得机器人控制权。还可以查看具体用户在控制机器人过程中所进行的记录操作。如此可以了解用户有没有进行不合理的操作，方便对系统的管理。

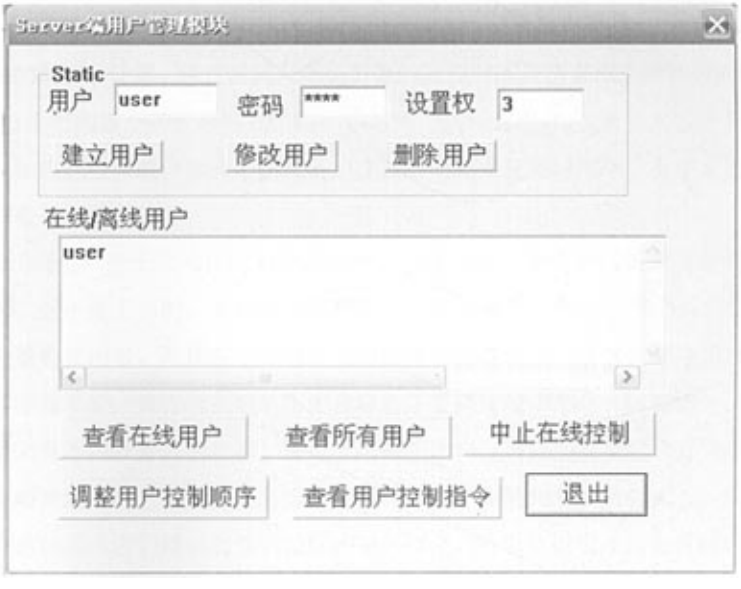


图 4.10 Server 端用户管理模块界面

4.7 网络时延的分析和影响

通过前面的研究与设计，我们已经对机器人网络控制平台系统有了全面认识，并且从技术角度而言我们可以实现面向各种要求的远程控制机器人系统。但实践证明机器人网络控制平台系统目前还存在着一些问题，最为突出的问题就是系统存在时间延迟。因此，本文有必要对上述问题的产生机理作进一步分析和探讨，以便于采取相应措施消除或削弱这些因素的影响，提高网络实时性和可靠性。

4.7.1 系统延迟的组成

在机器人网络控制平台系统使用中，用户从客户端发出指令后，在客户端上并不能立即观察到机器人的响应，而是需要等待一个延迟时间，并且这种延迟没有规律，很难用现有的控制方法或算法加以预测。实验结果显示，客户端在接受机器人状态信息时基本没有延时，但是在接受视频信息时，存在着 8～15 不定的延时。通过对机器人网络控制系统分析可知，整个系统控制回路都存在着不同程度的延迟。分析归纳起来，机器人网络控制系统总延迟时间 T_r，主要是由以下的几个延迟时间叠加：通讯延迟 T_c、执行延迟 T_P、数据延迟 T_d、扰动延迟 T_v 等，即

$$T_r = T_c + T_p + T_d + T_v$$

其中，通讯延迟 T_c 包括通讯初始化时间和在介质中的传输时间；执行延迟 T_P 包括控制指令的解释、计算、执行时间，现场图像的处理时间及仿真图像的运行时间等；数据延迟 T_d 指控制指令和视频等数据的传输时间；扰动延迟 T_v 主要指传输中不可预测的扰动，如信息丢失或信息次序的混乱。在以上的各类延迟时间中，有些是固定不变的，有些是不可预测的。

系统通讯延迟产生于信号传输的介质当中。一般来说，距离的远近对传输时间有一定的影响，但不是主要的。影响通讯延迟的主要因素是信号经过网络节点的数目。网络上有大量的使用者，并且在连接通路上的计算机的吞吐量、数据包调度策略以及路由策略存在着差异。数据包在到达指定目标前，要经过很多节点（计算机）。这些节点控制着来自不同数据源的数据。每个节点对到达自己节点的数据包进行排队，并根据路由策略决定它们的去向，使之更接近目标节点。如果此节点负载太大，则可能抛弃一些数据包或将它们路由到负载较低的服务器上。所以数据包在路由选择时会沿

不同的路径传输且排队时间变化不定，从而导致了通讯时间延迟的不确定性，甚至数据丢失。

执行时间是指系统设备处理信息的时间。系统客户端计算机处理用户数据的输入和反馈数据的显示存在一定的延迟；服务器端解释用户指令、机器人调用控制算法并执行以及图像采集并进行编码、压缩、存储也都会造成一定的时间延迟。这些延迟对于系统而言都基本固定不变。

数据延迟被定义为

$$T_d = \frac{(D_s + D_r)}{V_L} \qquad (4.1)$$

其中 D_s，D_r 是发送或返回的数据总量；V_L 是传输速率，与介质有关。

在网络上传输数据需要一个过程，所以必然存在延迟。影响数据延迟大小的因素是数据容量和网络带宽。扰动延迟的发生是随机的，因此也具有不确定性。

4.7.2 解决的途径

上述分析表明，影响机器人网络控制平台系统性能（稳定性、透明性、同步性）的因素是互联网的通讯协议、通讯质量和网络带宽。因此，一种解决途径是在现有的 Internet 网络技术环境下，设计合理可行的传输协议，最大限度地保障数据传输的实时性和可靠性，并通过设计控制器兼顾系统的鲁棒性和稳定性。在网络远程控制机器人系统研究领域中，有很多的研究人员正致力于此项工作，并取得了一定的收效。另一个被忽视的解决途径是开发新的网络体系结构，使之能够支持网络远程控制机器人系统所要求的网络 QOS（Quality of Service）。这一途径就是悄然兴起的 IP QOS 技术研究[40]。

IP 网的服务质量，简称 IP QOS。它是指 IP 网络所能够提供的各种特性对广大网络用户特定需求的满足程度。IP QOS 的提出是为了解决当前 IP 网络上多媒体业务对 IP 环境的新要求。具体的将讲，就是改善 IP 网络的业务可用性、带宽、时延、吞吐量和丢包率等指标，以满足不同业务对网络实时性和可靠性的要求。比如，通过设置优先权让实时性要求高的数据包比非实时性业务数据包优先通过路由器。因此，IP QOS 技术将在未来的 Internet 技术中占有重要的地位，成为促进未来 Internet 网络增长的关键技术。

目前的 IP 网络主要有两种结构体系，即集成业务体系结构（IntegratedServices

Architecture）和区分业务体系结构（Differentiated ServicesArchitecture ）。其中前者能够提供严格的端到端的 QOS 保证，更符合远程控制机器人系统对网络的要求。并且许多路由产品，如 Windows2000 Advanced Serve，已经按照集成业务体系结构实现了 QOS 功能部件。因此，IP QOS 技术在基于 Internet 的远程控制机器人系统中实施将是可行的。

虽然当前的IP QOS技术还不能在Internet中得到实施，但可以预测，未来的Internet必将对 IP QOS 技术提供更充分合理的支持。所以，采用 IP QOS 技术构建基于 Internet 的远程控制机器人系统具有理论前瞻性和实际意义，值得进一步探讨。

4.8 本章小结

本章对机器人网络控制平台系统各个模块进行了测试。实验证明客户端用户注册模块、机器人状态反馈模块、监督控制模块和自主控制模块都能很好的达到了设计的功能目标，服务器端模块也能够正常的运行。机器人网络控制平台系统作为机械设计与理论一机器人学专业方向研究生的实验系统，达到了很好的效果。

第 5 章 总结和展望

本文通过对国内外基于 Internet 的远程控制机器人系统的分析和研究，从系统硬件组成、软件框架、通信协议、控制方式，研究分类等方面进行了全面的归纳和总结，为系统的研究与开发提供重要的参考依据。本系统基于客户端/服务器端模式，采用分布式服务器架构，把视频服务器、应用程序服务器等分离出来，成功实现了系统的各个功能，并作为本学院机械设计理论机器人学专业方向的实验平台。

在系统的设计和开发中，体现了以下一些特点：

（1）控制系统的可扩展能力增强，可以接受不同机器人的接入，为机器人控制系统的改造和系统功能更新提供了方便。

（2）实现的机器人监督控制和自主任务控制。设计并实现了机器人数据控制协议。

（3）根据开放性机器人系统结构设计的设计思想，在软件设计机器人接入时，采用了抽象的设计概念，提取机器人共性，利用 VC++中的多态特性，实现多机器人载入的便利性。

以下是本系统中可以进行扩展和完善的地方：

（1）可以在系统中采用虚拟现实技术，在客户端实现机器人图形的三维仿真，进行预演预测方式控制，结合本系统已完成的现场视频反馈，在系统的抗延时和安全性将得到更大的提高。此项工作正在有其他研究生进行。

（2）网络时延的处理不但是改善系统性能的重要途径，而且是研究有关网控系统自我保护的关键内容。因此对网络延时的处理还需要融合更多的理论和技术进行研究，如延时预测技术。系统动态补偿理论等。

（3）可以在本系统下，扩展系统的功能，进行分布式多机器人协作的研究。

参考文献

[1] 赵长军.基于 Internet 的机器人网络控制研究[D].南京航天航空大学.2004,3:1

[2] Ken Goldberg 等.The Mercury Project: A Feasibility Study for Internet Robots， IEEE Robotics &Automation magazine. 2000(March)，35-40.

[3] Kenneth Taylor 等.Internet Robots: A New Robotics Niche. IEEE Robotics & Automation magazine.2000 (March)，27-34.

[4] 耿海霞等.基于 Web 的远程控制机器人研究.机器人，2002(4).375-379.

[5] 庄严等. 基于网络的机器人控制技术研究现状与发展.机器人[J].2002 (3).276-282

[6] http://www.pfu.fujitsu.com/lmaron

[7] http://www.omron.com/news/n_161001.html

[8] http://www.irobot.corn/home/default.asp

[9] 毕延军.基于 Internet 的机器人远程跟踪与控制系统的研究[D].华北电力大学。2005，12

[10] D.Westhoff，T.Scherer，J.Zhang 等. A flexible framework for task-oriented programming of service robots.VDI Verlag GMBH， dusseldorf， 40001，Germany.2004：737-744

[11] 钱江，苏剑波，席裕庚.基于 Java Applet 和服务器组件技术的 Web 机器人遥操作.计算机工程.2001. 27(6): 37-38

[12] 白剑，吴镇炜，刘振诗.用 Java 语言开发的机器人遥操作系统.机器人.2003，25(2):113-116

[13] 鉴萍.基于 Internet 的遥操作机器人系统时延控制研究[D].山东大学.2006.5

[14] Sherman A，Murat C C，Tendick F. Comparison of Teleoperation Control Architectures for Palpation Task[A]. Proc. of Symposium on Haptic interfaces for Virtual Environment and Teieoperator Systems[C]. Orlando， Florida， 2000: 1996-1923.

[15] Hannaford B. Stability and Performance Trade-offs in Bilateral Telemanipulation[A]. Proc. Of IEEE Int. Conf，on Robotics and Automation[C]. 1989: 1764-1767.

[16] 李万玉，董介春.移动机器人路径跟踪控制方法的研究.青岛大学学报(自然科学

版).2004，6. 17(2):36-40

[17] 朱徐华.网络环境下移动机器人的建模与控制方法研究[D]. 南京理工大学。2006，06

[18] 冯华山.基于 Internet 的远程控制机器人系统[D].西北工业大学.2004.03

[19] 王庆鹏，谈大龙.基于网络的机器人遥操作研究.中国科学院沈阳自动化研究所[D].2001，6

[20] 张君鸿，马玉林.基于 Internet 的机器人遥操作系统时间延迟的研究.哈尔滨工业大学[D].2003，6

[21] Sheridan T.B. Telerobotics, Automation and Human Supervisory Control. The MTT Press. 1992: 1 一 12,98-108

[22] Ando Noriaki， et al. Study on Influnce of Time Delay in Teleoperation.Proceedings of the IEEE International Conference on Systems， Man and Cybernetics， Oct 1999:1111-1116

[23] Sheridan T.B. Telerobotics，Automation and Human Supervisory Control. The MTT Press. 1992: 1-12，98-108

[24] Thomas B.Sheridan. Space Teleoperation Through Time Delay: Review and Prognosis. IEEE Transaction on robotics and automation， Vo1.9， No.S， 1993.pp592-606

[25] Brady K.， Tarn T.J. Internet-Based Remote Teleoperation. IEEE International Conference on Robotics&Automation， 1998，Leuven:65-70

[26] Hirzinger G，Brunner B，Dietrich J，Heindl J. Sensor-Based Space Robotics-ROTEX and Its Telerobotics Features. IEEE Transaction on Robotics&Automation， 1993，9(5): 649-663

[27] Brady K., Turn T.J, Xi Ning, etal. A Modular Telerobotics Architecture for Waste Retrieval and Remediation. The International Journal of Robotics Research, 1998, 17(4): 450-460

[28] Xi Ning, Tarn T.J., Bejczy K. Intelligent Planning and Control for Multirobot Coordination: An Event-Based Approach. IEEE Transaction on Robotics&Automation, 1996, 12(3): 439-452

[29] 孙启援.基于互联网的移动机器人网络控制系统研究[D] .天津大学机械工程学院.2004，06

[30] MOTOMAN NX100 说明及维护保养培训教材

[31] http://www.sg-motoman.com.cn/

[32] 丁展，刘海英等.Visual C++ 网络通信编程实用案例精选.人民邮电出版社.2004.4 第一版

[33] Steve Mack . 流媒体宝典.电子工业出版社.2003.1.第一版

[34] 袁鹏飞，孙军安.人民邮电出版社.SQL Server 2000 数据库系统管理.2001.5.第一版

[35] 刘刀柱，孟繁晶.Visual C++ 实践与提高数据库篇.中国铁道出版社.2001.3.第一版

[36] Stephen Prata. C++primer Plus 中文版. 人民邮电出版社.2005.5.第一版

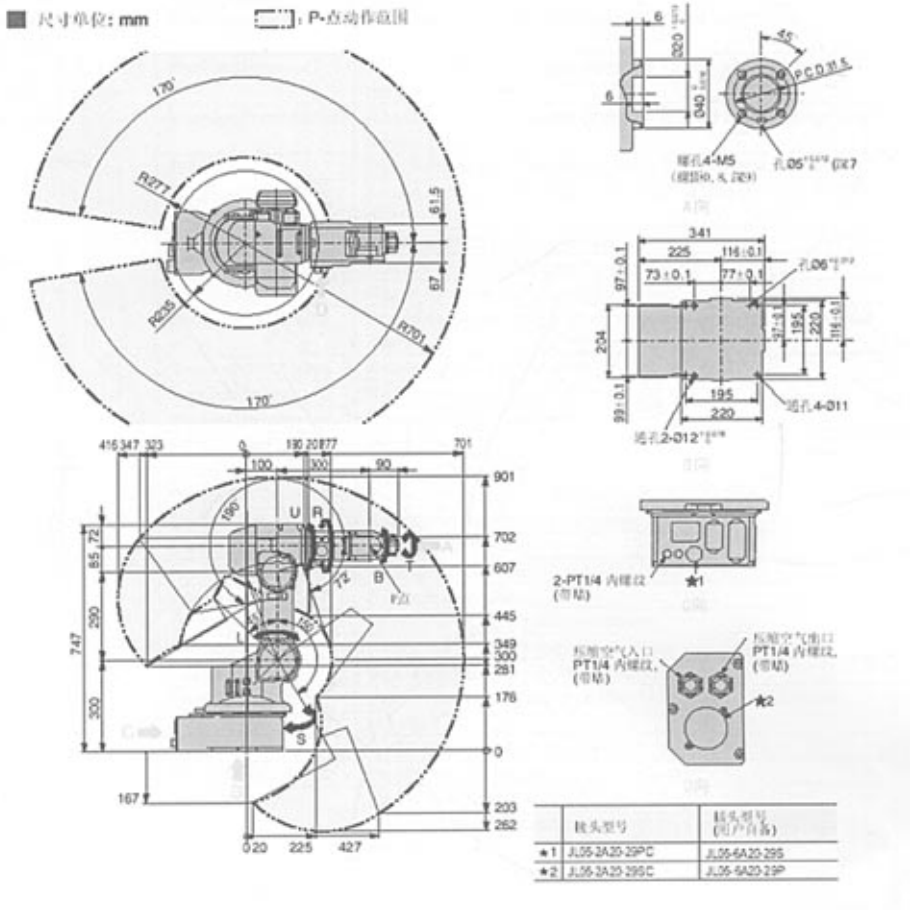
[37] Charles Wright. Visual C++ 程序员实用大全(精华版).2005.5.第一版

[38] 郭志刚，李治柱，赵春云.分布式对象计算技术及对象互连技术.上海交通大学学报，1998，32(4):135-139

[39] 黄海峰.CORBA 的通信机制及性能研究.计算机工程与应用.2003. 23: 157-159

[40] 史英海等.基于 IP QoS 技术的网络机器人遥操作研究.机器人.2003(4):363~366

附录 1 MOTOMAN-HP3 的空间工作范围和详细尺寸



	肘头型号	肘头型号 (用户自备)
★1	JL05-3A20-29PC	JL05-6A20-29S
★2	JL05-3A20-29SC	JL05-6A20-29P

附录 2 MOTOCOM32 的动态链接库中的函数

Function		Function Name	Description
File data transmission function		BscDownload	Sends a specified file to the robot controller.
		BscUpload	Receives a specified file from the robot controller.
Robot control function	Reading status	BscFindFirst	Reads the first job name from the all job list registered at the present time.
		BscFindFirstMaster	Reads the first job name from the job list that belongs to the master job.
		BscFindNext	Reads the next job name registered at the present time.
		BscFindNextMaster	Reads the next job name in the job list that belongs to the master job.
		BscGetCtrlGroup	Reads control group and task information.
		BscDownLoad	Sends a specified file to the robot controller.
		BscGetError	Reads an error code or alarm code.
		BscGetFirstAlarm	Reads an alarm code and returns the alarm code and alarm data.
		BscGetNextAlarm	Reads the next alarm code and alarm data.
		BscGetStatus	Reads the status information.
		BscGetUFrame	Reads specified user frame data.
		BscGetVarData	Reads variables.
		BscIsAlarm	Reads alarm status.
		BscIsCtrlGroup	Reads control group information.
		BscIsCycle	Reads playback mode information.
		BscIsError	Reads error status.
		BscIsErrorCode	Gets an error code.
		BscIsHold	Reads hold status.
		BscIsJobLine	Reads the current job line number.
		BscIsJobName	Reads the current job name.
		BscIsJobStep	Reads the current job step number.
		BscIsLoc	Reads the current robot position in pulse or XYZ frame system.
		BscIsPlayMode	Reads the operation mode.
		BscIsRemoteMode	Reads the command remote mode status.
		BscIsRobotPos	Reads the current robot position in a specified frame system. The existence of the external axis can also be specified.
		BscIsServo	Reads the servo status.
		BscIsTaskInf	Reads task information.
		BscIsTeachMode	Reads whether in the teach mode or play mode.
		BscJobWait	Waits for job completion until the robot motion stops or specified time expires.
	Control of System	BscCancel	Cancels an error.
		BscChangeTask	Changes a task.
		BscContinueJob	Starts job. (Execution starts from the current line of the current job.)
		BscConvertJobP2R	Converts a pulse job to a relative job in a specified frame system.
		BscConvertJobR2P	Converts a relative job in a specified frame system to a pulse job.
		BscDeleteJob	Deletes a job.
		BscHoldOff	Sets hold-OFF.
		BscHoldOn	Sets hold-ON.

		BscImov	Moves robot with linear motion from the current position for incremental value in a specified frame system.
		BscMDSP	Sends message data.
		BscMov	Moves robot with specified motion from the current position to a target posiiton in a specified frame system.
		BscMovj	Moves robot with joint motion to a target position in a specified frame system.
		BscMovl	Moves robot with linear motion to a target position in a specified frame system.
		BscOPLock	Interlocks the robot.
		BscOPUnLock	Releases the robot interlocked status.
		BscPMov	Moves robot to a specified pulse position.
		BscPMovj	Moves robot to a specified pulse position with joint motion.
		BscPMovl	Moves robot to a specified pulse position with linear motion.
		BscPutUFrame	Sets a specified user frame data.
		BscPutVarData	Sets variable data.
		BscStartJob	Starts job. (A job to be started has the job name which is selected last.)
		BscSelectJob	Selects a job.
		BscSelectMode	Selects mode. (Teach or Play)
		BscSelLoopCycle	Changes the cycle mode to auto mode.
		BscSelOneCycle	Changes the cycle mode to 1-cycle mode.
		BscSelStepCycle	Changes the cycle mode to step mode.
		BscSetLineNumber	Sets a line number of current job.
		BscSetMasterJob	Sets a job as a master job.
		BscReset	Resets a robot alarm.
		BscSetCtrlGroup	Sets a control group.
		BscServoOff	Sets servo power supply OFF.
		BscServoOn	Sets servo power supply ON.
		BscUpload	Receives a specified file from the robot controller.
Support of DCI function		BscDCILoadSave	Loads or saves a job with DCI instruction.
		BscDCILoadSaveOnce	Loads or saves a job with DCI instruction.
		BscDCIGetPos	Gets a variable with DCI function.
		BscDCIPutPos	Sets a variable with DCI function.
Read/Write of I/O signals		BscReadIO	Reads specified count of coil status.
		BscWriteIO	Writes specified count of coil status.
Other functions		BscClose	Release a communication handler.
		BscCommand	Sends a communication command.
		BscConnect	Connect a communication line.
		BscDisConnect	Disconnect a communication line.
		BscDiskFreeSizeGetReturns	specified drive vacant capacity.
		BscGets	Sends a character string by transmission in TTY level.
		BscInBytes	Returns the number of character which are received by transmission in TTY level.
		BscIsErrorCode	Gets an error code.
		BscOpen	Gets a communication handler.
		BscOutBytes	Returns the remaining number of characters which are sent by transmission in TTY level.
		BscPuts	Sends a character string by transmission in TTY level.
		BscSetBreak	Cancels transmission.
		BscSetCom	Sets a communication parameter.
		BscSetCondBSC	Sets a communication control timer or relay counter.
		BscStatus	Reads the status.

附录 3 机器人网络控制平台系统关键类的程序实现

```
// Robot.h: interface for the Robot class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_ROBOT_H__F59A748B_2768_466F_A388_46277C8A8B06__INCLUDED_)
#define AFX_ROBOT_H__F59A748B_2768_466F_A388_46277C8A8B06__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
struct STATUSINFORMATION                    //机器人返回的状态信息结构
{
    CString     curJobName;                 //BscIsJobName
    short       curJobLine;                 //BscIsJobLine
    short       curJobStep;                 //BscIsJobStep
    WORD        d1;                         //BscGetStatus
    WORD        d2;
    short       error;                      //BscError2
    double      xp[12];                     //BscIsLoc  :   xyz coordinate system
    double      jp[12];                     //BscIsLoc  :   joint coordinate system
    WORD        rconf;                      //BscIsLoc  :   form storage pointer

};
struct POSTION                              //单步控制时的位置参数结构
{
    bool    controlMode  ;
    //contorlMode=0；是用坐标控制 xyz；controlMode=1，使用关节脉冲控制；
    CString     movtype;
    CString     vtype;
    double      spd;
    CString     framename;
    short       rconf;
    short       toolno;
    double      p[12];
};


class Robot
{
public:
    void JobWait();
//此函数当任务等待任务完成后才返回
    bool ServoOff();                                    //关闭伺服电源
    void GetRobotStatus(STATUSINFORMATION &status);     //获得机器人的状态信息
    bool StartStep(POSTION &pos);                       //单指令执行
    void JobDelete(CString jobname);
//当任务下载到机器人控制柜运行完后，把任务删除，为了节省控制柜的内存。
    void Init(HWND hwd);                                //机器人初始化
    void ContinueRun();
//任务暂时停止运行后，重新运行任务
```

```
    void StopRun();                                          //暂时运行任务
    Robot();
    bool StartJob(CString tJobName);                         //开始一个任务。自主模式
    virtual ~Robot();

private:
    CString jobName;                                         //机器人运行的任务名，来自外界
的输入

    bool ServoOn();                                          //打开伺服电源
    bool JobDownLoad(CString jobName);                       //把任务下载到控制器上
    short disConnect();                                      //中断和机器人连接
    HWND inputHwnd;
//inputHwnd 是使用此 robot 类的窗口句柄
    short nCid;                                              //和控制柜连接的通道值
    short BuildConnect();                                    //建立和机器人的连接
};

#endif // !defined(AFX_ROBOT_H__F59A748B_2768_466F_A388_46277C8A8B06__INCLUDED_)


// Construction/Destruction
//////////////////////////////////////////////////////////////////////

Robot::Robot()
{

}

Robot::~Robot()
{
   short rc;
   rc=disConnect();
   if(rc!=0)
       AfxMessageBox("disClose()失败");
}

short Robot::BuildConnect()
{
    short ncid;
    short rc;
    char cur_dir[_MAX_DIR];                  // 当前的目录
    char *IPAddress= TEST_IP_ADDRESS;    // 机器人的 ip 地址

    _getcwd( cur_dir, sizeof(cur_dir) );
    if( TEST_TRANS_MODE==0 )
    {
        // 使用 com 口进行通讯
        ncid = BscOpen( cur_dir, PACKETCOM );
        if( ncid < 0 )
        {
            return( ncid );
        }

        // 设置串口的通讯参数
```

```
            rc = BscSetCom( ncid, 1, 9600, 2, 8, 0 );
            if( rc != 1 )
            {
                rc = BscClose( ncid );
                return( -1 );
            }

        } else
        {
            // 采用 Ethernet 进行通讯
            ncid = BscOpen( cur_dir, PACKETETHERNET );
            if( ncid < 0 )
            {
                return( ncid );
            }

            // Ethernet 的参数设置
            rc = BscSetEther( ncid, "211.82.118.115", 0, inputHwnd );
            if( rc != 1 )
            {
                rc = BscClose( ncid );
                return( -1 );
            }
        }
        // 与机器人建立连接
        rc = BscConnect( ncid );
        if( rc != 1 )
        {
            rc = BscClose( ncid );
            return( -1 );
        }

        return( ncid ); // <------- MOTOCOM32: Motohelp.hlp Visual C++

}

short Robot::disConnect()
{
        short rc;
        //取消和控制柜的连接
        rc = BscDisConnect( nCid );

        // 取消连接句柄
        rc = BscClose( nCid );

        return( rc );     // 0:Normal completion  Others:Failed to release
}

bool Robot::JobDownLoad(CString jobName)
{
        short rc;
        if(!jobName.IsEmpty())
        {
            //向机器人发送要执行的任务
            rc=BscDownLoad(nCid,(char *)(LPCSTR)jobName);
            if(rc!=0)
            {
```

```
            AfxMessageBox("JobDownLoad()中的 BscDownLoad()调用返回值有错。请注意！");
            return false;
        }
    }
    else
    {
        AfxMessageBox("JobDownLoad(CString jobName)任务名为空。请注意！");
        return false;
    }

    return true;

}

bool Robot::ServoOn()
 //07.04.06 为了改变单步运行老是弹出下面的一些对运行没有影响的对话框
{
    short rc,rc1;
/*
    rc=BscIsServo(nCid);
    if(rc==-1)
    {
        AfxMessageBox("Robot::ServoOn().BscIsServon(nCid)调用失败。请注意！");
        return false;
    }
    else if(rc==0)                          //servo is "off",open servo.
    {
        rc1= BscIsPlayMode(nCid);
        switch(rc1)
        {
        case -1:
            AfxMessageBox("Robot::ServoOn().BscIsPlayMode(nCid)调用失败。请注意！");
            return false;
            break;
        case 0:
//          AfxMessageBox("Robot::ServoOn().BscIsPlayMode(nCid) not operating。请注意！");
            BscSelectMode(nCid,2);                      //2=play,1=teach;
            break;
        default:
            break;                  //在 PLAY MODE 模式
        }
    */
        rc1=BscServoOn(nCid);
        if(rc1!=0)
        {
            AfxMessageBox("Robot::ServoOn().BscServoOn(nCid)没有打开 servo。请注意！");
            return false;

        }
        return true;

    //}
//  else if(rc==1)                  //servo have "on"
//      return true;
//  else
//      return false;
}
```

```
bool Robot::StartJob(CString tJobName)
{
    short rc;
    jobName=tJobName;
    if(!(JobDownLoad(jobName)))
    {
            AfxMessageBox("Robot::StartJob().JobDownLoad 没有成功加载。请注意！");
        //  return false;
    }

    rc=BscSelectJob(nCid,(char *) (LPCSTR)jobName);
    if(rc!=0)
    {
        AfxMessageBox("Robot::StartJob().BscSelectJob()失败！");
    //  return false;
    }


        if(!(ServoOn()))
    {
            AfxMessageBox("Robot::StartJob().ServoOn()执行没有成功。请注意！");
        //  return false;
    }
    rc=BscStartJob(nCid);
    switch(rc)
        {
        case 0:

            //开始任务
            break;
        case 1:
            AfxMessageBox("Robot::StartJob().BscStartJob(nCid) 当前指定的任务不存在于控制
柜中。请注意！");
            return false;
            break;
        default:
            AfxMessageBox("Robot::StartJob().BscStartJob(nCid) 发生错误,。请注意！");
            return false;
            break;

        }

    //等 JobDownLoad()函数成功时，在此 JobWait()添加 JobDelete()函数。

//  JobDelete(jobName);
    return true;
}

void Robot::StopRun()
{
    short rc;
    rc=BscHoldOn(nCid);
    if(rc!=0)
            AfxMessageBox("Robot::StopRun() 不能停止。请注意！");

}
```

```
void Robot::ContinueRun()
{
        short rc;
    rc=BscHoldOff(nCid);
    if(rc!=0)
            AfxMessageBox("Robot::ContinueRun().BscHoldOn(nCid) 发生错误。请注意！");

    rc=BscContinueJob(nCid);
    if(rc!=0)
            AfxMessageBox("Robot::ContinueRun().BscContinueJob(nCid) 发生错误。请注意！");

}

void Robot::Init(HWND hwd)
{
    inputHwnd=hwd;
    if(inputHwnd==NULL)
        AfxMessageBox("Robot::Init(HWND hwd)输入的窗口句柄为空");

    if((nCid=BuildConnect())<0)
            AfxMessageBox("Robot::BuildConnect()失败");
}

void Robot::JobDelete(CString jobname)
{
    short rc,rc1;
    if(!jobname.IsEmpty())
    {
        //向机器人发送要删除的任务
    rc=BscSelectJob(nCid,(char *) (LPCSTR)jobname);
    if(rc ==0)
    {
        ;
    }
   else  if(rc ==1)
    {
            AfxMessageBox("Robot::StartJob().BscSelectJob()失败,任务不能删除！");

        return ;
    }
   else
   {
            AfxMessageBox("Robot::StartJob().BscSelectJob()失败");


   }
        rc1=BscDeleteJob(nCid);
       if(rc!=0)
        {
            AfxMessageBox("JobDelete()中的 BscDownLoad()调用返回值有错。请注意！");
        }
    }
    else
    {
        AfxMessageBox("JobDelete(CString jobName)任务名为空。请注意！");
    }
```

```
}

bool Robot::StartStep(POSTION &pos)
{
    short rc;

        ServoOn();

    //contorlMode=0；是用坐标控制 xyz；controlMode=1，使用关节脉冲控制；
    if(pos.controlMode==0)
    {
        //pos.p 中，平需要的是 double 型的指针，可能存在问题
        rc= BscMov(nCid,(char*)(LPCSTR)pos.movtype,(char*)(LPCSTR)pos.vtype,pos.spd,(char
*)(LPCSTR)pos.framename,pos.rconf,pos.toolno,pos.p);
        if(rc!=0)
        {
            AfxMessageBox("tartStep(POSTION &pos)::BscMov()调用不成功，没有实现机器人的
坐标单步运行！");
            return false;
        }
        else
            return true;
    }
    else
    {
        rc=
    BscPMov(nCid,(char*)(LPCSTR)pos.movtype,(char*)(LPCSTR)pos.vtype,pos.spd,pos.toolno,pos.
p);
        if(rc!=0)
        {
            AfxMessageBox("tartStep(POSTION &pos)::BscPMov()调用不成功，没有实现机器人
的关节单步运行！");
            return false;
        }
        else
            return true;

    }

}

void Robot::GetRobotStatus(STATUSINFORMATION &status)
{

    short rc;       //读取机器人当前运行的任务名和任务的行数和运行点数
    rc=BscIsJobName(nCid,(char *)(LPCSTR)status.curJobName,20);
    if(rc==-1)
    {
        AfxMessageBox("Robot::GetRobotStatus.BscIsJobName 无法获取当前任务名 ");
        return;
    }


        rc=BscIsJobLine(nCid);
    if(rc==-1)
```

```
    {
        AfxMessageBox("Robot::GetRobotStatus.BscIsJobLine 无法获取当前任务行数 ");
        return;
    }
    else
        status.curJobLine=rc;

        rc=BscIsJobStep(nCid);
    if(rc==-1)
    {
        AfxMessageBox("Robot::GetRobotStatus.BscIsJobStep 无法获取当前任务点数");
        return;
    }
    else
        status.curJobStep=rc;

    //获取机器人的 xyz 和 joint 坐标值

    rc=BscIsLoc(nCid,0,&status.rconf,&status.xp[0]);

        if(rc==-1)
    {
        AfxMessageBox("Robot::GetRobotStatus.BscIsLocp 无法获取 xyz 坐标值");
        return;
    }
            rc=BscIsLoc(nCid,1,&status.rconf,&status.jp[0]);
        if(rc==-1)
    {
        AfxMessageBox("Robot::GetRobotStatus.BscIsLocp 无法获取 joint 坐标值");
        return;
    }

    rc=BscGetError2(nCid);
    if(rc==-1)
    {
        AfxMessageBox("Robot::GetRobotStatus.BscGetError2 无法获取机器人错误");
        return;
    }
    else
        status.error=rc;

    rc=BscGetStatus(nCid,&status.d1,&status.d2);
        if(rc==-1)
    {
        AfxMessageBox("Robot::GetRobotStatus.BscGetStatus 无法获取机器人状态");
        return;
    }

}

bool Robot::ServoOff()
{

    short rc;
    rc=BscServoOff(nCid);
    if(rc!=0)
    {
```

```
        AfxMessageBox("Robot::ServoOff().BscServoOff(nCid)伺服电源没有成功关闭");
        return false;
    }
    else
        return true;
}

void Robot::JobWait()
{
    short rc;
    rc=BscJobWait(nCid,-1);
    switch(rc)
        {
        case 0:
            //任务完成。
    //      AfxMessageBox("Robot::StartJob().BscJobWait(nCid,-1); 任务运行已完成。");
            break;
        case -1:
            ;
        case -2:
            AfxMessageBox("Robot::StartJob().BscJobWait(nCid,-1);
 //任务正在运行，不能停止。请注意！");
            break;
        default:
            AfxMessageBox("Robot::StartJob().BscJobWait(nCid,-1); 发生错误。请注意！");
            break;

        }

        //等 JobDownLoad()函数成功时，在此添加 JobDelete()函数。
    JobDelete(jobName);
}

/Define.h
#ifndef Define_h
#define Define_h
#include <stdlib.h>
#include "Robot.h"
#define COUNT 22              //表示传送的信息格式中有五个元素
#define MAXLENGTH   40        //表示每个元素最大可以占 20 个字符

/*
struct STATUSINFORMATION
{
    CString     curJobName;             //BscIsJobName
    short       curJobLine;             //BscIsJobLine
    short       curJobStep;               //BscIsJobStep
    WORD        d1;                     //BscGetStatus
    WORD        d2;
    short       error;                    //BscError2
    double      xp[6];                    //BscIsLoc  :     xyz coordinate system
    double      jp[6];                    //BscIsLoc  :     joint coordinate system
    WORD        rconf;                    //BscIsLoc  :     form storage pointer

};
struct POSTION                          //单步控制时的位置参数
{
```

```
    bool      controlMode  ;              //contorlMode=0; 是用坐标控制 xyz; controlMode=1,
使用关节脉冲控制;
    CString       movtype;
    CString       vtype;
    double        spd;
    CString       framename;
    short         rconf;
    short         toolno;
    double        p[12];
};
*/
enum   Privilege
{

Idle=0,CommandAndEmluator=1,CommandAndVideo,IndependenceAndEmluator,IndependenceAndVi
deo
};
//也可以用来在初始化时，用户选择的控制方式。

enum RobotType
{
    robot1=1,robot2,robot3
};
enum ControlCommand
{
    Up=1,Down,Right,Left,Fore,Back,Pause,Continue,Reset,Stop,Disconnect
};

enum ErrorInformation
{

    SUCCESSFUL=0,NOTEXISTUSERNAME=1,ERRORPOSSWORD,N0HIGHPRIVEILEGE,NOT
EXISTTTHISROBOT,HAVEANOTHERUSER，CONTROLTIMEOUT
};
class InfoFormat
{
public:
//      @InitialInformation:@userName@password@currentlyState@robotType@EerrorInfomation
    /*
    infoString=(CString)mm[0];
    userName=(CString)mm[1];
    password=(CString)mm[2];
    currentlyState=(Privilege) atoi(mm[3]);
    privilege=(Privilege) atoi(mm[4]);  //atoi（char *）是用来把字符串的数字转化为整数

    CString   movtype;
    int controlmode;
    int frmaename;
    int toolno;
    doble *xp;
    doble *jp;

    errorCode=(ErrorInformation) atoi(mm[5]);
```

```
CString jobname;
int d1;
int d2;
int error;
doble *xp;
doble *jp;

位置信息
CString   movtype
int controlmode;
int frmaename;
int toolno
doble *xp;
doble *jp;
*/
    CString   movType;
//当为位置信息时，表示的时采用“movj，movi”等；当为状态反馈信息时，为当前任务名//(jobname)
    WORD controlMode;
//为位置信息时，contorlMode=0；是用坐标控制 xyz；controlMode=1，使用关节脉冲控制；//当为当为状态反馈信息时
    WORD frameName;
    int toolNo;                                          //选择那个工具坐标系
    float xp[6];                                         //xyz 坐标系，六个点
    float jp[6];                                         //joint 坐标系，六个点


    CString         userName;
    CString         password;
    Privilege   currentlyState;          //表示当前用户处于那一个控制级别
    Privilege      privilege;            //表示用户可以享用的级别
    CString          infoString;
    ErrorInformation errorCode;          //如果发生错误，则返回错误码，负责返回 0

    void RecoverOriginData();            //初始赋值
    void AnalyzeString(CString s);
    void   ConveyToString(CString &str);

    void GetRobotStatus(STATUSINFORMATION &status);
    void GetRobotPosition(POSTION &position);
    void SetRobotStatus(STATUSINFORMATION &status);
    void SetRobotPosition(POSTION &position);

     InfoFormat();
     InfoFormat(CString str) ;
};
#endif

//defien.cpp
#include <iostream.h>
#include <stdlib.h>
#include "stdafx.h"

#include "Define.h"

 InfoFormat::InfoFormat()
{
```

```
    RecoverOriginData();
    errorCode =SUCCESSFUL;
}
 InfoFormat::InfoFormat(CString str)
{
    RecoverOriginData();
    infoString=str;                                    //str 可以是： UserInformation，
ControlInformation,InitialInformation。
    errorCode =SUCCESSFUL;
}

//InfoFormat::ConveyToString(CString &str)是用来把用户的信息转换成字符串格式，以方便发送。
//格式为：" infoString userName password currentlyState privilege"
void InfoFormat::ConveyToString(CString &str)
{

    CString  temp;
    /*这一段代码实现的功能和下一句的功能一样
    temp='@'+infoString+'@'+userName+'@'+password+'@';
    _itoa((int)currentlyState,&buff1,3);    //char *_itoa( int value, char *str, int radix );

    _itoa((int)privilege,&buff2,4);
    temp+=(CString)buff1+'@'+(CString)buff2;
    */

//
    temp.Format("@%s@%s@%s@%d@%d@%d",infoString,userName,password,(int)currentlyState,
(int)privilege,(int)errorCode);
    temp.Format("@%s@%s@%s@%d@%d@%s@%d@%d@%d@%f@%f@%f@%f@%f@%f@%
f@%f@%f@%f@%f@%f@%d"
                ,infoString,userName,password,(int)currentlyState,(int)privilege
                ,movType,controlMode,frameName,toolNo
                ,xp[0],xp[1],xp[2],xp[3],xp[4],xp[5]
                ,jp[0],jp[1],jp[2],jp[3],jp[4],jp[5]
                ,(int)errorCode);
    str=temp;

}

//InfoFormat::AnalyzeString(CString s)是用来把客户端发送的字符串转化为具体的变量值。
void InfoFormat::AnalyzeString(CString s)
{
    char *index[COUNT];
    char *result;
    char  mm[COUNT][MAXLENGTH];
    int count;
    CString temp=s;                    // CString temp='@'+s;为了取得每个字符之间的指针值
result=strpbrk(temp,"@");      //strpbtk()是返回字符串 temp 中第一次出现的" "空字符的地址
        index[0]=result++;

    for(int i=1;i<COUNT;i++)
    {
        result=strpbrk(result,"@");
        index[i]=result;
        result++;
    }
     for( i=0;i<COUNT;i++)
```

```
    {
        count=index[i+1]-index[i];
        if(i==COUNT-1)
            count=2;//因为最后一个空字符的指针值不知，但最后一个数长度肯定为 1
        for(int j=0;j<count-1;j++)
        {
            mm[i][j]=index[i][j+1];
        }

        mm[i][count-1]=NULL;
    }
    infoString=(CString)mm[0];
    userName=(CString)mm[1];
    password=(CString)mm[2];
    currentlyState=(Privilege) atoi(mm[3]);
    privilege=(Privilege) atoi(mm[4]);  //atoi（char *）是用来把字符串的数字转化为整数
    movType= (CString) mm[5];
    controlMode= atoi(mm[6]);
    frameName =atoi(mm[7]);
    toolNo = atoi(mm[8]);
    for(i =0;i<=5;i++)
    {
        xp[i] = atof(mm[i+9]);
        jp[i] = atof(mm[i+15]);
    }
    errorCode=(ErrorInformation) atoi(mm[21]);
}
//当服务器端反馈错误信息回来时，调用，errorCode 为相应的错误返回码。
void InfoFormat::RecoverOriginData()
{
    userName="userName";
    password="password";
    currentlyState=Idle;
    privilege=IndependenceAndVideo;
    infoString="invalidation information format";
    movType = "movj";
    controlMode = 1;
    frameName = 2;
    toolNo = 3 ;

    //下面的是在两种坐标下的第二原点位置，
    xp[0] = 399.443;
    xp[1] = 0;
    xp[2] = 285.002;
    xp[3] = 180;
    xp[4] = 0.35;
    xp[5] = 0;

    jp[0] = 0;
    jp[1] = 0;
    jp[2] = 0;
    jp[3] = 0;
    jp[4] = -82243;
    jp[5] = 0;

//  errorCode =SUCCESSFUL;
}
```

```
void InfoFormat::GetRobotStatus(STATUSINFORMATION &status)
{
    status.curJobName = movType;
    status.d1 = (WORD)controlMode;
    status.d2 = (WORD)frameName;
    status.error =toolNo;
    for(int i=0; i<6 ; i++)
    {
        status.xp[i] = xp[i];
        status.jp[i] = jp[i];
    }
}
void InfoFormat::GetRobotPosition(POSTION &position)
{
    position.movtype = movType;
    position.controlMode =(bool) controlMode;

    switch(frameName)
    {
    case 1:
        position.framename = "BASE";break;
    case 2:
        position.framename= "ROBOT";break;
    case 3:
        position.framename = "USER" ;break;
    case 4:
        position.framename   = "TOOL"; break;
    }

    position.toolno   = toolNo;
        if(position.controlMode == 0)
        {
            for(int i=0; i<6 ; i++)
            {
            position.p[i] = xp[i];
            }
        }
        else
        {
            for(int i=0; i<6 ; i++)
            {
                position.p[i] = jp[i];
            }
        }
                for(int i=6; i<=11 ; i++)
            {
                position.p[i] = 0;
            }
        position.spd = 20;
        position.rconf = 0;
        position.vtype = 'V';
}

void InfoFormat::SetRobotStatus(STATUSINFORMATION &status)
{

    movType=(CString) status.curJobName;
```

```
    controlMode= status.d1 ;
   (WORD)frameName = status.d2 ;
    toolNo=status.error;
   for(int i=0; i<6 ; i++)
   {
       xp[i] =status. xp[i];
       jp[i] =status. jp[i];
   }
}
void InfoFormat::SetRobotPosition(POSTION &position)
{
    movType      = position.movtype;
    controlMode=position.controlMode;

    if(position.framename=="BASE")
    { frameName= 1;}
   else if(position.framename=="ROBOT")
    { frameName= 2;}
   else if(position.framename=="USER")
    {frameName= 3;}
   else if(position.framename=="TOOL")
    { frameName= 4;}

   toolNo   =position.toolno;
       if(position.controlMode == 0)
       {
           for(int i=0; i<6 ; i++)
           {
           xp[i] =position. p[i];
           }
       }
       else
       {
           for(int i=0; i<6 ; i++)
           {
               jp[i] = position.p[i];
           }
       }

}

// Client.h: interface for the CClient class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_CLIENT_H__1C42589C_FB08_438C_8524_A5A40CB3B506__INCLUDED_)
#define AFX_CLIENT_H__1C42589C_FB08_438C_8524_A5A40CB3B506__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "winsock.h"
#include "NetTransmit.h"
#include "NetTransmitDlg.h"
class CNetTransmitDlg;

class CClient
{
```

```
public:
    void GetString(CString &str);
    void SendString(CString s);
    CString m_strServer;
    sockaddr_in m_addr;
    SOCKET m_hSocket;
    HWND m_hWnd;
    UINT m_uPort;
    bool InitAndConnect(HWND hwnd,UINT port,CString strServer);
    CClient();
    virtual ~CClient();

private:
    void ClientInit();
};

#endif
// !defined(AFX_CLIENT_H__1C42589C_FB08_438C_8524_A5A40CB3B506__INCLUDED_)

// Client.cpp: implementation of the CClient class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "NetTransmit.h"
#include "NetTransmitDlg.h"
#include "Client.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CClient::CClient()
{
    m_hSocket=NULL;
}

CClient::~CClient()
{

}

bool CClient::InitAndConnect(HWND hwnd,UINT port,CString strServer)
{
    m_hWnd=hwnd;
    m_uPort=port;
    m_strServer=strServer;

    if(m_hSocket!=NULL)
    {
        closesocket(m_hSocket);
        m_hSocket=NULL;
```

```
    }

    if(m_hSocket==NULL)
    {
        m_hSocket=socket(AF_INET,SOCK_STREAM,0);
        ASSERT(m_hSocket!=NULL);
        ClientInit();
    }

    m_addr.sin_family =AF_INET;
    m_addr.sin_port =htons(m_uPort);
    m_addr.sin_addr.S_un.S_addr=inet_addr(m_strServer.GetBuffer(0));
    int ret=0;
    ret=connect(m_hSocket,(LPSOCKADDR) &m_addr,sizeof(m_addr));
    if(ret==SOCKET_ERROR)
    {
        if(GetLastError()!=WSAEWOULDBLOCK)
        {
            AfxMessageBox(_T("请确认服务器已经打开并工作在同样的端口"));
            return FALSE;
        }
    }

    return TRUE;
}

void CClient::ClientInit()
{
    if(WSAAsyncSelect(m_hSocket,m_hWnd,CLI_MESSAGE,FD_ACCEPT|FD_READ|FD_WRITE|
FD_CLOSE)>0)
        AfxMessageBox("WASSyncSelect() make a mistake in select!");
}

void CClient::SendString(CString s)
{
    if(send(m_hSocket,s.GetBuffer(0),s.GetLength(),0)==SOCKET_ERROR)
        AfxMessageBox("socket send data error");
}

void CClient::GetString(CString &str)
{
    recv(m_hSocket,str.GetBuffer(0),1024,MSG_DONTROUTE);
 //MSG_DONTROUTE 的意义——即 flags 可以取几个值
}
```

硕士期间发表论文目录

[1]. 王文静，辛洪兵，刘振宝. 在包装流水线中边缘检测算法的比较. 包装工程. 2006. 6(27): 95-107

[2]. 刘振宝，辛洪兵，王文静. 网控机器人技术及其控制系统的研究状况. 北京工商大学学报. 2006.24(3)：32-36

[3]. 辛洪兵，刘振宝，王文静. 方便米饭生产线卸料机组设计. 食品科技. 2006.10（180）：138-140

[4]. 徐继峰，刘玉德，刘振宝.基于环形线圈的智能交通的动态数据采集与处理.北京工商大学学报.2006.5(21)

致　谢

首先，感谢我的导师辛洪兵教授，本论文是在导师的悉心指导下完成的，从论文的选题、研究、撰写到定稿，都得到辛老师细心的指导。在此向恩师表示最诚挚的感谢和深深的敬意。

导师辛洪兵教授平易近人，对学生关怀备至，在工作上精益求精，启发学生开拓视野，强调科研与实践相结合，鼓励学生学以致用，在实践中发现问题、解决问题。导师具有丰富的科研经验与雄厚的理论水平，其开明的学术思想、求实的科研作风和对科学研究及发展趋势的深刻认识给我以极大的影响、启迪和熏陶，令我终生受益。在三年的研究生学习期间，无论是在生活上还是学习上，辛老师都给了我无微不至的关怀，使我受益匪浅，并将激励我在今后的科研工作和学习中更加努力。对辛老师的恩情难以用言语表达，只希望自己以后刻苦钻研，不断进取，才能不辜负老师的期望。

在论文工作期间，王文静、邢亮、徐继峰、徐正兴同学在我的毕业设计中给了我很大的帮助，在此一并予以感谢。

最后，感谢母校北京工商大学的培养和教育，特别要感谢我的家人，是他们精神上的鼓励和物质上的帮助激励本人不断努力和前进。