第一章 概 论

§1.1 机器人技术

一、机器人的由来

机器人自从问世以来，经过20多年的发展，目前已被公认为是一种现代科学技术的典型产物。但是，无论是作为一种自动化机械已经在现代化生产中实际使用的机器人，还是作为人工智能学科的一个研究对象正处于实验阶段中的机器人，都难以符合"机器人"这个名词给人的印象。机器人其实是一种机器，它们既不齐备人类完美的器官和功能，也不具有而且也不一定具有现代人的外表(甚至相反，有些机器人的形态倒类似于某些动物)，尽管如此，"机器人"这一命名却是事出有因的。

1920年，捷克作家卡雷尔·查培克（Karel Capek）编写了一部幻想剧：《罗莎姆万能罗博特公司》("Rossum's Universal Robots")。剧中描写一家公司发明并制造了一大批能听命于人，能劳动而且形状象人的机器，公司驱使这些人造劳动者进行各种日常劳动，甚至取代了世界各国工人的工作，而进一步的研究竟能使这些机器富有感情，于是导致了它们反抗主人的暴乱。剧中的人造劳动者取名为捷克语 Robota，意为"苦力"、"劳役"，英语 Robot 系由此衍生而来。该剧哄动一时，很快译传国外。此后，种种"人形机器"见之于各类科学幻想作品。

机器人形象的产生充分说明了人类对于先进生产工具的创造性想象和勇敢追求。人们期待着诞生一种通用、柔软、灵活的自动机械，它与单能的传统机器不同，它能模仿人的器官的功能，从事那些只有人才能很好完成的工作。于是，人们这种美好的愿望给科学技术的研究提出了一个深入的课题——用工程的方法实现人体所特有的动作机能，以及完成这些动作所必要的智能。

二、机器人技术的进程

机器人从幻想世界真正走向现实世界是从自动化生产和科学研究的发展需要出发的。

遥控操纵器（Teleoperator）和数控机床的出现为机器人的产生准备了技术条件。

二次世界大战期间，在放射性材料的生产和处理过程中应用了一种简单的遥控操纵器。操作人员在一层很厚的混凝土防护墙外通过观察，用手操纵两个操作杆（主动部分），操作杆与墙内的一对机械抓手（从动部分）通过六个自由度的传动机构相连，于是机械抓手就能复现人手的动作位置和姿态，代替了操作人员的直接操作。1947 年，人们对这种遥控操纵器进行改进，采用电动伺服方式，使从动部分能相对于主动部分作跟随运动。

1949年，由于生产先进飞机的需要，美国麻省理工学院辐射实验室（MIT Radiation Laboratory）开始研制数控铣床，把复杂伺服系统的技术与最新发展的数字计算机技术结合起来，1953 年研制成功。切削模型以数字形式通过穿孔纸带输入机器，然后控制铣床的伺服轴按照模型的轨迹作切削动作。

1954 年，美国的 George C. Devol 设计并制作了世界上第一台机器人实验装置，发表了《适用于重复作业的通用性工业机器人》一文，并获得了专利。Devol 巧妙地把遥控操纵器的关节型连杆机构与数控机床的伺服轴连结在一起，预定的机械手动作一经

编程输入后，机械手就可以离开人的辅助而独立运行．这种机器人也可以接受示教而能完成各种简单任务．示教过程中操作者用手带动机械手依次通过工作任务的各个位置，这些位置序列记录在数字存储器内，任务的执行过程中，机器人的各个关节在伺服驱动下再现出那些位置序列．因此，这种机器人的主要技术功能就是"可编程"以及"示教再现"．

60 年代，机器人产品正式问世，机器人技术开始形成．

1960 年，美国的 Consolidated Control 公司根据 Devol 的技术专利研制出第一台机器人样机，并成立了 Unimation 公司，定型生产了 Unimate（意为"万能自动"）机器人．同时，美国"机床与铸造公司"（AMF）设计制造了另一种可编程的机器人 Versatran（意为"多才多艺"）．这两种型号的机器人以"示教再现"的方式在汽车生产线上成功地代替工人进行传送、焊接、喷漆等作业，它们在工作中表现出来的经济效益、可靠性、灵活性，使其它发达国家工业界为之倾倒．于是 Unimate 和 Versatran 作为商品开始在世界市场上销售，日本、西欧也纷纷从美国引进机器人技术．

在机器人崭露头角于工业生产的同时，研究领域不断地把机器人技术引向深入发展．1961 年，美国麻省理工学院 Lincoln 实验室把一个配有接触传感器的遥控操纵器的从动部分与一台计算机连结在一起，这样形成的机器人可以凭触觉决定物体的状态．随后，用电视摄像头作为输入机理的计算机图像处理、物体辨识的研究工作也陆续取得成果．1968 年，美国斯坦福人工智能实验室（SAIL）的 J. McCarthy 等人研究了新颖的课题：研制带有手、眼、耳的计算机系统．于是，智能机器人的研究形象逐渐丰满起来．

70 年代以来，机器人产业蓬勃兴起，机器人技术发展为专门的学科．

1970 年，第一次国际工业机器人会议在美国举行．工业机器人各种卓有成效的实用范例促成了机器人应用领域的进一步扩

展，同时，又由于不同应用场合的特点，导致了各种坐标系统、各种结构的机器人相继出现。而随后的大规模集成电路技术的飞跃发展及微型计算机的普遍应用，则使机器人的控制性能大幅度地得到提高，成本不断降低。于是，导致了数百种类的不同结构、不同控制方法、不同用途的机器人终于在 80 年代的今天，真正进入了实用化的普及阶段。

据统计，1980 年，世界上可编程机器人的总数量已超过万台，预计 1990 年将接近 15 万台(图 1.1)。就机器人的世界年产量和产值来说，1983 年的产量为 2.4 万台，产值相当于 10 亿美元，预计 1990 年将达 7.8 万台，产值约 44 亿美元(图 1.2)。在 1982 年全世界拥有的 57427 台工业机器人(不含固定程序上下料机)中，日本的占有量绝对领先(图 1.3)，这使它享有"机器人王国"之美称；而就机器人台数与产业工人数的比例而言(图 1.4)，瑞典则为目前世界上使用机器人最为普遍的国家。

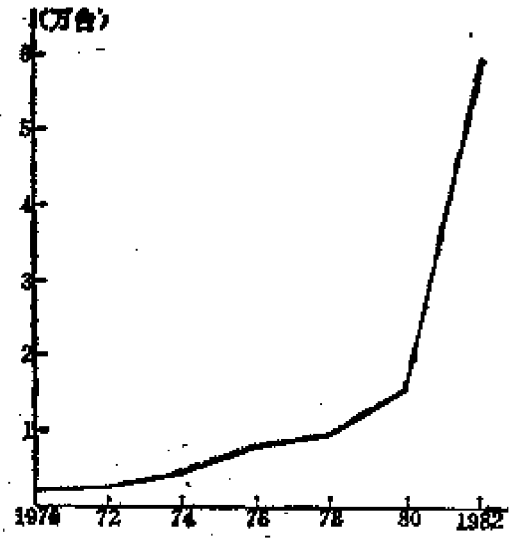


图 1.1 世界机器人数量的增长情况

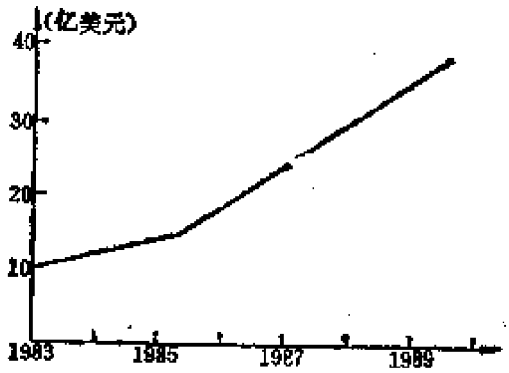


图 1.2 世界机器人的年产量和年产值

70 年代电子计算机在可靠性、信息加工能力等方面的高度发

展，还使得人工智能等前沿学科的技术成果在机器人智能化的研究领域中得到实现的可能。60年代后期的智能机器人多数是在实验室中做些积木的分类、堆放动作。进入70年代，研究重点开始移到工业应用上。例如，配有视觉、触觉的机器人可以用于铸件、泵体的识别和检查，用于集成电路的装配、焊接。再有，早期的智能机器人研究还从对象物识别转向对环境识别，例如景物分析，任务规划及通过自然语言与人对话等课题。另一方面，用机械实现人的手、足及各种动物的爬行机构等仿生机械研究（主要在日本）非常活跃，而用仿生学的方法研究生物视觉、触觉、听觉的实现方法，探索"类大脑"机器的可能，则作为研制智能机器人的一派，也有了一些阶段性成果。

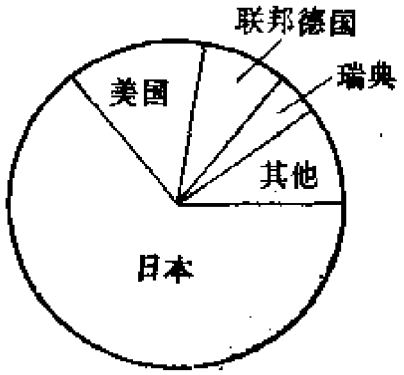


图 1.3　机器人在各国的分布

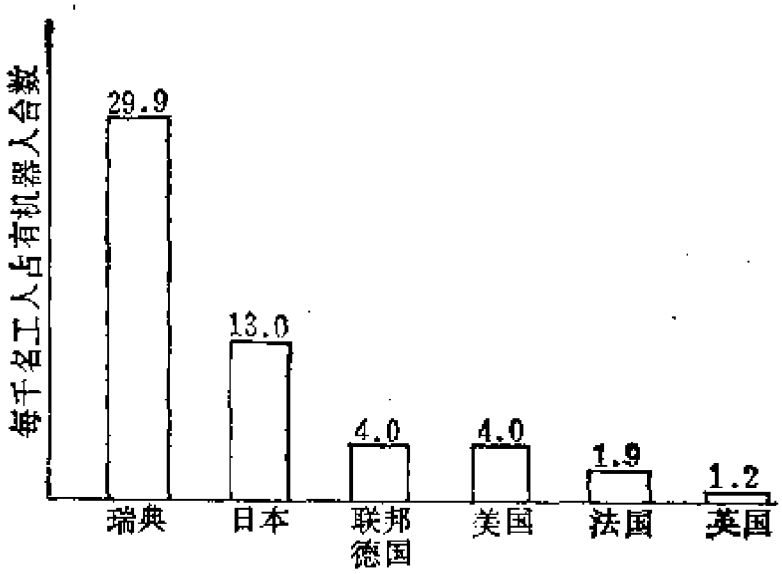


图 1.4　机器人相对于工人数量的分布

经历20多年的发展过程，机器人技术终于形成了一门综合性

学科——机器人学（Robotics）。一般地说，机器人学的研究目标是以智能计算机为基础的机器人的基本组织和操作，它包括基础研究和应用研究两方面内容，课题有：（1）机械手设计；（2）机器人运动学、动力学和控制；（3）轨迹设计和规划；（4）传感器；（5）机器人视觉；（6）机器人控制语言；（7）装置与系统结构；（8）机器智能等。由于机器人学综合了力学、机械学、电子学、生物学、控制论、计算机、人工智能、系统工程等多种学科领域的知识，因此，也有人认为机器人学实际上是一个可分成若干学科的学科类。

§1.2 机器人的定义和分类

一、机器人的定义

机器人的称谓难以顾名思义，国际学术界至今也没有对机器人作出统一公认的，文字严格的定义。不同的专家往往给以各有侧重的说法，不同的国家也往往沿用各自习惯的解释。

有一些描述可以看作对机器人的总括性的、理想化的定义。例如，日本早稻田大学加藤一郎认为：机器人是由能工作的手、能行动的脚和有意识的头脑组成的一个个体，同时具有非接触传感器（相当于耳、目），接触传感器（相当于皮肤），固有感及平衡感等感觉器官和能力。

但是，按照这种定义，当今世界上已有的机器人无疑都要受到求全责备，即使未来的机器人，恐怕也只须具备适应不同场合所必需的功能。

于是，也有一些针对不同型式的机器人分别给以具体解释的定义，而机器人则是一种总称。例如，日本工业机器人协会（JIRA）就列举描述了六种型式的机器人：

（1）手动操纵器——人操纵的机械手。

(2) 固定程序机器人——按照预先设定的顺序、条件与位置逐个进行动作的操作器，设定的信息不容易改变。

(3) 可变程序机器人——按照预先设定的顺序、条件与位置逐个进行动作的操作器，设定的信息容易改变。

(4) 示教再现机器人——由人事先进行示教、存储其作业顺序、位置、时间信息，根据要求读出并进行作业的操作器。

(5) 数控机器人——把作业顺序、位置、时间信息编成数字指令，按指令进行作业的机器人。

(6) 智能机器人——能用感觉和识别功能做决策行动的机器人。

但由于上述型式 (1) 缺乏独立性，型式 (2) 缺乏通用性，而型式 (3) 一般都为非伺服控制，因此有的国家认为它们不属机器人范围，或者最多只能称作原始的机器人。

综合诸家的解释，概括各种机器人的性能，可以按如下特点描述机器人：

(1) 机器人的动作机构具有类似于人或其它生物体某些器官(肢体、感官等)的功能。

(2) 机器人具有通用性，工作种类多样，动作程序灵活易变。

(3) 机器人具有不同程度的智能性，如记忆、感知、推理、决策、学习等。

(4) 机器人具有独立性，完整的机器人系统，在工作中可以不依赖于人的干预。

二、机器人的分类

可以从多方面对机器人进行分类。

按照机器人从低级到高级的发展程度，一般认为有三代机器人。

第一代机器人，主要指只能以“示教-再现”方式工作的机器

人。示教内容为机器人操作机构的空间轨迹、作业条件、作业顺序等.示教方法或是操作员"手把手"地直接做,或是与计算机编程相结合。目前国际上商品化、实用化的机器人大都属于第一代机器人。

第二代机器人具有一定的感觉装置,能获取作业环境、操作对象的简单信息,通过计算机处理、分析,机器人作出一定的推理,对动作进行反馈控制,表现出低级的智能。由于信息处理系统的庞大与昂贵,第二代机器人目前只有少数可以普及应用。

第三代机器人是指具有高度适应性的自治机器人。它具有多种感知功能,可进行复杂的逻辑思维,判断决策,在作业环境中独立行动。第三代机器人与第五代计算机关系密切,而且目前都处于实验探索阶段。

按照性能指标,结构形态等技术内容,机器人的分类很多,例如,机器人的负载能力和动作空间有不同的等级:

超大型机器人　1000kg 以上

大型机器人　100kg ～ 1000kg，10m² 以上

中型机器人　10kg ～ 100kg，1m² ～ 10m²

小型机器人　0.1kg ～ 10kg，0.1m² ～ 1m²

超小型机器人　0.1kg 以下，0.1m² 以下

按照开发内容和目的区分，基本上有三类机器人:

工业机器人 (Industrial Robot)

操纵型机器人 (Teleoperator Robot)

智能机器人 (Intelligent Robot)

研究这三类机器人的系统构成和控制功能具有重要的意义，以下着重给予分别介绍。

§1.3 工业机器人

工业机器人是一类根据预先编制在存储装置内的操作程序，

自动地重复进行作业的机器人，因而也称重复型机器人（Repeatable Robot）。它实际上就是前面所说的第一代机器人。少数工业机器人兼有第二代机器人的特征。

一、系统构成

工业机器人的系统构成如图 1.5 所示。

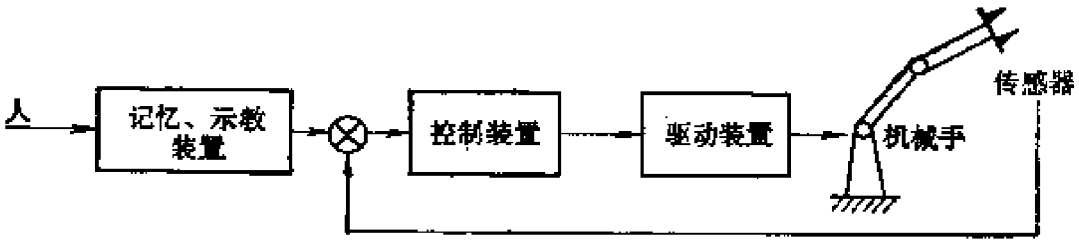


图 1.5　工业机器人的系统构成

工业机器人的本体主要是一只类似于人的上肢功能的机械手臂，或者是无关节结构，或者是关节式结构。如果要在三维空间对物体进行作业，一般需要六个自由度，即沿笛卡尔坐标三轴的直线移动及围绕这三个轴的转动。

工业机器人直接操作对象物的是机械手末端的手爪（亦称 End Effector），它随对象物的形状和材料不同而制成各种各样，例如用以夹持工件的无指手掌，用以抓取棒料的有指手爪（一般为二指，有关节或无关节），用以吸附平板的真空吸盘或电磁吸盘，用以吊挂重物的钩爪等。

早期的工业机器人的示教、记忆、控制装置利用凸轮、挡块、插销板、穿孔纸带、磁鼓、继电器等机电元件构成，而 80 年代的工业机器人则主要使用微处理机系统综合实现上述装置的功能。

驱动装置最为普遍的是伺服电机。大型作业的机器人往往使用液压传动。较为简单的或要求防爆的机器人可采用气动执行机构。

工业机器人的传感器包括外部信息传感器和内部信息传感器。外部信息传感器用以检测、判断工作对象的位置、形状、接触状态等，例如从各种原理的接触开关到完善的视觉、触觉处理系统。内部信息传感器是指机器人驱动系统中的反馈控制信号检测元件。

二、控制方式

工业机器人的工作过程亦如图 1.5 所示。在使用机器人以前，操作人员通过示教装置把作业内容编成程序，输入到记忆装置。在从外部给出启动命令后，机器人从记忆装置中读出信息，并送到控制装置，发出控制信号，由驱动机构控制机械手，(在一定的精度范围之内）按照记忆装置中的内容完成给定的动作。如果再次启动，机器人重复上述作业。

由此看出，工业机器人的控制方式与传统的自动机械的最大区别在于采用了"示教-再现"方式，因而表现出通用、柔软、灵活等特点。在这种控制方式下，工业机器人的控制过程分为四步：示教—存储—再现—操作。

1. 控制信息

机器人控制中必须有三种信息。

顺序信息，即机器人各种单元动作的先后次序，其中包括机器人对外围设备(如传送带、焊接机等)作业条件的检测、设定等步骤。

位置信息，即机器人应到达作业空间各点的坐标值，其中包括手爪在到达点上的姿态。

时间信息，即机器人各顺序步所用的时间，这也可表示为机器人完成各动作的速度。

控制过程中的示教、存储和再现都围绕这三种信息进行。

2. 示教、存储和再现

示教是为了使机器人按照人的要求进行操作，由人把控制信息分离地或集中地输入到机器人中的过程。示教方法有两种，一种是直接示教方式，即操作人员直接带动机器人的手臂依次通过预定的轨迹，这时，顺序、位置和时间三种信息可以做到综合示教。另一种是间接示教方式，即操作人员通过操作手动控制盒上的按键，编制机器人的动作顺序，确定位置、设定速度或限时。这种方式中三种信息的示教一般是分离进行的。在计算机控制的情况下，用特定的语言编制示教程序，实际上也是一种间接示教方式，其中位置信息往往仍需通过手动控制盒设定。

存储是指在必要的期限内保存示教的信息。存储容量的大小决定机器人完成作业的复杂程度。存储方式也分为分离存储与集中存储两种。集中存储控制信息适于存储量大的复杂作业。分离存储方式可将三种控制信息单独存放在不同的装置里，它要求分离示教，这使示教复杂，但便于再现时灵活组合控制信息。

再现是指根据需要读出存储信息，向执行机构发出具体指令。相应于集中存储方式，再现时只是"原样照搬"，而对于分离的存储方式，再现时就可根据外部传感器的输出信号或通过人工干预改变动作顺序，这使机器人对于工作环境的变化具有一定的适应性。

操作是指根据再现时所发出的一条条指令，驱使机器人的各个自由度产生相应的动作，最终使机器人手爪从空间一点移动到另一点。这里，存在着如何实现机器人手爪的空间位置的控制问题。

3. 位置控制方式

工业机器人一般采用如下两种位置控制方式。

(1) 点位控制——PTP (Point To Point)方式。这种方式只关心起始点和目标点的位置，而不考虑两点间的移动路径。于是，机器人的运动轨迹就取决于各个自由度在移动到目标点的过程中所采取的动作形态。PTP 控制比较简单，适用于上下料、点焊、搬

运等作业.

(2) 连续路径控制——CP (Continuous Path) 方式. 这种方式不仅要求机器人以一定的精度到达目标点，而且对移动轨迹型式有一定精度范围的要求. 例如机器人进行喷漆、连续电弧焊作业就是这种情况. CP 方式要求示教的各位置点必须是连续的，因而也比较麻烦，示教效率低，目前大多数工业机器人实际上采用 PTP 示教(对于直线轨迹为两点指定，对于圆弧或圆，则为三点或四点指定). 而点间的轨迹则由微处理器按轨迹型式作插补运算(直线插补或圆弧插补)，再发出相应的再现指令.

三、PUMA 机器人

PUMA (Programmable Universal Manipulator for Assembly) 系列机器人是一类先进的计算机控制工业机器人. 它是美国 Unimation 公司的最新产品，后转产日本、联邦德国等国. 适用于传送、焊接、装配、堆放等多种作业，畅销于世界. 图 1.6 示出的为 PUMA 560 型机器人.

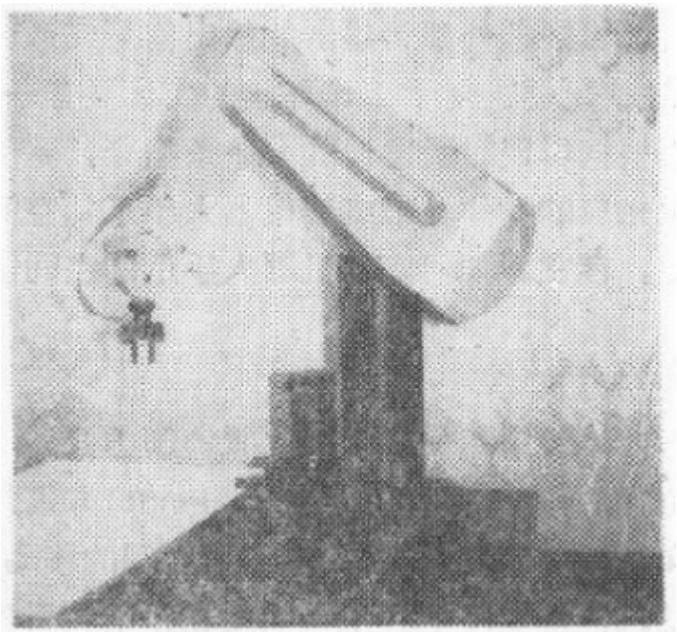


图 1.6 PUMA 560 机器人

以日本川崎重工业公司生产的 PUMA 560 型机器人为例，机器人的系统构成如图 1.7 所示。

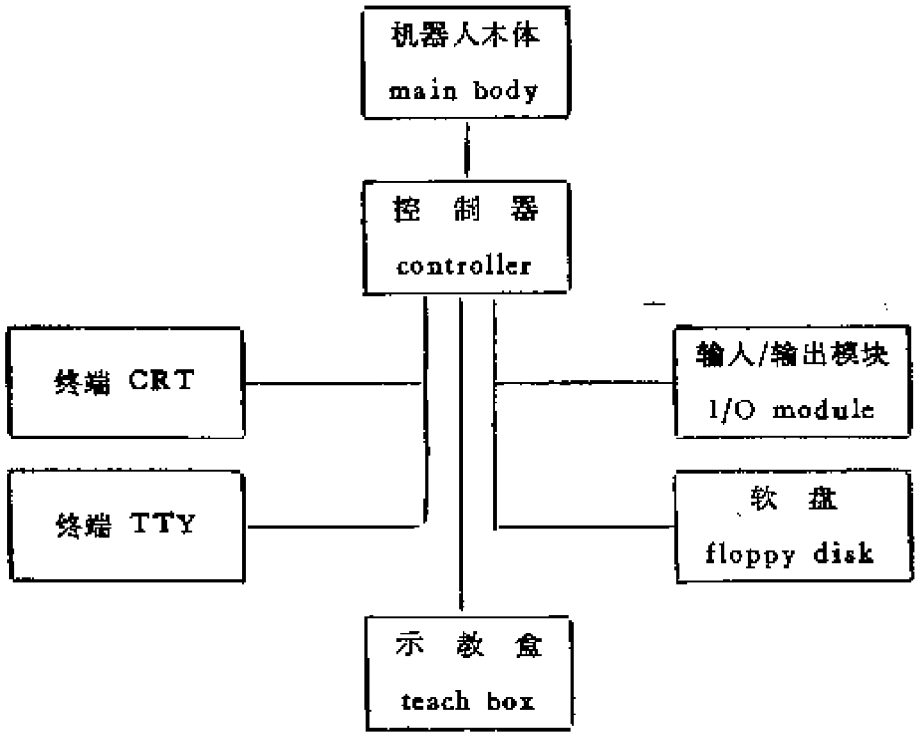


图 1.7 PUMA 机器人系统构成

机器人本体是一台六自由度关节型电动机械手臂。各关节旋转范围分别为：JT1（“腰”），320°；JT2（“肩”），250°；JT3（“肘”），270°；JT4（“腕”1），280°；JT5（“腕”2），200°；JT6（“腕”3），520°。作业空间达 0.8m²。负重在动态时为 2.5kg，静态时为 6kg。最大速度 0.5m/s。重复精度为 ±0.1mm。机械手腕部可配置两指（无关节）气动开闭手爪或其它选件。

PUMA 系列机器人的控制器由以集成电路 LSI-11 为 CPU 的专用微处理机系统以及各关节的伺服控制电路组成，其中 RAM 容量为 8kW。PUMA 机器人的最大特点是配有一个兼有控制运算与编程语言功能的 VAL 软件系统，可用以进行离线或联机编程，对机械手进行实时控制。

PUMA 机器人的示教控制方式有直接与间接两种方法。使机械臂处于“Free”状态后，各关节自由松驰，操作人员可一边带动机械手按需要的轨迹移动，一边按动示教盒上的“Record”钮，VAL 控制系统自动记录按动按钮时机器人手腕的坐标位置，同时，自动编辑运动过程的程序。操作人员也可运用 VAL 系统的编辑功能间接编制运行程序，经 CRT 终端键入计算机，随即运行，或存入软盘，需要时输入。但程序中的位置变量一般则是另外通过手动控制机械手实际地设定赋值的，示教盒上有相应的按钮能操纵机器人相对于“Joint”坐标系、“World”坐标系或“Tool”坐标系运动。

VAL 系统对于位置控制采用两种轨迹插补方式。关节插补运动（Joint-interpolated motion）：根据当前位置点与下一位置点间坐标差值，按照一定的算法，给每个关节分配应变化的角度。各关节同时转动，以不同速度同时到达。机器人手爪端部轨迹为一复杂的曲线。直线插补运动（Straight-line motion）：实际上是将两位置点间的直线轨迹分成很小的间隔，然后再对各个关节的角度变量进行复杂的变换运算。其中必须保证各插补点附近轨迹要平滑，即速度和加速度的连续性。以上两种轨迹运动的控制方式基本上就是前面所说的 PTP 和 CP 方式。对于复杂曲线的连续轨迹要求，可以设定足够多的关键点，再运用直线插补运动得到实现。对于焊接等作业，VAL 系统也有专门的指令调用生成圆弧轨迹的子程序。

PUMA 机器人配备的输入/输出模块（I/O Module）运用八路双向线路与外围设备，如传送带、外部传感装置等交换逻辑信号，实现连锁控制（VAL 系统有专门的指令管理输入输出信号）。目前，美国 Unimation 公司已研制了 UNIVISION I、II 两型机器人视觉系统，它们可与 PUMA 机器人配套运行。

§1.4 操纵型机器人

操纵型机器人是一类由人操纵进行工作的机器人。

操纵型机器人的工作系统实际上是一种人机系统。操纵人员处于联机控制回路之中。一方面，操纵人员在工作中不停地向机器人发送操作指令，在智能和适应能力方面辅助机器人完成复杂的作业；另一方面，机器人把操作对象和作业环境的状态直接地或间接地(通过监视装置)反馈给操纵人员，作为操纵人员控制机器人行为的根据。这样，人与机器人之间相互传递信息的问题就成为操纵型机器人的研究重点。

操纵型机器人的控制方式实际上是用"操纵"代替了工业机器人的"示教"方式。简单的操纵型机器人的动作可以看作是处于示教阶段的直接示教工业机器人的动作。复杂的操纵型机器人具有适应控制方式，即操作人员只给予"宏指令"，并不指示机器人的动作细节，机器人能根据本身的认识、学习机能自动适应作业情况。这种操纵型机器人接近于智能机器人。

操纵型机器人既具备机器人的一般结构和性能特点，而又不能离开人的操纵，这完全是因为它具有特定的应用需要。

操纵型机器人大体分为以下两种类型。

1. 能力扩大式机器人

这种机器人用以扩大人的体力和活动范围，或弥补人的肢体功能，例如装着式机器人和各种假肢(义肢)等。

装着式机器人也称"体外骨骼 (Exoskeleton)"。这种装置往往有几十个关节和相应的电动或液动机构。它们"披挂"或装定在人体身上，数倍或数十倍地"放大"各个部位的动作力量，代替人从事重体力工作，执行机构同时配有力传感器，使操纵者感觉到操作对象的反作用力，调整控制作用。

为伤残者研制的各种动力假手是典型的操纵型机器人．假手结构是一种关节式机械手臂，自备驱动源，以便按照需要适当增加自由度．大脑皮质运动中枢产生的兴奋脉冲传到截肢端部的肌肉，假手的控制信号就来源于肌肉作机械收缩的"应变"信号，或者肌肉在兴奋脉冲到达时产生的"肌电位"信号．假手装着者可以用眼睛监视调整假手的动作，而假手也可以配备人工触觉装置，向皮肤感觉系统反馈动作过程中受到的刺激．

机器人的手是为了代替人手的工作而研制的．因此，假手和机器人，两者的最终目标是一致的．而且，假手既然要"以假乱真"，它的运动轨迹、动作姿态就需要自然、美观、协调．因此，它不但要用到一般机器人的基本技术，而控制性能方面还有着许多独特的研究课题．多年来，动力假肢在福利事业的促进下已形成了一个专门的研究方向．

2．遥控机器人

这种机器人一般用于特殊的作业环境，例如：放射性物质、真空、有毒气体等隔离工作情况；造船、铁塔、建筑等危险工作条件；特别是宇宙、海洋开发用的探查工作环境．

苏联1970年向月球发送了"月球探测器1号"机器人，进行土壤分析、摄影、观测等任务，如果碰到石头，可自行拒绝地面人员的挖土命令．美国由"阿波罗12号"向月球发送了"探测者3号"机器人，它在空中实验室操作人员的控制下伸出约1.5m的机械手，采集月球岩土样品，在实验室中进行化验，把结果发回地球．

用于海洋开发的遥控机器人实际上是安上了机械手、足、眼等器官的深水作业机器．美国建造的"可控水下回收装置" Curv 和"海洋机器人" Mobot，装有作为视觉用的声纳和摄象机，作为听觉的水听器，检测方向用的陀螺罗盘等．它的机械手通过电缆按照岸上观测站的指令进行动作，曾在西班牙洋面回收了掉入海底的核弹头．这是一种"无人有缆"的机器人．另外，还有"有人无

缆”的情况，操作人员在机器人——深海调查船内操纵多个机械手从事搬运物体和照像摄影等作业.

遥控机器人，特别是远距离操纵的机器人，由于作业环境的复杂，以及与人的通讯联系方面的困难，正在由简单遥控式、监控式，向智能式的方向发展.

§1.5 智能机器人

智能机器人是这样一类机器人：机器人本身能认识工作环境、工作对象及其状态，它根据人给予的指令和“自身”认识外界的结果来独立地决定工作方法，利用操作机构和移动机构实现任务目标，并能适应工作环境的变化.

智能机器人即所谓的“第三代机器人”，它与工业机器人是两种可以同时并存的自动机械，但它的研究目标在于从工程上模拟人(或其它生物体)的复杂动作及其相应的智能行为，并获得综合的机器实现. 因此，智能机器人是工业机器人从无智能发展到有智能、从低智能水平发展到高度智能化的产物. 它更接近于人们事先对于“机器人”的理想要求.

智能机器人应该具备四种机能：运动机能——施加于外部环境的相当于人的手、脚的动作机能；感知机能——获取外部环境信息以便进行自我行动监视的机能；思维机能——求解问题的认识、推理、判断机能；人-机通讯机能——理解指示命令、输出内部状态，与人进行信息交换的机能.

由此可见，智能机器人的“智能”特征就在于它具有与外部世界——对象、环境和人相协调的工作机能.

从控制方式看，智能机器人不同于工业机器人的“示教-再现”以及操纵机器人的“操纵”，而是一种“认知-适应”的方式.

一、智能机器人的硬件系统

根据研制目的不同，智能机器人的系统构成不尽一致，比较完整的典型结构可如图 1.8 所示。

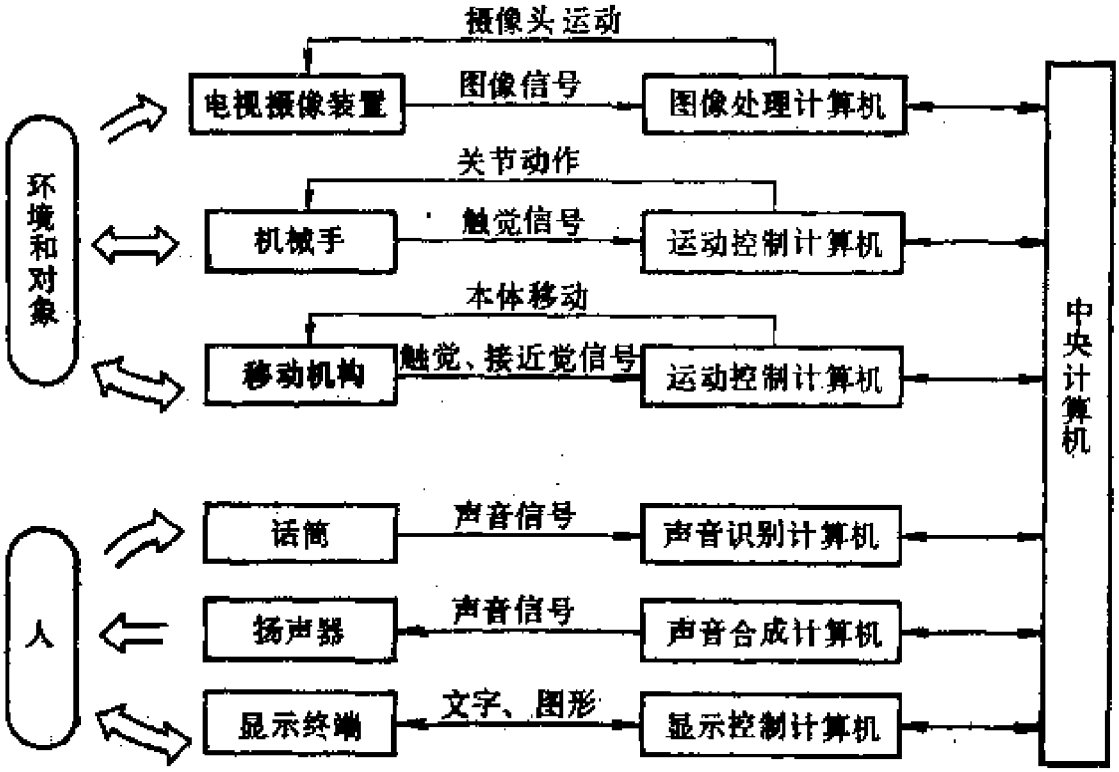


图 1.8　智能机器人的硬件系统

由图可见，智能机器人的系统综合运用了多种智能模拟技术，其目标是建立起一个“人”的模型。

眼——电视摄像装置和图像处理机。“看”和“动”相互关连，是人使用最频繁的基本动作。机器人的视觉是它最主要的感知手段。视觉装置可获取目标物的明暗、距离和颜色三种信息，据以识别它们的形状、姿态、位置、色别等特征参数。

手——多关节机械手、多关节机械手指及其控制系统。与工业机器人相比，智能机器人的“手”一是需要增加自由度，二是需要

配备接触觉、压觉、滑觉、力觉等感觉以便产生柔软、灵活、可靠的动作，完成复杂作业．而且触觉信息本身就可用来配合或代替视觉识别物体．

脚——车轮、连杆式、履带式或爬行式的机构及其控制系统．本体可以自由地"摸索"移动，这是智能机器人与传统的自动机械的一大区别，这可使它在人难以达到的地方完成作业，还可使它获得第一性的环境认识信息．

耳和口——话筒和扬声器以及语音识别和合成系统．两者用于人-机之间的听觉通讯，能以自然语言与人会话，将使人机联系大为通畅，这是机器人的智能水平极高的表现．显示终端或者用于文字、图形的视觉通讯，或者用于机器人专用计算机语言的运行．智能机器人专用语言的高级形式接近于人的自然语言，要求具有沟通人与机器人的思维方式的功能，这是目前研究价值最实际的通讯手段．

脑——中央计算机．整个智能机器人的结构是一个多级计算机系统．中央计算机担负着运动、感知、思维、人-机通讯这四种机能所涉及到的信息处理和管理控制任务．它必须：具有大容量内存，以便建立包括环境模型，对象数据、推理机制等内容的知识库；具有并行实时处理能力，以便改善下属各子系统之间存在的时间不平衡性，求得协调行为的高速实现．即使如此，理想的智能机器人对于用现有的顺序型冯·诺曼计算机作为"脑"仍感不足，而是寄希望于以非顺序型理论为基础的第五代计算机．其原因就在于智能计算机需要一套"第五代"计算机可配备的人工智能软件系统．

二、智能机器人的软件系统

智能机器人的软件系统（图 1.9）实际上就是人工智能主要技术对于机器人的综合运用．

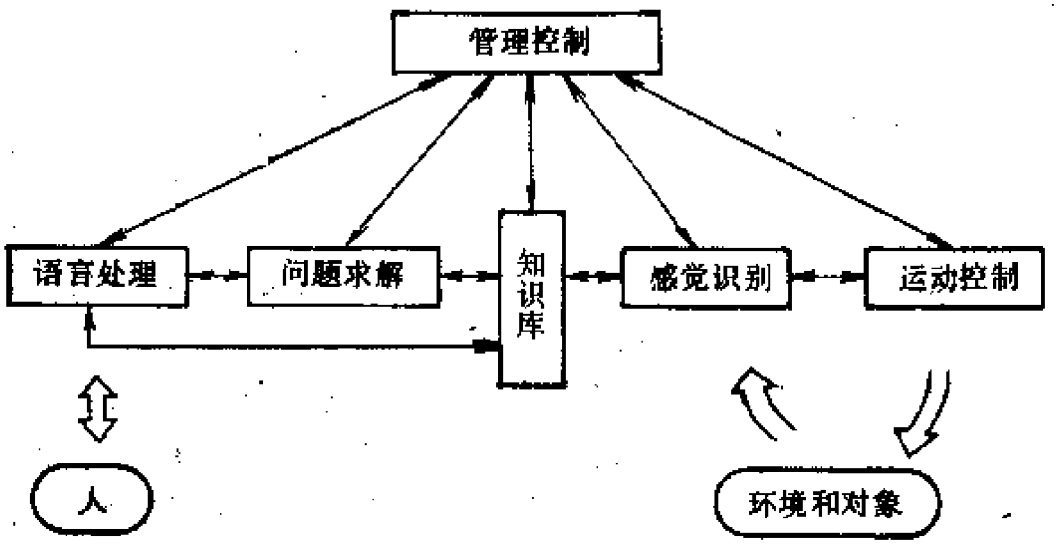


图 1.9　智能机器人的软件系统

机器人问题求解称之为机器人规划，即要求机器人自动寻求控制动作的某种有序组合，把初始的作业状态转变成满足一定条件的目标状态，所求得的动作序列叫做规划的解。例如把分散放置的零件组装成某个部件，穿行障碍物到达某个目的地都属于规划问题。监督、调整已知规划的实际执行过程也是机器人规划系统应有的功能。

自然语音与书面文字都是自然语言处理的对象。语音的识别与合成固然要涉及到信息处理的各种方法，但语言处理的关键是对语句(声音或文字)的语义理解；这需要先对语句的结构进行分析，然后抽取其中的"意义"信息并加以表示，最后做到能对语句作出解释，回答有关的问题，或者把它们翻译成其它语言。

视觉与触觉的识别手段不同，但共同的问题也都在于"理解"，机器人的感觉装置只是获取、处理景物的各种特征信息，(例如，物体的边缘、形状等)，而最终目标是要建立起景物的"模型"，即把感受数据压缩为一种容易处理的、明确而有意义的描述(例如，景物中的物体是什么，处于什么状态，相互关系如何等)。

知识库是机器人软件系统的核心。人工智能问题中所谓知识，通常是指描述各种客观环境、对象、条件的，组织成一定结构的“数据”，以及解释、运用这些“数据”的，反映有关领域客观规律或主观判断过程的推理机制。于是，机器人规划问题中对于状态和控制动作的表示，以及寻求规划的解的搜索与推理过程，自然语言理解问题中建立句法、语义“词典”及其检索、推理的过程，感觉识别问题中描述预期的景物模型及其与假设模型的匹配判断过程等等，都要依赖于知识库的功能，牵涉到知识获取、知识表示以及如何使用知识的方法。

机器人的运动控制问题的特点在于机器人具有复杂的机械结构。例如，多关节的机械手需要建立多个坐标系统，进行繁杂的坐标变换运算，而处于动态过程中的各个关节还存在着非线性参数、相互耦合等问题，这需要采用近似而有效的轨迹控制方法。对于步行式的机器人移动机构，则需要解决运动稳定性问题，手、脚协调控制问题。

总之，软件系统是机器人的“生命”，体现了机器人的“智能”，上述问题求解、自然语言处理和感觉识别等技术，都是人工智能学科中的主要课题。随着研究的进步，智能机器人软件系统必将逐步完善、逐步实用。

三、智能机器人的研制实例

由于技术上的困难或出于不同的研究目的，智能机器人的实际研制工作一般都是局部地进行的，或者是“眼-车”系统，或者是“手-眼”系统，或者是带有多种感觉的机械手。

美国斯坦福研究所（Stanford Reserch Institute，缩写为 SRI）60 年代研制的机器人 Shakey 是一种典型的“眼-车”系统（图 1.10）。

Shakey 的感觉器官主要是“眼”——安装在可动头部的电视

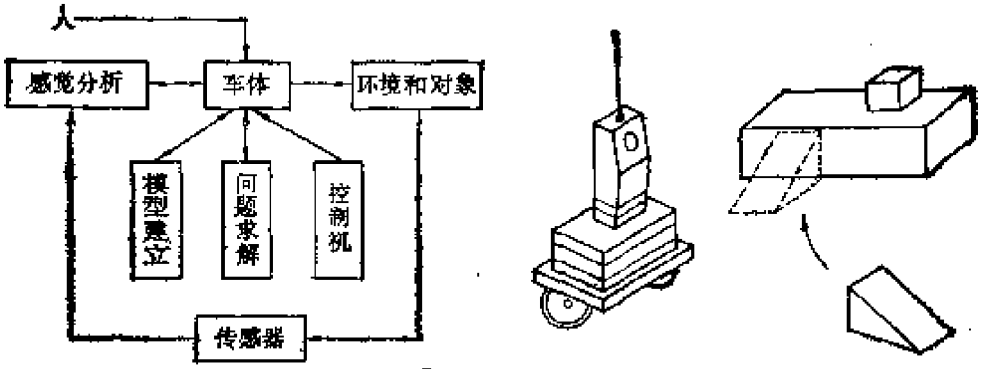


图 1.10 "眼-车"系统 (Shakey)

摄像机和光学测距器，以及车体下部的用来确认障碍物的触觉器"猫胡子"；移动机构是两个独立驱动的车轮；操作手段是靠车体推动物体。 Shakey 由体外的 SDS 940 中型计算机通过电缆或电波进行控制、通讯，体内只备控制电路。Shakey 的环境是形状简单的、有门相通的若干房间，操作对象是不同大小的长方体箱子和楔形箱子。

Shakey 的软件功能主要是视野范围内的对象识别，依靠积累经验求解行动规划，以及运用逻辑推理的问-答能力。

Shakey 可以穿行房间，搜索、识别指定的对象，并进行"智能"的操作。当人命令它把放在大箱子上的小箱子推下来时，它在分析了对象的高度位置后，首先寻找楔形箱子，推动它靠拢长方箱子，然后沿此斜坡登上木箱，把小箱子顺斜坡推下(图 1.11)。

Shakey 虽然属于智能机器人的初期研究成果，但作为一种实验对象，它开创了人工智能技术应用于机器人的局面。其中的规划程序 STRIPS 把定理证明的方法与问题求解方法相结合，形成了机器人规划问题的研究基础。

"眼-手"系统的典型例子是日本日立中央研究所 1971 年研究的装配用智能机器人(图 1.11)，它采用两个摄像管作为眼睛，一个看图纸，一个看零件，多关节的手臂带有触觉，成功地进行了印

刷板检查，电动机、晶体管的装配工作。

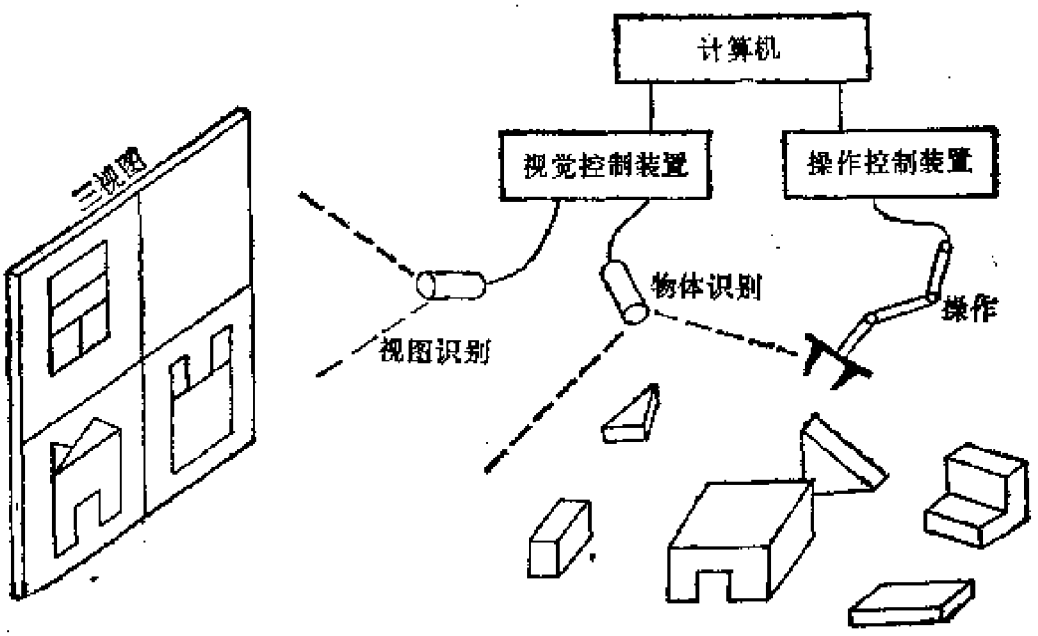


图 1.11 “眼-手”系统

§ 1.6 发展机器人技术的意义

机器人技术从诞生到现在，虽然只有短短 30 年的历史，但是它却显示了旺盛的生命力。近年来，世界上对于发展机器人的呼声更是有增无减，发达国家竞相争先，发展中国家急起直追，其中的根本原因在于机器人技术所具有的重要的应用价值和研究意义，以及人们对于机器人技术的深刻认识。

一、机器人技术极大地提高生产效益

由于机器人是一种省人化、合理化的自动化设备，它能大幅度地降低生产费用，提高生产效率，特别是它基本上能够独立作业，而不依赖于人的干预，可以免受疲劳、情绪或环境条件的影响，使

产品质量稳定、成品率提高。

例如，美国一家工厂使用机器人为歼击机加工部件，在金属板上钻250种大小不同、精度很高的孔眼，每班可完成24至30个零件而无一次品。而由工人操作钻孔机只能完成6件，次品率为10%，工厂一举就节约9300美元。喷漆机器人只要经过熟练工人一个小时的示教，便可掌握3～8年以上实践的工人才能达到的技术质量。日本FANUC公司采用机器人与CNC机床组合成加工单元，与原有的数控机床相比，总产量提高180～300%，每个操作人员的生产效率可提高四倍。

二、机器人技术是新技术革命的重要标志

与以往的技术革命相比，当前世界范围的新技术革命的特征是信息化、知识化、分散化，在自动化生产方面，则是生产过程由刚性自动化发展为柔性自动化，"大规模、大批量"的生产方式正面临着"小批量、多品种"的市场需求的挑战。

机器人由于它本身的固有特点——通用性，适应了新技术革命的需要。机器人可以发挥计算机高速、大量处理信息的优势，使计算机程序技术转化为机器的柔性应变能力，对于同一产品的不同变形体，甚至不同的加工作业，机器人只须改变一下程序就能迅速胜任，从而顺应了技术和产品频繁更新的要求。目前一些国家纷纷发展的"柔性制造系统"（FMS），都是以机器人技术为中心设计运行的。

机器人技术本身也是新技术革命的重要内容。例如，为了开发机器人系统的机械特性，需要提供轻质、高强度，或者是有耐腐蚀、耐辐射等性能的材料，这将会给新的材料工艺提出课题；为了利用与机器人相关的制造技术，将需要进一步完善机械-电子等新型的综合技术；随着智能机器人的研制工作的深入，将会促进计算机系统技术与人工智能技术的发展，事实上，智能机器人与"第五

代”计算机——智能计算机已经成为一些发达国家之间科学技术竞争的一大焦点。

机器人的应用和机器人技术的发展，还可能在新技术革命中打开一些“禁区”，开拓一些原来无法占领的新领域。例如在海洋开发、宇宙开发、微电子、精密加工以及原子能研究等方面，就可望使用特殊的机器人去探索一些新的奥秘。

三、机器人是机器进化的必然产物

人类在改造自然的历史进程中，随着对材料、能源和信息这三者的认识和利用，不断创造出各种工具（机器），满足并推动生产力的发展。

漫长的农业社会中，人们借助于人力、畜力、风力、水力这些自然力，创造出纺车、织机、风车、水磨这些简单工具机，形成“人-工具机-劳动对象”这种生产系统。

进入工业社会，开发了化石燃料这一重要能源，形成蒸汽机时代，尔后又有电机的出现，生产系统变为“人-动力机-传动机-工作机-劳动对象”。同时机器进化到一个复杂的系统。

随着机械化程度的提高、机器运转速度的提高，以及电力、无线电技术的发展，人们面对远距离控制的要求，又体会到信息对于控制生产过程的作用。于是产生了各种控制机，直至发展到具有信息处理能力的电子计算机出现，生产系统变为“人-控制机-动力机-传动机-工作机-劳动对象”。

工业社会向信息社会发展，生产的自动化、应变性要求越来越高，原有机器系统就显得庞杂而不灵活，这时人们就仿造自身的机体和功能，把控制机、动力机、传动机、工作机综合集中成一体，创造了“集成化”的机器系统——机器人。从而引起了生产系统的巨大变革，成为“人-机器人-劳动对象”，或者“人-机器人-动力机-工作机-劳动对象”。

由此可见，机器人的出现完全是生产工具和机器不断进化的产物。从工业机器人到智能机器人，机器系统还将向更高级的水平发展。

第二章　机器人的力学分析

智能机器人最主要的操作机构是各种形态的机械手．在脱开各种智能感受装置和计算机之后，智能机器人的机械手与工业机器人的机械手本质上是一样的，都是一种模拟人臂功能的机械结构．本章的内容就是扼要地介绍这种机械结构的运动学、动力学问题，建立起机器人机械手空间运动的概念。

§ 2.1　机械手的形态和自由度

机械手的动作形态是由三种不同的单位动作——旋转、回转、伸缩组合而成的。

如图 2.1 所示，旋转或回转是指运动机构产生相对转动，两者的不同仅在于转动部件的轴线与转动轴线是否同轴，因而常常把

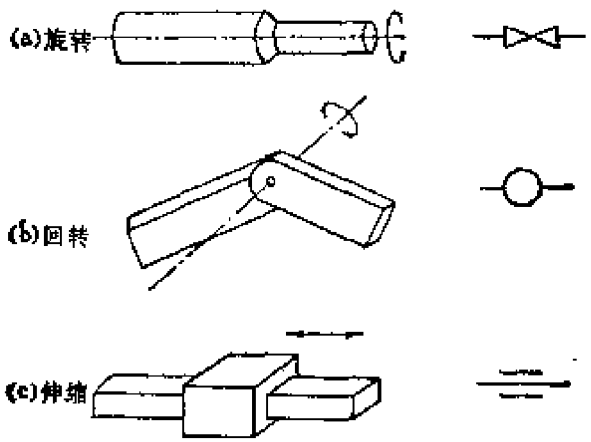


图 2.1　机械手的单位动作

它们笼统地称为转动。伸缩是指运动机构产生直线运动，这在人臂的动作中是不存在的，但机械手引入了伸缩动作，运动范围就可以得到扩大。

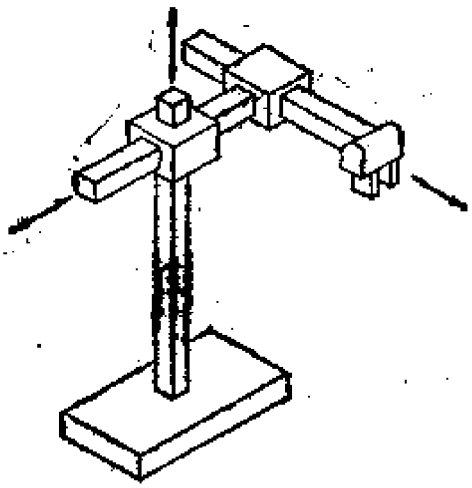


图 2.2 直角坐标型机械手

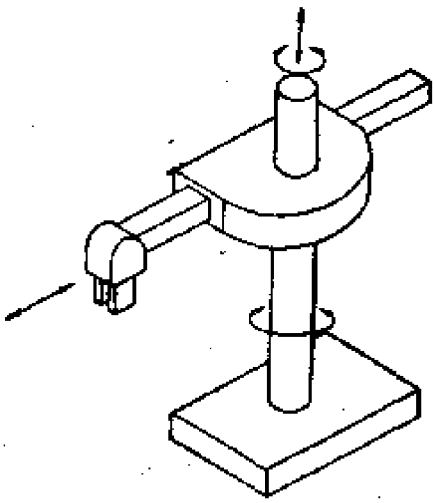


图 2.3 圆柱坐标型机械手

根据单位动作组合方式的不同，机械手的动作形态一般归纳为以下四种类型：(1) 直角坐标型(图 2.2)，(2) 圆柱坐标型(图 2.3)，(3) 极坐标型(图 2.4)，(4) 多关节型(图 2.5)。

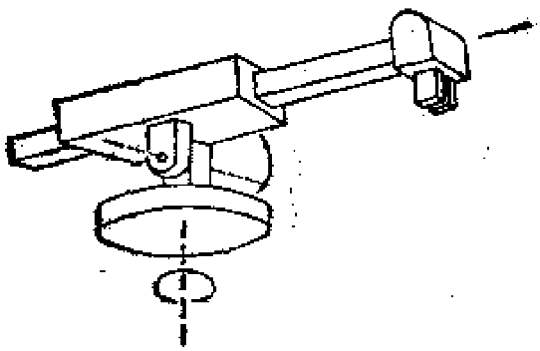


图 2.4 极坐标型机械手

直角坐标型机械手可以在三个相互正交的方向上作直线伸缩运动，机械手的手爪位于一个笛卡尔坐标系内。有的机器人还利用旋转关节控制手爪的姿态。这类机械手各个方向的运动是独立

的，计算比较方便，末端位置和精度也是一定的，但由于占地面积大，往往限于特定的应用场合。

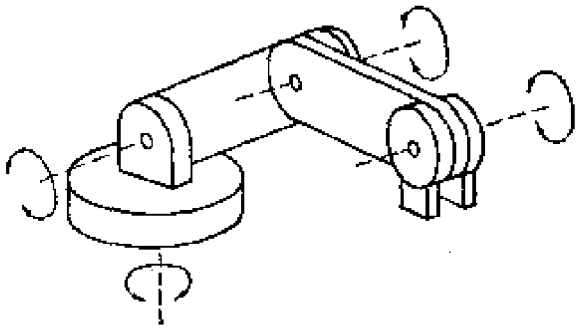


图 2.5　多关节型机械手

圆柱坐标型机械手有一个围绕基座轴的旋转运动和两个在相互正交的方向上的直线伸缩运动。它适宜于采用油压（或气压）驱动机构，在操作对象位于机器人四周的情况下，操作最为方便。

极坐标型机械手类似于一个坦克炮塔，它的动作形态包括围绕基座轴的旋转，一个回转和一个直线伸缩运动，其特点类似于圆柱型机械手。

多关节型机械手最接近于人臂的构造，它主要由多个回转或旋转关节所组成，一般都采用电机驱动机构。运用不同的关节连接方式，它可以完成各种复杂的操作运动，例如，抓取物体背面的东西，绕过某些障碍物等。由于具有占地面积小，动作范围大，空间移动速度快而灵活等特点，多关节型机械手广为各种智能机器人所采用。

机器人机械手具有的独立的单位动作组合数称之为自由度。

由于一个物体在三维空间中的位置和姿态分别需要三个独立的坐标才能得到确定。所以，能使手爪达到任意空间位置并具有任意姿态的机械手至少需要六个自由度。

自由度是表达机器人通用性、灵活性的主要指标。智能机器人往往需要更多的机动性（例如，当工作区存在障碍时，机械手手爪可以进入难以达到的地方），它的机械手可能具有六个以上的自由度。当然自由度愈多，机械结构愈复杂，从而控制也愈困难。

前述四种型式的机械手，是根据实际的机械结构来分类的，这

种分类便于我们直观地认识各种机械手．如果把机械手的各个部件抽象为一系列刚性连杆，它们通过一个个关节连接在一起，其中，能使两个相邻连杆作旋转(包括回转)运动的关节称之为旋转关节，能使两个相邻连杆产生直线位移的关节称之为柱关节，那末所有的机械手都可以看作是一种开链式多连杆机构．而机械手的自由度也就是它各个独立的关节位移变量的总数．本章将以上述这种机械手模型讨论它的各种力学问题．

§2.2 空间描述和坐标变换

一、刚体位置和姿态的描述

在机器人的操作过程中，无论是机械手的各个连杆，或是操作工具和被操作对象，都将在空间产生复杂的运动．如果把这些物体看成是刚体，那么我们首先必须描述一个刚体在空间的位置和姿态．这里的描述规定在直角坐标系内进行．

设在一个固定的参考坐标系 $\{A\}$ 中有一刚体 G，G 上任选一

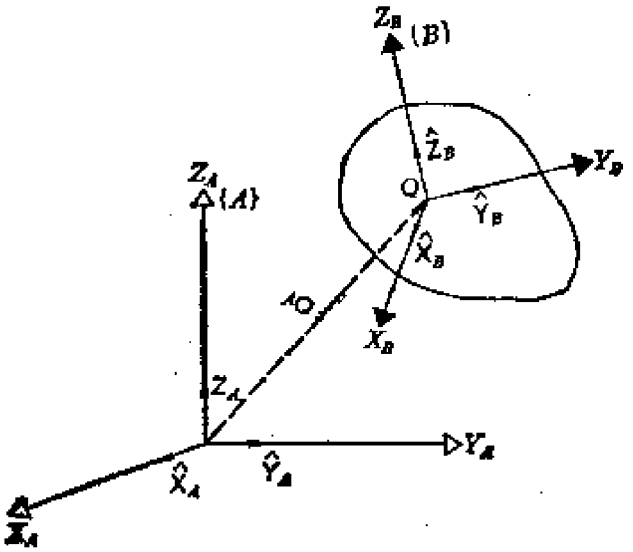


图 2.6 刚体位置和姿态的描述

点 Q，那么，G 在空间相对于 $\{A\}$ 的位置就可以由三维向量 ${}^{A}\boldsymbol{Q}$ 唯一地表示出来(图 2.6)。

$$ {}^{A}\boldsymbol{Q} = [Q_{X_A}\ Q_{Y_A}\ Q_{Z_A}]^T \tag{2-1} $$

式中上标 A 说明向量 ${}^{A}\boldsymbol{Q}$ 相对于 $\{A\}$ 而言，Q_{X_A}、Q_{Y_A}、Q_{Z_A} 则分别为 ${}^{A}\boldsymbol{Q}$ 在 $\{A\}$ 中的三个坐标值。

如果在 G 上建立另一个坐标系 $\{B\}$，为方便起见，$\{B\}$ 的原点就设为 Q，由于 $\{B\}$ 与 G 固联在一起，那么 $\{B\}$ 的三个坐标轴相对于 $\{A\}$ 的方向就完全确定了 G 在空间相对于 $\{A\}$ 的姿态。这可表示为一个 3 × 3 矩阵

$$ {}^{A}_{B}R = [{}^{A}\hat{X}_B\ {}^{A}\hat{Y}_B\ {}^{A}\hat{Z}_B] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2-2} $$

式中 R 的上标 A 和下标 B 说明 R 所表示的是 $\{B\}$ 相对于 $\{A\}$ 的关系，列向量 ${}^{A}\hat{X}_B$、${}^{A}\hat{Y}_B$、${}^{A}\hat{Z}_B$ 的分量分别为 $\{B\}$ 的三个坐标轴上单位向量 $\hat{X}_B$、$\hat{Y}_B$、$\hat{Z}_B$ 投影到 $\{A\}$ 中的方向余弦，即

$$ \begin{cases} \hat{X}_B = r_{11}\hat{X}_A + r_{21}\hat{Y}_A + r_{31}\hat{Z}_A \\ \hat{Y}_B = r_{12}\hat{X}_A + r_{22}\hat{Y}_A + r_{32}\hat{Z}_A \\ \hat{Z}_B = r_{13}\hat{X}_A + r_{33}\hat{Y}_A + r_{33}\hat{Z}_A \end{cases} \tag{2-3} $$

由上可知，刚体的空间描述完全等价于与之固联的坐标系相对于同一参考坐标系的描述，固联坐标系的原点位置和三个轴的方向分别表示了刚体的位置和姿态。这样，我们就可以置具体的刚体于不顾，而抽象地讨论不同坐标系之间的关系了。

二、坐标系的平移变换和旋转变换

设有两个坐标系 $\{A\}$ 和 $\{B\}$，它们的原点位置不同，而姿态完全相同，即 $\{A\}$、$\{B\}$ 的三个坐标轴对应平行(图 2.7)。在这种情况下，显然，$\{B\}$ 可以看作是先与 $\{A\}$ 重合，然后沿向量 ${}^{A}\boldsymbol{Q}$ 平

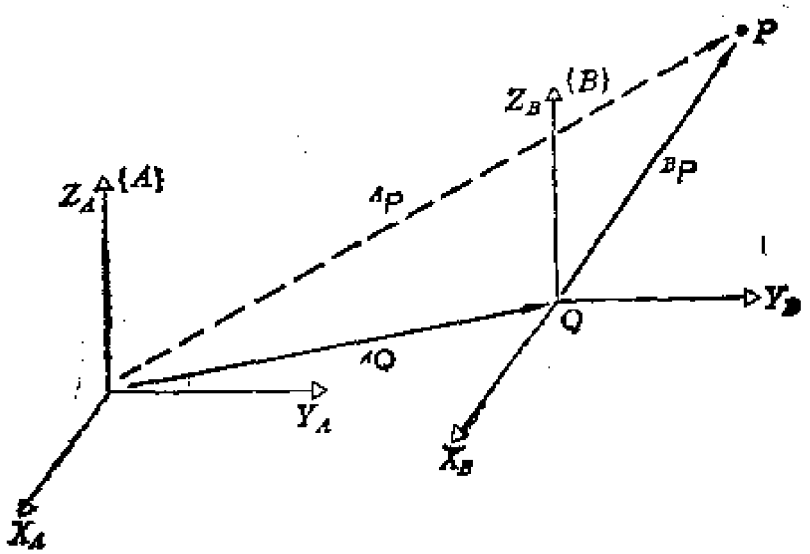


图 2.7　平移变换

移而得到的一个坐标系。

假定 P 为空间任意一点，它在 $\{A\}$、$\{B\}$ 中的表示分别为 ^AP、^BP，那么，由于 ^AQ 和 ^BP 是在具有相同姿态的坐标系中定义的，根据向量的加法，我们就有

$$^AP = {}^AQ + {}^BP \tag{2-4}$$

于是，向量 ^AQ 表征了两个坐标系的平移变换关系，依靠它，可以把一个坐标系中点向量的描述映射为另一个坐标系中的描述。我们称 ^AQ 为平移向量。

现在假设坐标系 $\{A\}$ 和 $\{B\}$ 的两个原点重合在一起，而姿态不同(图 2.8)。在这种情况下，$\{B\}$ 可以看作是先与 $\{A\}$ 重合，然后原点不动，按一定顺序绕某些轴在空间旋转得到的一个坐标系。

先讨论绕 $\{A\}$ 的某轴作一次旋转的情况。

设有一与 $\{A\}$ 重合的坐标系 $\{C\}$，如果 $\{C\}$ 绕 $\{A\}$ 的 X 轴旋转一个角度 α (图 2.9)，根据式 (2-2)，相对于 $\{A\}$ 描述 $\{C\}$ 的矩阵为

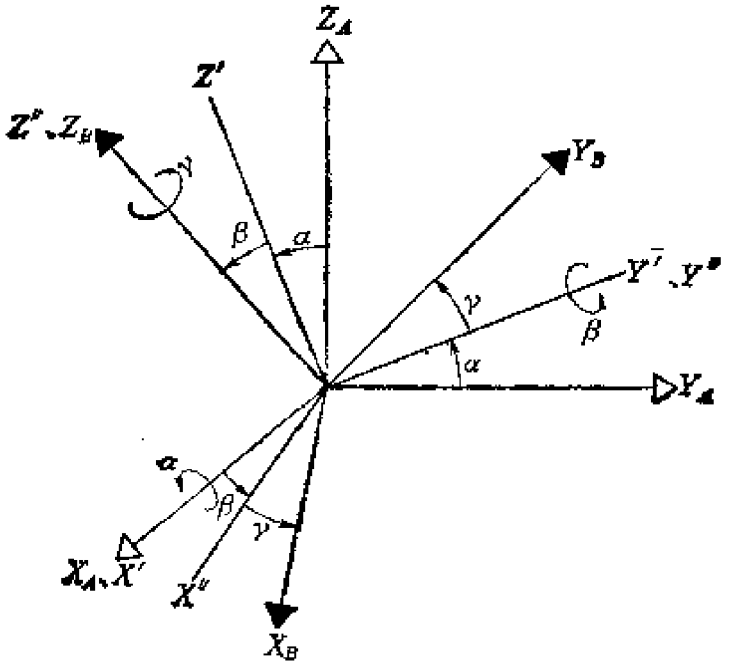


图 2.8　旋转变换

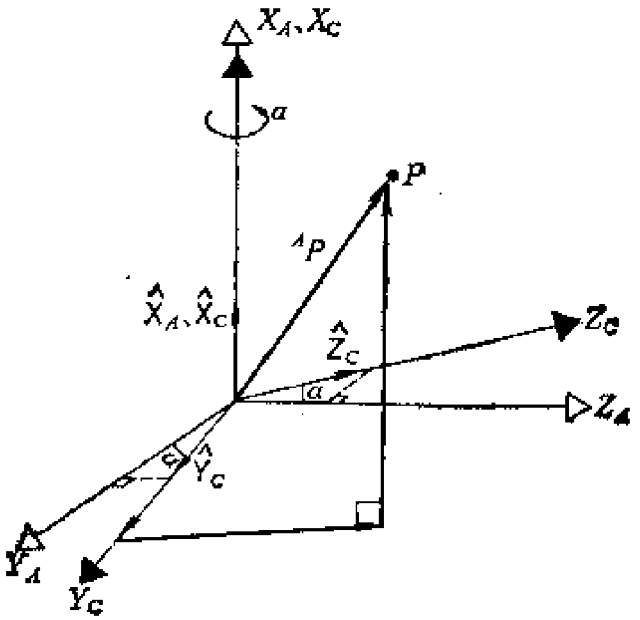


图 2.9　$\{C\}$ 绕 $\{A\}$ 的 X 轴旋转 α 角

$$^A_C R(X, \alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \qquad (2\text{-}5)$$

式中 ${}^{A}_{C}R(X,\alpha)$ 的表示注明了旋转轴和相应的旋转角度.

如果 $\{C\}$ 分别绕 $\{A\}$ 的 Y、Z 轴旋转角度 β、γ，类似地，我们有

$$ {}^{A}_{C}R(Y,\beta)=\begin{bmatrix}\cos\beta & 0 & \sin\beta\\ 0 & 1 & 0\\ -\sin\beta & 0 & \cos\beta\end{bmatrix} \tag{2-6} $$

$$ {}^{A}_{C}R(Z,\gamma)=\begin{bmatrix}\cos\gamma & -\sin\gamma & 0\\ \sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix} \tag{2-7} $$

现在再看这样一个坐标系 $\{B\}$：与 $\{A\}$ 重合的一个坐标系绕 X 轴旋转 α 得到 $\{B'\}$，$\{B'\}$ 再绕当前的 Y 轴旋转 β 得到 $\{B''\}$，最后 $\{B''\}$ 绕当前的 Z 轴旋转 γ 得到 $\{B\}$（图 2.8）.

对于空间任意一点 P，当得到 $\{B'\}$ 时，参照图 2.9，可以根据向量的加法得到

$$ \begin{aligned} {}^{A}P &= P_{X_{B'}}\hat{X}_{B'}+P_{Y_{B'}}\hat{Y}_{B'}+P_{Z_{B'}}\hat{Z}_{B'}\\ &= P_{X_{B'}}(1\cdot\hat{X}_{A}+0\cdot\hat{Y}_{A}+0\cdot\hat{Z}_{A})\\ &\quad +P_{Y_{B'}}(0\cdot\hat{X}_{A}+\cos\alpha\cdot\hat{Y}_{A}+\sin\alpha\cdot\hat{Z}_{A})\\ &\quad +P_{Z_{B'}}(0\cdot\hat{X}_{A}-\sin\alpha\cdot\hat{Y}_{A}+\cos\alpha\cdot\hat{Z}_{A}) \end{aligned} $$

写成矩阵形式

$$ {}^{A}P=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\alpha & -\sin\alpha\\ 0 & \sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}P_{X_{B'}}\\ P_{Y_{B'}}\\ P_{Z_{B'}}\end{bmatrix} $$

参照式 (2-5)，即有

$$ {}^{A}P={}^{A}_{B'}R(X,\alpha){}^{B'}P \tag{2-8} $$

类似地，当得到 $\{B''\}$，$\{B\}$ 时，可有

$$ {}^{B'}P={}^{B'}_{B''}R(Y,\beta){}^{B''}P \tag{2-9} $$

$$ {}^{B''}P={}^{B''}_{B}R(Z,\gamma){}^{B}P \tag{2-10} $$

将式 (2-10) 代入式 (2-9)，再代入式 (2-8)，就得到

$$^{A}\boldsymbol{P} = {}^{A}_{B'}R(X,\alpha)\,{}^{B'}_{B''}R(Y,\beta)\,{}^{B''}_{B}R(Z,\gamma)\,{}^{B}\boldsymbol{P} \qquad (2\text{-}11)$$

我们将式 (2-11) 右边三个矩阵的乘积记为 ${}^{A}_{B}R$，式(2-11) 变为

$$^{A}\boldsymbol{P} = {}^{A}_{B}R\,{}^{B}\boldsymbol{P} \qquad (2\text{-}12)$$

实际上，可以验证，${}^{A}_{B}R$ 的列向量就是 $\{B\}$ 的三个轴的单位向量在 $\{A\}$ 中的方向余弦，也就是说，${}^{A}_{B}R$ 是 $\{B\}$ 直接相对于 $\{A\}$ 的描述，这正好符合我们用 R 的上、下标 A、B 来说明 R 的意义。由于 ${}^{A}_{B}R$ 同时也表示了坐标系的旋转过程，我们称之为旋转矩阵，而象式 (2-5)、(2-6)、(2-7) 中的三个矩阵又称为基本旋转矩阵。

总之，旋转矩阵 ${}^{A}_{B}R$ 表征了两个坐标系的旋转变换关系，依靠它，可以把一个坐标系中点向量的描述映射为另一个坐标系中的描述。

值得注意的是，式 (2-12) 的推导表明，如果把 $\{B\}$ 看作是先与 $\{A\}$ 重合，然后逐次绕当前坐标系的某轴作旋转得到的坐标系，那么，不管每次旋转围绕当前坐标系的哪一个轴，只要从左到右乘上相应的基本旋转矩阵，都可以得到一个旋转矩阵 ${}^{A}_{B}R$。

另外，由于 ${}^{A}_{B}R$ 的列向量都具有单位长度，而且两两正交，说明 ${}^{A}_{B}R$ 为一正交矩阵，根据正交矩阵的性质和我们对旋转矩阵上、下标的约定，就有 ${}^{A}_{B}R^{-1} = {}^{A}_{B}R^{T} = {}^{B}_{A}R$，即，${}^{A}_{B}R$ 的逆表示了 $\{A\}$ 相对于 $\{B\}$ 的描述和旋转变换。

三、齐次变换

空间任意两个坐标系 $\{A\}$ 和 $\{B\}$ 之间的关系都可以看作是平移变换和旋转变换的合成结果，如图 2.10 所示。不难看出，$\{B\}$ 就是先与 $\{A\}$ 重合，然后沿向量 ${}^{A}\boldsymbol{Q}$ 平移得到与 $\{A\}$ 的姿态相同的中间坐标系 $\{B'\}$，再从 $\{B'\}$ 出发，经过若干次旋转得到的坐标系。

对于空间任意一点 P，已知 ${}^{B}\boldsymbol{P}$，我们可以利用向量的加法来求得 ${}^{A}\boldsymbol{P}$。由于只有姿态相同的坐标系中的向量才能相加，因此

量 ${}^{A}\boldsymbol{Q}$ 所确定的平移变换以及旋转矩阵 ${}^{A}_{B}R$ 所确定的旋转变换的合成坐标变换。对于仅有平移变换的情况，${}^{A}_{B}T$ 记为

$$ {}^{A}_{B}\mathrm{Trans}(Q_{X_A}\ Q_{Y_A}\ Q_{Z_A}) = \begin{bmatrix} 1 & 0 & 0 & Q_{X_A} \\ 0 & 1 & 0 & Q_{Y_A} \\ 0 & 0 & 1 & Q_{Z_A} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-17} $$

对于仅有基本旋转变换的情况，${}^{A}_{B}T$ 记为

$$ {}^{A}_{B}\mathrm{Rot}(X,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & 0 \\ 0 & s\theta & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-18} $$

$$ {}^{A}_{B}\mathrm{Rot}(Y,\theta) = \begin{bmatrix} c\theta & 0 & s\theta & 0 \\ 0 & 1 & 0 & 0 \\ -s\theta & 0 & c\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-19} $$

$$ {}^{A}_{B}\mathrm{Rot}(Z,\theta) = \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-20} $$

上面三式中，我们采用了 $c\theta$、$s\theta$ 表示 $\cos\theta$、$\sin\theta$ 的缩写记法。

（3）齐次变换的乘积运算类似于旋转矩阵的乘积运算，即若 ${}^{A}_{B}T$ 为 $\{B\}$ 相对于 $\{A\}$，${}^{B}_{C}T$ 为 $\{C\}$ 相对于 $\{B\}$ 的齐次变换，那么

$$ {}^{A}_{B}T\,{}^{B}_{C}T = \begin{bmatrix} {}^{A}_{B}R\ {}^{B}_{C}R & {}^{A}_{B}R\ {}^{B}\boldsymbol{Q}_C + {}^{A}\boldsymbol{Q}_B \\ 0\ 0\ 0 & 1 \end{bmatrix} = {}^{A}_{C}T \tag{2-21} $$

式 (2-21) 中 ${}^{A}_{B}R\,{}^{B}_{C}R = {}^{A}_{C}R$ 表示了 $\{C\}$ 相对于 $\{A\}$ 的姿态描述，${}^{B}\boldsymbol{Q}_C$ 为 $\{C\}$ 相对于 $\{B\}$、${}^{A}\boldsymbol{Q}_B$ 为 $\{B\}$ 相对于 $\{A\}$ 的平移向量，${}^{A}_{B}R\,{}^{B}\boldsymbol{Q}_C + {}^{A}\boldsymbol{Q}_B$ 则表示了 $\{C\}$ 相对于 $\{A\}$ 的平移向量。因而

${}^A_BT\,{}^B_CT = {}^A_CT$，它表示了 $\{C\}$ 相对于 $\{A\}$ 的描述和变换。

(4) 齐次变换 ${}^A_BT$ 的逆的一般形式为

$$ {}^B_AT^{-1} = \begin{bmatrix} {}^A_BR^T & -{}^A_BR^T\,{}^AQ \\ 0\ 0\ 0 & 1 \end{bmatrix} = {}^B_AT \qquad (2\text{-}22) $$

在 §2.2 中，我们已指出 ${}^A_BR^T = {}^B_AR$，于是式 (2-22) 中，${}^A_BR^T$ 表示了 $\{A\}$ 相对于 $\{B\}$ 的姿态描述，${}^AQ$ 为 $\{B\}$ 相对于 $\{A\}$ 的平移向量，$-{}^A_BR^T\,{}^AQ = -{}^B_AR\,{}^AQ$ 则为由 $\{B\}$ 原点指向 $\{A\}$ 原点的向量相对于 $\{B\}$ 的表示，即为 $\{A\}$ 相对于 $\{B\}$ 的平移向量。因而，${}^A_BT^{-1} = {}^B_AT$，它表示了 $\{A\}$ 相对于 $\{B\}$ 的相反描述和逆变换。

§2.3 机械手的运动学方程

我们在 §2.1 中指出，机械手可以看作是一个开链式多连杆机构。始端连杆即为机械手的基座，或称基杆，末端连杆则与机械手的手爪相连，相邻连杆之间用一个旋转关节或柱关节连结在一起。机械手的运动学要研究的有两个问题：运动学正问题——已知各关节位移变量的值，要求手爪在空间的位置和姿态，这实际上是建立运动学方程的过程；运动学逆问题——指定手爪的空间位置和姿态，要求出各关节位移变量的相应值，这实际上是求解运动学方程的过程。我们首先根据 §2.2 的讨论，建立机械手的运动学方程。

一、连杆的坐标系

机械手的任一连杆 $i\ (i = 0, 1, 2, \cdots, n)$ 可看作是一个刚体，它确定了两个相邻关节 J_i、J_{i+1} 的轴线 i 与轴线 $i+1$ 之间的关系。我们可以用下述四个参数描述连杆 i（参看图 2.11）。

(1) 连杆长度 a_i：轴线 i 与轴线 $i+1$ 之间公垂线的长度，若两轴线相交，$a_i = 0$；若两轴线平行，可对 a_i 的垂足有所选择。

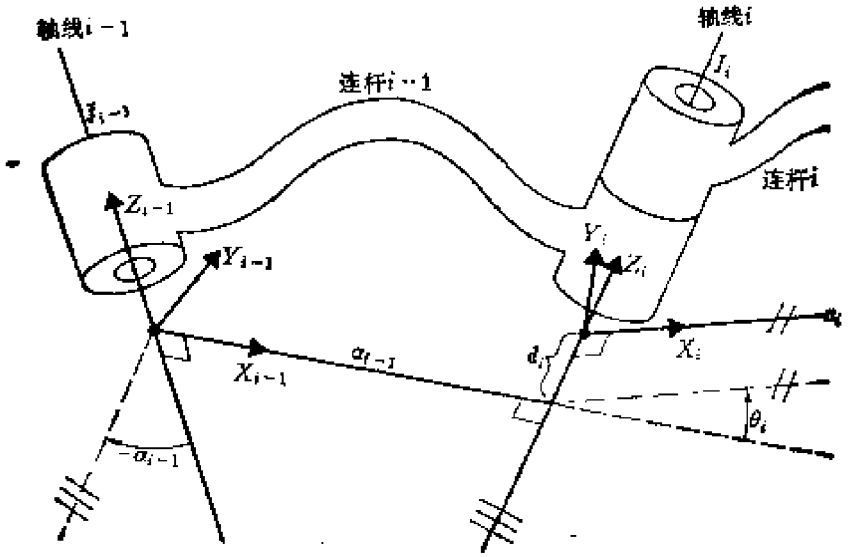


图 2.11　连杆参数和连杆坐标系

(2) 连杆扭角 α_i：以从轴线 i 指向轴线 $i+1$ 的 a_i 为法线，经过 a_i 在轴线 i 上的垂足作一平面，将轴线 i 按相对于 a_i 的右手规则转向轴线 $i+1$ 在该平面上的投影，这样所成的转角即为 α_i。若两轴线相交，可对 α_i 的正负号任意选择，若两轴线平行，$\alpha_i=0$。

(3) 连杆间距 d_i：从 a_{i-1} 在轴线 i 上的垂足沿轴线 i 指向 a_i 在轴线 i 上的垂足的长度。若 J_i 为柱关节，则 d_i 为关节位移变量，d_i 的零位置任选。若 J_1 为旋转关节，规定 $d_1=0$。

(4) 关节转角 θ_i：把 a_i 平移到与 a_{i-1} 的延长线相交的位置上，这两条公垂线所成的夹角。若 J_i 为旋转关节，则 θ_i 为关节位移变量，θ_i 的零位置任选。若 J_1 为柱关节，规定 $\theta_1=0$。

对于机械手的始端连杆 0（基杆）和末端连杆 n，我们自然地规定 $a_0=a_n=0$，$\alpha_0=\alpha_n=0$。

现在对连杆 i 定义一个与之相固联的坐标系 $\{i\}$，定义的规则如下(参看图 2.11)：$\{i\}$ 的原点设为 a_i 在轴线 i 上的垂足。$\{i\}$ 的 Z 轴 (Z_i) 与轴线 i 相重合。$\{i\}$ 的 X 轴 (X_i) 沿 a_i 的方向，即

从轴线 i 指向轴线 $i+1$，当 $a_i=0$ 时，X_i 选为 Z_i 与 Z_{i+1} 所成平面的法线，于是 X_i 的选择有两种可能性，这对于 α_i 的正负号有影响。$\{i\}$ 的 Y 轴 (Y_i) 按右手规则得到。

基杆坐标系 $\{0\}$ 的选择是任意的，我们规定 Z_0 沿轴线 1，这使得 $\theta_1=0$ 时，$\{0\}$ 和 $\{1\}$ 重合。

末端连杆坐标系的确定取决于 J_n，若 J_n 为旋转关节，$\{n\}$ 的原点选择要保证 $d_n=0$，而 X_n 的选择要使它在 $\theta_n=0$ 时与 X_{n-1} 重合；若 J_n 为柱关节，$\{n\}$ 的原点选择要使它在 $d_n=0$ 时位于 X_{n-1} 在轴线 n 上的垂足处，而 X_n 的选择要保证 $\theta_n=0$。

二、运动学方程

运用 §2.2 和 §2.3 的讨论结果，我们很容易确定连杆坐标系 $\{i\}$ 相对于 $\{i-1\}$ 的变换。

把图 2.11 简明地重画于图 2.12，$\{i\}$ 可以看作是先与 $\{i-1\}$ 重合，然后经过四次变换得到一个坐标系。设中间坐标系依次为 $\{u\}$、$\{v\}$、$\{w\}$，其变换顺序为：绕 X_{i-1} 旋转 α_{i-1} 得到 $\{u\}$，

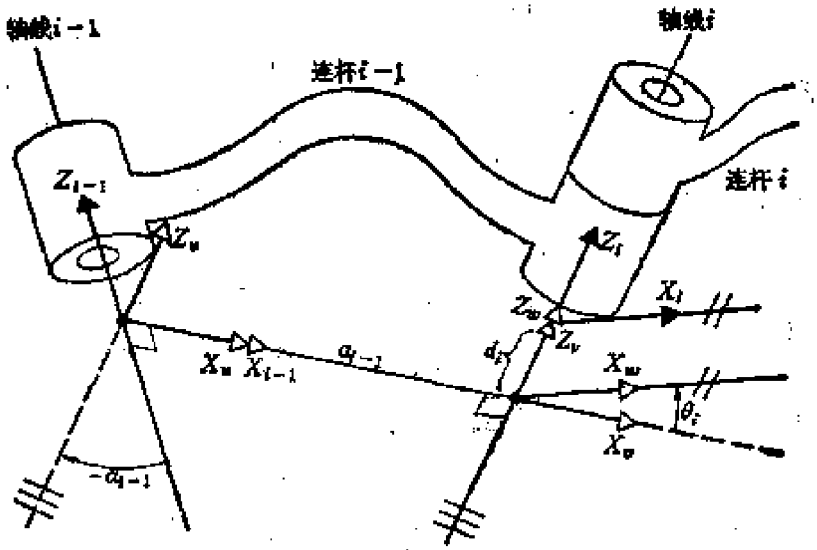


图 2.12 $\{i\}$ 相对于 $\{i-1\}$ 的变换

$\{u\}$ 沿 X_{i-1} 平移 a_{i-1} 得到 $\{v\}$，$\{v\}$ 绕 Z_i 旋转 θ_i 得到 $\{w\}$，$\{w\}$ 沿 Z_i 平移 d_i 得到 $\{i\}$。

于是，可以直接写出

$$
\begin{aligned}
{}^{i-1}_{i}T &= {}^{i-1}_{u}\mathrm{Rot}(X,\alpha_{i-1}){}^{u}_{v}\mathrm{Trans}(a_{i-1},0,0){}^{v}_{w}\mathrm{Rot}(Z,\theta_i)\\
&\quad\cdot{}^{w}_{i}\mathrm{Trans}(0,0,d_i)\\
&=\begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{2-23}
$$

如果已知连杆参数 a_{i-1}、α_{i-1}、θ_i（对于柱关节）或 d_i（对于旋转关节），那么，${}^{i-1}_{i}T$ 仅为一个变量的函数。

对于 n 个连杆的机械手，运动学方程是要确定与末端坐标系 $\{n\}$ 固联的手爪相对于基杆坐标系 $\{0\}$ 的变换，这只要把 n 个连杆变换矩阵逐次右乘就可得到

$$
{}^{0}_{n}T = {}^{0}_{1}T{}^{1}_{2}T{}^{2}_{3}T\cdots{}^{i-1}_{i}T\cdots{}^{n-1}_{n}T \tag{2-24}
$$

${}^{0}_{n}T$ 是 n 个关节位移变量的函数，如果得到了机械手关节位置传感器的读数，运动学正问题也就解决了。

例 2.1 PUMA 560 是一个全部为旋转关节的六自由度机器人，如图 2.13（所有关节角都处于零位置）所示，J_4、J_5、J_6 的轴线都相交，交点即为 $\{4\}$、$\{5\}$、$\{6\}$ 的原点。对应的连杆参数列于表 2.1。

表 2.1 PUMA 560 的连杆参数

J_i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	−90°	0	0	θ_2
3	0	a_2	d_3	θ_3
4	−90°	a_3	d_4	θ_4
5	90°	0	0	θ_5
6	−90°	0	0	θ_6

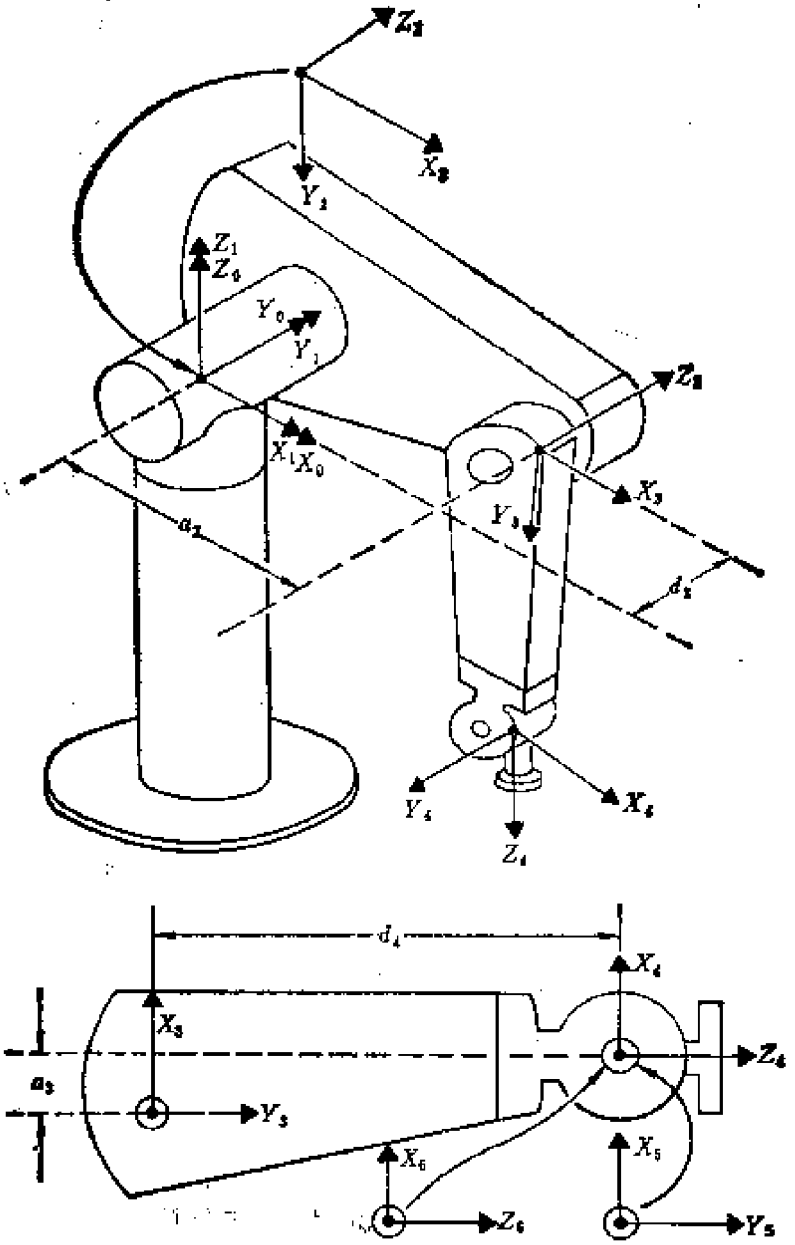


图 2.13 PUMA 560 的坐标系

利用式(2-23)可求出每个连杆变换如下：

$$
{}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_2 & -c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{2}_{3}T=\begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},\quad {}^{3}_{4}T=\begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -s\theta_4 & -c\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{4}_{5}T=\begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\quad {}^{5}_{6}T=\begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

(2-25)

以下我们采用进一步的简略符号进行计算，如 $c_3=c\theta_3$，$s_4=s\theta_4$ 等等。

$$
{}^{4}_{6}T={}^{4}_{5}T{}^{5}_{6}T=\begin{bmatrix} c_5c_6 & -s_5s_6 & -s_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-26}
$$

$$
{}^{3}_{6}T={}^{3}_{4}T{}^{4}_{6}T=\begin{bmatrix} c_4c_5c_6-s_4s_6 & -c_4c_5s_6-s_4c_6 & -c_4s_5 & a_3 \\ s_5c_6 & -s_5s_6 & c_5 & d_4 \\ -s_4c_5c_6-c_4s_6 & s_4c_5s_6-c_4c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

(2-27)

由于 J_2、J_3 的轴线总是平行的，先计算 ${}^{1}_{3}T{}^{2}_{3}T$，并运用和角公式 $c_{23}=c_2c_3-s_2s_3$、$s_{23}=c_2s_3+s_2c_3$，以使结果表达式简单一些。

$$
{}^{1}_{3}T={}^{1}_{2}T{}^{2}_{3}T=\begin{bmatrix} c_{23} & -s_{23} & 0 & a_2c_2 \\ 0 & 0 & 1 & 0 \\ -s_{23} & -c_{23} & 0 & -a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-28}
$$

$$ {}^{1}_{6}T = {}^{1}_{3}T\,{}^{3}_{6}T = \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} & q'_X \\ r'_{21} & r'_{22} & r'_{23} & q'_Y \\ r'_{31} & r'_{32} & r'_{33} & q'_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-29} $$

其中

$$ \left.\begin{aligned} r'_{11} &= c_{23}[c_4c_5c_6 - s_4s_6] - s_{23}s_5s_6 \\ r'_{21} &= -s_4c_5c_6 - c_4s_6 \\ r'_{31} &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \\ r'_{12} &= -c_{23}(c_4c_5c_6 + s_4c_6) + s_{23}s_5s_6 \\ r'_{22} &= s_4c_5s_6 - c_4c_6 \\ r'_{32} &= s_{23}(c_4c_5s_6 + s_4c_6) + c_{23}s_5s_6 \\ r'_{13} &= -c_{23}c_4c_5 - s_{23}c_5 \\ r'_{23} &= s_4s_5 \\ r'_{33} &= s_{23}c_4s_5 - c_{23}c_5 \\ q'_X &= a_2c_2 - a_3c_{23} - d_4s_2 \\ q'_Y &= d_3 \\ q'_Z &= -a_3s_{23} - a_2s_2 - d_4c_{23} \end{aligned}\right\} \tag{2-30} $$

最后求出

$$ {}^{0}_{6}T = {}^{0}_{1}T\,{}^{1}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & q_X \\ r_{21} & r_{22} & r_{23} & q_Y \\ r_{31} & r_{32} & r_{33} & q_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-31} $$

其中

$$ \left.\begin{aligned} r_{11} &= c_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] + s_1(s_4c_5c_6 + c_4s_6) \\ r_{21} &= s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] - c_1(s_4c_5c_6 + c_4s_6) \\ r_{31} &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \\ r_{12} &= c_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] \\ &\quad + s_1(c_4c_6 - s_4c_5c_6) \end{aligned}\right| $$

$$
\begin{aligned}
r_{22} &= s_1[c_{23}(-c_4c_5c_6 - s_4c_6) + s_{23}s_5s_6] \\
&\quad - c_1(c_4c_6 - s_4c_5c_6) \\
r_{32} &= -s_{23}(-c_4c_5c_6 - s_4c_6) + c_{23}s_5s_6 \\
r_{13} &= -c_1(c_{23}c_4c_5 + s_{23}c_5) - s_1s_4s_5 \\
r_{23} &= -s_1(c_{23}c_4c_5 + s_{23}c_5) + c_1s_4s_5 \\
r_{33} &= s_{23}c_4s_5 - c_{23}c_5 \\
q_X &= c_1(a_2c_2 + a_3c_{23} - d_4s_{23}) - d_3s_1 \\
q_Y &= s_1(a_2c_2 + a_3c_{23} - d_4s_{23}) + d_3c_1 \\
q_Z &= -a_2s_{23} - a_3s_2 - d_4c_{23}
\end{aligned}
\qquad (2\text{-}32)
$$

三、求解运动学方程

求解运动学方程也就是所谓机械手的运动学逆问题：已知 ${}^0_6T$，求出 $\theta_1, \theta_2, \cdots, \theta_n$ 的值。

只要 ${}^0_6T$ 表示的末端连杆坐标系的位置和姿态位于机械手的可达工作空间内，则运动学方程至少有一个解存在。但在可达空间内，有一部分是灵活工作空间，即机械手的末端不但可以达到，而且具有任意姿态，因此，运动学方程有可能出现重解，一般地说，不为零的连杆参数愈多，解的个数愈多。

机械手的运动学方程是一组非线性方程式，没有通用的解法。如果采用数值迭代程序，求解过程显然很慢，而且在重解的情况下，也不能保证求出所有的解。因此，一般都希望采用各种技巧，求出运动学方程的解的解析表达式，即封闭形式的解。已经证明，对于六自由度的机械手的运动学方程，只有在多个关节轴线正交或平行的情况下，才有可能求出封闭解。而实际的机械手结构一般都设计得比较简单，以便满足求得封闭解的特殊要求。

下面，我们介绍求取封闭解的两种方法，并举例说明其中的技巧。

1. 代数解法

例 2.2 PUMA 560 的运动学方程为

$$
{}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & q_X \\ r_{21} & r_{22} & r_{23} & q_Y \\ r_{31} & r_{32} & r_{33} & q_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
= {}^0_1T(\theta_1){}^1_2T(\theta_2){}^2_3T(\theta_3){}^3_4T(\theta_4){}^4_5T(\theta_5){}^5_6T(\theta_6) \qquad (2\text{-}33)
$$

在求解过程中，我们逐次在式 (2-33) 的两边同时左乘一个齐次变换的逆，达到分离变量的目的。关节转角的表达式中采用了双变量反正切函数 $\mathrm{Atan2}(y, x) = \tan^{-1}(y/x)$，这样可以通过 x、y 的正负号来确定转角的象限。

(1) 求 θ_1　对式 (2-33) 两边同时左乘 ${}^0_1T^{-1}$ 得到

$$
({}^0_1T(\theta_1))^{-1}\,{}^0_6T = {}^1_2T(\theta_2){}^2_3T(\theta_3){}^3_4T(\theta_4){}^4_5T(\theta_5){}^5_6T(\theta_6)
$$

即为

$$
\begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & q_X \\ r_{21} & r_{22} & r_{23} & q_Y \\ r_{31} & r_{32} & r_{33} & q_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^1_6T \qquad (2\text{-}34)
$$

在上式两边的矩阵中寻找简单的表达式或常数 [${}^1_6T$ 的元素见式 (2-30)]，使之对应相等，在此选择元素 (2, 4)，就有

$$
-s_1q_X + c_1q_Y = d_3 \qquad (2\text{-}35)
$$

解这类方程式，可作三角代换

$$
q_X = r\cos\phi,\quad q_Y = r\sin\phi \qquad (2\text{-}36)
$$

其中 $r = \sqrt{q_X^2 + q_Y^2}$，$\phi = \mathrm{Atan2}(q_Y, q_X)$，这时式 (2-35) 变为

$$
c_1s_\phi - s_1c_\phi = \frac{d_3}{r},\quad \text{即}\quad \sin(\phi - \theta_1) = \frac{d_3}{r}
$$

而

$$
\cos(\phi - \theta_1) = \pm\sqrt{1 - \frac{d_3^2}{r^2}}
$$

于是

$$\phi - \theta_1 = \text{Atan2}\left(\frac{d_3}{r},\ \pm\sqrt{1-\frac{d_3^2}{r^2}}\right)$$

$$\theta_1 = \text{Atan2}(q_Y, q_X) - \text{Atan2}\left(d_3,\ \pm\sqrt{1-\frac{d_3^2}{r^2}}\right) \quad (2\text{-}37)$$

(2) 求 θ_3　现在，式 (2-34) 左边已为定值.

类似地，我们继续通过式 (2-34) 寻找简单的方程. 令元素 (1, 4) 和 (3, 4) 对应相等，就有

$$\begin{cases} c_1 q_X + s_1 q_Y = a_3 c_{23} - d_4 s_{23} + a_2 c_2 \\ -q_Z = a_3 s_{23} + d_4 c_{23} + a_2 s_2 \end{cases}$$

将上面两式平方后相加，代入 θ_1 的已知值，可得

$$a_3 c_3 - d_4 s_3 = K \quad (2\text{-}38)$$

式中

$$K = \frac{q_X^2 + q_Y^2 + q_Z^2 - a_2^2 - a_3^2 - d_3^2 - d_4^2}{2a_2}$$

式 (2-38) 与式 (2-35) 形式相同，于是可采用类似的三角代换求得

$$\theta_3 = \text{Atan2}(a_3, d_4) - \text{Atan2}(K, \pm\sqrt{a_3^2 + d_4^2 - K^2}) \quad (2\text{-}39)$$

(3) 求 θ_2　对式 (2-33) 再次分离变量

$$({}^0_3T(\theta_2))^{-1}\,{}^0_6T = {}^3_4T(\theta_4){}^4_5T(\theta_5){}^5_6T(\theta_6)$$

即为

$$\begin{bmatrix} c_1c_{23} & s_1c_{23} & -s_{23} & -a_2c_3 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3 \\ -s_1 & c_1 & 0 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & q_X \\ r_{21} & r_{22} & r_{23} & q_Y \\ r_{31} & r_{32} & r_{33} & q_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^3_6T \quad (2\text{-}40)$$

令上式两边 (${}^3_6T$ 的元素见式 (2-27)) 的元素 (1,4) 和 (2,4) 对应相等，就有

$$\begin{cases} c_1c_{23}q_X + s_1c_{23}q_Y - s_{23}q_Z - a_2c_3 = a_3 \\ -c_1s_{23}q_X - s_1s_{23}q_Y - c_{23}q_Z + a_2s_3 = d_4 \end{cases} \tag{2-41}$$

对上式联立求解得到 s_{23} 和 c_{23} 的值，这样

$$\theta_{23} = \text{Atan2}[(-a_3 - a_2c_3)q_Z - (c_1q_X + s_1q_Y)(d_4 - a_2s_3),$$
$$(a_2s_3 - d_4)q_Z + (a_3 + a_2c_3)(c_1q_X + s_1q_Y)] \tag{2-42}$$

因而

$$\theta_2 = \theta_{23} - \theta_3 \tag{2-43}$$

(4) 求 θ_4　现在式 (2-40) 的左边已为定值，令它两边的元素 (1，3) 和 (3，3) 对应相等，就有

$$\begin{cases} r_{13}c_4c_{23} + r_{23}s_4c_{23} - r_{33}s_{23} = -c_4s_5 \\ -r_{13}s_4 + r_{23}c_4 = s_4s_5 \end{cases}$$

只要 $s_5 \neq 0$，我们可由上式求出

$$\theta_4 = \text{Atan2}(-r_{13}s_4 + r_{23}c_4, -r_{13}c_4c_{23} - r_{23}s_4c_{23} + r_{33}s_{23}) \tag{2-44}$$

当 $\theta_5 = 0$ 时，机械手的 J_4 和 J_6 在同一条轴线上，这时，只能求出 θ_4 与 θ_6 的和或差，如果选取 θ_4 的值，可求得 θ_6 的相应值.

(5) 求 θ_5　对式 (2-33) 再次分离变量

$$[{}^4_3T(\theta_4)]^{-1}\,{}^4_0T = {}^5_3T(\theta_5){}^6_5T(\theta_6) \tag{2-45}$$

式中

$$[{}^4_3T(\theta_4)]^{-1} =$$

$$\begin{bmatrix} c_1c_{23}c_4 + s_1s_4 & s_1c_{23}c_4 - c_1s_4 & -s_{23}c_4 & -a_2c_3c_4 + d_3s_4 - a_3c_4 \\ -c_1c_{23}s_4 + s_1c_4 & -s_1c_{23}s_4 - c_1c_4 & s_{23}c_4 & a_2c_3s_4 + d_3c_4 + a_3s_4 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

令式 (2-45) 两边 [${}^4_0T$ 的元素见式 (2-26)] 的元素 (1.3) 及 (3,3) 对应相等，就有

$$\begin{cases} r_{13}(c_1c_{23}c_4 + s_1s_4) + r_{23}(s_1c_{23}c_4 - c_1s_4) - r_{33}(s_{23}c_4) = -s_5 \\ r_{13}(-c_1s_{23}) + r_{23}(-s_1s_{23}) + r_{33}(-c_{23}) = c_5 \end{cases}$$

于是，可由上式表出

$$\theta_5 = \text{Atan2}(s_5, c_5) \tag{2-46}$$

(6) 求 θ_6　再次对式 (2-33) 分离变量

$$({}^0_5T)^{-1}\,{}^0_6T = {}^5_6T(\theta_6) \tag{2-47}$$

其中 ${}^5_6T$ 见式 (2-25)．求出 $({}^0_5T)^{-1}$，令两边元素 (3，1) 和 (1，1) 对应相等，可求得

$$\theta_6 = \text{Atan2}(s_6, c_6) \tag{2-48}$$

式中

$$s_6 = -r_{11}(c_1c_{23}s_4 - s_1c_4) - r_{21}(s_1c_{23}s_4 + c_1c_4) + r_{31}(s_{23}s_4)$$

$$c_6 = r_{11}[(c_1c_{23}c_4 + s_1s_4)c_5 - c_1s_{23}s_5] + r_{21}[(s_1c_{23}c_4 - c_1s_4)c_4 - s_1s_{23}s_5] - r_{31}(s_{23}c_4c_5 + c_{23}c_5)$$

与式 (2-37)、(2-39) 中的正负号相对应，运动学方程就有四组解，如图 2.14 所示．此外，由于 PUMA 560 的手腕可以"翻转"，

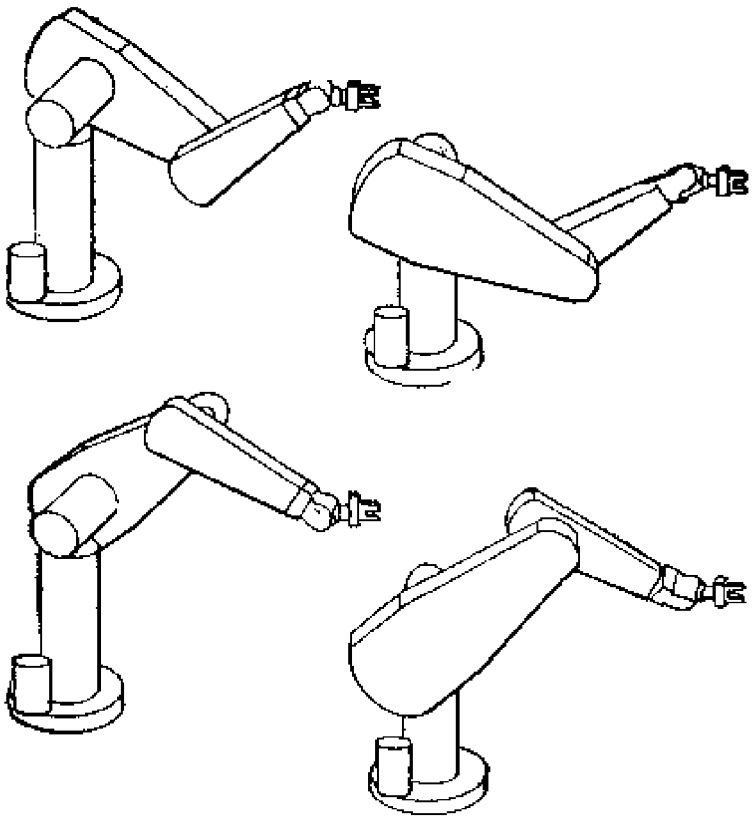


图 2.14　PUMA 560 的四组解

而"翻转" 180° 之后，末端连杆坐标系(手爪)的位置和姿态仍然符合 ${}^0_6T$ 的描述，于是，对于已求得的每一组解 θ_1、θ_2、θ_3、θ_4、θ_5、θ_6，都可以分别求得一组"翻腕"解 θ'_1、θ'_2、θ'_3、θ'_4、θ'_5、θ'_6，而且

$$\theta'_1 = \theta_1, \quad \theta'_2 = \theta_2, \quad \theta'_3 = \theta_3,$$

$$\theta'_4 = \theta_4 + 180°, \quad \theta'_5 = -\theta_5, \quad \theta'_6 = \theta_6 + 180°$$

在这样得到的八组解中，必须去掉那些超出关节转动限制范围的解，在其余的有效解中，通常选取最接近机械手当前形态的解。

2. 几何解法

所谓几何解法，就是设法把机械手的空间几何问题分解成若干个平面几何问题。这样，不建立机械手的运动学方程，而是直接应用平面几何的方法，就可以求解出关节角。当机械手的连杆扭角 $\alpha_i = 90°$ 或 $0°$ 时，几何解法常常是简便可行的。

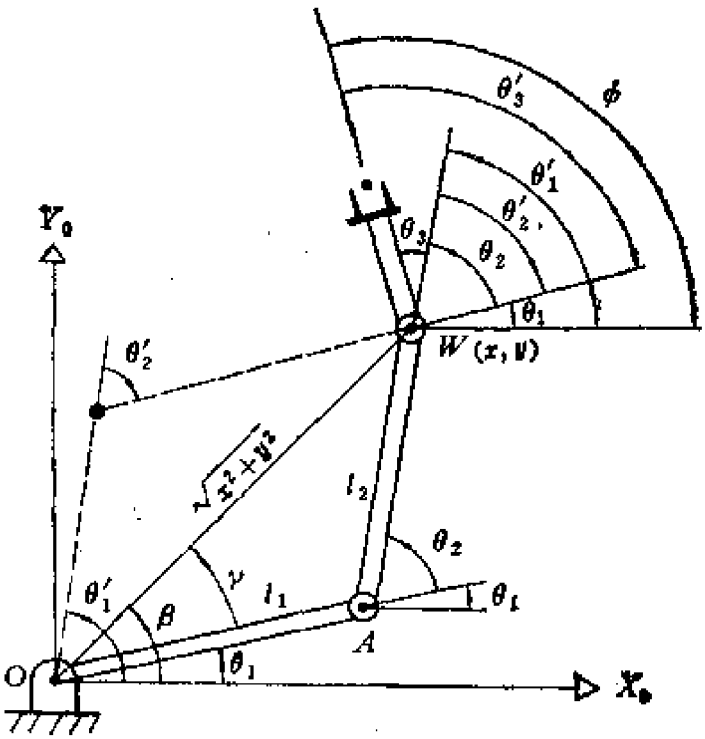


图 2.15　平面机械手的几何解法

例 2.3 图 2.15 示意了一个三自由度平面机械手。实线表示连杆长度 l_1、l_2，对应的关节角为 θ_1、θ_2、θ_3；为实现给定的机械手末端目标，l_1、l_2 也可以有另一种形态，由图中的虚线表示，对应的关节角为 θ_1'、θ_2'、θ_3'。机械手的末端目标可以用三个参数表示：W 点（腕）的坐标 x、y 和连杆 l_3 的姿态角 ϕ，即正 X_0 轴与 l_3 的夹角。由于 ϕ、l_3 为已知，经过变换，很容易得到 x、y。

考虑三角形 OAW，利用余弦定理，有

$$x^2 + y^2 = l_1^2 + l_2^2 - 2l_1l_2\cos(180° - \theta_2) \tag{2-49}$$

因而

$$\begin{cases} \theta_2 = \cos^{-1}\dfrac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2}, & 0° \leqslant \theta_2 \leqslant 180° \\ \theta_2' = -\cos^{-1}\dfrac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2}, & -180° \leqslant \theta_2' \leqslant 0° \end{cases} \tag{2-50}$$

利用辅助角 β 和 γ 可求出 θ_1、θ_1'。首先，β 可以在任何象限，这取决于 x、y 的正负号，因而 β 可以表为双变量反正切，即

$$\beta = \mathrm{Atan2}(y, x) \tag{2-51}$$

再利用余弦定理

$$l_2^2 = x^2 + y^2 + l_1^2 - 2l_1\sqrt{x^2 + y^2}\cos\gamma \tag{2-52}$$

解得

$$\begin{cases} \gamma = \cos^{-1}\dfrac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}}, & 0° \leqslant \gamma \leqslant 180° \\ \gamma' = -\cos^{-1}\dfrac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}}, & -180° \leqslant \gamma' \leqslant 0° \end{cases} \tag{2-53}$$

对应地，可得

$$\begin{cases} \theta_1 = \beta - \gamma \\ \theta_1' = \beta + \gamma' \end{cases} \tag{2-54}$$

最后，由图可知，三个关节角之和必等于杆 l_3 的姿态角，即

$$\begin{cases} \theta_1 + \theta_2 + \theta_3 = \phi \\ \theta_1' + \theta_2' + \theta_3' = \phi' \end{cases} \tag{2-55}$$

由上式解得 θ_3 和 θ_3'，我们就可以得到平面机械手的两组关节角的解。

§2.4 机械手的速度分析

要分析机械手运动的瞬时状况，必须涉及它的线速度和角速度，以及线加速度和角加速度。为方便起见，我们把加速度问题归入动力学方程一节（§2.6）中讨论。

首先说明有关速度表示符号的约定。

空间任意一点的速度表示一般要涉及到两个坐标系，即，既要指明速度是相对于哪个坐标系的运动所造成的，又要指明在哪个坐标系中描述这一速度。为此，我们用两个上标记号加以注明，例如，${}^A({}^B\boldsymbol{V}_P)$ 表示 P 点相对于坐标系 $\{B\}$ 的速度在坐标系 $\{A\}$ 中的描述。

如果两个上标相同，可以只写出一个，例如，${}^B({}^B\boldsymbol{V}_P) = {}^B\boldsymbol{V}_P$。

由于速度也是一个向量，如同位置向量一样，我们可以通过旋转矩阵把速度的描述从一个坐标系变换到另一个坐标系，例如 ${}^A({}^B\boldsymbol{V}_P) = {}^A_B R({}^B\boldsymbol{V}_P)$。

对于坐标系的原点，一般总是考虑它相对于固定的参考坐标系（在机械手问题中，即为基杆坐标系 $\{0\}$）的速度，我们约定速度表示只用一个上标指明在哪个坐标系中描述速度，例如 ${}^B\boldsymbol{V}_Q$ 表示某个坐标系原点 Q 相对于固定参考坐标系的运动速度在坐标系 $\{B\}$ 中的描述。

在讨论加速度问题时，我们仍然沿用上述约定。

一、刚体运动的速度

由于刚体的描述等价于与它固联的一个坐标系，因此我们只需要研究坐标系之间的相对运动，就可以讨论刚体的运动情况。

设有两个原点不重合的坐标系 $\{A\}$ 和 $\{B\}$，其中 $\{A\}$ 是固定不动的，$\{B\}$ 的姿态 ${}^A_BR$ 不随时间而改变，但 $\{B\}$ 相对于 $\{A\}$ 作直线运动，如图 2.16 所示。

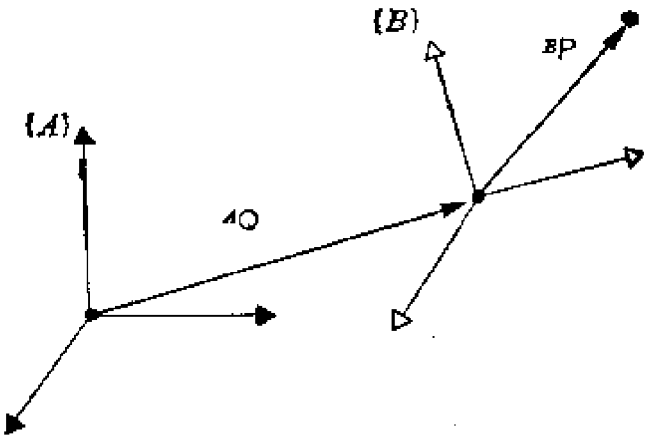


图 2.16　坐标系的平移运动

空间一点 P 相对于 $\{A\}$ 的运动是由于 ${}^AQ$ 与 ${}^BP$ 随着时间的改变。显然 P 点在 $\{A\}$ 中的线速度就为

$$ {}^AV_P = {}^AV_Q + {}^A_BR\,{}^BV_P \tag{2-56} $$

现在设 $\{B\}$ 与固定坐标系 $\{A\}$ 的原点始终重合，但 $\{B\}$ 相对于 $\{A\}$ 绕角速度向量 ${}^A\Omega_B$ 旋转，(这里用向量 ${}^A\Omega_B$ 表示坐标系的旋转，与 §2.2 中的基本旋转的合成结果实际上是等价的)，${}^A\Omega_B$ 的大小表示旋转速度的值。如图 2.17 所示，空间一点 P 相对于 $\{A\}$ 的运动速度有两个分量，下面分别说明。

假定 ${}^BP$ 相对于 $\{B\}$ 不动，由于 $\{B\}$ 的旋转，在 $\{A\}$ 中看起来，P 的速度为

$$ {}^A\Omega_B \times {}^AP $$

这一结果可从图 2.18 中得到直观的解释。${}^AP$ 的微分变化 ΔP 垂直于 ${}^A\Omega_B$ 和 ${}^AP$，其大小为

$$ |\Delta P| = (|{}^AP| \sin\theta)(|{}^A\Omega_B|\Delta t) $$

若考虑 ${}^BP$ 相对于 $\{B\}$ 的速度，P 点在 $\{A\}$ 中的速度就为

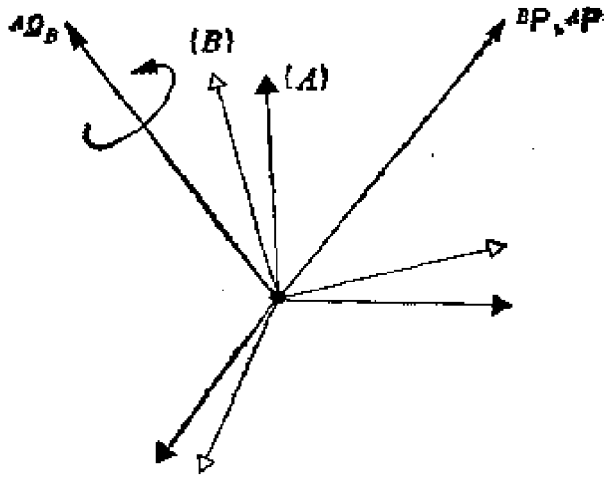


图 2.17　坐标系的旋转运动

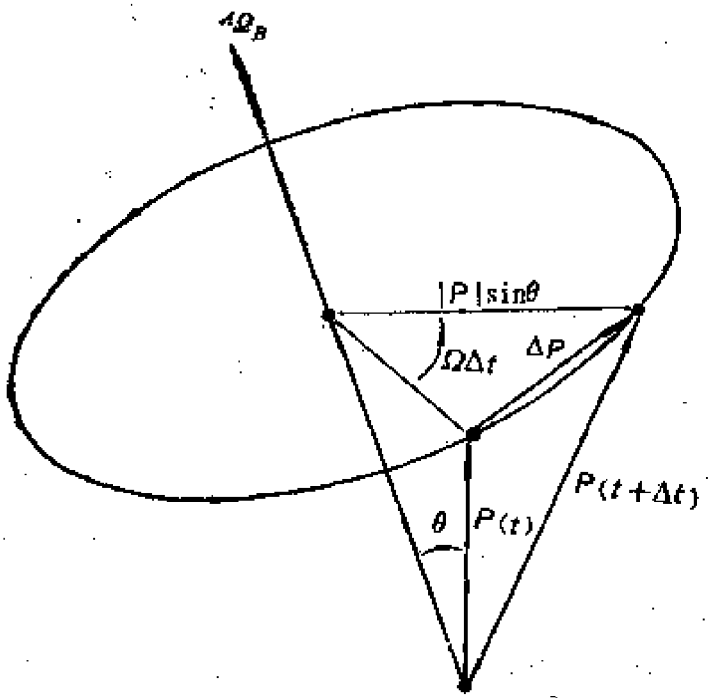


图 2.18　坐标系旋转造成的速度

$$ {}^AV_P = {}^A({}^BV_P) + {}^A\Omega_B \times {}^AP \qquad (2\text{-}57) $$

用旋转矩阵去掉双重上标，又由于在任何瞬时 ${}^AP$ 都可描述为 ${}^A_BR\,{}^BP$，上式就为

$$ {}^AV_P = {}^A_BR\,{}^BV_P + {}^A\Omega_B \times {}^A_BR\,{}^BP \qquad (2\text{-}58) $$

最后，考虑 $\{B\}$ 相对于 $\{A\}$ 既有直线运动又有旋转运动的一般情形。显然，只要在式(2-58)中加入原点线速度一项，我们就可得到 P 点相对于固定坐标系的速度

$$^{A}\boldsymbol{V}_{P} = {}^{A}\boldsymbol{V}_{Q} + {}^{A}_{B}\boldsymbol{R}\,{}^{B}\boldsymbol{V}_{P} + {}^{A}\boldsymbol{\Omega}_{B} \times {}^{A}_{B}\boldsymbol{R}\,{}^{B}\boldsymbol{P} \qquad (2\text{-}59)$$

二、连杆间速度的变换

连杆 i 相对于参考坐标系 $\{0\}$ 的速度可表为连杆坐标系 $\{i\}$ 的角速度向量 ${}^{0}\boldsymbol{\Omega}_{i}$ 和原点线速度向量 ${}^{0}\boldsymbol{V}_{i}$，简记为 $\boldsymbol{\omega}_{i}$ 和 $\boldsymbol{v}_{i}$，如果把这两个向量在 $\{i\}$ 中描述就为 ${}^{i}\boldsymbol{\omega}_{i}$，${}^{i}\boldsymbol{v}_{i}$。如图 2.19 所示。

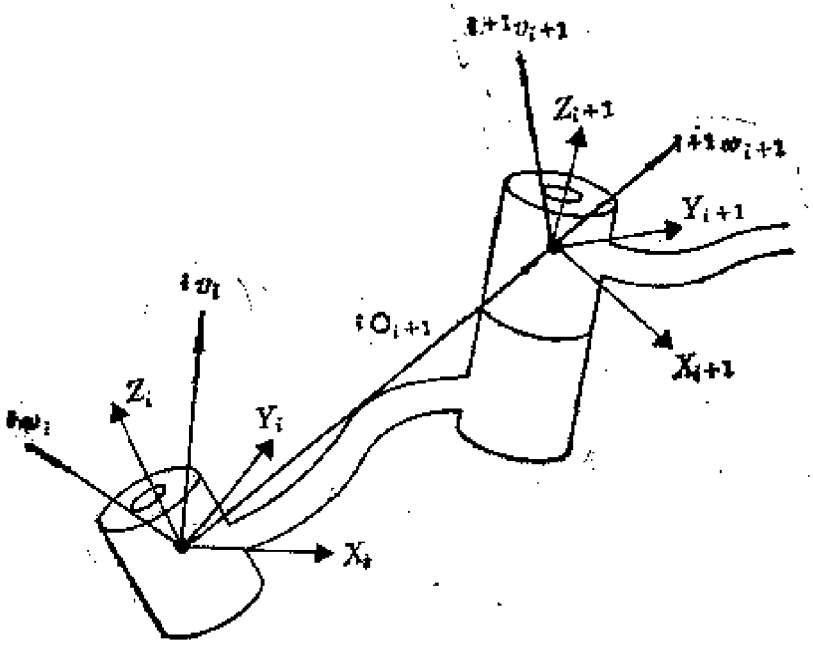


图 2.19　连杆的速度

同一坐标系中的向量可以相加。因此连杆 $i+1$ 的角速度就等于它与连杆 i 相同的角速度加上它在关节 $i+1$ 处的旋转速度所引起的分量，这在 $\{i\}$ 中就描述为

$$^{i}\boldsymbol{\omega}_{i+1} = {}^{i}\boldsymbol{\omega}_{i} + {}^{i}_{i+1}\boldsymbol{R}\,\dot{\theta}_{i+1}\,{}^{i+1}\hat{\boldsymbol{Z}}_{i+1} \qquad (2\text{-}60)$$

其中，θ_{i+1} 为关节转角，${}^{i+1}\hat{\boldsymbol{Z}}_{i+1}$ 为 $\{i+1\}$ 的 Z 轴的单位向量在

$\{i+1\}$ 中的描述，从而

$$\dot{\theta}_{i+1}{}^{i+1}\hat{Z}_{i+1} = {}^{i+1}\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix} \tag{2-61}$$

把式 (2-60) 两边左乘 ${}^{i+1}_{i}R$，就得到连杆 $i+1$ 的角速度在 $\{i+1\}$ 的描述：

$${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}{}^{i+1}\hat{Z}_{i+1} \tag{2-62}$$

坐标系 $\{i+1\}$ 原点的线速度则为与 $\{i\}$ 原点相同的速度再加上由于 $\{i\}$ 的旋转速度而产生的新分量。这与式 (2-59) 类似，只是由于 ${}^{i}Q_{i+1}$ 在 $\{i\}$ 中为常数，因而有一项消失了，于是

$${}^{i}v_{i+1} = {}^{i}v_i + {}^{i}\omega_i \times {}^{i}Q_{i+1} \tag{2-63}$$

而

$${}^{i+1}v_{i+1} = {}^{i+1}_{i}R({}^{i}v_i + {}^{i}\omega_i \times {}^{i}Q_{i+1}) \tag{2-64}$$

例 2.4 对于图 2.20 所示的带有两个旋转关节的平面机械手，可以运用上述的递推公式 (2-62) 和 (2-64) 求出机械手末端的速度。求解过程中，我们认为机械手有一个固定不动的关节 3，即有 $\theta_3 = 0$，$\dot{\theta}_3 = 0$。

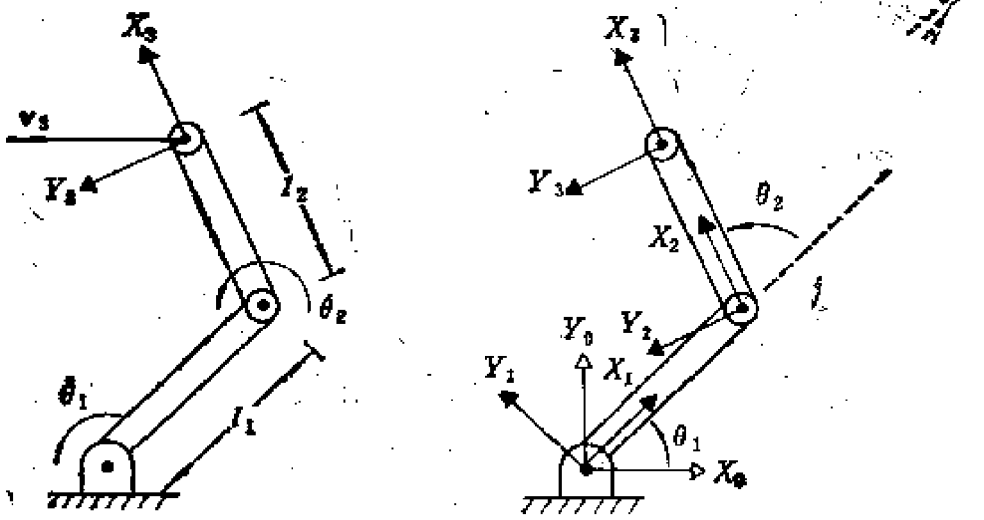


图 2.20 二自由度平面机械手

首先，求出旋转变换 ${}^{i+1}_{i}R$（即 ${}^{i}_{i+1}R^{T}$）和平移变换 ${}^{i}Q_{i+1}$，并把它们写成齐次变换的形式：

$$\left.\begin{aligned}
{}^{0}_{1}T &= \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{1}_{2}T &= \begin{bmatrix} c_2 & -s_2 & 0 & l_1 \\ s_2 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{2}_{3}T &= \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}\right\} \quad (2\text{-}65)$$

然后，逐次运用式 (2-62) 和 (2-64)。由于基杆坐标系 {0} 固定不动，因而

$${}^{0}\omega_0 = 0, \qquad {}^{0}v_0 = 0$$

其余坐标系的速度为

$${}^{1}\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}, \qquad {}^{1}v_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$${}^{2}\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix}, \quad {}^{2}v_2 = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{bmatrix}$$

$${}^{3}\omega_3 = {}^{2}\omega_2, \qquad {}^{3}v_3 = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 + l_2(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix}$$

为把这些速度相对于基杆坐标系{0}来描述，可先通过式(2-65)求出旋转变换

$$ {}^0_3R = {}^0_1R\,{}^1_2R\,{}^2_3R = \begin{bmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2-66} $$

于是，

$$ \left.\begin{aligned} {}^0\omega_3 &= {}^0_1R\,{}^3\omega_3 = \begin{bmatrix} 0 \\ 0 \\ \dot\theta_1 + \dot\theta_2 \end{bmatrix} \\ {}^0v_3 &= {}_1R^0_3v_3 = \begin{bmatrix} -l_1s_1\dot\theta_1 - l_2s_{12}(\dot\theta_1 + \dot\theta_2) \\ l_2c_1\dot\theta_2 + l_2c_{12}(\dot\theta_1 + \dot\theta_2) \\ 0 \end{bmatrix} \end{aligned}\right\} \tag{2-67} $$

三、雅可比矩阵

一般地，对于 n 个自由度的机械手，如果逐次运用式(2-62)和(2-64)，就可以求出末端连杆的角速度 ${}^n\omega_n$ 和线速度 ${}^nv_n$，借助于 ${}^0_nR$，则可以得到这些速度在基杆坐标系中的描述，即 ${}^0\omega_n$ 和 ${}^0v_n$，或简记为 ω_n、v_n。如果把 ω_n、v_n 写成一个向量

$$ \dot\chi = \begin{bmatrix} v_n \\ \omega_n \end{bmatrix} \tag{2-68} $$

具体的推导结果可表示为一个雅可比矩阵形式

$$ \dot\chi = J(\Theta)\dot\Theta \tag{2-69} $$

其中，Θ 为 $n \times 1$ 的机械手关节(旋转关节或柱关节)的位移向量，$\dot\chi$ 为 6×1 直角坐标速度向量，于是，$6 \times n$ 雅可比矩阵 $J(\Theta)$ 表明了机械手关节速度与末端(手爪)直角坐标速度之间的线性变换关系。

例如，例 2.4 中的式(2-67)可以写成

$$\begin{bmatrix} v_{3X} \\ v_{3Y} \\ v_{3Z} \\ \omega_{3X} \\ \omega_{3Y} \\ \omega_{3Z} \end{bmatrix} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -c_2 s_{12} \\ l_1 c_1 + l_2 c_{13} & l_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{2-70}$$

于是，机械手的雅可比矩阵就为

$$J(\boldsymbol{\Theta}) = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

需要指出，雅可比矩阵是关节位移的函数，因此，它所表达的关节速度与末端直角坐标速度之间的线性关系只是一种瞬时的关系，随着机械手形态的变化，雅可比矩阵将有所不同。

雅可比矩阵实际上可以通过对机械手坐标系进行微分平移和微分旋转来推导，各种推导方法和计算方法在此不再讨论。

当机械手有六个自由度时，雅可比矩阵为 $J(\boldsymbol{\Theta})$ 为 6×6 方阵。如果 $J(\boldsymbol{\Theta})$ 可逆，那末只要给定机械手末端的直角坐标速度，就可以求得相应的关节速度

$$\dot{\boldsymbol{\Theta}} = J^{-1}(\boldsymbol{\Theta})\dot{\boldsymbol{X}} \tag{2-71}$$

但是，雅可比矩阵 $J(\boldsymbol{\Theta})$ 是随着机械手的形态变化的，某些形态下的 $\boldsymbol{\Theta}$ 值就可能使 $J(\boldsymbol{\Theta})$ 成为奇异，这时的机械手末端位置称之为机械手的奇异点。当机械手处于奇异形态时，它在直角坐标空间的自由度就有所减少，这意味着在直角坐标空间的某些方向上，无论选取什么样的关节速度，机械手都不能沿着那些方向运动。奇异点可能处于机械手工作空间的边界或工作空间内部。

例 2.5 对于例 2.4 的平面机械手，如果只考虑末端的线速度，雅可比矩阵可简写为

$$J(\Theta)=\begin{bmatrix}-l_1s_1-l_2s_{12} & -l_2s_{12}\\ l_1c_1+l_2c_{12} & l_2c_{12}\end{bmatrix} \tag{2-72}$$

对上式求逆

$$J^{-1}(\Theta)=\left[\frac{1}{l_1l_2s_2}\right]\begin{bmatrix}l_2c_{12} & l_2s_{12}\\ -l_1c_1-l_2c_{12} & -l_1s_1-l_2s_{12}\end{bmatrix} \tag{2-73}$$

显然，当 $s_2=0$ 即 $\theta_2=0°$或 $180°$ 时，$J^{-1}(\Theta)$ 不存在，机械手处于奇异形态。$\theta_2=0°$ 表示机械手完全伸直，$\theta_2=180°$ 表示机械手完全折迭起来，两种情况都使机械手末端只能沿一个直角坐标方向运动，相应的奇异点都位于工作空间的边界上。

假如机械手末端以单位速度沿轴 X_0 运动（图 2.21），我们来考察它接近奇异点时的情况。

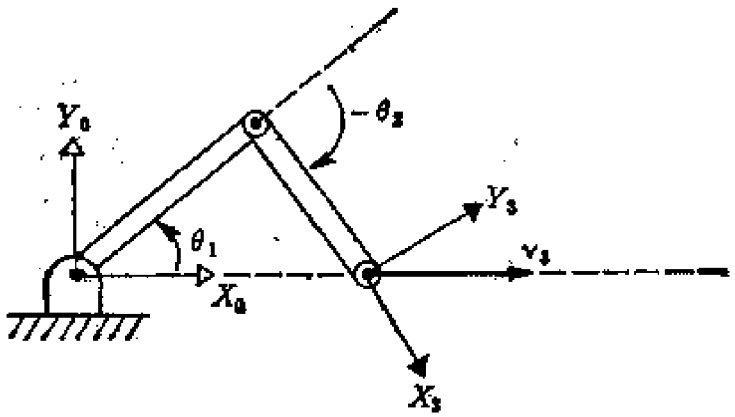


图 2.21 平面机械手向奇异点运动

由式 (2-73) 可求得机械手的关节速度

$$\dot{\theta}_1=\frac{c_{12}}{l_1s_2},\quad \dot{\theta}_2=-\frac{c_1}{l_2s_2}-\frac{c_{12}}{l_1s_2}$$

显然，当 θ_2 趋向于 0 时，即当机械手接近奇异点时，关节速度将趋向于无穷大。

§2.5 机械手的静力学分析

当机器人的机械手通过与末端连杆相连的手爪推动物体或支撑负荷时，或者在考虑机械手自身结构的设计时，都需要分析机械手在静态下的力或力矩的平衡问题，也就是要求解为了维持平衡，机械手的各个关节必须给出的力或力矩。

一、刚体受到的静态力和力矩

一个刚体 G 在任何时刻都可能受到几个力和力矩的作用，它们可以合成为作用在 G 上某点 C 的一个合力 $\boldsymbol{F}_C$ 和合力矩 $\boldsymbol{N}_C$，如果我们用来表示 G 的固联坐标系的原点位于 A，那么我们也可以在 A 点得到一个与 $\boldsymbol{F}_C$ 和 $\boldsymbol{N}_C$ 等价的静态力 $\boldsymbol{F}_A$ 和静态力矩 $\boldsymbol{N}_A$ (图 2.22)。

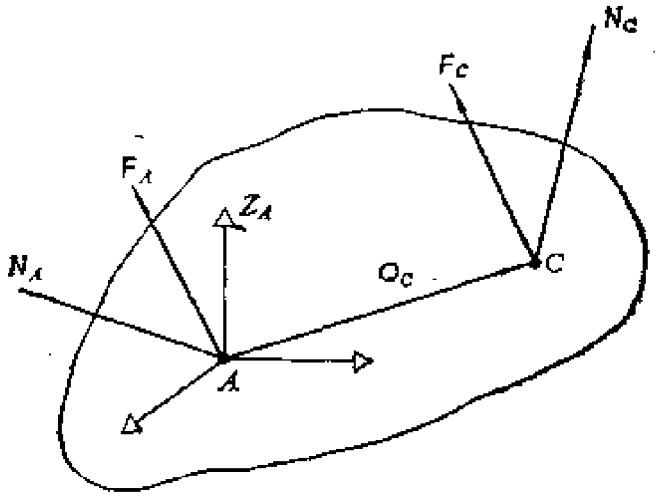


图 2.22 刚体的静态力和力矩

根据静力学知识，就有

$$\boldsymbol{F}_A = \boldsymbol{F}_C \tag{2-74}$$

$$\boldsymbol{N}_A = \boldsymbol{N}_C + \boldsymbol{Q}_C \times \boldsymbol{F}_C \tag{2-75}$$

式(2-75)中，$Q_c \times F_c$ 为 F_c 由于"杠杆" Q_c 而产生的力矩。

二、连杆间静态力、力矩的变换

在静态平衡的条件下，我们可以把机械手看作是它的各个关节都被"锁定"的一个结构。于是，根据式(2-74)和(2-75)，就可以把作用在连杆 $i+1$ 上的力和力矩等价地描述为作用在连杆 i 上的力和力矩。

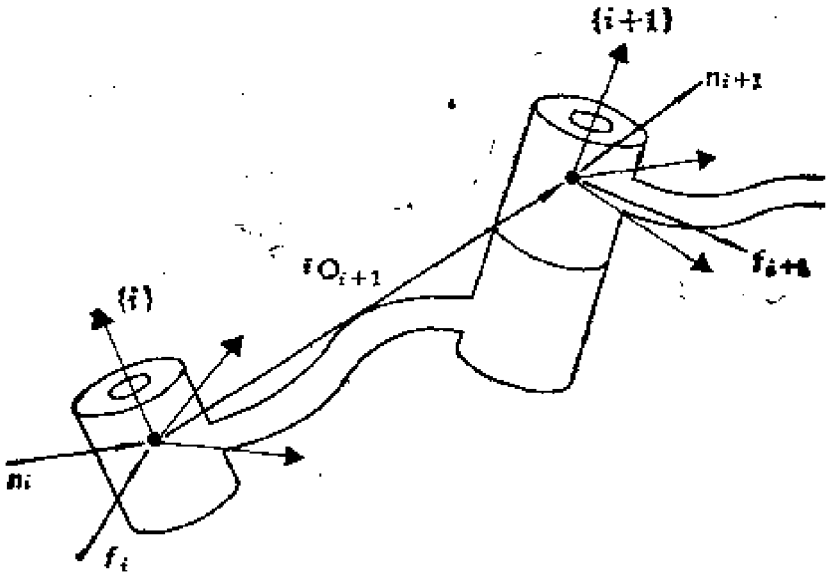


图 2-23　连杆的静态力和力矩

如图 2.23 所示，就有

$$^iF_i = {}^iF_{i+1} \tag{2-76}$$

$$^iN_i = {}^iN_{i+1} + {}^iQ_{i+1} \times {}^iF_{i+1} \tag{2-77}$$

运用旋转变换矩阵 ${}^i_{i+1}R$，上面两式右边的力和力矩可以写成都定义在自身坐标系中的描述形式

$$^iF_i = {}^i_{i+1}R\,{}^{i+1}F_{i+1} \tag{2-78}$$

$$^iN_i = {}^i_{i+1}R\,{}^{i+1}N_{i+1} + {}^iQ_{i+1} \times {}^iF_i \tag{2-79}$$

这样，我们就可以逐次运用式(2-78)和(2-79)，求出高序号连杆受到的力和力矩向低序号连杆传递的结果。

最后，我们考虑机械手的静态平衡问题。静态力向量和力矩向量的所有分量中，一部分为机械手本身的连杆结构所平衡，一部分则为各个关节的力矩或力所平衡。对于旋转关节，关节平衡力矩的 Z 分量应为

$$t_i = {}^iN_i^T\ {}^iZ_i \tag{2-80}$$

对于柱关节，关节平衡力的 Z 分量应为

$$s_i = {}^iF_i^T\ {}^i\hat{Z}_i \tag{2-81}$$

类似于速度的雅可比矩阵形式，我们也可以得到一个力域中的雅可比矩阵形式，而且可以证明，在此，雅可比矩阵是以转置的形式出现的

$$\tau = J^T(\Theta)\mathcal{F} \tag{2-82}$$

其中，τ 为 $n \times 1$ 向量，表示 n 个关节上的平衡力/平衡力矩，而 $\mathcal{F}$ 为作用在手爪上的直角坐标力/力矩形成的 6×1 向量。因此，$J^T(\Theta)$ 实际上表示把手爪上的直角坐标力/力矩映射为等价的关节力/关节力矩。

§2.6 机械手的动力学方程

机械手动力学研究的是机械手的运动与产生这种运动的力和力矩之间的关系。这里也有两个问题需要解决：动力学正问题——施加一组关节力矩或力，计算出机械手的运动，即关节位移变量、速度和加速度；动力学逆问题——已知轨迹点所对应的关节位移变量、速度和加速度，求出所需要的关节力矩或力。正问题为机械手的仿真研究所用，而逆问题直接与机械手的控制研究有关。在此我们仅讨论逆问题的一些求解方法。

一、刚体运动的加速度

类似于刚体运动的速度，我们也通过两个坐标系 $\{A\}$ 和 $\{B\}$

间的相对运动来讨论刚体的加速度。

对于{B}与固定坐标系{A}的原点重合但姿态有相对变化的情形，式(2-58)表明了空间一点 ${}^{B}\boldsymbol{P}$ 在{A}中的速度，即

$$^{A}\boldsymbol{\Omega}_{C} = {}^{A}\boldsymbol{\Omega}_{B} + {}^{A}_{B}R\,{}^{B}\boldsymbol{\Omega}_{C} \tag{2-89}$$

而角加速度为

$$^{A}\dot{\boldsymbol{\Omega}}_{C} = {}^{A}\dot{\boldsymbol{\Omega}}_{B} + \frac{d}{dt}\left({}^{A}_{B}R\,{}^{B}\boldsymbol{\Omega}_{C}\right)$$

仿照着式(2-84)的导数展开式，直接写出上式右边的后一项得到

$$^{A}\dot{\boldsymbol{\Omega}}_{C} = {}^{A}\dot{\boldsymbol{\Omega}}_{B} + {}^{A}_{B}R\,{}^{B}\dot{\boldsymbol{\Omega}}_{C} + {}^{A}\boldsymbol{\Omega}_{B} \times {}^{A}_{B}R\,{}^{B}\boldsymbol{\Omega}_{C} \tag{2-90}$$

二、刚体的惯性张量

三维空间中自由运动的刚体是用惯性张量来描述它的质量分布的。我们以刚体的质心C为原点定义一个坐标系$\{C\}$，惯性张量在$\{C\}$中表示为一个3×3对角矩阵

$$C_I = \begin{bmatrix} I_{XX} & I_{XY} & I_{XZ} \\ I_{XY} & I_{YY} & I_{YZ} \\ I_{XZ} & I_{YZ} & I_{ZZ} \end{bmatrix} \tag{2-91}$$

其中对角线元素是刚体以X、Y、Z三轴为旋转轴的转动惯量，它们依次为

$$I_{XX} = \iiint_{v} (y^2 + z^2)\rho dv$$

$$I_{YY} = \iiint_{v} (x^2 + z^2)\rho dv$$

$$I_{ZZ} = \iiint_{v} (y^2 + z^2)\rho dv$$

其余元素为惯性积：

$$I_{XY} = \iiint_{v} xy\rho dv$$

$$I_{XZ} = \iiint_{v} xz\rho dv$$

$$I_{YZ} = \iiint_{v} yz\rho dv$$

其中，ρ为密度，dv为微分体积单元，dv的位置由向量$^{C}P =$

$(x, y, z)^T$ 确定，P 为构成刚体的点。

惯性张量是参考坐标系位置和姿态的函数。根据力学中的平行轴定理，质心坐标系 $\{C\}$ 中表示的惯性张量可变换到另一参考坐标系，例如 $\{A\}$，而且

$$\begin{cases} {}^{A}I_{ZZ} = {}^{C}I_{ZZ} + m(x_C^2 + y_C^2) \\ {}^{A}I_{XY} = {}^{C}I_{XY} + mx_Cy_C \end{cases} \tag{2-92}$$

式中 x_C、y_C、z_C 为 ${}^{A}C$ 的三个分量。其它的转动惯量和惯性积可参照式 (2-92) 通过置换求得。

三、牛顿-欧拉方程

如果把图 2.24 所示的机械手的连杆 i 看作刚体，它的质心加速度 V_C、总质量 m 与引起这一加速度的力 F 之间的关系满足牛顿第二运动定律

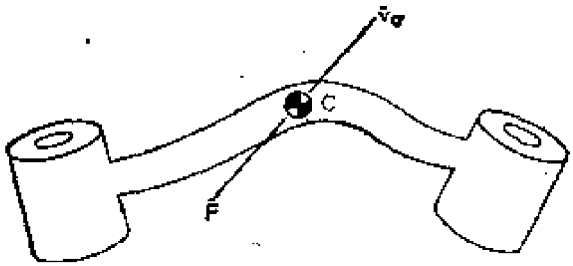


图 2.24　作用力与连杆的质心加速度

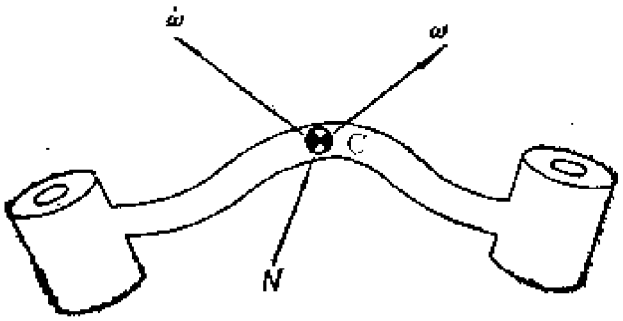


图 2.25　作用力矩与连杆的角加速度

$$\boldsymbol{F} = m\boldsymbol{V}_C \tag{2-93}$$

当连杆围绕通过质心的转轴旋转时，角速度 $\boldsymbol{\omega}$、角加速度 $\dot{\boldsymbol{\omega}}$、惯性张量 ${}^{C}I$、与作用力矩 $\boldsymbol{N}$（图 2.25）满足欧拉方程

$$\boldsymbol{N} = {}^{C}I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times {}^{C}I\boldsymbol{\omega} \tag{2-94}$$

四、动力学逆问题

已知机械手各关节的位置、速度和加速度 $(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}, \ddot{\boldsymbol{\Theta}})$，我们分两步求出产生给定运动的关节转矩。

第一步，为了得到作用在每个连杆上的惯性力，必须求出各个连杆、质心的角速度、线加速度和角加速度，这可从连杆 1 开始，逐次向高序号连杆迭代计算。

回顾式 (2-62)，连杆 $i+1$ 的角速度为

$${}^{i+1}\boldsymbol{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\boldsymbol{\omega}_i + \dot{\theta}_{i+1}\,{}^{i+1}\boldsymbol{Z}_{i+1} \tag{2-95}$$

根据式 (2-90)，可求得连杆 $i+1$ 的角加速度

$$\begin{aligned}{}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} = {}&{}^{i+1}_{i}R\,{}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i+1}_{i}R\,{}^{i}\boldsymbol{\omega}_i \times \dot{\theta}_{i+1}\,{}^{i+1}\boldsymbol{Z}_{i+1} \\ &+ \ddot{\theta}_{i+1}\,{}^{i+1}\boldsymbol{Z}_{i+1}\end{aligned} \tag{2-96}$$

对于柱关节，上式简化为

$${}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\boldsymbol{\omega}}_i \tag{2-97}$$

根据式 (2-88)，可求得 $\{i+1\}$ 原点的线加速度

$$\begin{aligned}{}^{i+1}\dot{\boldsymbol{v}}_{i+1} = {}&{}^{i+1}_{i}R[{}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\boldsymbol{Q}_{i+1} + {}^{i}\boldsymbol{\omega}_i \times ({}^{i}\boldsymbol{\omega}_i \times {}^{i}\boldsymbol{Q}_{i+1}) \\ &+ {}^{i}\dot{\boldsymbol{v}}_i]\end{aligned} \tag{2-98}$$

式中 ${}^{i}\boldsymbol{Q}_{i+1}$ 为 $\{i+1\}$ 原点在 $\{i\}$ 中描述的向量。对于柱关节，根据式 (2-87)，原点线加速度则为

$$\begin{aligned}{}^{i+1}\dot{\boldsymbol{v}}_{i+1} = {}&{}^{i+1}_{i}R[{}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\boldsymbol{Q}_{i+1} + {}^{i}\boldsymbol{\omega}_i \times ({}^{i}\boldsymbol{\omega}_i \times {}^{i}\boldsymbol{Q}_{i+1}) \\ &+ {}^{i}\dot{\boldsymbol{v}}_i] + 2\,{}^{i+1}\boldsymbol{\omega}_{i+1} \times \dot{\boldsymbol{d}}_{i+1}\,{}^{i+1}\boldsymbol{Z}_{i+1} \\ &+ \ddot{\boldsymbol{d}}_{i+1}\,{}^{i+1}\boldsymbol{Z}_{i+1}\end{aligned} \tag{2-99}$$

再根据式 (2-88)，可求得质心 C_{i+1} 的线加速度

$${}^{i+1}\dot{\boldsymbol{v}}_{C_{i+1}} = {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} \times {}^{i+1}\boldsymbol{Q}_{C_{i+1}} + {}^{i+1}\boldsymbol{\omega}_{i+1}$$

$$\times ({}^{i+1}\omega_{i+1} \times {}^{i+1}Q_{C_{i+1}}) + {}^{i+1}\dot{v}_{i+1} \qquad (2\text{-}100)$$

在此我们规定了质心坐标系 $\{C_{i+1}\}$ 的姿态与 $\{i+1\}$ 相同。式(2-100)对于柱关节也适用。

注意，上述方程式用于坐标系 $\{1\}$ 时特别简单，因为 ${}^{0}\omega_{0} = {}^{0}\dot{\omega}_{0} = 0$。

到此，我们可以运用牛顿-欧拉方程式计算出作用在质心 C_{i+1} 上的惯性力和力矩：

$${}^{i+1}F_{i+1} = m_{i+1}{}^{i+1}\dot{v}_{C_{i+1}} \qquad (2\text{-}101)$$

$${}^{i+1}N_{i+1} = {}^{C_{i+1}}I_{i+1}{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}{}^{i+1}\omega_{i+1} \qquad (2\text{-}102)$$

第二步，对连杆 i 写出力和力矩的平衡方程式，最后从末端连杆 n 开始，逐次向较低序号的连杆迭代计算关节力和力矩。

连杆 i 所受的是其相邻连杆所施加的作用力和力矩，以及惯性力和力矩。我们规定：

f_i — 连杆 $i-1$ 作用在连杆 i 上的力，

n_i — 连杆 $i-1$ 作用在连杆 i 上的力矩。

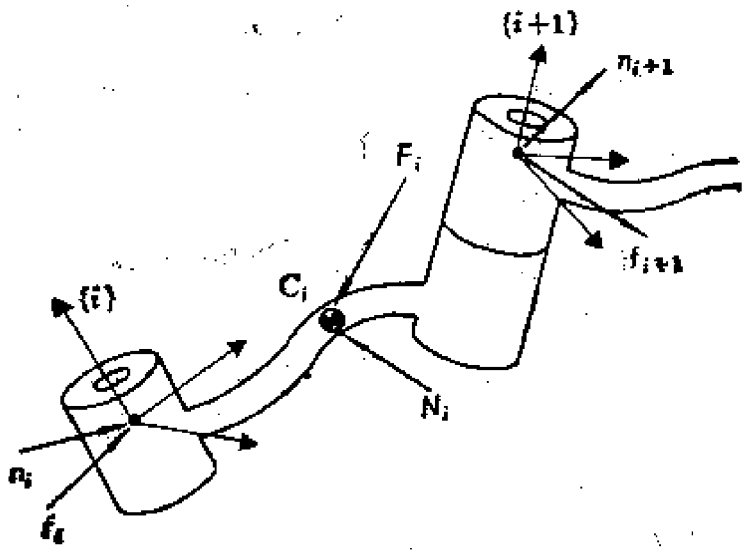


图 2.26 力和力矩的平衡

如图 2.26 所示，我们写出连杆 i 上的力平衡方程式

$$^iF_i = {}^if_i - {}_{i+1}^iR^{i+1}f_{i+1} \tag{2-103}$$

把关于质心的力矩加起来，使之为零，得到力矩平衡方程式

$$\begin{aligned}{}^iN_i = {}^in_i - {}^in_{i+1} + (-{}^iQ_{C_i}) \times {}^if_i \\ - ({}^iQ_{i+1} - {}^iQ_{C_i}) \times {}^if_{i+1}\end{aligned} \tag{2-104}$$

根据式 (2-103) 的结果，再运用旋转矩阵，式 (2-104) 可写为

$$\begin{aligned}{}^iN_i = {}^in_i - {}_{i+1}^iR^{i+1}n_{i+1} - {}^iQ_{C_i} \times {}^iF_i - {}^iQ_{i+1} \\ \times {}_{i+1}^iR^{i+1}f_{i+1}\end{aligned} \tag{2-105}$$

最后，把式 (2-103) 和 (2-105) 整理成迭代形式：

$${}^if_i = {}_{i+1}^iR^{i+1}f_{i+1} + {}^iF_i \tag{2-106}$$

$$\begin{aligned}{}^in_i = {}^iN_i + {}_{i+1}^iR^{i+1}n_{i+1} + {}^iQ_{C_i} \times {}^iF_i + {}^iQ_{C_i} \\ \times {}_{i+1}^iR^{i+1}f_{i+1}\end{aligned} \tag{2-107}$$

如同静态情况一样，所需要给出的关节力/力矩应为一个连杆作用于相邻连杆的力/力矩的 Z 分量，对于旋转关节

$$\tau_i = {}^in_i^T \hat{Z}_i \tag{2-108}$$

对于柱关节

$$s_i = {}^iF_i^T \hat{Z}_i \tag{2-109}$$

注意，如果机械手在自由空间运动，则 ${}^{n+1}f_{n+1} = {}^{n+1}n_{n+1} = 0$，否则应将它与外界接触的力和力矩包括在平衡方程式中。

另外，如果考虑连杆的重力，只要让 $|{}^0\dot{v}_0| = g$（重力加速度），方向向上，这等于说机械手的基座以加速度 g 向上作加速运动，从而相当于重力造成的影响。

§ 2.7 封闭形式的动力学方程

一、封闭形式动力学方程的推导

上一节介绍了利用牛顿-欧拉方程求解机械手逆动力学的迭代关系式，这些关系式一方面可作为数值计算方法，另一方面还可

以用来解析地推导封闭形式的动力学方程，即清楚地描述关节力/力矩与关节位移、速度、加速度之间的函数关系式。下面通过一个简单的例子加以说明。

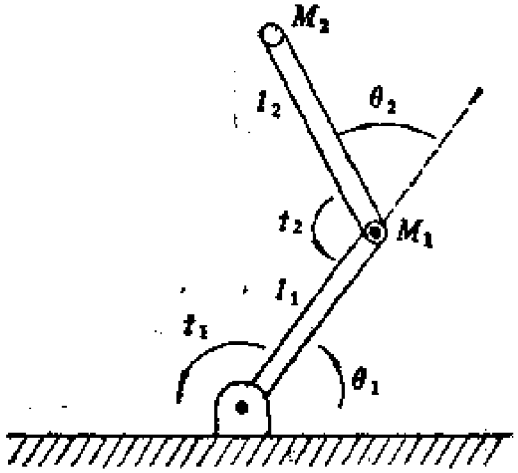


图 2.27　质量集中在连杆末稍端点的平面机械手

例 2.6　在图 2.27 中示意了一个带有两个旋转关节的平面机械手，假设连杆的质量都集中于它们的末稍端点处，这样可使质心坐标系中表示的连杆惯性张量为零矩阵，以便简化封闭形式动力学方程的推导。

首先确定出现在牛顿-欧拉迭代关系式中的各个量。

连杆质心向量为

$$^1Q_{C_1} = l_1\hat{X}_1, \quad ^2Q_{C_2} = l_2\hat{X}_2$$

根据假设，连杆质心坐标系中的惯性张量为

$$^{C_1}I_1 = 0, \quad ^{C_2}I_2 = 0$$

由于机械手末端没有受到任何外力，因此

$$f_3 = 0, \quad n_3 = 0$$

由于基座固定不动，因此

$$^0\omega_0 = 0, \quad ^0\dot{\omega}_0 = 0$$

考虑到重力影响，有

$$^0\dot{v}_0 = g\hat{Y}_0$$

相邻连杆坐标系之间的旋转矩阵为

$$^{i+1}_{i}R = \begin{bmatrix} c_{i+1} & -s_{i+1} & 0 \\ s_{i+1} & c_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad ^{i}_{i+1}R = \begin{bmatrix} c_{i+1} & s_{i+1} & 0 \\ -s_{i+1} & c_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

注意上式中 c_{i+1} 和 s_{i+1} 表示余弦函数 $\cos\theta_{i+1}$ 和正弦函数 $\sin\theta_{i+1}$。

现在运用 § 2.6 中的式 (2-95)、(2-96)、(2-98)、(2-100)、(2-101) 和 (2-102)，由连杆 1 到连杆 2 迭代计算. 对于连杆 1，有

$$
{}^1\boldsymbol{\omega}_1 = \dot{\theta}_1{}^1\hat{\boldsymbol{Z}}_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}
$$

$$
{}^1\dot{\boldsymbol{\omega}}_1 = \ddot{\theta}_1{}^1\hat{\boldsymbol{Z}}_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix}
$$

$$
{}^1\dot{\boldsymbol{v}}_1 = \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix} = \begin{bmatrix} gs_1 \\ gc_1 \\ 0 \end{bmatrix}
$$

$$
{}^1\dot{\boldsymbol{v}}_{C_1} = \begin{bmatrix} 0 \\ l_1\ddot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} -l_1\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} gs_1 \\ gc_1 \\ 0 \end{bmatrix} = \begin{bmatrix} -l_1\dot{\theta}_1^2 + gs_1 \\ l_1\ddot{\theta}_1 + gc_1 \\ 0 \end{bmatrix}
$$

$$
{}^1\boldsymbol{F}_1 = \begin{bmatrix} -m_1l_1\dot{\theta}_1^2 + m_1gs_1 \\ m_1l_1\ddot{\theta}_1 + m_1gc_1 \\ 0 \end{bmatrix}
$$

$$
{}^1\boldsymbol{N}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
$$

对于连杆 2，有

$$
{}^2\boldsymbol{\omega}_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix}
$$

$$
{}^2\dot{\boldsymbol{\omega}}_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 + \ddot{\theta}_2 \end{bmatrix}
$$

$$
{}^2\dot{v}_2 = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -l_1\dot{\theta}_1^2 + gs_1 \\ l_1\ddot{\theta}_1 + gc_1 \\ 0 \end{bmatrix}
$$

$$
= \begin{bmatrix} l_1\ddot{\theta}_1 s_2 - l_1\dot{\theta}_1^2 c_2 + gs_{12} \\ l_1\ddot{\theta}_1 c_2 + l_1\dot{\theta}_1^2 s_2 + gc_{12} \\ 0 \end{bmatrix}
$$

$$
{}^1\dot{v}_{C_2} = \begin{bmatrix} 0 \\ l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix} + \begin{bmatrix} -l_2(\dot{\theta}_1 + \dot{\theta}_2)^2 \\ 0 \\ 0 \end{bmatrix}
$$

$$
+ \begin{bmatrix} l_1\ddot{\theta}_1 s_2 - l_1\dot{\theta}_1^2 c_2 + gs_{12} \\ l_1\ddot{\theta}_1 c_2 + l_1\dot{\theta}_1^2 s_2 + gc_{12} \\ 0 \end{bmatrix}
$$

$$
{}^2F_2 = \begin{bmatrix} m_2 l_1\ddot{\theta}_1 s_2 - m_2 l_1\dot{\theta}_1^2 c_2 + m_2 gs_{12} - m_2 l_2(\dot{\theta}_1 + \dot{\theta}_2)^2 \\ m_2 l_1\ddot{\theta}_1 c_2 + m_2 l_1\dot{\theta}_1^2 s_2 + m_2 gc_{12} + m_2 l_2(\ddot{\theta}_1 + \ddot{\theta}_2) \\ 0 \end{bmatrix}
$$

$$
{}^2N_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
$$

然后，运用 § 2.6 中的式 (2-106)、(2-107) 和式 (2-108)，由连杆 2 到连杆 1 进行迭代计算。对于连杆 2，有

$$
{}^2f_2 = {}^2F_2
$$

$$
{}^2n_2 = \begin{bmatrix} 0 \\ 0 \\ m_2 l_1 l_2 c_2\ddot{\theta}_1 + m_2 l_1 l_2 s_2\dot{\theta}_1^2 + m_2 l_2 gc_{12} + m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) \end{bmatrix}
$$

对于连杆 1，有

$$
{}^1f_1 = \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$\times\begin{bmatrix} m_2l_1s_2\ddot{\theta}_1 - m_2l_1c_2\dot{\theta}_1^2 + m_2gs_{12} - m_2l_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ m_2l_1c_2\ddot{\theta}_1 + m_2l_1s_2\dot{\theta}_1 + m_2gc_{12} + m_2l_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ 0 \end{bmatrix}$$

$$+\begin{bmatrix} -m_1l_1\dot{\theta}_1^2 + m_1gs_1 \\ m_1l_1\ddot{\theta}_1 + m_1gc_1 \\ 0 \end{bmatrix}$$

$${}^1\boldsymbol{n}_1 = \begin{bmatrix} 0 \\ 0 \\ m_2l_1l_2c_2\ddot{\theta}_1 + m_2l_1l_2s_2\dot{\theta}_1^2 + m_2l_2gc_{12} + m_2l_2^2(\ddot{\theta}_1+\ddot{\theta}_2) \end{bmatrix}$$

$$+\begin{bmatrix} 0 \\ 0 \\ m_1l_1^2\ddot{\theta}_1 + m_1l_1gc_1 \end{bmatrix}$$

$$+\begin{bmatrix} 0 \\ 0 \\ m_2l_1^2\ddot{\theta}_1 - m_2l_1l_2s_2(\dot{\theta}_1+\dot{\theta}_2)^2 + m_2l_1gs_2s_{12} + m_2l_1l_2c_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ + m_2l_1gc_2c_{12} \end{bmatrix}$$

最后，提取 ${}^1\boldsymbol{n}_1$ 和 ${}^2\boldsymbol{n}_2$ 的 $\hat{\boldsymbol{Z}}$ 分量，得到关节力矩

$$\begin{cases} \tau_1 = m_2l_2^2(\ddot{\theta}_1+\ddot{\theta}_2) + m_2l_1l_2c_2(2\ddot{\theta}_1+\ddot{\theta}_2) + (m_1+m_2)l_1^2\ddot{\theta}_1 \\ \qquad - m_2l_1l_2s_2\dot{\theta}_2^2 - 2m_2l_1l_2s_2\dot{\theta}_1\dot{\theta}_2 + m_2l_2gc_{12} \\ \qquad + (m_1+m_2)l_1gc_1 \\ \tau_2 = m_2l_1l_2c_2\ddot{\theta}_1 + m_2l_1l_2s_2\dot{\theta}_1^2 + m_2l_2gc_{12} + m_2l_2^2(\ddot{\theta}_1+\ddot{\theta}_2) \end{cases} \tag{2-110}$$

式(2-110)清楚地表示了关节力矩与关节转角、速度和加速度之间的函数关系，即为平面机械手的封闭形式动力学方程。

对于上述例子，如果我们令

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}, \qquad \boldsymbol{\Theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$M(\Theta)=\begin{bmatrix} m_1l_1^2+m_2(l_1^2+l_2^2+2l_1l_2c_2) & m_2l_2^2+m_2l_1l_2c_2 \\ m_2l_2^2+m_2l_1l_2c_2 & m_2l_2^2 \end{bmatrix} \tag{2-111}$$

$$V(\Theta,\dot{\Theta})=\begin{bmatrix} -m_2l_1l_2s_2\dot{\theta}_2^2-2m_2l_1l_2s_2\dot{\theta}_1\dot{\theta}_2 \\ m_2l_1l_2s_2\dot{\theta}_1^2 \end{bmatrix} \tag{2-112}$$

$$G(\Theta)=\begin{bmatrix} m_2l_2gc_{12}+(m_1+m_2)l_1gc_1 \\ m_2l_2gc_{12} \end{bmatrix} \tag{2-113}$$

那么式 (2-110) 可以改写为

$$\tau=M(\Theta)\ddot{\Theta}+V(\Theta,\dot{\Theta})+G(\Theta) \tag{2-114}$$

上式表示了机械手的封闭形式动力学方程的一般结构，对于 n 个自由度的机械手，$M(\Theta)$ 为 $n\times n$ 矩阵，$V(\Theta,\dot{\Theta})$ 和 $G(\Theta)$ 都为 $n\times 1$ 向量，Θ 为表示旋转关节或柱关节位移的 $n\times 1$ 向量。

二、动力学方程的物理解释

式 (2-114) 所表示的动力学方程中的各项都有明确的物理意义，下面结合例 2.6 加以说明。

$G(\Theta)$ 表示了重力效应。由式 (2-113) 可以看出，$G(\Theta)$ 的两个分量分别为连杆质量 m_1、m_2 对于它们各自关节轴的力矩。$G(\Theta)$ 与机械手的形态有关，当手臂沿 X 轴完全伸开时，$\theta_2=0$，c_{12} 有最大值，从而重力矩最大。

对于 $M(\Theta)$ 和 $V(\Theta,\dot{\Theta})$ 的意义分析，可忽略重力效应。

$M(\Theta)\ddot{\Theta}$ 表示了惯性力矩，$M(\Theta)$ 因而称为惯性矩阵。例如式 (2-111) 中，元素 (1, 1) 表示了假定关节 2 "固定" 不动 ($\dot{\theta}_2=0$，$\ddot{\theta}_2=0$) 时，机械手围绕关节 1 的转动惯量，其中第一项 $m_2l_1^2$ 表示连杆 1 相对于关节 1 的转动惯量，而第二项 $m_2(l_1^2+l_2^2+2l_1l_2c_2)$ 则表示关节 2 相对于关节 1 的转动惯量；类似地，式 (2-111) 的元素 (2, 2) 表示了假定关节 1 "固定" 不动 ($\dot{\theta}_1=0$，$\ddot{\theta}_1=0$) 时，机械手围绕关节 2 的转动惯量。当 $\dot{\theta}_1=\dot{\theta}_2=0$、$\ddot{\theta}_1=0$ 时，连杆 2

的加速运动将对关节 1 产生耦合作用力矩，可以证明，式 (2-111) 的元素 (1，2) 表示了与这一耦合力矩相应的转动惯量，而且，元素 (2，1) 与元素 (1，2) 相等。

$\boldsymbol{V}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})$ 称为离心和哥氏项。例如式 (2-112) 的第一个元素中，第一项 $-m_2 l_1 l_2 s_2 \dot{\theta}_2^2$ 与关节 2 的速度成正比，它表示了假定关节 1 "固定"不动 ($\dot{\theta}_1 = 0$，$\ddot{\theta}_1 = 0$)、关节 2 以恒速 $\dot{\theta}_2$ 旋转时，所产生的离心力对关节 1 的力矩；类似地，式 (2-112) 的第二个元素 $-m_2 l_1 l_2 s_2 \dot{\theta}_1^2$ 则表示了假定关节 2 "固定" 不动 ($\dot{\theta}_2 = 0$，$\ddot{\theta}_2 = 0$)、关节 1 以恒速 $\dot{\theta}_1$ 旋转时所产生的离心力对关节 2 的力矩。式 (2-112) 的第一个元素中的第二项 $-2m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2$ 与关节 1、2 的速度乘积成正比，按照哥氏定理，它表示了 m_2 受到的哥氏力对于关节 1 产生的力矩；由于哥氏力的方向与连杆 2 一致，因而对关节 2 不产生力矩，式 (2-112) 的第二个元素中没有相应的项。

§ 2.8 直角坐标动力学方程

上一节推导的动力学方程

$$\boldsymbol{\tau} = M(\boldsymbol{\Theta})\ddot{\boldsymbol{\Theta}} + \boldsymbol{V}(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}) + \boldsymbol{G}(\boldsymbol{\Theta}) \qquad (2\text{-}115)$$

实际上是动力学方程在机械手关节空间中的表示形式，这是因为它所描述的是关节力矩与关节位置变量及其导数的函数关系，它与机械手的结构特征密切相关。

在以后讨论机械手的控制问题时，我们还希望借助于直角坐标把机械手的动力学方程表示成类似于关节空间中的形式：

$$\boldsymbol{\mathscr{F}} = M_x(\boldsymbol{\Theta})\ddot{\boldsymbol{\chi}} + \boldsymbol{V}_x(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}}) + \boldsymbol{G}_x(\boldsymbol{\Theta}) \qquad (2\text{-}116)$$

其中，$\boldsymbol{\chi}$ 为机械手末端位置、姿态的直角坐标向量，$M_x(\boldsymbol{\Theta})$、$\boldsymbol{V}_x(\boldsymbol{\Theta}, \dot{\boldsymbol{\Theta}})$、$\boldsymbol{G}_x(\boldsymbol{\Theta})$ 分别为相对于直角坐标空间的惯性矩阵、离心和哥氏项向量、重力项向量。而 $\boldsymbol{\mathscr{F}}$ 表示作用在机械手末端的直角坐标力/力矩向量，我们在 § 2.5 中已经指明[式 (2-82)] 它与

作用在关节上的力/力矩之间的关系，即

$$\tau = J^T(\Theta)\mathcal{F} \tag{2-117}$$

式中的雅可比矩阵 $J(\Theta)$ 与 $\mathcal{F}$ 和 $\mathcal{X}$ 在相同的坐标系中描述（通常为工具坐标系 $\{T\}$）。

一个标量质量 m 具有的动能为 $\frac{1}{2}mv^2$，机械手的动能可以类似地表示为一个二次型

$$\frac{1}{2}\dot{\Theta}^T M_x(\Theta)\dot{\Theta} \tag{2-118}$$

在直角坐标空间中描述上式，为

$$\frac{1}{2}\dot{\mathcal{X}}^T M_x(\Theta)\dot{\mathcal{X}} \tag{2-119}$$

令式 (2-118) 和式 (2-119) 相等，并利用我们在 § 2.4 中给出的关系式(式 (2-69))

$$\dot{\mathcal{X}} = J(\Theta)\dot{\Theta} \tag{2-120}$$

经过推导可得

$$M_x(\Theta) = [J^T(\Theta)]^{-1}M(\Theta)[J(\Theta)]^{-1} \tag{2-121}$$

其余的直角坐标向量也可以类似地表示为关节空间的向量与雅可比矩阵的组合，我们不加证明地列出结果表达式

$$V_x(\Theta,\dot{\Theta}) = [J^T(\Theta)]^{-1}\{V(\Theta,\dot{\Theta}) - M(\Theta)[J(\Theta)]^{-1}\dot{J}(\Theta)\dot{\Theta}\} \tag{2-122}$$

$$G_x(\Theta) = [J^T(\Theta)]^{-1}G(\Theta) \tag{2-123}$$

第三章　运动轨迹生成

本章讨论多维空间中机械手运动轨迹的生成方法。所谓轨迹，即指机械手的每一个自由度在运动过程中每时每刻的位置、速度和加速度；所谓轨迹生成，即根据这些轨迹参数，计算出期望的运动轨迹。

轨迹生成涉及到下列三个问题。

首先，要描述机械手的空间运动。写出描述作业空间和时间的复杂函数并非容易，应该允许用户用比较简单的方式描述机械手的运动，而复杂的细节问题则由计算机系统来解决。例如，用户可以只给出手爪的目标位置和姿态，让系统由此确定到达目标的途经点、持续时间、速度等轨迹参数。确定轨迹参数的过程称之为轨迹的规划。

其次，根据所确定的轨迹参数，在计算机内部描述期望的轨迹。这主要是选择习惯规定以及合理的软件数据结构问题。

最后，对内部描述的轨迹要进行实际计算，即根据位置、速度和加速度生成出整个轨迹。计算是实时进行的，每一个轨迹点的计算速度称为轨迹更新速率，在典型的机械手系统中，这一速率在20Hz到200Hz之间。

§3.1　轨迹描述和生成的一般考虑

一般，机械手的运动可以看作是工具坐标系$\{T\}$相对于基坐

标系 $\{S\}$ 的运动。这不但符合机械手使用者考虑问题的角度，而且有利于描述、生成机械手的运动轨迹。

用工具坐标系相对于基坐标系的运动描述轨迹，是一种通用的描述方法。运动的描述与特定的机械手、手爪或工具无关，即，这种描述方法既适用于不同的机械手，也适用于同一个机械手拿着不同尺寸的工具的情况，此外，还可以描述和规划相对于移动工作台(例如传送带)的运动轨迹，这时，基坐标系的位置将随着时间变化。

如图 3.1 所示，要使机械手从起始状态移动到某个预期的终止状态，实际上可以看作是把工具坐标系从当前的起始值 $\{T_{起始}\}$ 变化为终止值 $\{T_{终止}\}$。一般情况下，这种变化不但包括工具坐标系的位置，而且包括坐标系的姿态。以下，我们将使用"点"这个词来表示机械手的状态，或工具坐标系的位置及姿态，例如起始点、终止点等。

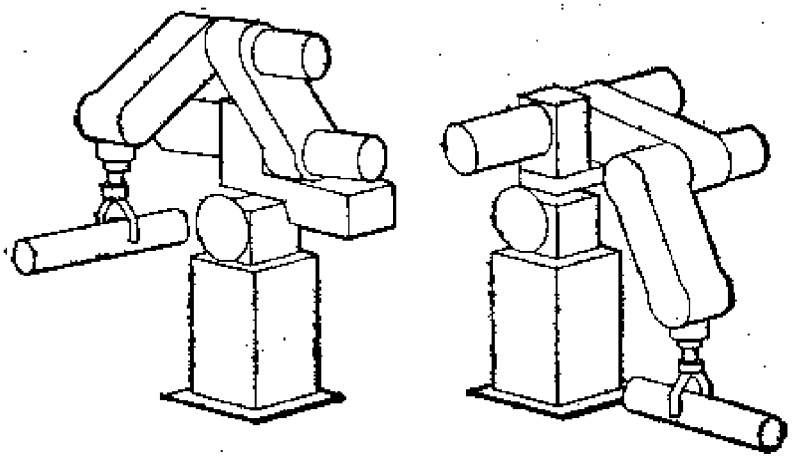


图 3.1　机械手的初始位置和目标位置

在需要更详细地描述运动时，不仅要指出期望的机械手终止点，而且要给出介于起始点与终止点之间的中间点，或称途经点。相应地，工具坐标系必然存在一组由途经点决定的中间值 $\{T_i\}$。所有途经点以及起始点和终止点总称为轨迹点。运动轨迹除了空

间约束外，还存在着时间分配问题。例如，在规定路径的同时，还必须给出两个途经点之间的运动时间。

显然，机械手的运动应当平稳，不平稳的运动将造成机械部件的迅速磨损，并导致机械手的振动。为了保证提供一个平滑的轨迹，必须选择连续的描述运动轨迹的函数，而且描述函数的一阶导数（速度），有时甚至二阶导数（加速度）也应当是连续的。

以下分别介绍两种运动轨迹的规划和生成方法。

§3.2 关节空间法

关节空间法借助于机械手关节（不失一般性，以后的讨论中，都假设为旋转关节）角度的函数来描述运动轨迹。

上一节指出，轨迹点通常用工具坐标系 $\{T\}$ 相对于基坐标 $\{S\}$ 的位置和姿态来表示。在每一个途经点上，根据机械手运动学的逆运算可以得到一组关节的角度，如果能为每一个关节找到一个平滑函数，就可以描述整个机械手从起始点，通过途经点，最后到达终止点的运动轨迹。只要在任何一段轨迹内每一个关节的运动时间都相同，那么所有的关节都将同时到达途经点和终止点。相应地，工具坐标系在每个途经点和终止点上，也就具有了预期的直角坐标位置和姿态。尽管每个关节的运动时间相同，但关节角度函数之间是相互独立的。

关节空间法不必在直角坐标系中描述两个途经点之间的轨迹形状，计算过程简单、容易，而且由于关节空间与直角坐标空间之间并不存在连续的对应关系，因此基本上不会发生机构的奇异点问题。

在关节空间法中，可以选取不同类型的关节角度函数，因而生成不同的轨迹。

一、三次多项式插值

在机械手的运动过程中，由于相应于起始点的关节角度 θ_0 是已知的，而终止点的关节角度 θ_f 也可以通过逆运动学的解得到，因此，运动轨迹的描述，实际是要选取一个平滑函数 $\theta(t)$，把它插补在起始点关节角度值与终止点关节角度值之间，连成一段平滑的运动轨迹(参看图 3.2)。

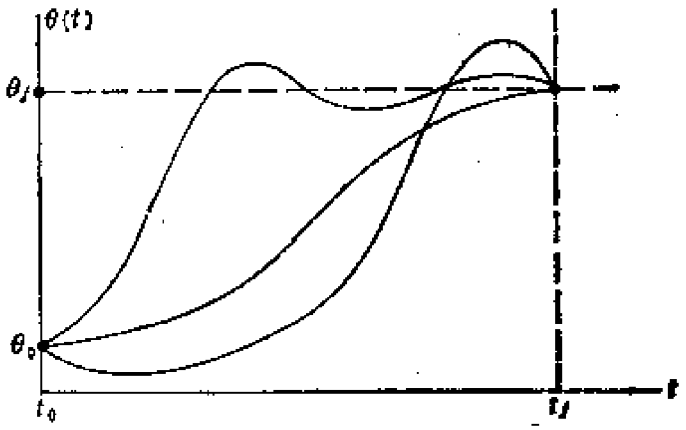


图 3.2　单个关节的轨迹形状

为实现单个关节的平稳运动，$\theta(t)$ 至少需要四个约束条件。其中两个约束条件是关节的起始值和终点值：

$$\begin{cases}\theta(0)=\theta_0\\ \theta(t_f)=\theta_f\end{cases} \tag{3-1}$$

为满足关节运动速度的连续性要求，另外还需要两个约束条件。在当前的简单情况下，只需规定起始点的速度和终止点速度都为零，即

$$\begin{cases}\dot{\theta}(0)=0\\ \dot{\theta}(t_f)=0\end{cases} \tag{3-2}$$

式 (3-1) 和 (3-2) 给出的四个约束条件唯一地确定了一个三次多项式：

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3-3}$$

运动轨迹上关节的速度和加速度则为

$$\begin{cases} \dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \\ \ddot{\theta}(t) = 2a_2 + 6a_3 t \end{cases} \tag{3-4}$$

式 (3-3) 和 (3-4) 与四个约束条件联立，产生四个关于 a_0、a_1、a_2、a_3 的方程：

$$\begin{cases} \theta_0 = a_0 \\ \theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ 0 = a_1 \\ 0 = a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{cases} \tag{3-5}$$

求解上述方程组可得

$$\begin{cases} a_0 = \theta_0 \\ a_1 = 0 \\ a_2 = \dfrac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 = -\dfrac{2}{t_f^3}(\theta_f - \theta_0) \end{cases} \tag{3-6}$$

需要再次指出：这组解只适用于关节起始、终止速度为零的运动情况。

例 3.1 设只有一个旋转关节的单链机械手处于静止状态时，$\theta_0 = 15°$，要求它在 3 s 之内平稳地达到 $\theta_f = 75°$ 的终止位置，从而使机械手运动到速度为零的终点。

把 $\theta(0)$、$\theta(t_f)$ 代入式 (3-6)，可以求出三次多项式的系数：

$$a_0 = 15.0,\quad a_1 = 0.0,\quad a_2 = 20.0,\quad a_3 = -4.44$$

由式 (3-3) 和 (3-4)，则可确定机械手的位置、速度和加速度，即有

$$\theta(t) = 15.0 + 20.0t^2 - 4.44t^3$$

$$\dot{\theta}(t) = 40.0t - 13.33t^2$$

$$\theta(t) = 40.0 - 26.66t$$

图 3.3 表示了描述运动轨迹的曲线。显然，任何三次函数的速度曲线均为抛物线，相应的加速度曲线均为直线。

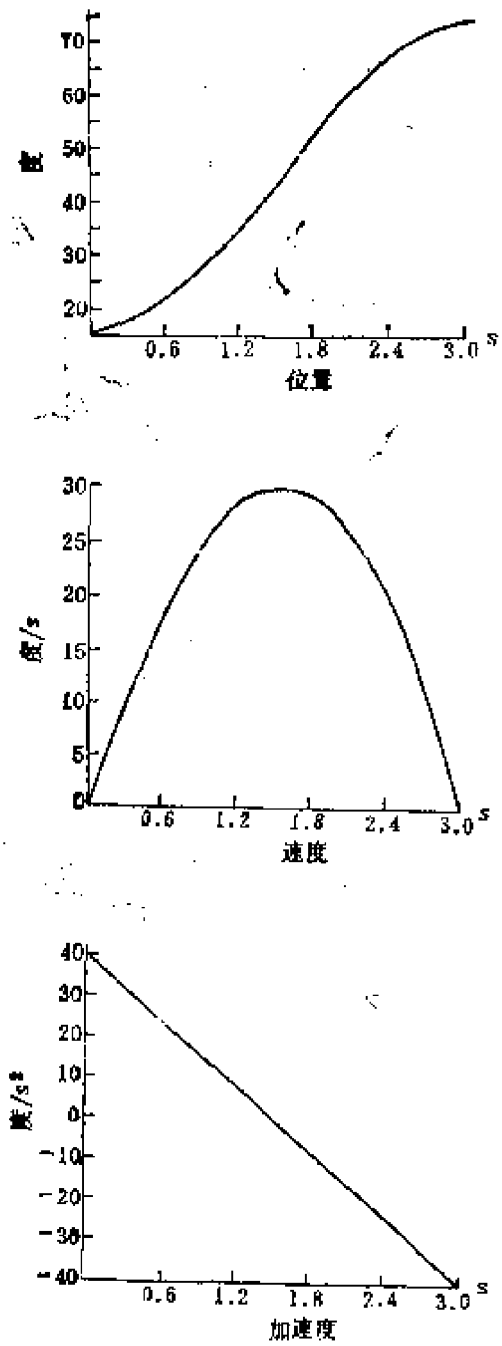


图 3.3 三次多项式轨迹

确定三次多项式的四个方程为

二、包括途经点的三次多项式插值

一般情况下，我们希望规划包含了途经点的轨迹。如果机械手在途经点停留，可以直接使用前面讲的三次多项式的解，如果途经点上确实只是"一经而过"，则有必要推广上述方法。

推广过程是显然的，只要把所有的途经点也看作是一些"终止点"或"起始点"，求解逆运动学，得到一组关节角度值，然后就可以确定相应的三次多项式，从而把途经点平滑地联接起来。但是，在这些"终止点"或"初始点"上关节运动的速度不再是零。

途经点上关节速度设为已知，则约束式 (3-2) 变为

$$\begin{cases}\dot{\theta}(0) = \dot{\theta}_0 \\ \dot{\theta}(t_f) = \dot{\theta}_f\end{cases} \qquad (3\text{-}7)$$

$$\begin{cases} \theta_0 = a_0 \\ \theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ \dot{\theta}_0 = a_1 \\ \dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{cases} \tag{3-8}$$

对以上方程组求 a_i，可得

$$\begin{cases} a_0 = \theta_0 \\ a_1 = \dot{\theta}_0 \\ a_2 = \dfrac{3}{t_f^2}(\theta_f - \theta_0) - \dfrac{2}{t_f}\dot{\theta}_0 - \dfrac{1}{t_f}\dot{\theta}_f \\ a_3 = -\dfrac{2}{t_f^3}(\theta_f - \theta_0) + \dfrac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0) \end{cases} \tag{3-9}$$

实际上，由上式确定的三次多项式描述了起始点和终止点具有任意位置和速度的运动轨迹。

根据式（3-9）求相邻途经点之间的三次多项式，必须已知途经点的速度。有以下三种方法可以用来生成通过途经点时的关节速度：

（1）根据工具坐标系在直角坐标系中的瞬时线速度和角速度来确定每个途经点的速度。

（2）在直角坐标空间或者关节空间采用适当的启发式方法，由控制系统自动地选择途经点的速度。

（3）按照保证每个途经点的加速度连续的原则，由控制系统自动地选择途经点的速度。

对于方法（1），可使用在途经点求出的机械手雅可比的逆，把直角坐标系中该点的速度"映射"为期望的关节速度。当然，如果机械手的某个途经点是一个奇异点，这时就不能任意设置速度值。按照方法（1）生成的轨迹虽然能满足用户设置速度的需求，但是逐点设置速度值毕竟需要很大的工作量。所以，较好的机器人控制系统应当具有方法（2）或（3）的功能，或者二者兼而有之。

对于方法 (2)，图 3.4 例示了一种利用启发式信息选择途经点速度的过程。图中，θ_0 为初始点，θ_D 为终止点，θ_A、θ_B 和 θ_C 为途经点，细实线表示关节运动到途经点时的速度。这里所用的启发式信息从概念到计算方法都很简单，即，假设用直线段把这些途经点连接起来，如果这些线段的斜率在途经点处改变符号，就把速度选定为零；如果这些线段的斜率不改变符号，则选取途经点两侧的线段斜率的平均值作为该点的速度。因此，只要给出了期望的途经点，系统就能用这种方法自动地生成相应的速度。

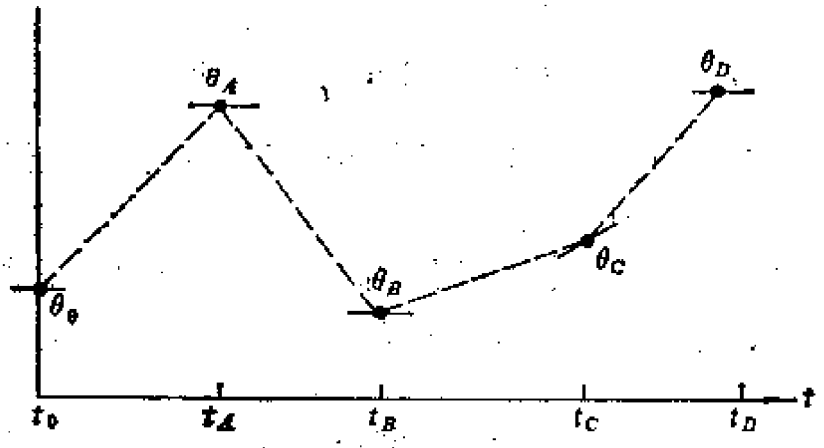


图 3.4　途经点的启发式选择

对于方法 (3)，为了保证途经点处的加速度连续，可以设法用两条三次曲线在途经点处联接起来，拼凑成期望的轨迹。拼凑的约束条件是：联接处不但速度连续，而且要求加速度也连续，下面具体地说明这种方法。

设所考虑的途经点处关节角度为 θ_v，与该点相邻的前后两点的关节角度分别为 θ_0，θ_E。相应于从 θ_0 运动到 θ_v 的三次多项式为

$$\theta(t) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3 \qquad (3\text{-}10)$$

相应于从 θ_v 运动到 θ_E 的三次多项式为

$$\theta(t) = a_{20} + a_{21}t + a_{22}t^2 + a_{23}t^3 \qquad (3\text{-}11)$$

上述两个三次多项式的时间区间分别为 $[0, t_{f1}]$ 和 $[0, t_{f2}]$。

对这两个多项式的约束是

$$\begin{cases} \theta_0 = a_{10} \\ \theta_v = a_{10} + a_{11}t_{f1} + a_{12}t_{f1}^2 + a_{13}t_{f1}^3 \\ \theta_v = a_{20} \\ \theta_g = a_{20} + a_{21}t_{f2} + a_{22}t_{f2}^2 + a_{23}t_{f2}^3 \\ 0 = a_{11} \\ 0 = a_{21} + 2a_{22}t_{f2} + 3a_{23}t_{f2}^2 \\ a_{11} + 2a_{12}t_{f1} + 3a_{13}t_{f1}^2 = a_{21} \\ 2a_{12} + 6a_{13}t_{f1} = 2a_{22} \end{cases} \tag{3-12}$$

这些约束构成了有八个未知数的八个线性方程。令 $t_f = t_{f1} = t_{f2}$，解这个方程组可以得到

$$\begin{cases} a_{10} = \theta_0 \\ a_{11} = 0 \\ a_{12} = \dfrac{12\theta_v - 3\theta_g - 9\theta_0}{4t_f^2} \\ a_{13} = \dfrac{-8\theta_v + 3\theta_g + 5\theta_0}{4t_f^3} \\ a_{20} = \theta_v \\ a_{21} = \dfrac{3\theta_g - 3\theta_0}{4t_f} \\ a_{22} = \dfrac{-12\theta_v + 6\theta_g + 6\theta_0}{4t_f^2} \\ a_{23} = \dfrac{8\theta_v - 5\theta_g - 3\theta_0}{4t_f^3} \end{cases} \tag{3-13}$$

一般情况下，一个完整的轨迹由多个三次多项式表示，约束条件(关节经过途经点时加速度连续)构成的方程可以表示成矩阵的形式，用这矩阵来求出途经点的速度。

三、高阶多项式插值

关节运动的轨迹段往往需要用高于三阶的多项式表示。例如，对于某段轨迹的起始点和终止点都规定了关节的位置、速度和加速度，则需要一个五次多项式：

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \qquad (3\text{-}14)$$

其中，要求满足的约束为

$$\begin{cases} \theta_0 = a_0 \\ \theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\ \dot{\theta}_0 = a_1 \\ \dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\ \ddot{\theta}_0 = 2a_2 \\ \ddot{\theta}_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \end{cases} \qquad (3\text{-}15)$$

这六个方程构成了具有六个变量的线性方程组，它们的解为：

$$\begin{cases} a_0 = \theta_0 \\ a_1 = \dot{\theta}_0 \\ a_2 = \dfrac{\ddot{\theta}_0}{2} \\ a_3 = \dfrac{20\theta_f - 20\theta_0 - (8\dot{\theta}_f + 12\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^3} \\ a_4 = \dfrac{30\theta_0 - 30\theta_f + (14\dot{\theta}_f + 16\dot{\theta}_0)t_f + (3\ddot{\theta}_0 - 2\ddot{\theta}_f)t_f^2}{2t_f^4} \\ a_5 = \dfrac{12\theta_f - 12\theta_0 - (6\dot{\theta}_f + 6\dot{\theta}_0)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^5} \end{cases} \qquad (3\text{-}16)$$

四、用抛物线过渡的线性插值

也可以选择轨迹的形状为线性函数。即，当关节由初始点运动到终止点时，只使用线性插值(图 3.5)。值得指出，在这种情况

下尽管每个关节都作线性运动，而手爪的运动轨迹则一般不是线性的。

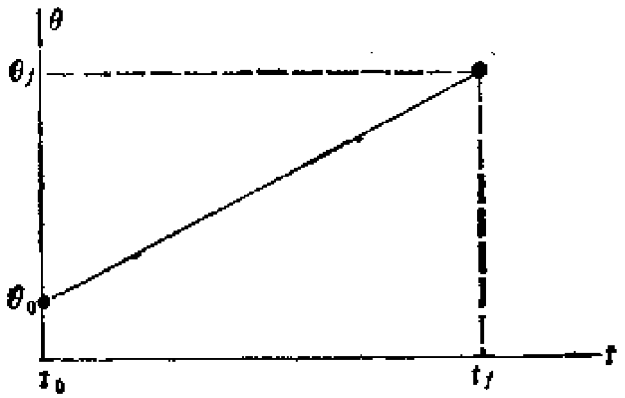


图 3.5　线性插值需要无穷大加速度

显然，简单的线性插值将导致关节在运动的起点和终点的速度不连续。为了生成一条位置和速度都连续的平滑轨迹，在使用线性插值时，把每一个轨迹点的某个邻域内的轨迹设计成相同的抛物线。由于抛物线对于时间的二阶导数为常数，即相应区间内的加速度恒定不变，这样就可以使速度平滑地变化，因而整个轨迹上的位置和速度都是连续的。图 3.6 表示了用这种方法设计的一条简单轨迹。线性函数与两个抛物线函数平滑地衔接在一起形成的轨迹，称为带有抛物线过渡域的线性轨迹。

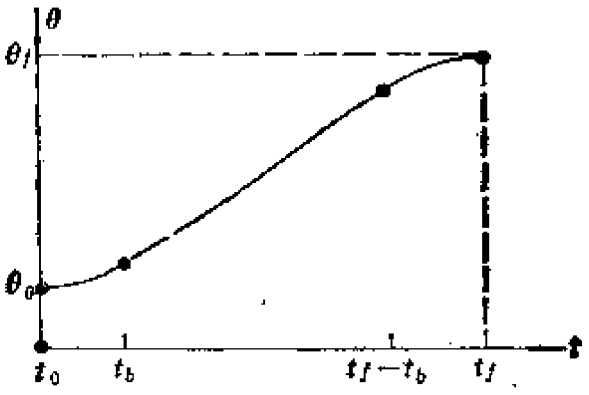


图 3.6　带有抛物线过渡域的线性轨迹

为了构成单一的轨迹，假设两个过渡域都具有相同的持续时间，因此在这两个域中采用相同的加速度值(冠以正负号)．正如图 3.7 所示，这样设计的轨迹并不唯一，但是需要注意，每一个结果都对称于时间中点 t_h 和位置中点 θ_h。

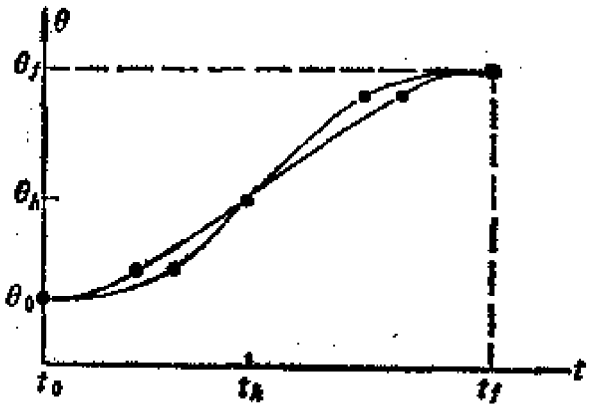


图 3.7　带有抛物线过渡域的线性轨迹的对称性

由于过渡域 $[t_0, t_b]$ 的终点速度必须等于线性域的速度，所以

$$\ddot{\theta} t_b = \frac{\theta_h - \theta_b}{t_h - t_b} \tag{3-17}$$

式中，θ_b 为过渡域终点 t_b 处的关节角度，$\ddot{\theta}$ 为过渡域的加速度，θ_b 的值可以按下式解得：

$$\theta_b = \theta_0 + \frac{1}{2}\ddot{\theta} t_b^2 \tag{3-18}$$

令 $t = 2t_h$，据式 (3-17) 和 (3-18) 可得

$$\ddot{\theta} t_b^2 - \ddot{\theta} t t_b + (\theta_f - \theta_0) = 0 \tag{3-19}$$

其中，t 为所期望的运动持续时间。

到此，对已知的任意 θ_f、θ_0 和 t，可以选择满足式 (3-19) 的 $\ddot{\theta}$ 和 t_b 来获得相应的轨迹。通常的作法是先选择加速度 θ 的值，然后用式 (3-19) 求出相应的 t_b，即

$$t_b = \frac{t}{2} - \frac{\sqrt{\ddot{\theta}^2 t^2 - 4\ddot{\theta}(\theta_f - \theta_0)}}{2\ddot{\theta}} \tag{3-20}$$

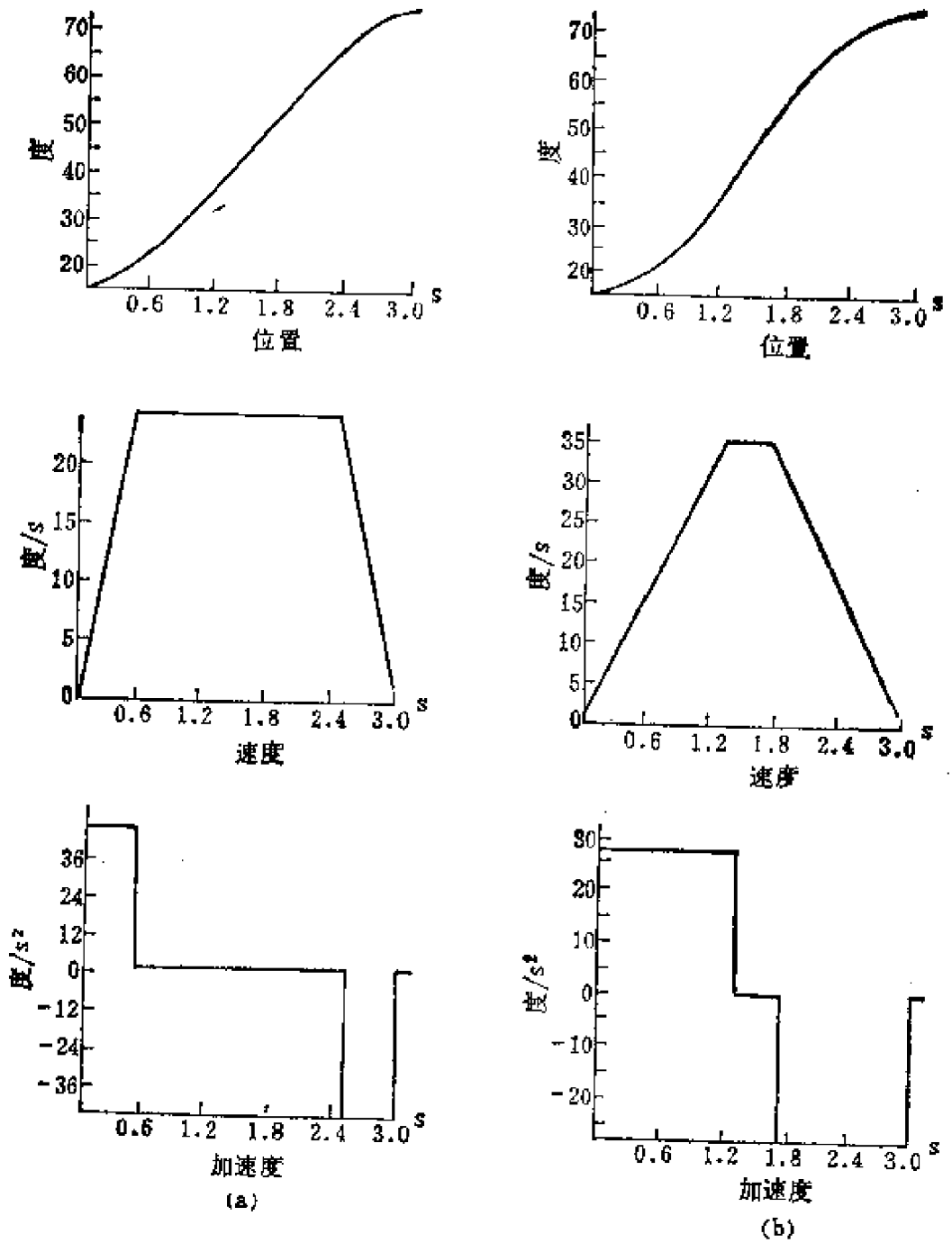


图 3.8　带有抛物线过渡域的线性轨迹的比较

由上式可知，为保证 t_b 有解，过渡域加速度 θ 必须选得足够大，即须

$$\ddot{\theta} \geqslant \frac{4(\theta_f - \theta_0)}{t^2} \tag{3-21}$$

当式 (3-21) 中等号成立时，线性域的长度缩减为零，整个轨迹由两个过渡域构成，这两个过渡域在衔接处的斜率相等。当加速度的选值越来越大时，过渡域的长度会越来越短。如果加速度选为无穷大，轨迹又返回到简单的线性插值的情况。

例 3.2 对于例 3.1 给出的 θ_0、θ_f 和 t，设计出两条带有抛物线过渡域的线性轨迹。

图 3.8a 表示了一种可能，其中加速度 $\ddot{\theta}$ 选得相当高。在这种情况下，关节迅速加速，然后转为匀速运动，最后减速。而图 3.8b 所示的轨迹，由于所选的加速度相当小，所以线性域几乎消失了。

五、包括途经点的带有抛物线过渡域的线性轨迹

如图 3.9 所示，某个关节的运动中设有 n 个途经点，其中三个相邻的途经点表示为 j，k 和 l，每两个相邻途经点之间都以线性函数相连，而所有途经点附近都具有过渡域。

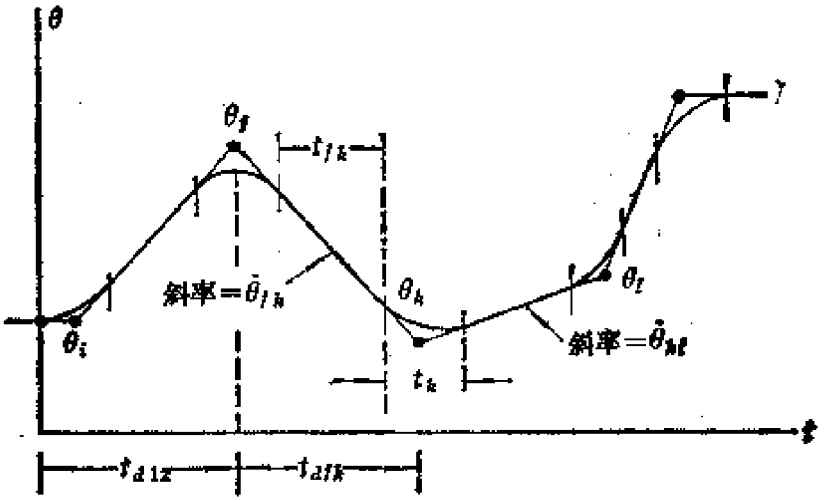


图 3.9 多段带有抛物线过渡域的线性轨迹

图中，在 k 点的过渡域的持续时间为 t_k，点 j 和点 k 之间线性域的持续时间为 t_{jk}，连接 j 与 k 点的轨迹的全部持续时间为

t_{dik}。另外，i 与 k 之间线性域的速度为 $\dot{\theta}_{ik}$，i 点过渡域的加速度为 $\ddot{\theta}_i$。现在的问题是在具有途经点的情况下，如何确定带有抛物线过渡域的线性轨迹。

与求两点之间的轨迹的情况一样，这个问题有很多解，这些解是由每个过渡域的加速度值决定的。给定任意轨迹点的位置 θ_k，期望的持续时间 t_{dik} 以及加速度的绝对值 $|\ddot{\theta}_k|$，我们可以计算出过渡域的持续时间 t_k。对于那些内部的轨迹点（j，$k \neq 1$，2；i，$k \neq n-1$，n），根据下述方程求解：

$$\begin{cases} \dot{\theta}_{ik} = \dfrac{\theta_k - \theta_j}{t_{dik}} & \text{(a)} \\ \theta_k = \mathrm{SGN}(\dot{\theta}_{kl} - \dot{\theta}_{ik})|\ddot{\theta}_k| & \text{(b)} \\ t_k = \dfrac{\dot{\theta}_{kl} - \dot{\theta}_{ik}}{\theta_k} & \text{(c)} \\ t_{ik} = t_{dik} - \dfrac{1}{2}t_j - \dfrac{1}{2}t_k & \text{(d)} \end{cases} \tag{3-22}$$

第一个轨迹段和最后一个轨迹段的处理与式（3-22）略有不同，因为在轨迹端部的整个过渡域都必须计入轨迹段的持续时间。

对于第一段，令线性域速度的两个表达式相等，就可求出 t_1，即

$$\frac{\theta_2 - \theta_1}{t_{d12} - \dfrac{1}{2}t_1} = \ddot{\theta}_1 t_1 \tag{3-23}$$

用上式可以算出起始点过渡域的时间 t_1，进而求出 $\dot{\theta}_{12}$ 和 t_{12}：

$$\begin{cases} \theta_1 = \mathrm{SGN}(\theta_2 - \theta_1)|\ddot{\theta}_1| & \text{(a)} \\ t_1 = t_{d12} - \sqrt{t_{d12}^2 - \dfrac{2(\theta_2 - \theta_1)}{\theta_1}} & \text{(b)} \\ \dot{\theta}_{12} = \dfrac{\theta_2 - \theta_1}{t_{d12} - \dfrac{1}{2}t_1} & \text{(c)} \\ t_{12} = t_{d12} - t_1 - \dfrac{1}{2}t_2 & \text{(d)} \end{cases} \tag{3-24}$$

对于连结途经点 $n-1$ 与终止点 n 的最后一段，情况与第一段类似，即有

$$\frac{\theta_{n-1}-\theta_n}{t_{d(n-1)n}-\frac{1}{2}t_n}=\ddot{\theta}_n t_n \tag{3-25}$$

根据上式又可以求出

$$\begin{cases}\theta_n=\mathrm{SGN}(\theta_{n-1}-\theta_n)|\ddot{\theta}_n| \\ t_n=t_{d(n-1)n}-\sqrt{t_{d(n-1)n}^2+\dfrac{2(\theta_n-\theta_{n-1})}{\ddot{\theta}_n}} \\ \dot{\theta}_{(n-1)n}=\dfrac{\theta_n-\theta_{n-1}}{t_{d(n-1)n}-\dfrac{1}{2}t_n} \\ t_{(n-1)n}=t_{d(n-1)n}-t_n-\dfrac{1}{2}t_{n-1}\end{cases} \tag{3-26}$$

使用式 (3-22) 到 (3-26)，就可以求出多段轨迹中各个过渡域的时间和速度．通常，用户仅仅给定途经点以及各段所期望的持续时间，在这种情况下，系统使用每个关节的加速度的默认值．如果为了更省事，还可以使系统根据默认的速度值来计算持续时间．对于所有的过渡域，加速度值必须足够大，以便各段有足够长的线性域．

例 3.3 设关节的轨迹点分别为 10°，35°，25° 和 10°，三个轨迹段的持续时间分别为 2s，1s 和 3s，各个过渡域默认的加速度绝对值为 50°/s²，各段的速度，过渡域时间以及线性域时间分别计算如下．

从第一段开始，先由式 (3-24a) 确定起始点处调和域的加速度

$$\ddot{\theta}_1=50.0$$

由式 (3-24b) 可以求得这个过渡域的持续时间

$$t_1=2-\sqrt{4-\frac{2(35-10)}{50.0}}=0.27$$

由式（3-24c）可以求出第一个线性域的速度

$$\dot{\theta}_{12} = \frac{35 - 10}{2 - 0.5(0.27)} = 13.50$$

由式（3-22a）求出第二个线性域的速度

$$\dot{\theta}_{25} = \frac{25 - 35}{1} = -10.0$$

再由式（3-22b）确定第二个过渡域的加速度

$$\ddot{\theta}_2 = -50.0$$

然后由式（3-22c）求出这个过渡域的持续时间

$$t_2 = \frac{-10.0 - 13.50}{-50.0} = 0.47$$

由式（3-24d）可以计算第一个线性域时间

$$t_{12} = 2 - 0.27 - \frac{1}{2}(0.47) = 1.50$$

再下一步，由式（3-26a）可得第四个过渡域的加速度

$$\ddot{\theta}_4 = 50.0$$

最后一段的时间可由式（3-26b）得到，

$$t_4 = 3 - \sqrt{9 + \frac{2(10 - 25)}{50.0}} = 0.102$$

第三段线性域的速度 $\dot{\theta}_{34}$，由式（3-26c）得到，

$$\theta_{34} = \frac{10 - 25}{3 - 0.050} = -5.10$$

再用式（3-22b）计算第三个过渡域的加速度

$$\ddot{\theta}_3 = 50.0$$

由式（3-22c）算出这个过渡域的时间

$$t_3 = \frac{-5.10 - (-10.0)}{50} = 0.098$$

最后，由式（3-22d）我们计算出第二、三个线性域的持续时间

$$t_{23} = 1 - \frac{1}{2}(0.47) - \frac{1}{2}(0.098) = 0.716$$

$$t_{34} = 3 - \frac{1}{2}(0.098) - 0.102 = 2.849$$

上述这些计算的结果实际上构成了轨迹的一个"规划". 在执行的时候，控制系统的轨迹生成器将使用这些数据以轨迹更新的速率来求出 θ, $\dot{\theta}$ 和 $\ddot{\theta}$.

值得指出的是，按照这些带有抛物线过渡域的线性轨迹，机械手在运动中并没有经过那些途经点，即使加速度足够大，机械手的实际轨迹也只是充分接近途经点. 如果希望机械手的轨迹一定要经过某个途经点，那么只要将整个轨迹分成两大段，把这个途经点作为前一段的终止点而同时又是后一段的起始点就可以了.

如果用户希望机械手在运行过程中不但要准确地通过某个途

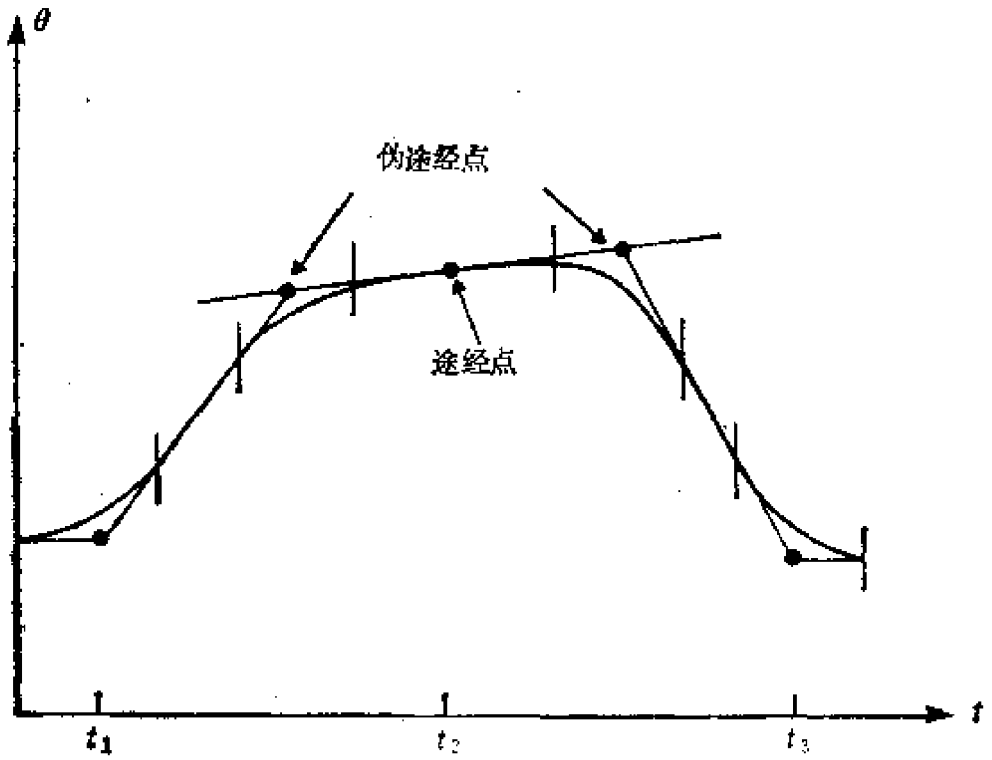


图 3.10 用两个伪途经点造成一个"穿透"点

经点，而且通过时的速度不为零，这时，可以通过在指定途经点两侧设置两个“伪途经点”的方法来满足这一要求。即，使指定途经点位于两个伪途经点的连线上，成为两个相应过渡域之间的线性域上的一点，例如图 3.10 所示。这样，再用前面讲的方法所生成的轨迹势必能以一定的速度“穿透”指定的途经点。穿透速度可由用户确定，也可以由控制系统根据适当的启发信息来选择。

§ 3.3 直角坐标空间法

前面曾经指出，关节空间法生成的轨迹虽然可以保证机械手通过途经点达到终止点，然而手爪在三维空间中运行的实际轨迹形状却非常复杂，它取决于机械手所用的运动学方程。本节考虑这样的轨迹生成方法：轨迹的形状由它在直角坐标系中的位置和姿态来表示。由于位置和姿态是时间的函数，这样就便于对轨迹的空间形状提出一定的设计要求。例如，要求轨迹是直线、圆、正弦曲线或其他规则曲线。上述方法称之为直角坐标空间法。

§ 3.1 中已经指出，机械手的轨迹点通常用工具坐标系相对于基坐标系的位置和姿态来表示，在直角坐标空间法中，连接各个轨迹点的平滑函数不再是关节角度的时间函数，而是直角坐标变量关于时间的函数。这些轨迹可以直接根据指定的、即相对于基坐标系 $\{S\}$ 描述的工具坐标系 $\{T\}$ 的位置和姿态变化来进行规划，而不必一开始就要通过求解逆运动学去计算关节的角度。换句话说，当轨迹在直角坐标空间生成以后，作为最后一步，要按照轨迹更新速度进行逆运动学计算，以便求出相应的关节角度。显然，直角坐标空间法在运行时需要更大的计算量。

直角坐标空间法的实现方案很多，这一节只介绍一种，即利用关节空间法中提出的带有抛物线过渡域的线性函数生成运动轨迹。

一、直角坐标直线运动

手爪在空间的直线运动轨迹比较容易描述．如果在空间的一条直线上选出一组间距非常小的途经点，那么，当手爪沿着这组点运动时，不论在相邻两点间选什么样的平滑函数，从宏观上看，手爪是沿着一条直线在运动．然而，如果在途经点之间的距离设置得很大的时候，手爪仍然能够沿着空间的直线运动，这将给用户带来很大的方便．这种表示轨迹和驱动手臂运动的模式称为直角坐标直线运动．用直线来定义的运动是直角坐标运动功能的一个子集．在更一般的直角坐标运动中，直角坐标对于时间的任意函数都可以用来描述轨迹，例如椭圆、正弦曲线等等．

在规划和生成直角坐标直线轨迹的时候，使用带有抛物线过渡域的线性函数比较合适．在每一轨迹段的线性域中，由于位置的三个元素都按线性方式运动，所以手爪将沿着空间的直线轨迹运动．然而，如果把每一个途经点的姿态都表示成第二章中所说的旋转矩阵，那就不能对它的元素进行线性插值，因为任何旋转矩阵必须由规范化正交列组成，如果在两个矩阵元素间进行线性插值，所得到的旋转矩阵的列就难以保证满足规范化正交的要求．这里，我们将采用一种"角-轴"表示法代替旋转矩阵描述姿态．

设有原点重合的两个坐标系 $\{S\}$ 和 $\{A\}$，力学中已经证明，$\{A\}$ 相对于 $\{S\}$ 的任何姿态都可以通过适当地选择一个单位向量 ${}^S\hat{K}=[k_x, k_y, k_z]^T$，和一个角度 θ 得到表示，即，$\{A\}$ 可以看作是开始与 $\{S\}$ 重合，然后绕 ${}^S\hat{K}$ 按右手规则旋转 θ 角所形成的坐标系（图 3.11）．$\{A\}$ 相对于 $\{S\}$ 的姿态记为 ROT $({}^S\hat{K}, \theta)$ 或 ${}^S_AR({}^S\hat{K}, \theta)$，它实际上也是一种坐标变换，而且与旋转变换是等价的．${}^S\hat{K}$ 称为有限转动的等价轴，由于它是单位向量，$|{}^S\hat{K}|=1$，因而用 k_X, k_Y, k_Z 中任意两个参数就可说明，于是，"角-轴"表示法只需要三个参数（加上角度 θ）就可描述一个坐标系的姿态．

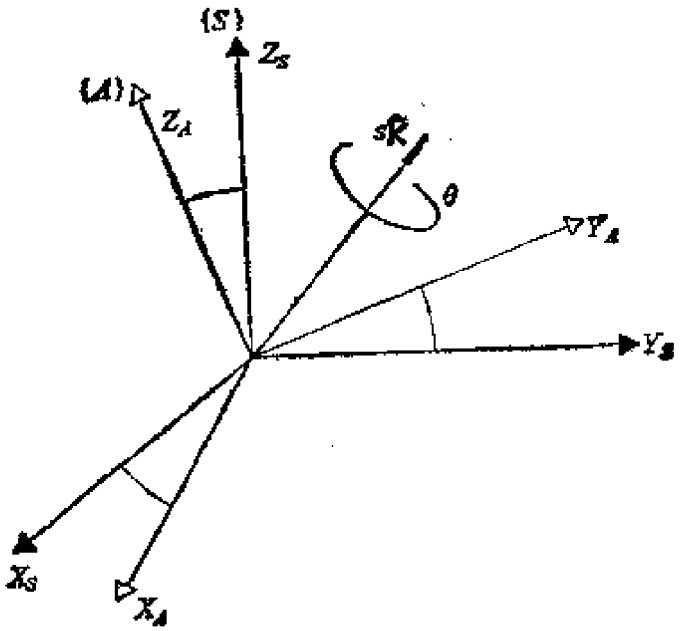


图 3.11 "角-轴"表示法

如果我们把"角-轴"表示法与表示直角坐标位置的 3 × 1 向量结合在一起，就得到一个描述直角坐标位置和姿态的 6 × 1 向量。

考虑一个途经点，它在基坐标系 $\{S\}$ 中表示成 ^S_AT，即，描述途经点的末端坐标系 $\{A\}$，它的位置由原点向量 $^SP_{AO}$ 确定，而它的姿态则由 ^S_AR 确定。把旋转矩阵 ^S_AR 变换成角-轴表达式 ROT $(^S\hat{K}_A, \theta)$，或简写成 ^SK_A。我们把表示途经点在直角坐标系中的位置和姿态的 6 × 1 向量记为 $^S\chi_A$，于是就有

$$^S\chi_A = \begin{bmatrix} ^SP_{AO} \\ ^SK_A \end{bmatrix} \tag{3-27}$$

当每一个轨迹点都用上式表示后，下一步的问题就是选择适当的时间函数，把这六个量从一个轨迹点平滑地移到另一个轨迹点。如果使用具有抛物线过渡域的线性函数，两个轨迹点之间的轨迹形状将是线性的，当经过途经点时，手爪运行的线速度和角速度将平稳变化。

应该注意，角-轴表达式中的旋转角不是唯一的：

$$({}^{S}\boldsymbol{K}_{A}, \theta) = ({}^{S}\boldsymbol{K}_{A}, \theta + n360°) \qquad (3\text{-}28)$$

其中 n 是任意整数。在从途经点 $\{A\}$ 移向途经点 $\{B\}$ 的时候，显然，总的旋转量应该取最小值，即，使它小于 180°。假定 $\{A\}$ 的旋转表达式为 ${}^{S}\boldsymbol{K}_{A}$，必须选择特定的 ${}^{S}\boldsymbol{K}_{B}$，使得 $|{}^{S}\boldsymbol{K}_{B} - {}^{S}\boldsymbol{K}_{A}|$ 最小。例如，图 3.12 画出了 ${}^{S}\boldsymbol{K}_{B}$ 的四种可能及其与给定 ${}^{S}\boldsymbol{K}_{A}$ 的关系。比较各个差向量(虚线)可以决定选取哪一个 ${}^{S}\boldsymbol{K}_{B}$，从而获得最小的旋转量。在图 3.12 的情况下选择结果是 ${}^{S}\boldsymbol{K}_{B(-1)}$。

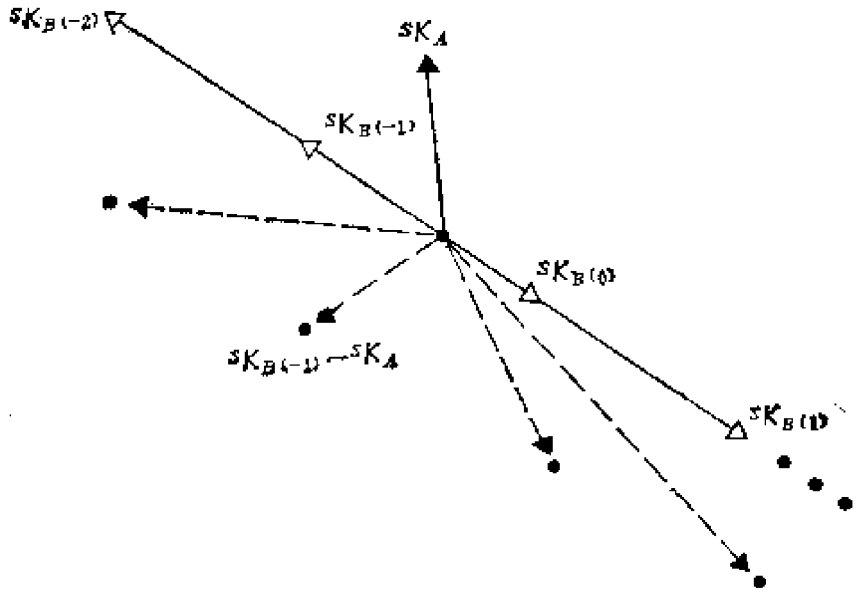


图 3.12　最小旋转量的确定

另外，一旦选择了每个途经点的六个 $\boldsymbol{X}$ 分量，在用 § 3.2 中介绍的数学方法生成具有抛物线过渡域的线性轨迹时，必须再加上一个约束条件：每一个自由度的过渡域时间必须相同。这样才能保证各个自由度构成的复合运动在空间形成一条直线。而所有的过渡时间相同又将导致每个自由度过渡域的加速度不同。因此，我们需要规定过渡域的持续时间，使用式 (3-22c) 计算(而不是用其他方式确定)需要的加速度。过渡时间是可以选择的，所以可

以保证各个自由度的加速度不会超过上界.

二、直角坐标轨迹的几何问题

由于直角坐标空间法在直角坐标空间中的轨迹形状与关节位置之间作出了连续的对应关系,因而,直角坐标轨迹比较容易出现工作空间的问题以及奇异点问题.

虽然机械手的起始点和终止点都在它的工作空间中,但是轨迹上的某些点很可能超出了机械手的工作空间.这就导致有的情况下使用关节空间轨迹很容易运行,而使用直角坐标直线运动轨迹就无法运行了.

另外,如果机械手沿直角坐标直线轨迹朝着机械上的奇异点运动,当接近这个奇异点时,为了保证手爪以恒速运行,某个或某些关节的速度将会趋于无穷大.由于关节的最高速度是有上限的,所以一般情况下这将导致机械手偏离预期的轨迹.解决这个问题的办法之一是减小手爪的运行速度,使每个关节的运行速度都不超过各自的速度范围.这种方式将失去轨迹应有的时间特征,但至少保证了轨迹的空间特性.

鉴于直角坐标空间法存在上述问题,大部分机械手的控制系统都兼有用关节空间和直角坐标空间生成轨迹的功能.一般情况下尽量使用关节空间轨迹,只有在特殊需要的情况下才使用直角坐标空间轨迹.

§3.4 轨迹的实时生成

一、关节空间轨迹的生成

§3.2 中介绍了几种在关节空间规划轨迹的方法,其中任何一种方法计算出的结果都是各个轨迹段的一组数据.控制系统的轨迹生成器使用这些数据以轨迹更新速度具体计算出 θ, $\dot{\theta}$ 和 $\ddot{\theta}$.

对于三次多项式轨迹，轨迹生成器仅仅在已知 t 的情况下求解式(3-3)。当到达某轨迹的终点时，调用一组新的三次多项式系数，t 重新赋成零，继续进行轨迹的生成。

对于具有抛物线过渡域的线性轨迹，每次生成新的轨迹段时，首先检测时间 t 的值以判断当前处于各段的线性域还是过渡域。处于线性域时，每个关节的轨迹用下式计算：

$$\begin{cases}\theta = \theta_i + \dot{\theta}_{ik}t \\ \dot{\theta} = \dot{\theta}_{ik} \\ \ddot{\theta} = 0\end{cases} \tag{3-29}$$

其中，t 是从第 i 个途经点起算的时间，$\dot{\theta}_{ik}$ 是在规划轨迹时由式(3-22a)算出。处于过渡域的时候，每个关节的轨迹由下式算出：

$$\begin{cases} t_{inb} = t - \left(\dfrac{1}{2}\ t_i + t_{ik}\right) \\ \theta = \theta_i + \dot{\theta}_{ik}(t - t_{inb}) + \dfrac{1}{2}\ \ddot{\theta}_k t_{inb}^2 \\ \dot{\theta} = \dot{\theta}_{ik} + \ddot{\theta}_k t_{inb} \\ \ddot{\theta} = \ddot{\theta}_k \end{cases} \tag{3-30}$$

其中 $\dot{\theta}_{ik}$，$\ddot{\theta}_k$，t_i 和 t_{ik} 在轨迹规划过程中用式(3-22)至式(3-26)算出。当进入一个新的线性域时，重新把 t 置成 $\dfrac{1}{2}\ t_i$，直到把表示轨迹段的数据都处理完为止。

二、直角坐标空间轨迹的生成

在§3.3介绍的直角坐标法中，我们采用了具有抛物线过渡域的线性轨迹，所标出的数据是直角坐标系中的位置和姿态的值，而不是关节变量的值，所以我们要把式(3-29)和(3-30)重新表示成 χ，用以表示直角坐标的位置和姿态的向量。在线性域中，χ 的每一个自由度用下式计算：

$$\begin{cases} x = x_j + \dot{x}_{jk}t \\ \dot{x} = \dot{x}_{jk} \\ \ddot{x} = 0 \end{cases} \tag{3-31}$$

其中，t 是从第 j 个途经点起算的时间，$\dot{x}_{jk}$ 在轨迹规划过程中由类似于式（3-22a）的方程求出．在过渡域中，每一个自由度的轨迹由下式计算：

$$\begin{cases} t_{\text{inb}} = t - \left(\dfrac{1}{2}t_j + t_{jk}\right) \\ x = x_j + \dot{x}_{jk}(t - t_{\text{inb}}) + \dfrac{1}{2}\ddot{x}_k t_{\text{inb}}^2 \\ \dot{x} = \dot{x}_{jk} + \ddot{x}_k t_{\text{inb}} \\ \ddot{x} = \ddot{x}_k \end{cases} \tag{3-32}$$

其中 $\dot{x}_{jk}$、$\ddot{x}_k$、t_j 和 t_{jk} 的值是在轨迹规划的过程中计算出来的，这与关节空间的情况完全相同．

最后，这些直角坐标空间的轨迹（$\boldsymbol{\chi}$，$\dot{\boldsymbol{\chi}}$ 和 $\ddot{\boldsymbol{\chi}}$）必须转换成等价的关节空间量．对此，可以通过求解逆运动学得到关节的位置，用雅可比的逆求出速度，用雅可比的逆以及它的导数求出加速度，所得结果可为解析解．在实际工作中常常用简化了的方法，即根据逆运动学以轨迹更新速率把 $\boldsymbol{\chi}$ 转换成关节角向量 $\boldsymbol{\Theta}$，然后用数值微分计算出 $\dot{\boldsymbol{\Theta}}$ 和 $\ddot{\boldsymbol{\Theta}}$：

$$\begin{cases} \dot{\boldsymbol{\Theta}}(t) = \dfrac{\boldsymbol{\Theta}(t) - \boldsymbol{\Theta}(t - \delta t)}{\delta t} \\ \ddot{\boldsymbol{\Theta}}(t) = \dfrac{\dot{\boldsymbol{\Theta}}(t) - \dot{\boldsymbol{\Theta}}(t - \delta t)}{\delta t} \end{cases} \tag{3-33}$$

第四章　机器人控制

§4.1　位置控制

一、位置控制问题

上一章介绍了如何生成机械手的运动轨迹，本章将讨论如何使机械手执行那些预期的运动。

从当前已有的机器人系统看，很少采用步进电机等开环方式控制机械手，大部分机械手的驱动器都是对各关节施加力或力矩。这样就必须使用某种反馈控制系统，使机械手按要求运动。

在我们所讨论的机械手模型中，每个关节都装有角度量测元件，有时候还装有速度量测元件。关节的驱动和传动方法可以多种多样，但作为模型，就认为每一个关节只由一个驱动器单独驱动。

为了使每个关节按预期的方式运动，必须选择一种控制法则(即控制算法)给驱动器传送力矩命令，其根据通常是从量测元件得到的反馈信息，但是反馈信息同时也给控制算法的综合带来不少困难。与其他控制问题相比，机械手的控制是相当复杂的。

机械手的控制问题是非线性的，这意味着大部分线性控制理论不能直接在这里使用，无论是"经典的"方法或是"现代的"方法都是如此。不过，对于大多数非线性问题的求解，我们都只限于采用线性控制理论的方法。

二、单自由度的位置控制

首先讨论一个最简单的机械系统的位置控制问题。质量为m的物体位于没有摩擦力的轨道上，它能够沿单一的轨道方向X自由运动，驱动器可以在X方向给物体施加任意大小的力f，量测元件可以量测出物体在X方向上的位置和速度，系统的开环运动方程为

$$f = m\ddot{x} \tag{4-1}$$

为了进一步简化问题，可以假定物体的质量为一个单位，即 $m = 1$，我们把这样的系统称之为"单位质量系统"。于是，位置控制问题就是建立一个合适的控制器，使物体不受随机的干扰力的影响而始终维持在预期的位置上。

1. 简单的位置调节

对于前面所说的单位质量系统，可以用按下式表示的控制法则计算驱动器应该施加到物体上的力：

$$f = -k_v\dot{x} + k_p(x_d - x) \tag{4-2}$$

其中，x_d为物体的期望位置，x为实际位置，k_p和k_v为控制系统的位置及速度增益。如果适当地选择增益，就可以抑制干扰力，使物体维持在预期的位置x_d上。

根据式(4-2)，当考虑对外部干扰的响应时，可以把系统看作由单位质量的物体与一个刚度为k_p的虚拟弹簧联接而成。物体以及弹簧的振动是衰减的，就象是物体处于粘滞摩擦系数为k_v的情况。于是，k_p决定了系统的闭环刚度。

2. 跟踪轨迹的位置控制

现在把上述位置控制深入一步，不仅要求把物体维持在预期的位置上，还要求物体跟踪指定的轨迹，即，使物体的运动变化满足指定的时间函数 $x_d(t)$。假定 $x_d(t)$ 存在一阶、二阶导数 $\dot{x}_d(t)$、$\ddot{x}_d(t)$，跟踪轨迹的位置控制法则可选为

$$f = \ddot{x}_d + k_v(\dot{x}_d - \dot{x}) + k_p(x_d - x) \tag{4-3}$$

把上式与系统运动方程式(4-1)结合起来，取 $m = 1$，就可得到系统运动的误差方程：

$$\ddot{e} + k_v\dot{e} + k_p e = 0 \tag{4-4}$$

其中，e 表示位置误差，定义为 $x_d - x$。

由于我们恰当地选择了控制法则式(4-3)，因而导出了闭环系统的微分方程式(4-4)，这样通过选取 k_v 和 k_p 的值，就可以很容易地确定系统对于误差的抑制特性，例如，可以使这个二阶系统达到临界阻尼状态，没有超调，误差可以得到最快的抑制。

式(4-4)的特征方程为

$$s^2 + k_v s + k_p = 0 \tag{4-5}$$

它的根为

$$\begin{cases} s_1 = -\dfrac{k_v}{2} + \dfrac{\sqrt{k_v^2 - 4k_p}}{2} \\ s_2 = -\dfrac{k_v}{2} - \dfrac{\sqrt{k_v^2 - 4k_p}}{2} \end{cases} \tag{4-6}$$

当

$$k_v^2 - 4k_p = 0 \tag{4-7}$$

的时候方程式(4-5)处于临界阻尼状态。闭环特征方程的根 s_1 和 s_2 称作系统的闭环极点。这些极点在复平面的位置决定了干扰抑制的特性。如果要求系统处于临界阻尼状态，那么就有一对特定的 s_1、s_2 与之对应。在以下的有关讨论中，我们都把临界阻尼作为预期的状态。

以上所述的轨迹跟踪控制器可表为图 4.1 所示的方块图。控制对象是只有一个自由度的单位质量系统。

3. 引入积分项

在很多情况下，控制法则中往往加入一个误差的积分项，此时式(4-3)变为

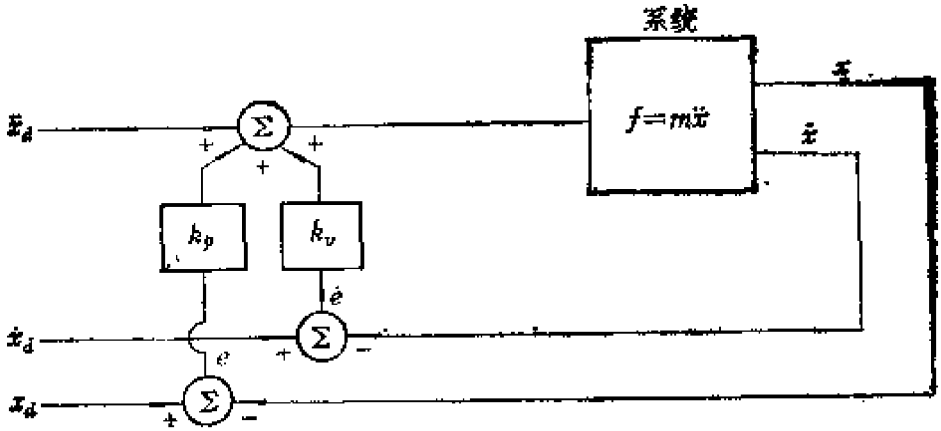


图 4.1　单位质量系统的控制方块图

$$f = \ddot{x}_d + k_v\dot{e} + k_p e + k_i\int e\,dt \qquad (4\text{-}8)$$

引入积分项以后，任何稳态误差都在积分项中得到反应，从而产生足够大的反馈作用力使系统运动，直至消除误差．选择这样的控制法则，系统就变成了三阶系统．求解式(4-8)表示的三阶微分方程，可以确定系统对预期输入信息或者扰动信息的响应．一般情况下 k_i 相当小，因而这个三阶系统与没有积分项的二阶系统很相似．上述这种控制法则称为比例-积分-微分控制，即所谓的 PID 控制．

三、控制法则的分解

为了考虑更复杂的机械系统控制法则设计问题，我们先来讨论图 4.2 所示的"质量、弹簧、摩擦"系统，通过一种比较简单的控制法则分解方法，可以把系统简化为基本的单位质量系统．

图 4.2 所示系统的开环运动方程为

$$m\ddot{x} + b\dot{x} + kx = f \qquad (4\text{-}9)$$

我们希望把相应于系统控制器的控制法则分解成两部分．一个部分是"基于模型"的控制法则部分，它使用特定的受控系统的参数，对于所讨论的系统使用 m、b 和 k 的假设条件．　我们通过下面的

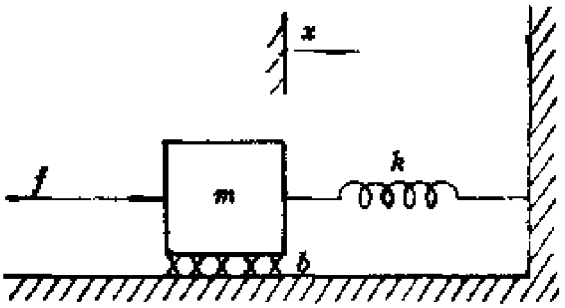


图 4.2 "质量、弹簧、摩擦"系统

举例讨论，将会清楚地看到，建立这一部分控制法则，就可以把系统看作一个单位质量系统。控制法则的另一个部分称为"误差驱动"控制法则，它对期望的和实际的变量取微分，再把这些误差乘以增益，最后形成误差信号。有时候也把误差驱动部分称作"伺服"部分。由于基于模型的部分使人们可以把系统看作为单位质量系统，伺服部分的设计就很简单了，即选择增益时，只要把受控系统看作是一个既没有摩擦、又没有刚度的单位质量系统就可以了。

基于模型的控制法则可有如下形式：

$$f = \alpha f' + \beta \tag{4-10}$$

其中 α 和 β 是待定的函数或常量，选择 α 和 β 的原则是，把 f' 作为新的输入量，系统就表现出单位质量系统的特征。

对于这种结构的控制法则，由式 (4-9) 和式 (4-10) 可以得到系统方程

$$m\ddot{x} + b\dot{x} + kx = \alpha f' + \beta \tag{4-11}$$

这时，如果把 α 和 β 选定为

$$\begin{cases} \alpha = m \\ \beta = b\dot{x} + kx \end{cases} \tag{4-12}$$

将上式代入 (4-11)，就可得

$$\ddot{x} = f' \tag{4-13}$$

这正是单位质量系统的运动方程。

现在，设计控制法则的伺服部分就很容易了，即只需把 f' 作为误差函数来计算：

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e \tag{4-14}$$

上式表示的系统前面已经讨论过，其特征方程为式(4-5)，式(4-7)给出了临界阻尼时的解。图 4.3 一般地表示了经过分解后，跟踪轨迹输入的伺服控制法则方块图，图中虚线表示 α、β 是系统状态 x 和 $\dot{x}$ 的函数。

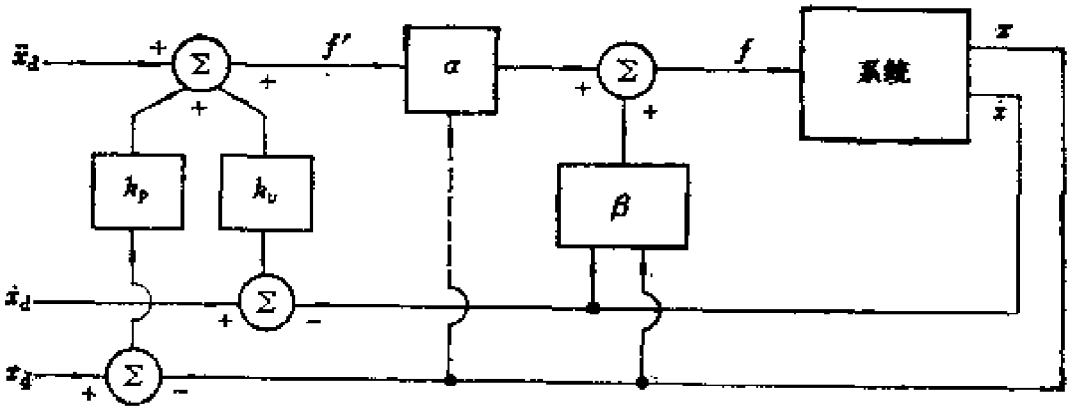


图 4.3　分解后的伺服控制法则方块图

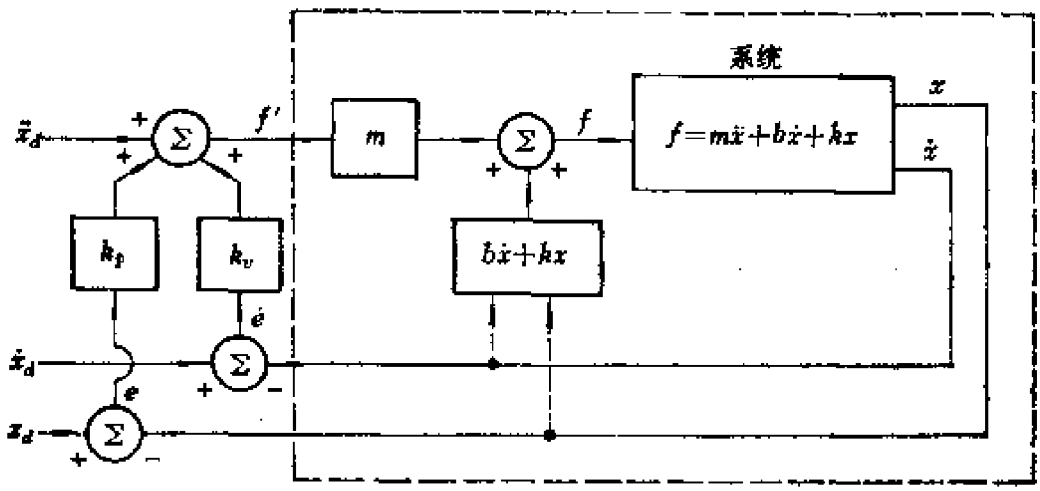


图 4.4　使用分解伺服控制方法的控制系统

例 4.1　通过对式(4-14)表示的伺服控制法则确定恰当的反

馈增益，使系统达到闭环刚度为 k_{CL}N/m 的临界阻尼状态.

对于一个单位质量系统，如果式(4-7)得到满足，系统则处于临界阻尼状态. 因此我们可以很容易地计算出增益:

$$\begin{cases} k_p = k_{CL} \\ k_v = 2\sqrt{k_{CL}} \end{cases}$$

完整的控制系统框图如图 4.4 所示. 虚线内的部分可以看成一个"黑盒子"，它具有单位质量系统的动力学特性. "黑盒子"内部是实际系统加上控制法则基于模型的部分.

四、非线性和时变系统

在上一节的讨论中，只是把"质量、弹簧、摩擦"系统看作线性非时变系统. 对于非线性和时变的情况，求解过程就比较困难.

如果非线性不严重，可以用局部线性化的方法导出线性模型，它在工作点的某个邻域内与非线性方程近似. 如果系统在其变量的小范围内工作，则当系统运动时，工作点可以和它一起运动. 在每一个新的工作点都重新进行线性化工作. 这种动态线性化的方法能使系统线性化，但是系统变成了时变系统.

虽然在某些分析和设计技术中，对原有系统加以准静态线性化处理是很有用的，但是在我们的控制法则综合过程中不准备用它. 我们将直接研究非线性运动方程，而不借助线性化来导出控制器.

如果图 4.2 中的弹簧不是线性的，它有某种非线性特性，我们可以把系统看作为准静态系统，并且每时每刻决定系统极点的位置. 这样，极点就在复平面上运动，它是物体位置的函数. 因此我们不能选择固定的增益使极点保持在预期的位置上(例如，临界阻尼). 为此，需要考虑更复杂的控制法则，其中要选取作为物体位置的函数的时变增益，从而可使系统总是处于临界阻尼状态. 从这种考虑出发，实质上是要计算 k_p，使得控制法则中有一个非线

性项正好抵消掉弹簧的非线性效应，从而整个刚度始终保持不变．这样的控制方式可以称作线性化控制法则，因为它用一个非线性控制项"消除"了系统的非线性，使得整个闭环系统可作为线性系统处理．

对上述线性化控制法则进行分解，伺服法则部分总是保持不变的，但是基于模型的部分现在将包含非线性模型．因此，基于模型的控制部分就要完成线性化的功能．下面我们通过具体例子加以说明．

例 4.2 考虑"质量、弹簧、摩擦"系统（图 4.2），有如图 4.5 所示的非线性弹簧特性．与通常的线性弹簧关系 $f = kx$ 不同，这个弹簧由 $f = qx^3$ 来描述．现在要确定一种控制法则，使系统保持处于临界阻尼状态，而且刚度为 k_{CL}．

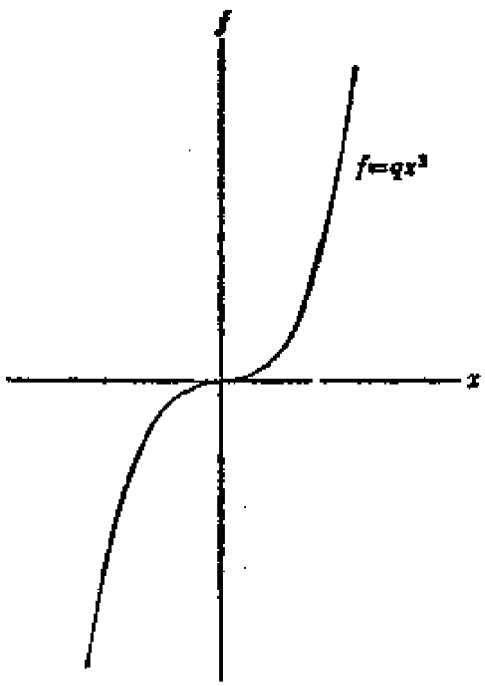


图 4.5 非线性弹簧的力-位移特性

开环方程为

$$m\ddot{x} + b\dot{x} + qx^3 = f$$

基于模式的控制部分为 $f = \alpha f' + \beta$，其中

$$\begin{cases} \alpha = m \\ \beta = b\dot{x} + qx^3 \end{cases}$$

与前面讲的一样，伺服部分为

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e$$

根据某种期望的性能指标可以计算出增益值．图 4.6 表示了这个控制系统的方块图．

例 4.3 考虑"质量、弹簧、摩擦"系统（图4.2），有如图 4.7 所示的非线性摩擦特性，与线性摩擦 $f = b\dot{x}$ 的描述不同，这个库伦

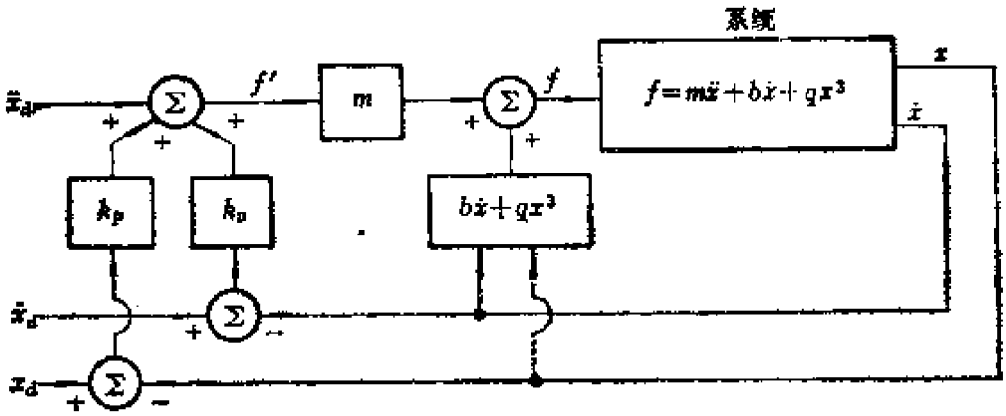


图 4.6　含有非线性弹簧的控制系统方块图

摩擦由 $f = b_c$ SGN $(\dot{x})$ 来描述。对于当前大部分机械手来说，不论是旋转关节还是移动关节，用这种非线性模型表示关节轴摩擦都比用线性模型表示更为精确。现在要设计一个控制法则，使用非线性的基于模型的部分使系统总是处于临界阻尼状态。

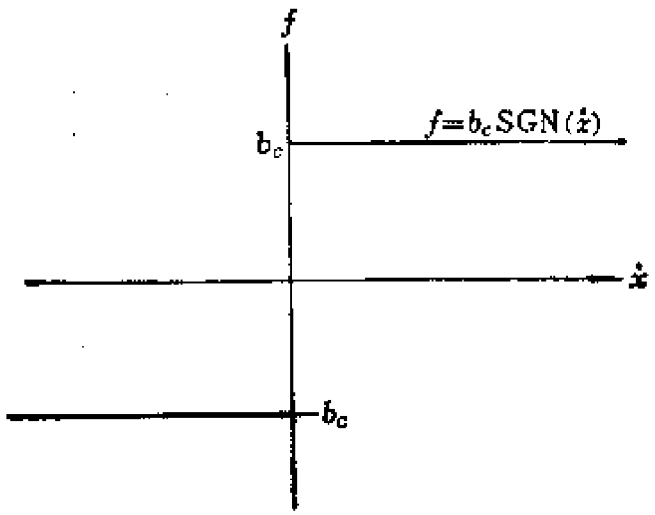


图 4.7　库伦摩擦的力-速度特性

开环方程为

$$m\ddot{x} + b_c \text{ SGN}(\dot{x}) + kx = f$$

基于模型的控制部分为 $\tau = \alpha f' + \beta$，其中

$$\begin{cases} \alpha = m \\ \beta = b_c \ \mathrm{SGN}\ (\dot{x}) + k\dot{x} \end{cases}$$

伺服部分仍是

$$f' = \ddot{x}_d + k_v \dot{e} + k_p e$$

式中的增益值可根据某种期望的性能指标来计算。

现在我们已经看到，虽然非线性控制理论中的问题处理起来是相当困难的，但是，在特定的简单情况下设计一个非线性控制器并不那么复杂。在上述简单的例子中我们所用到的方法，实际上具有一般性，对于机械手的控制问题也同样适用。这种方法可以总结为：

（1）计算一个非线性的基于模型的控制法则，用它来抵消被控制系统的非线性；

（2）把系统简化为线性系统，它可以用单位质量系统中导出的简单的线性伺服法则来进行控制。

从某种意义上说，线性化控制法则是提供了一个受控系统的“反模型”。系统中的非线性与反模型中的非线性相抵消，这一点与伺服法则一起构成了一个线性闭环系统。很显然，要完成这种抵消，我们必须知道非线性系统的参数与结构，在实际应用时这常常是个问题。

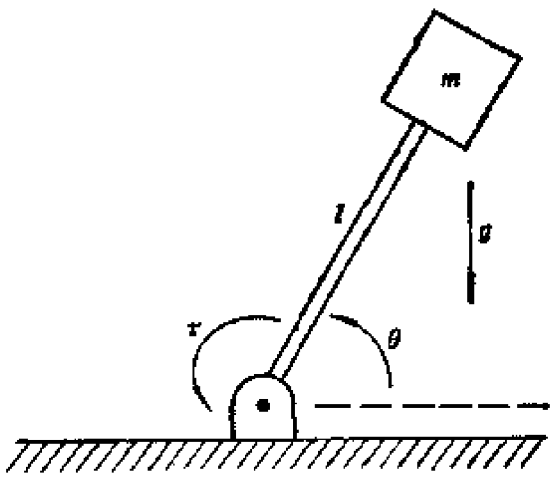


图 4.8　单连杆机械手

例 4.4　考虑图 4.8 所示单连杆机械手，它有一个旋转关节，假设质量集中在连杆末端的一个点上，因而惯性矩为 ml^2。在关节处存在有库伦摩擦和粘滞摩擦，还有由于重力产生的

负荷.

机械手的模型为

$$\tau = ml^2\ddot{\theta} + v\dot{\theta} + c\ \mathrm{SGN}\ (\dot{\theta}) + mlg\cos(\theta)$$

与往常一样,控制系统分为两部分,线性化的基于模型的控制部分和伺服控制部分.

基于模型的控制部分为 $\tau = \alpha\tau' + \beta$, 其中

$$\begin{cases}\alpha = ml^2\\ \beta = v\dot{\theta} + c\ \mathrm{SGN}\ (\dot{\theta}) + mlg\cos(\theta)\end{cases}$$

当然,伺服部分为

$$\tau' = \ddot{\theta}_d + k_v\dot{e} + k_p e$$

根据某种期望的性能指标可以计算出增益的值.

§ 4.2 机械手的位置控制系统

一、多输入/多输出的控制法则

与前面讨论的简单例子不同,机械手的控制问题是一个多输入/多输出的问题,因此,需要用向量表示位置、速度和加速度,控制法则所计算的则是关节驱动信号的向量. 前面讲述的控制法则的分解方法仍然可以使用,不过将以矩阵-向量的形式出现.

基于模型的控制法则可表为

$$\boldsymbol{F} = \alpha\boldsymbol{F}' + \boldsymbol{\beta} \qquad (4\text{-}15)$$

对于 n 个自由度的系统来说,上式中的 $\boldsymbol{F}$、$\boldsymbol{F}'$ 和 $\boldsymbol{\beta}$ 都是 $n \times 1$ 的向量; α 是 $n \times n$ 的矩阵(不一定要求是对角阵),其作用是对 n 个运动方程进行解耦. 如果正确地选择 α 和 $\boldsymbol{\beta}$, 那么, 系统对于输入 $\boldsymbol{F}'$ 表现为 n 个独立的单位质量系统. 由于这个原因,在多维的情况下,控制法则基于模型的部分往往称作"线性化和解耦法则".

多维系统的伺服法则形式为

$$F' = \ddot{x}_d + K_v\dot{E} + K_pE \tag{4-16}$$

其中，K_v 和 K_p 都是 $n \times n$ 的矩阵，它们通常选为对角阵，对角线上的元素为常数增益值，E 和 $\dot{E}$ 是 $n \times 1$ 的误差向量，分别表示位置和速度误差.

二、机械手的位置控制系统

在第二章中，我们讨论了机械手的力学模型及其相应的运动学方程和动力学方程，封闭形式的动力学方程为

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) \tag{4-17}$$

其中，$M(\Theta)$ 是机械手的 $n \times n$ 惯性矩阵，离心项和哥氏项 $V(\Theta, \dot{\Theta})$ 为 $n \times 1$ 向量，重力项 $G(\Theta)$ 为 $n \times 1$ 向量. $M(\Theta)$ 和 $G(\Theta)$ 中的每一项都是与 Θ 有关的复杂的函数，Θ 为机械手所有的关节位置. $V(\Theta, \dot{\Theta})$ 的每一项都是与 Θ 和 $\dot{\Theta}$ 有关的复杂函数.

此外，我们可以加进一个摩擦(或者其他非刚体效应)模型，摩擦模型设为关节位置和速度的函数，于是式 (4-17) 变为

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}) \tag{4-18}$$

现在对上式表示的复杂系统运用控制法则的分解方法. 基于模型的控制法则为

$$\tau = \alpha\tau' + \beta \tag{4-19}$$

其中 τ 是关节力矩，为 $n \times 1$ 向量. 我们选择

$$\begin{cases} \alpha = M(\Theta) \\ \beta = V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}) \end{cases} \tag{4-20}$$

伺服法则为

$$\tau' = \ddot{\Theta}_d + K_v\dot{E} + K_pE \tag{4-21}$$

其中

$$E = \Theta_d - \Theta$$

控制法则经过分解后，控制系统的方块图如图 4.9 所示.

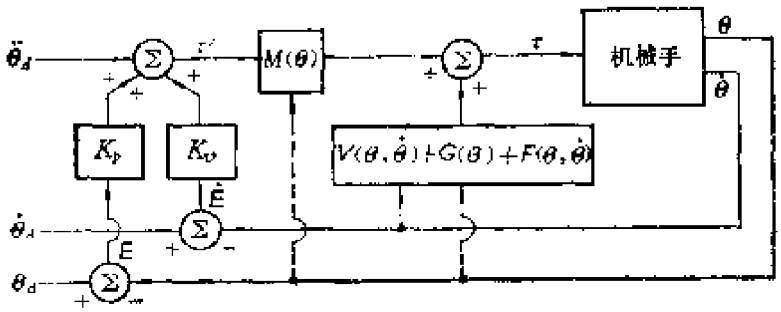


图 4.9　机械手控制系统方块图

在上述的式 (4-17) 中，由于含有表示离心项和哥氏项的 $\boldsymbol{V}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})$，它既与关节位置 ($\boldsymbol{\Theta}$) 有关，又与关节速度 ($\dot{\boldsymbol{\Theta}}$) 有关，因而可以把式 (4-17) 称为状态空间动力学方程。如果把 $\boldsymbol{V}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}})$ 的形式重新分为两项，可有

$$\boldsymbol{V}(\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}}) = B(\boldsymbol{\Theta})[\boldsymbol{\Theta},\dot{\boldsymbol{\Theta}}] + C(\boldsymbol{\Theta})[\dot{\boldsymbol{\Theta}}^2] \qquad (4\text{-}22)$$

式中，第一项为哥氏项，$B(\boldsymbol{\Theta})$ 是由哥氏系数构成的 $n \times [n(n-1)/2]$ 矩阵，$[\boldsymbol{\Theta}\dot{\boldsymbol{\Theta}}]$ 是由关节速度的乘积构成的 $[n(n-1)/2] \times 1$ 向量，即

$$[\boldsymbol{\Theta}\dot{\boldsymbol{\Theta}}] = [\dot{\theta}_1\dot{\theta}_2\ \dot{\theta}_1\dot{\theta}_3 \cdots \dot{\theta}_{n-1}\dot{\theta}_n]^T \qquad (4\text{-}23)$$

式 (4-22) 中的第二项为离心项，$C(\boldsymbol{\Theta})$ 是由离心系数构成的 $n \times n$ 矩阵，$[\dot{\boldsymbol{\Theta}}^2]$ 是由各关节速度的平方构成的 $n \times 1$ 向量，即

$$[\dot{\boldsymbol{\Theta}}^2] = [\dot{\theta}_1^2\dot{\theta}_2^2 \cdots \dot{\theta}_n^2] \qquad (4\text{-}24)$$

于是，式 (4-17) 就变为

$$\boldsymbol{\tau} = M(\boldsymbol{\Theta})\ddot{\boldsymbol{\Theta}} + B(\boldsymbol{\Theta})[\boldsymbol{\Theta}\dot{\boldsymbol{\Theta}}] + C(\boldsymbol{\Theta})[\dot{\boldsymbol{\Theta}}^2] + \boldsymbol{G}(\boldsymbol{\Theta}) \qquad (4\text{-}25)$$

由于上式中各矩阵元素仅为机械手关节位置 $\boldsymbol{\Theta}$ 的函数，因而可以把式 (4-24) 称为位姿空间动力学方程。

位姿空间动力学方程的表示形式，方便了实际应用中的控制法则计算过程，因为所有动力学参数仅表现为关节位置的函数，因而可以随着机械手的位置和姿态的变化而得到及时的修正。图 4-10 表示了相应于位姿空间动力学方程的控制系统方块图。系统已经解耦并进行了线性化处理。按照这种控制结构，假设动力

学系数的计算可另由负责"后台过程"的一台计算机完成，那么，动力学参数的计算速率可低于闭环控制过程的伺服计算速率。例如，前者可为 60Hz，而后者可达 200Hz。

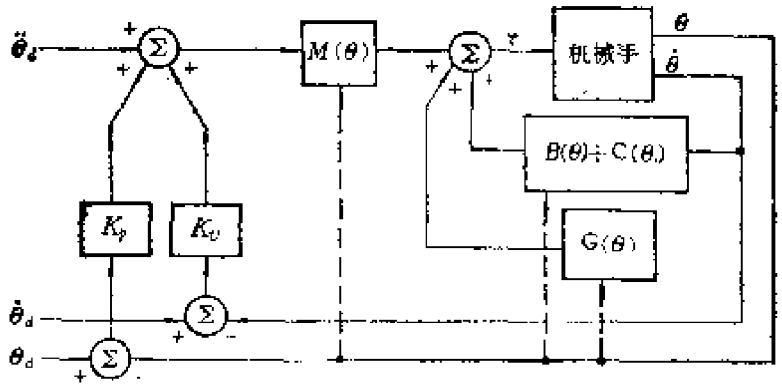


图 4.10　基于模型的机械手控制系统

如果机械手的全部模型参数，如杆长、惯性、摩擦系数等都为已知，那么可由图 4.10 实现一个控制系统，对于手臂的任意位置和姿态，它都能使各个关节处于临界阻尼状态。当然，由于我们的模型不会很精确，所以这一点难以做到。但是，能做到满足具体的实际应用还是可能的。对于模型中误差效果的分析，我们的方法是指明作用在关节上的干扰力矩向量。在图 4.10 中我们已经指明这些作为输入的干扰。这些干扰力矩的产生是由于有些因素在动力学方程中没有考虑到或者模型化不正确。如果摩擦项没有包括在基于模型的控制法则中，干扰力矩就主要源于摩擦效应，另外，这些未知的力矩也包括了共振以及所有的"噪声"效应。

考虑到未知的扰动力矩，系统的误差方程为

$$\ddot{\boldsymbol{E}} + K_v\dot{\boldsymbol{E}} + K_p\boldsymbol{E} = M^{-1}(\boldsymbol{\Theta})\boldsymbol{\tau}_d \qquad (4\text{-}26)$$

其中 $\boldsymbol{\tau}_d$ 表示各个关节的干扰力矩向量。式 (4-26) 的左边没有耦合问题，但从右边可以看出，任意一个关节的扰动都将给所有其他关节引进误差，因为在一般情况下 $M(\boldsymbol{\Theta})$ 不是对角阵。

三、实用考虑

在前面几部分的讨论中，实际上都作了一些假设，推导了线性的，解耦的控制系统。下面提出一些系统设计和求解过程中需要考虑的实际问题。

1. 参数问题

实用过程中的主要问题是人们对模型参数值掌握的不够准确，特别是摩擦效应，甚至选取什么样的模型结构都极为困难，更不用说参数值了。另外，机械手的动力学模型从整体上说并不是一成不变的，由于机械手也在不断老化，参数就会随着变化，因而不能重复使用。

在实际工作中，大部分机械手要抓取各种各样的零件或者工具，当机械手拿着这些物体的时候，物体的重量和惯性改变了机械手原有的动力学特性。如果工具的质量特性是已知的，可以把它们包含在控制法则的模型部分中。当抓取工具的时候，惯性矩阵和机械手最后一个连杆的质量中心可以修改成新的值，这个值表示了最后一个连杆及工具的组合效应。然而在许多应用中物体的质量特性是不知道的，所以保持精确的动力学模型是不可能的。

为此，我们用符号 $\hat{M}(\Theta)$ 表示机械手惯性矩阵 $M(\Theta)$ 的模型。类似地，$\hat{V}(\Theta,\dot{\Theta})$，$\hat{G}(\Theta)$ 和 $\hat{F}(\Theta,\dot{\Theta})$ 表示实际机构中速度项，重力项和摩擦项的模型。所谓完整而精确的模型知识指的是

$$\begin{cases}\hat{M}(\Theta)=M(\Theta)\\ \hat{V}(\Theta,\dot{\Theta})=V(\Theta,\dot{\Theta})\\ G(\Theta)=G(\Theta)\\ \hat{F}(\Theta,\dot{\Theta})=F(\Theta,\dot{\Theta})\end{cases} \tag{4-27}$$

因此，虽然给出的机械手的动力学为

$$\tau=M(\Theta)\ddot{\Theta}+V(\Theta,\dot{\Theta})+G(\Theta)+F(\Theta,\dot{\Theta}) \tag{4-28}$$

而我们的控制法则将计算

$$\begin{cases}\tau = \alpha\tau' + \beta \\ \alpha = \hat{M}(\Theta) \\ \beta = \hat{V}(\Theta, \dot{\Theta}) + \hat{G}(\Theta) + \hat{F}(\Theta, \dot{\Theta})\end{cases} \tag{4-29}$$

参数不完整不精确时，解耦和线性化的工作将不能正确地完成．写出系统的闭环方程(略去自变量符号)，我们有

$$\ddot{E} + K_v\dot{E} + K_pE = \hat{M}^{-1}[(M - \hat{M})\ddot{\Theta} + (V - \hat{V}) + (G - \hat{G}) + (F - \hat{F})] \tag{4-30}$$

如果式(4-27)成立，这意味着模型是正确的，那么式(4-30)的右侧将为零，误差将消失；反之，实际参数与模型参数之间的偏差将产生可由式(4-30)表示的伺服误差．

2.计算时间问题

在前面对于控制法则分解的讨论中，我们隐含地假设了整个系统是一个连续系统，而且，计算控制法则所需要的计算时间为零．实际上，机械手控制系统按控制法则计算式(4-29)去计算需要相当大的计算工作量，因此必须解决快速计算的问题．

现在几乎所有的机械手控制系统都采用了数字电路，并且以固定的采样速度运行，即，位置(或者其他)量测元件的读数方式是离散的，它向驱动器传送命令也不是连续进行的，而是以有限的采样频率进行的．为了分析由于计算和有限的采样频率所产生的延时影响，必须使用离散控制领域内的某些方法．离散控制系统中许多概念和方法在机械手的控制中都是非常必要的．

尽管如此，离散系统的控制理论还是难以用于机械手，如果通过求级数近似解的办法来建立动力学方程的离散模型，也许不如干脆采用连续模型，同时以足够快的计算速度，达到连续模型的合理近似目标．但是，多快算是足够快？在选择足够快的伺服速度的时候，有许多因素需要考虑，例如，采样频率与参考输入信号带宽的关系，采样周期与干扰噪声的相关时间的关系，以及采样速度

与机械手机构共振频率的关系等等。这些都是属于较深层次的问题，在此不再仔细讨论。

四、机器人位置控制的简单方法

在目前商品化的机器人控制器中，由于缺乏参数知识、难于高速计算等原因，都没有采用我们在前面介绍过的那种完善的机械手模型，而一般都采用比较简单的控制法则——误差驱动。在我们提出的分解的控制器结构中，也可以采用这些简单的控制方法。

1. 单个关节的 PID 控制

对于大部分现有的工业机器人的控制方案，可以描述为

$$\begin{cases} \alpha = I \\ \beta = 0 \end{cases} \tag{4-31}$$

其中 I 是 $n \times n$ 的单位矩阵。伺服部分为

$$\tau' = \ddot{\Theta}_d + K_v \dot{E} + K_p E + K_i \int E dt \tag{4-32}$$

式中 K_v，K_p 和 K_i 是常数对角阵。在很多情况下不用 $\ddot{\Theta}_d$ 这一项，即简单地把这一项置为零。这就是说，大部分简单机器人控制器所采用的控制法则，根本不使用基于模型的部分。因为每个关节都是用单独的控制系统控制，所以这种 PID 控制方案比较简单，常常是每个关节用一台微处理器来处理式 (4-32)。

要想描述以这种方式控制的机械手的性能也是不容易的。因为关节之间没有解耦，一个关节的运动就会影响到其他的关节。这些交互作用引起的误差是由误差驱动这个控制法则抑制的。要选择一个固定的增益，使机械手在任何形态下都处于临界阻尼状态是不可能的。因此选择“平均”增益，使得当机械手的形态处于工作空间的中心位置时，系统接近于临界阻尼状态。在机械手处于各种极端的形态时，系统变成了欠阻尼或者过阻尼。由于在机械手的设计过程采取了一些细节措施，这种影响有可能比较小，达

到较好的控制效果．在这种系统中，保持比较高的增益很重要，这样可以使那些无法避免的扰动被迅速地抑制掉．当然，增益的上限值是由许多因素决定的，例如伺服速度，量测元件的噪声和结构共振等等．

这种系统的控制器抑制误差的确切过程虽然很难直观地表示出来，但可以用式（4-30）加以说明，其中 $\hat{V}=G=\hat{F}=0$，$\hat{M}=I$．在这里我们还假设 $K_i=0$，因此可以得到下式（略去自变量）：

$$\ddot{E}+K_v\dot{E}+K_pE=(M-I)\ddot{\Theta}+V+G+F \tag{4-33}$$

以这种方式控制的机械手将以不同的方式把干扰抑制掉，这取决于机械手的形态．另外，由于关节之间的耦合，机械手自身的运动将不断地产生伺服误差．从理论上看，式（4-33）是不能令人满意的，但是它几乎表示了当前所有的工业机械手的控制方法．使用足够高的 K_v 和 K_p 可以使这些机械手具有比较好的控制特性．

2．单个关节"有效关节惯量" PID 控制

下式描述了另外一种机械手的控制方案

$$\begin{cases}\alpha=M'(\Theta)\\ \beta=0\end{cases} \tag{4-34}$$

式中 M' 为 $n\times n$ 的对角阵，对角线上的元素为机械手形态（位置和姿态）的函数．伺服部分为

$$\tau'=\ddot{\Theta}_d+K_v\dot{E}+K_pE+K_i\int E\,dt \tag{4-35}$$

式中 K_v，K_p 和 K_i 为常数对角阵．这里机械手的质量矩阵为简化形式，不用计算非对角线上的元素．在这个模型中，每个关节被看作是独立的系统，其惯量随着机械手形态的变化而变化．构造这种可变惯量的模型是为了消除（至少是部分消除）惯量变化的影响．

这样的系统可以比较容易地保持在接近临界阻尼的状态．但

是我们仍然需要不断地抑制由于关节间的耦合产生的扰动。

3. 惯量解耦

有些机械手的控制方案，用我们的表示方法可以表示为

$$\begin{cases}\alpha = \hat{M}(\Theta) \\ \beta = 0\end{cases} \tag{4-36}$$

式中 $\hat{M}$ 是机械手的 $n \times n$ 惯性矩阵模型．伺服部分为

$$\tau' = \ddot{\Theta}_d + K_v\dot{E} + K_pE + K_i\int E dt \tag{4-37}$$

式中 K_v，K_p 和 K_i 是常数对角阵．假设质量矩阵模型是准确的，这类机械手控制器的闭环误差空间方程(略去自变量)为

$$\ddot{E} + K_v\dot{E} + K_pE = \hat{M}^{-1}(V + G + F) \tag{4-38}$$

由于动力学项将造成静态位置误差，有些机器人制造厂家在控制法则中加入了重力模型 $G(\Theta)$[相当于 $\beta = \hat{G}(\Theta)$]。

4. 解耦控制的近似方法

对于特定的机械手，有许多方法可用来简化其动力学方程，简化以后可以得到一个比较简单的解耦和线性化法则．通常的做法是把速度项产生的力矩分量忽略掉，即模型只包括惯量项和重力项．一般情况下控制器中不考虑摩擦项，因为很难建立正确的摩擦模型．有时候也对惯性矩阵进行简化，只考虑轴与轴之间的主要耦合，不考虑次要的交叉耦合效应。

五、基于直角坐标的控制系统

在这一部分我们将介绍基于直角坐标控制的概念，这种方法还处于研究阶段。

1. 与基于关节控制方式的比较

到现在为止我们讨论的机械手的控制方式中，都假定那些期望的轨迹是由关节的位置、速度和加速度的时间顺序表示的．对于给定的期望输入，基于关节空间的控制方案的做法是，求出关节

空间期望值和实际值之间的差，从而得到轨迹误差．但是，正如第三章所指出的，我们常常期望机械手的手爪沿着直角坐标空间的直线（或者其他曲线）轨迹运动．为此，可以计算出对应于直角坐标空间直线轨迹的关节空间轨迹．图 4.11 给出了这种机械手轨迹控制的方式．这种方案的基本特点是具有一个“轨迹变换”的过程，用来计算关节的轨迹．随后的控制问题，我们在前面已经讨论过了．

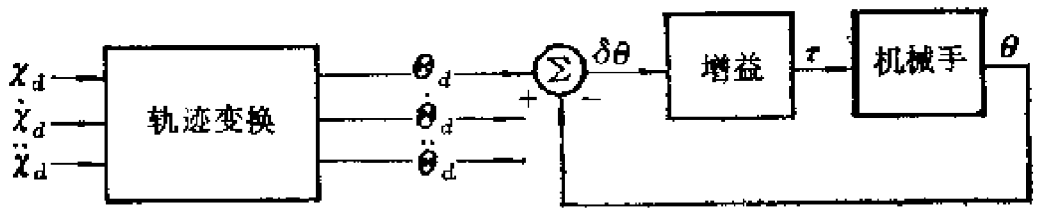


图 4.11　输入为直角坐标轨迹的基于关节的控制方块图

轨迹变换的计算量比较大，如果用解析的方法，则需要计算

$$\begin{cases} \boldsymbol{\Theta}_d = \mathrm{INVKIN}\ (\boldsymbol{\chi}_d) \\ \dot{\boldsymbol{\Theta}}_d = J^{-1}(\boldsymbol{\Theta})\dot{\boldsymbol{\chi}}_d \\ \ddot{\boldsymbol{\Theta}}_d = \dot{J}^{-1}(\boldsymbol{\Theta})\dot{\boldsymbol{\chi}}_d + J^{-1}\ddot{\boldsymbol{\chi}}_d \end{cases} \tag{4-39}$$

式中 INVKIN $(\boldsymbol{\chi}_d)$ 表示对 $\boldsymbol{\chi}_d$ 求解逆运动学．目前所有的系统都进行这种计算，通常，先求解运动学逆问题得出 $\boldsymbol{\Theta}_d$，然后用一阶、二阶差分求解出关节的速度和加速度．但是，除非运用无源滤波器，否则这种数值微分将引入噪声和延迟．因此，我们可以另外考虑解决问题的方法，或者设法寻找工作量比较小的计算方法来计算式 (4-39)，或者提出一种不同的控制方案，使它不需要这些信息．

图 4.12 表示了另外一种方案．其中，检测到的机械手各关节位置立即通过运动学方程（即图中的 Kin$(\boldsymbol{\Theta})$，以后的有关图中也这样表示）转换成直角坐标系中的位置描述，然后把这种描述与期望的直角坐标位置相比较，以形成直角坐标空间的误差信息．

这种以生成直角坐标空间误差为基础的控制方案称作基于直角坐标的控制方案。

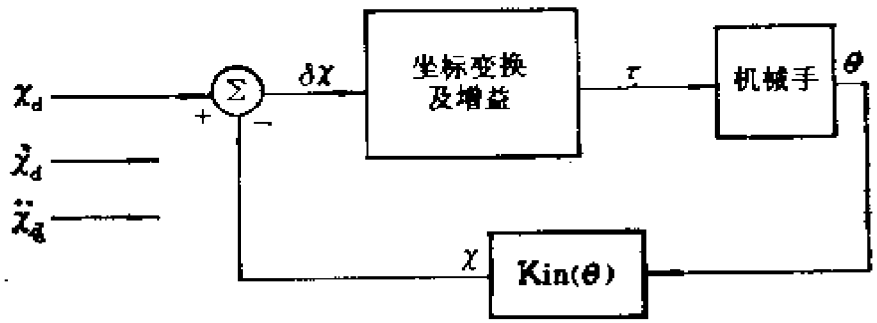


图 4.12　基于直角坐标控制方式的概念框图

在上述控制方案中，轨迹变换过程被伺服回路中的某种坐标变换所替代。值得指出，由于在这种方案中，运动学问题和其他的变换现在都包含在回路之中，因而基于直角坐标的控制器必须在回路中完成大量的计算工作量，这可能是基于直角坐标方法的一种"倒退"，因为与基于关节的系统比较，在使用相同的计算机的情况下，这种系统将以比较低的采样频率运行。一般情况下，这将会降低系统的稳定性和抑制干扰的能力。

2. 直角坐标解耦方式

与基于关节的控制器一样，基于直角坐标的控制器也应该对于机械手的所有可能的形态，都能做到以临界阻尼的方式抑制直角坐标误差。

基于关节的控制器之所以具有较好的控制性能，就在于建立了线性化和解耦的手臂模型。在基于直角坐标的方案中，也可以设计类似的控制器。但是在这里，我们必须用直角坐标变量写出机械手的动力学方程。即

$$\mathcal{F} = M_x(\Theta)\chi + V_x(\Theta, \dot{\Theta}) + G_x(\Theta) \qquad (4\text{-}40)$$

式中，$\mathcal{F}$ 是作用于机械手手爪上的力/力矩向量，χ 是在直角坐标系中表示手爪位置和方位的向量。其他的参数与在关节空间中

的情况类似： $M_x(\Theta)$ 为质量矩阵，$V_x(\Theta, \dot{\Theta})$ 为速度项向量，$G_x(\Theta)$ 为重力项向量。 应该说明的是这些参数都是直角坐标空间中的参数。

由于式 (4-40) 可计算作用于手爪上的直角坐标力向量，我们将使用雅可比的转置以便完成控制功能。这就是说，用式 (4-40) 计算出 $\mathcal{F}$ 以后，由于无法直接把直角坐标空间的力加到手爪上，因而必须用下式计算出与这个力等效的施加在各个关节上的力矩

$$\tau = J^T(\Theta)\mathcal{F} \qquad (4\text{-}41)$$

图 4.13 画出了使用动力学解耦的直角坐标控制系统方块图。注意，图中雅可比转置这一环节位于“手臂”环节之前，所表示的控制器允许直接描述直角坐标轨迹，而无需进行轨迹转换。

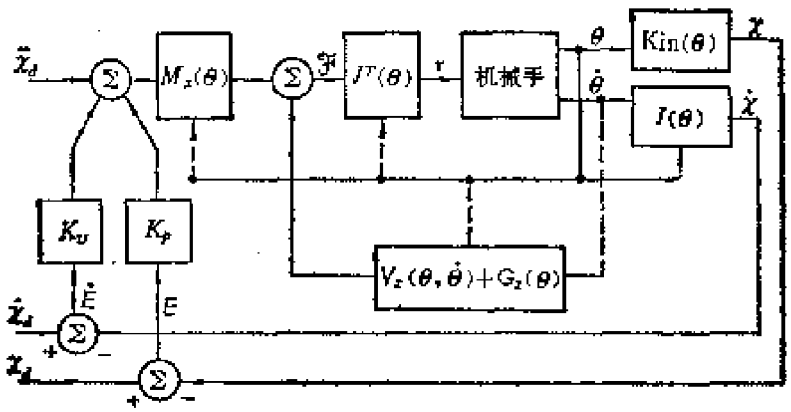


图 4.13　基于直角坐标的控制方块图

在基于关节的控制方案中，我们提出过把动力学参数修正计算与伺服计算分开的实际做法，基于直角坐标的控制方案也可以这样实现，图 4.14 画出了这种实现方案。其中，动力学参数仅仅是机械手位置的函数，这些参数的修正可以通过“后台处理过程”以低于伺服速度的速度得到完成。这样安排是比较合理的，因为我们希望通过快速伺服来最大限度地抑制干扰和提高稳定性。由于动力学参数仅仅是机械手位置的函数，其修正速度仅取决于机

械手形态的变化速度．修正速度一般不超过 100Hz，而期望的伺服速度有可能达到 500Hz．

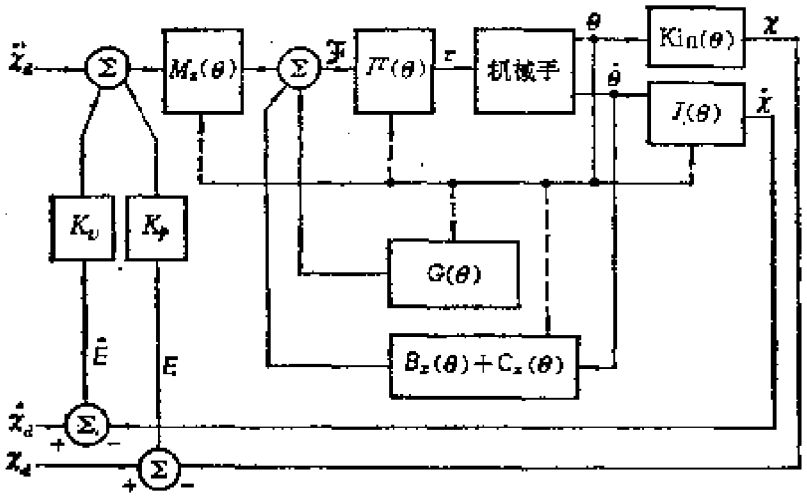


图 4.14　基于直角坐标控制的一种实现方案

§4.3　力控制

一、力控制问题

机械手的位置控制，适用于指挥机械手沿空间的某个轨迹运动的情况．但是一旦遇到手爪与环境物体接触的情况，位置控制就不够用了．例如当机械手拿着海绵擦玻璃时，如果海绵很软并且知道玻璃的精确位置，那么通过控制手爪相对于玻璃的位置，来调整它对于玻璃的作用力当然是可行的．然而，如果用一个刚性工具刮玻璃表面的油漆，而且玻璃表面的空间位置不准确，或是机械手位置伺服的误差比较大，这就不可能靠位置控制完成刮油漆的工作，其结果要么会把玻璃弄碎，要么机械手只是拿着工具在玻璃上晃来晃去，刮不到油漆．

显然，在擦玻璃和刮油漆两项工作中，描述玻璃平面的空间位置是不合适的，一种比较好的方法是描述工具与玻璃表面的接触力．相应地，机器人的控制系统就必须要具有位置/力混合控制的

功能.

具备了力控制功能，机器人可以表现出一种智能化特征，因而能够胜任更为复杂的操作任务. 例如，在典型的部件装配工作中，要求机器人能够在不确定的环境条件下进行柔顺控制. 如果在机器人的手爪上装上力传感元件，就可以及时提供关于操作状态的变化信息，由控制器加以处理，指挥机械手作出灵活的反应. 实际上，机械手的力控制问题与智能机器人的力觉是密切相关的.

具备了力控制功能还可以在一定程度上放宽机械手的精度指标，进而降低对整个机器人系统的体积、重量以及造价方面的要求. 由于采用了相对测量的方法，所以机械手和操作对象的绝对位置误差不像单纯的位置控制系统那么重要. 当中等硬度的物体互相作用时，相对位置的微小变化都会生成很大的接触力，利用这些力进行控制，可以在很大程度上弥补机械手绝对精度的不足.

二、力传感器

力传感器的作用是量测机械手的手爪与外界接触时的受力大小. 大多数力传感器是用应变片作为敏感元件的，这些应变片贴在金属骨架上，金属架的应力大小决定了应变片的输出量. 从控制的角度看，力传感器的设计通常需要考虑以下问题：

（1）为了获得预期的信息，需要几个敏感元件？

（2）如何把敏感元件装到金属骨架上？

（3）在维持一定刚度的前提下，采用什么样的骨架结构对力更敏感？

（4）怎样把机械过载保护加进装置中？

一般情况下，机械手有三种部位可以安装这些敏感元件：

（1）在关节驱动器上安装. 敏感元件量测驱动器本身输出的力和力矩，这对有些控制方案将很有用. 但是，在一般情况下它无法直接提供手爪与外界接触力的信息.

(2) 在手爪与机械手的最后一个关节之间安装. 即构成所谓的"腕部力传感器". 这种力传感器量测作用在手爪上的力以及力矩,得到具有六个分量的力/力矩向量.

(3) 在手爪指尖上安装. 通常这种"力敏感手指"上装的应变片可以量测作用在每个手指上的一到四个分量.

在此,我们不打算涉及力传感器的内部结构、工艺等问题,而只考虑在控制法则中如何使用这些传感器的力信息.

三、"部分约束"任务的控制结构

当机械手的手爪与环境物体接触以后,它的运动就部分地受到约束. 下面介绍一种针对"部分约束"情况的控制结构.

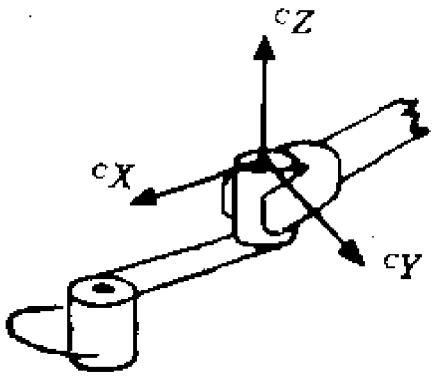
每一项控制任务都可以分解成若干项子任务,这些子任务是由手爪(或工具)与工作环境接触状况来定义的. 根据任务的机械特征和几何特征,我们可以把这样的子任务与一组约束联系起来,这组约束称作"自然约束". 例如,当手爪与静止的刚体表面接触时,它不能自由地通过这一表面,这就存在一种自然的位置约束;如果接触面没有摩擦力,手爪就不能沿着与接触面相切的方向自由地施加作用力,这样就存在着一种自然的力约束.

一般地说,我们可以对每一项子任务定义一个"广义接触面":与这个表面相垂直的方向上存在有位置约束,与它相切的方向上存在有力约束. 这两类约束把手爪运动的自由度分成两组正交的集合,我们必须根据不同的准则对这两组集合进行控制.

图 4.15 分别画出了两个具有代表性的任务以及与其相关的自然约束. 对于每一种情况,我们都用所谓的约束坐标系 $\{C\}$ 来描述任务. $\{C\}$ 的位置随任务位置而定,即 $\{C\}$ 可能在环境中固定不动,也可能随着机械手的手爪运动. 如图 4.15 (a) 所示,约束坐标系依附于曲柄,并且随着曲柄移动,而 X 方向总是指向曲柄的轴尖,当手爪(依靠摩擦)紧紧握住曲柄手把的时候也能摇着曲柄

转动．在图 4.15 (b) 中，约束坐标系依附于螺丝刀的顶端，工作时它随着螺丝刀转动．需要注意，由于螺钉上有槽，要避免沿着槽的 Y 方向上产生滑动，因为这个方向上的力被约束为零．在上述这两个例子里，整个工作过程中两组约束保持不变．在比较复杂的情况下，一个任务可以分解成若干个子任务，对于每个子任务的自然约束集保持不变．

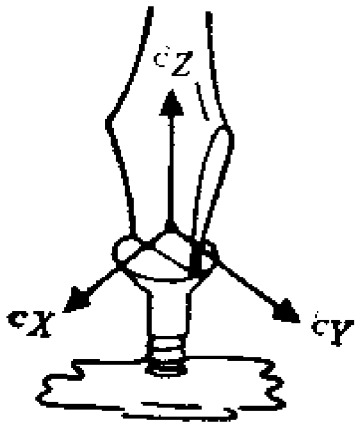
(a)摇手柄



自然约束

$v_x=0$　　$f_y=0$
$v_z=0$　　$n_z=0$
$\omega_x=0$
$\omega_y=0$

(b)拧螺丝刀



自然约束

$v_x=0$　　$f_y=0$
$\omega_x=0$　　$n_z=0$
$\omega_y=0$
$v_z=0$

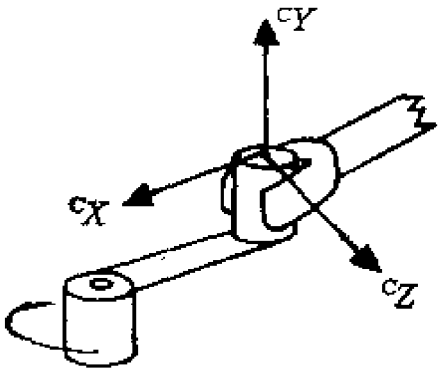
图 4.15　两种不同任务的自然约束

在图 4.15 中，位置约束是用坐标系 $\{C\}$ 中手爪速度 v 的各个分量描述的．当然我们也可以直接用位置量来表示，但是，在很多

情况下用"速度等于零"来表示位置约束会简单一些．类似地，力的约束可用作用在手爪上的力/力矩向量 $\mathcal{F}$ 的各个分量表示．需要说明一下，当我们说位置约束的时候其中也包含了姿态约束，当我们说力约束的时候也包含了力矩约束．这里用自然约束这个词是因为当手爪(或工具)接触外界环境的时候自然生成了这些约束条件，它们与预期的机械手运动毫无关系．

还有另外一种约束，称作"人为约束"．这种约束用来描述预

(a)摇手柄



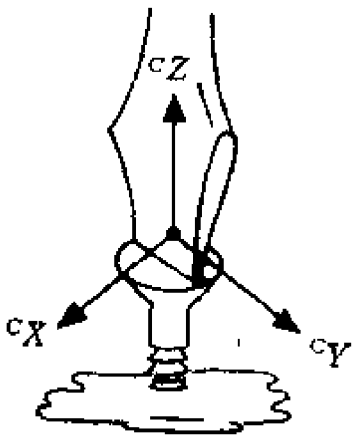
自然约束

$v_x=0$　　$f_y=0$
$v_z=0$　　$n_z=0$
$\omega_x=0$
$\omega_y=0$

人为约束

$v_y=0$　　$f_x=0$
$\omega_z=\alpha_1$　　$f_z=0$
$n_x=0$
$n_y=0$

(b)拧螺丝刀



自然约束

$v_x=0$　　$f_y=0$
$\omega_x=0$　　$n_z=0$
$\omega_y=0$
$v_z=0$

人为约束

$v_x=0$　　$f_x=0$
$\omega_z=\alpha_2$　　$n_x=0$
$n_y=0$
$f_z=\alpha_3$

图 4.16　两项任务的自然约束和人为约束

期的运动或者施加的力．这就是说，每当描述预期的位置或者力的轨迹时，就要定义一组人为的约束条件．这种约束的作用方向与自然约束不同，人为的力约束条件是沿着广义接触面的垂直方向，显然，这样定义才能与自然约束保持一致．

图 4.16 给出了两个任务的自然约束和人为约束坐标系．从图中可以看到，当对 $\{C\}$ 中某个自由度给定一个位置的自然约束时，就应该相应地说明一个力的人为约束，反之亦然．约束坐标系中任何一个自由度都要受到相应的控制，以便满足位置或者力的约束．

下面简单介绍装配策略问题．装配策略指的是规划好的人为约束序列，按照这个序列，任务将按预期的方式运行．在这种策略中必须能够检测接触状况的变化，这样才能跟踪自然约束条件的变迁．随着自然约束条件的变化，装配策略调用新的人为约束条件，并且由控制系统实施．关于装配任务约束条件的自动选择方法，是需要另外研究的问题．在后面的有关讨论中，我们假定已经对任务进行了分析，确定了自然约束条件，并且假定人们已经选择好了用于控制机械手的装配策略．

另外，在我们分析任务的过程中，忽略了接触面之间的摩擦力，这并不影响我们讨论问题．通常滑动摩擦力作用在位置受控的方向上，从位置伺服的角度可以把它看作一种扰动，控制系统可以克服它的影响．

例 4.5 把一根圆棒插进一个圆孔中，这是一种很简单的装配操作，图 4.17 画出了这种装配的动作序列：(a) 机械手的手爪拿着圆棒朝着工件运动；(b) 在圆孔的左边接触了工件表面；(c) 手爪沿着工件表面滑动，直至圆棒与孔对准；(d) 往圆孔内插圆棒，直至与圆孔底部接触后停止动作，整个装配操作结束．图中每一种接触状况可以定义为一个子任务，然后分别给出自然约束和人为约束，同时还可以说明操作过程中系统如何敏感出自然约束

条件的变化。

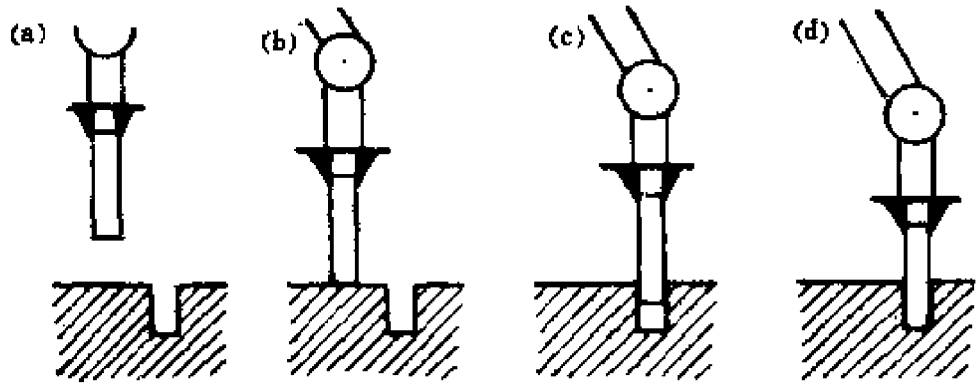


图 4.17　插棒动作序列中的四种接触情况

对于图 4.17(a)，由于圆棒还没有与工件接触，因而它的运动没有受到限制，自然约束条件可表为

$$^{C}\mathcal{F} = 0 \tag{4-42}$$

在这种情况下，手爪是可以全方位运动的。根据当前的任务要求，人为约束条件即为控制圆棒沿着 ^{C}Z 方向，朝着工件表面运动。例如

$$^{C}v = \begin{bmatrix} 0 \\ 0 \\ v_{接近} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4-43}$$

其中 $v_{接近}$ 是圆棒朝工件方向运动的速度。

对于图 4.17(b)，圆棒接触到了工件表面。这时如果敏感出 ^{C}Z 方向上的力超过了某一阈值，就认为发生了接触，这意味着现在存在了一种新的自然约束，即圆棒不能再沿着 ^{C}Z 方向自由运动，也不能自由地绕 ^{C}X 和 ^{C}Y 轴转动。在另外三个自由度上不能自由地施加力，于是自然约束条件可表为

$$\begin{cases} {}^{c}v_{Z}=0 \\ {}^{c}\omega_{X}=0 \\ {}^{c}\omega_{Y}=0 \\ {}^{c}f_{X}=0 \\ {}^{c}f_{Y}=0 \\ {}^{c}n_{Z}=0 \end{cases} \tag{4-44}$$

当前的人为约束条件要体现这样一种装配策略：从下面给出的人为约束条件，我们可以体会出，使圆棒沿着工件表面朝 ${}^{c}X$ 方向滑动，在滑动的同时，在 ${}^{c}Z$ 方向要施加一个小小的力 $f_{接触}$，以便确保圆棒与工件表面的接触。于是我们有

$$\begin{cases} {}^{c}v_{X}=v_{滑动} \\ {}^{c}v_{Y}=0 \\ {}^{c}\omega_{Z}=0 \\ {}^{c}f_{Z}=f_{接触} \\ {}^{c}n_{X}=0 \\ {}^{c}n_{Y}=0 \end{cases} \tag{4-45}$$

式中 $v_{滑动}$ 为圆棒沿工件表面滑动的速度。

对于图 4.17 (c)，圆棒已经在圆孔中插入了一小段距离。这时，如果通过量测发现圆棒沿 ${}^{c}Z$ 方向的速度超过了某一阈值（理想的情况下应为非零值），就可以认为插棒任务达到了图 4.17(c) 所示状态。至此，自然约束发生了变化，进而控制策略（体现在人为约束之中了）也必须改变。新的自然约束为

$$\begin{cases} {}^{c}v_{X}=0 \\ {}^{c}v_{Y}=0 \\ {}^{c}\omega_{X}=0 \\ {}^{c}\omega_{Y}=0 \\ {}^{c}f_{Z}=0 \\ {}^{c}n_{Z}=0 \end{cases} \tag{4-46}$$

人为约束条件选择为

$$\begin{cases} {}^{c}v_Z = v_{插入} \\ {}^{c}\omega_Z = 0 \\ {}^{c}f_X = 0 \\ {}^{c}f_Y = 0 \\ {}^{c}n_X = 0 \\ {}^{c}n_Y = 0 \end{cases} \tag{4-47}$$

其中，$v_{插入}$ 是把棒插进圆孔的速度。

最后，要检测图 4.17(d) 所示状态。很明显，当沿 ${}^{c}Z$ 方向的力增加到超过某个阈值的时候，就认为达到了这种状态。

从上面的例子可以发现一个很有意思的规律：自然约束的每一次变化，总是通过监测当时没有受到控制的位置或力的变量来检测的。例如，为了检测观察从图 4.17 (b) 到 (c) 的自然约束变化，我们监测的是 ${}^{c}Z$ 方向上的速度量，而当时的控制量却是沿 ${}^{c}Z$ 方向的力；又如，为了检测圆棒是否已经插到圆孔的底部，我们监测的是 ${}^{c}f_Z$，然而当时的控制量却是 ${}^{c}v_Z$。

对于综合的部件装配任务，装配策略的确定是相当复杂的，尤其在考虑到摩擦力作用的情况下更是如此，在机器人装配策略的自动规划系统中，这是一个需要研究的实际问题。

§ 4.4 机械手的位置/力混合控制

一、位置/力混合控制方式

图 4.18 表示了机械手的手爪与外界接触的两种极端状态。在 (a) 中，机械手可以在空间自由运动。这时，自然约束完全是关于接触力的约束，即由于手爪与外界没有力的互相作用，所以全部的力约束条件都为零。对于有六个自由度的手臂来说，可以在位置的六个自由度上运动，但是我们不能在任何方向上施加力。而

(b) 说明了另一种极端情况，手爪紧紧固定在墙上．这时，机械手要满足六个位置自然约束，因为这时不能自由地改变它的位置，然而机械手可以自由地在六个自由度上向对象施加力和力矩．

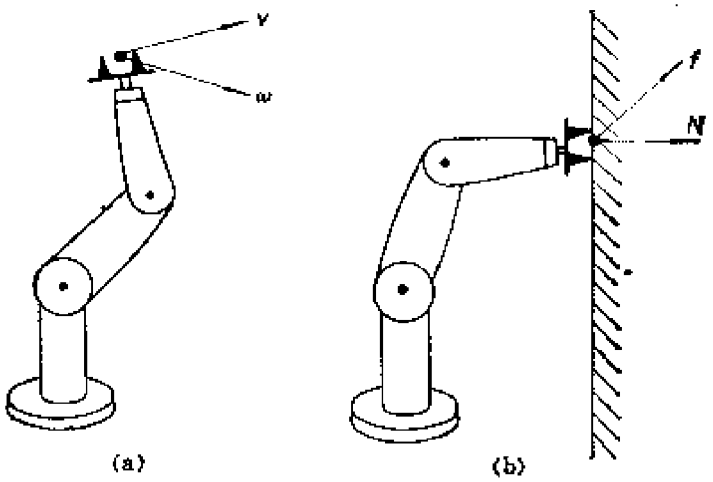


图 4.18　手爪与外界接触的两种极端状态

我们可以把前面讨论的位置控制问题应用于图 4.18 (a) 的情况．由于图 4.18 (b) 的情况在实际中很少出现，因而我们可以把它看作局部约束任务的情况，其中，有些自由度服从于位置控制，而另一些自由度则服从于力控制．这样就需要采用一种位置/力混合控制的方式．

位置/力混合控制器必须解决下述三个问题：

(1) 对于机械手存在力自然约束的方向进行位置控制；

(2) 对于机械手存在位置自然约束的方向进行力控制；

(3) 在任意约束坐标系 $\{C\}$ 的正交自由度上实现位置与力的任意混合控制．

二、“质量-弹簧”的力控制

回顾位置控制的讨论，我们从研究单个物体的位置控制问题出发，尔后把 n 个关节的整个机械手的控制问题等效为 n 个独立

物体的控制问题。类似地，这里也首先讨论单自由度的力控制问题。

在考虑接触力的时候，必须设计某种环境模型。为使概念明确，这里使用很简单的弹簧模型表示受控物体和环境之间的接触作用，即，假设系统是刚性的，而环境具有的刚度为 k_e。

现在讨论图 4.19 所示连有弹簧的单个物体的控制。考虑到未知的干扰力，我们用 $f_{干扰}$ 表示摩擦力或者机械手传动装置的齿槽效应力。希望控制的变量是作用在环境，即弹簧上的力 f_e，而且

$$f_e = k_e x \tag{4-48}$$

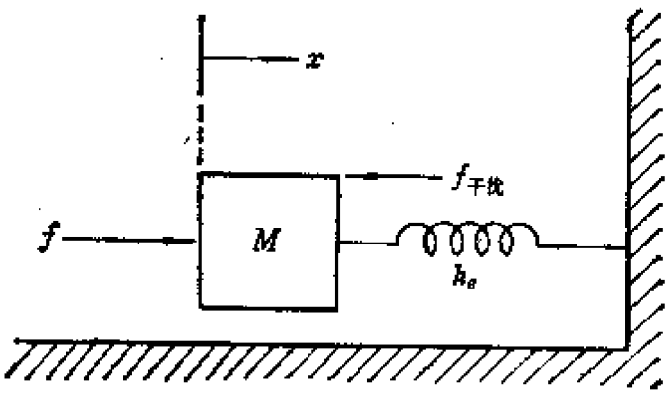


图 4.19 "质量-弹簧"系统

描述这一物理系统的方程为

$$f = m\ddot{x} + k_e x + f_{干扰} \tag{4-49}$$

如果用控制变量 f_e 表示，则有

$$f = mk_e^{-1}\ddot{f}_e + f_e + f_{干扰} \tag{4-50}$$

运用控制法则的分解方法，选定

$$\begin{cases} \alpha = mk_e^{-1} \\ \beta = f_e + f_{干扰} \end{cases} \tag{4-51}$$

得出伺服法则

$$f = mk_e^{-1}[\ddot{f}_d + k_{v1}\dot{e}_1 + k_{p1}e_1] + f_e + f_{干扰} \tag{4-52}$$

式中 $e_f = f_d - f_e$，即为期望力 f_d 与在环境检测出的力 f_e 之间的误差。如果能计算式(4-52)，当有闭环系统

$$\ddot{e}_f + k_{vf}\dot{e}_f + k_{pf}e_f = 0 \tag{4-53}$$

但是，由于 $f_{干扰}$ 是未知的，因而式(4-52)并不可行。当然也可以在伺服法则中简单地舍去这一项，但是稳态分析表明，还有更好的解决办法，特别是当环境的刚度 k_e 很高(常常如此)的时候，我们可以设法改写伺服法则，使之做到可行而又实用，下面具体讨论这一过程。

如果我们舍去 $f_{干扰}$ 这一项，使式(4-50)与式(4-52)相等，而且通过把各阶导数设为零，进行稳态误差分析可以得到

$$e_f = -\frac{f_{干扰}}{\alpha} \tag{4-54}$$

其中 $\alpha = mk_e^{-1}k_{kf}$，为有效的力反馈增益。然而，如果我们在伺服法则式(4-52)中用 f_d 代替 $f_e + f_{干扰}$ 这一项，稳态误差为

$$e_f = -\frac{f_{干扰}}{1+\alpha} \tag{4-55}$$

一般情况下环境是刚性的，α 比较小，所以由式(4-55)计算出来的稳态误差比由式(4-54)计算出的误差小，这是一个明显的

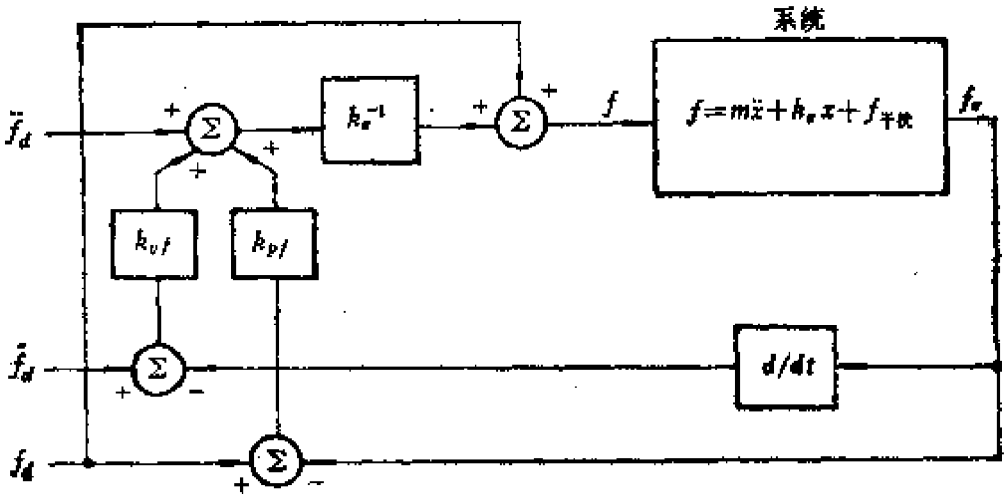


图 4.20 "质量-弹簧"力控制系统方块图

改进。为此，我们提出下式表示的伺服法则，即

$$f = mk_e^{-1}[\ddot{f}_d + k_{vf}\dot{e} + k_{pf}e_f] + f_d \qquad (4\text{-}56)$$

图 4.20 画出了使用伺服法则式 (4-56) 的闭环系统方块图。

图 4.20 只表示一个原理框图，在实际应用中往往会有比较大的变化。首先，力的轨迹通常都是一些常数，即实际应用中常把接触力控制为某个常数值，而很少把它设置为任意的时间的函数。因此控制系统的 $\dot{f}_d$、$\ddot{f}_d$ 两项输入一般都为零。另一个实际问题是敏感出的力噪声很大，因此用数值微分的方法计算 $\dot{f}_e$ 不太方便。由于 $f_e = k_e x$，我们可以由此获得作用于环境的力的导数 $\dot{f}_e = k_e\dot{x}$。这样做非常现实，因为绝大多数机械手都能很好地量测速度。根据以上两个实用的选择，我们可以把伺服法则写成

$$f = m[k_{pf}k_e^{-1}e_f - k_{vf}\dot{x}] + f_d \qquad (4\text{-}57)$$

相应的方块图如图 4.21 所示。

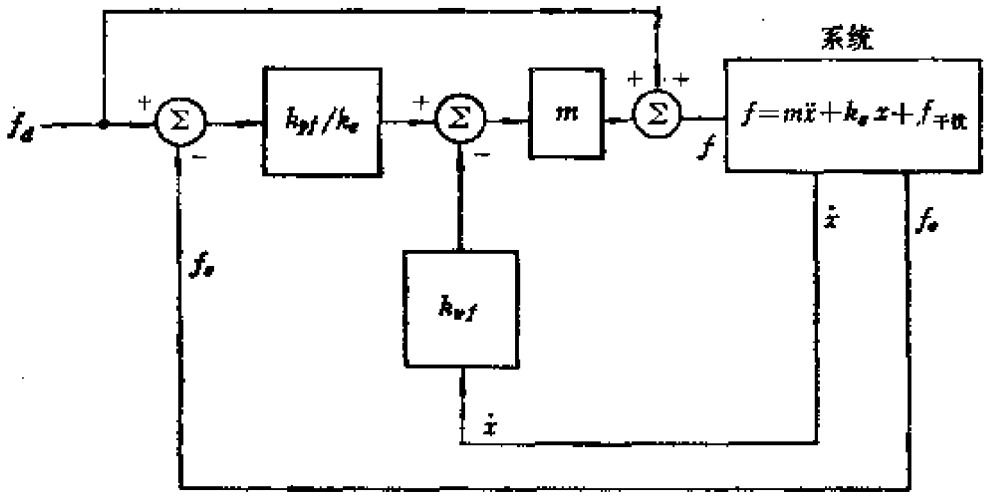


图 4.21　一个实际的"质量-弹簧"力控制系统

还有一个很重要的问题需要再说明一下。伺服法则所用到的环境刚度 k_e 在实际系统中往往是未知的，或许它还是时变的。但由于装配机器人的处理对象常常是刚性部件，因此可以认为 k_e 相当高。一般情况下都作这种假设，在选择增益的时候，要考虑到 k_e

变化的情况下系统能够正常地工作.

三、机械手的位置/力混合控制系统

这部分介绍一种控制系统的结构,作为机械手位置/力混合控制器的实现方案.

1. 与约束坐标系轴向一致的直角坐标机械手

首先讨论如图 4.22 所示的简单情况. 假定有一台三自由度机械手,每个关节都是柱关节,轴线分别沿 Z、Y 和 X 方向,而且每个连杆的质量都为 m, 连杆滑动时摩擦力为零. 还假设关节的运动方向完全与约束坐标系 $\{C\}$ 的轴向一致. 手爪与刚度为 k_e 的表面接触, 作用在 ${}^{C}Y$ 方向上. 所以在 ${}^{C}Y$ 方向需要进行力控制, 而在 ${}^{C}X$ 和 ${}^{C}Z$ 方向则需要进行位置控制.

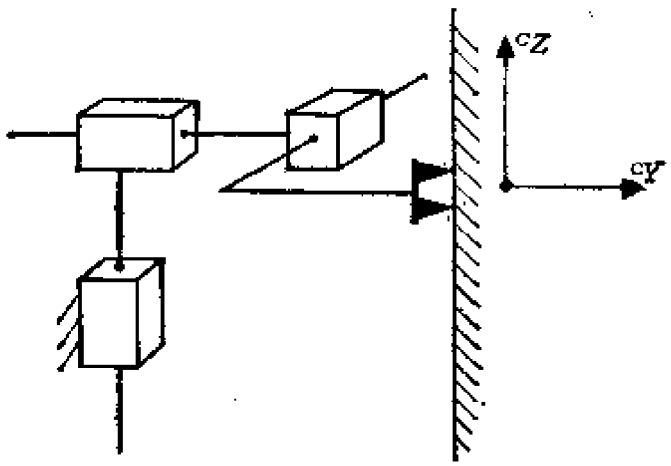


图 4.22 三自由度直角坐标机械手与环境表面接触

这种情况下,位置/力混合控制问题的解比较清楚. 关节 1 和关节 3 应该使用 § 4.1 讲的单位质量系统位置控制器, 关节 2 (作用于 Y 方向) 应该使用前面讲到的力控制器. 于是我们可以在 X 和 ${}^{C}Z$ 方向设定位置轨迹, 而在 ${}^{C}Y$ 方向独立地设定力的轨迹 (或许是一个常数).

往往有这样的情况, 对于机械手的某个自由度有时候需要进

行位置控制，有时候又需要进行力控制，这样，就要有相应的直角坐标机械手控制器。它对于机械手的每个自由度既能进行位置控制，又能进行力控制。由于不可能在任一时刻同时满足这六个约束，因而可以设置一个工作模式，用来指明每个自由度在给定时刻究竟跟踪什么样的轨迹。

图 4.23 画出了三自由度直角坐标机械手的混合控制器方块图。三个关节既有位置控制器，又有力控制器，图中引入的 3×3 对角阵 S 和 S' 实际上是两组开关，用来设置约束坐标系 $\{C\}$ 中每个自由度所要求的控制模式。如要求对第 i 个关节进行位置控制（或力控制）则矩阵 S（或矩阵 S'）对角线上第 i 个元素为 1，否则为零，反之亦然。例如相应于图 4.21 的 S 和 S' 应为

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

与 S 的设置相一致，系统总是有三个轨迹分量受控，这三个分量由位置轨迹和力轨迹任意组合而成。当系统的某个关节以位置控制（力控制）方式工作的时候，这个关节的力控制（位置控制）的误差信息就被忽略掉。

图 4.23 所示混合控制器是针对关节轴线与约束坐标系轴向完全一致的特定情况，下面我们将使用类似于前面讨论的方法，把这一特定情况推广到一般的机械手，而且要适用于任意约束坐标系 $\{C\}$，使得在理想情况下，系统运行时机械手好象有一个与约束坐标系 $\{C\}$ 中的每一个自由度都一致的驱动器。

2. 一般机械手的混合控制器

要把图 4.23 所示的混合控制器推广到一般机械手，可以直接

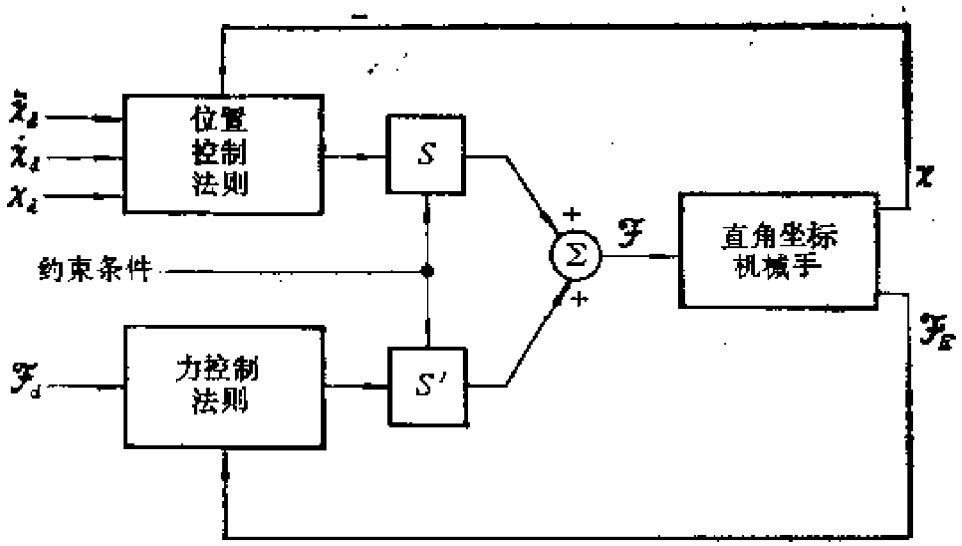


图 4.23　三自由度直角坐标机械手的混合控制器

使用基于直角坐标控制的概念．基本思想是，通过使用第二章中介绍的直角坐标空间的动力学模型，有可能把实际机械手的组合系统以及计算模型等效为一组独立的、没有耦合的单位质量系统，一旦完成了解耦和线性化的工作，我们就可以运用本节前面介绍的简单的伺服系统．

图 4.24 说明了基于直角坐标空间的机械手动力学的解耦形式，机械手看起来象一组没有耦合的单位质量系统．为了用于混合控制方案，直角坐标动力学的各项以及雅可比矩阵都在约束坐标系 $\{C\}$ 中描述，运动学方程也相对于 $\{C\}$ 进行计算．

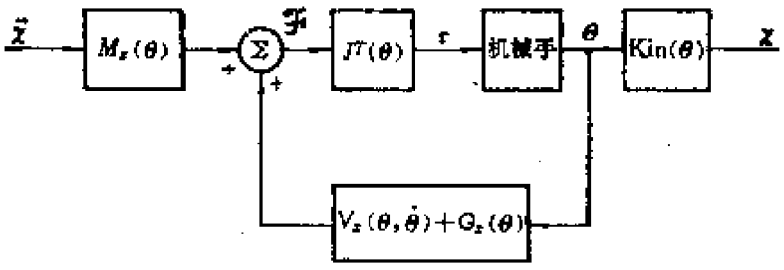


图 4.24　直角坐标解耦型式

由于前面已经为与约束坐标系一致的直角坐标机械手设计了

混合控制器(图 4.23)，而且直角坐标解耦型式给我们提供了具有同样的输入-输出特性的系统，现在我们只要把二者结合起来，就可以生成一般的位置/力混合控制器，结果方块图如图 4.25 所示.

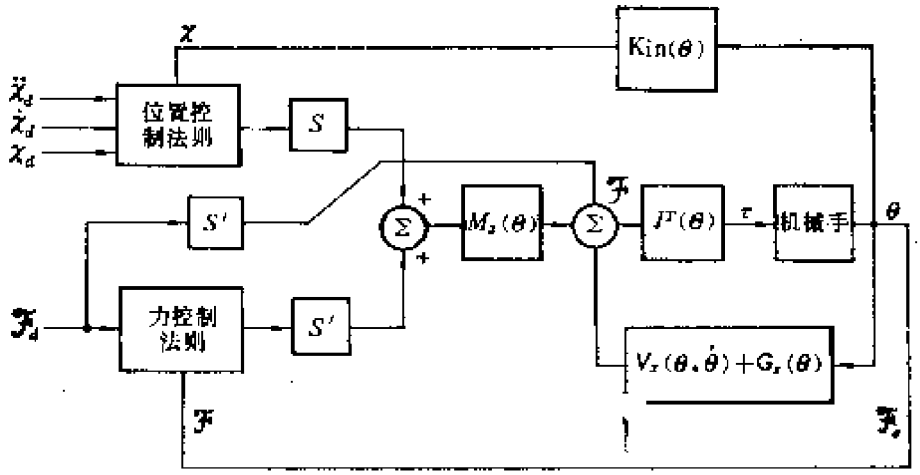


图 4.25　一般机械手的位置/力混合控制器(略去速度反馈闭环)

不妨再次说明一下，图 4.25 中，动力学各项、雅可比阵都在约束坐标系 $\{C\}$ 中描述；运动学方程、以及敏感力都要变换到 $\{C\}$ 中，伺服误差也要在 $\{C\}$ 中计算；当然，还要对 S、S' 作适当的取值.

图 4.25 所示的系统原理已经在 PUMA 560 上实现 (Stanford 大学).

3. 伺服刚度的控制

对某个自由度进行严格的位置控制或者力控制，就伺服刚度的范围而论，这是两种控制极端. 即，理想的位置伺服控制的刚度为无穷大，从而抑制了作用于系统所有的力干扰；而理想的力伺服的刚度则为零，从而不论位置干扰多大，系统都维持期望的力. 如果能够控制手爪的刚度，使它或者为零，或者为无穷大，那将是很有用的. 一般情况下，我们可能希望控制手爪的"机械阻抗".

在分析手爪与环境接触的时候，我们曾经假设环境是刚性的.

如果我们对于接触环境为高刚度的情况采用零刚度的力控制，而对于接触环境为零刚度的情况(在空间自由运动)则采用高刚度的位置控制，即总是用与环境刚度相反的值作为手爪的伺服刚度，这种做法也许是一种比较好的控制策略。这样，在处理象塑料或者弹簧这类部件的时候，伺服刚度的设置值就应选为零与无穷大以外的适当值。在混合控制器中这一点是很容易做到的，只要降低{C}中某个自由度相应的位置控制增益就行了。如果这样做了，一般情况下速度增益也会降低，从而使那个自由度保持为临界阻尼状态。

四、柔顺控制

机器人的力控制很难实现，其中的主要问题是：计算量相当大，动力学模型缺乏完整、准确的参数，缺乏可靠的力传感器以及难于拟定位置/力的控制策略等等。下面介绍几种当前在机器人中实用的简单的方法。

1. 被动式柔顺控制

位置伺服刚度很大的机械手并不适合被操作部件相互接触从而产生作用力的任务，以免造成部件的挤压损坏，例如装配任务就是这种情况。于是，实际运用中就设法求助于部件或机械手本身的柔顺性，从而有可能在一定程度上完成那些任务，这样就需要专门设计一些柔顺的装置。最成功的是所谓的 RCC (Remote Center Compliance) 装置 (MIT Draper 实验室研制)，它的主要构成是在机械手的腕与手爪之间插入的一个六自由度的弹簧装置。每个自由度上有一根弹簧，通过选择适当的刚度，使得与整个装置相连的手爪具有一定的柔顺性。这样，在进行某些装配操作时，可使动作平滑、迅速而又不致损伤部件。

2. 利用软化位置增益的柔顺控制

被动式柔顺控制中，柔顺性的实现采用被动的、固定的方式，

其实通过调节位置系统的增益，也可能使机械手的"视在刚度"得到主动的改变，表现出一定的柔顺性．这种方式可用于类似于研磨的操作，机械手只需要保持与操作表面的接触，而不需要很精细的力控制．

有一种方法(由 J. K. Salisbury 提出)很值得介绍：在基于关节的伺服系统中改变位置增益，使得手爪沿着直角坐标的自由度表现出一定的刚度．具体地说，假设整个机械手可看作一个有六个自由度的弹簧，它的特性可描述为

$$\mathcal{F} = K_{px}\delta\mathcal{X} \tag{4-58}$$

式中 K_{px} 为 6×6 对角阵，对角线上元素依次为弹簧的三个线性刚度和三个扭转刚度，对于机械手，即为手爪表现出来的三个平移刚度和三个旋转刚度．

回顾机械手雅可比阵的定义，有

$$\delta\mathcal{X} = J(\Theta)\delta\Theta \tag{4-59}$$

将上式代入式 (4-58) 可得

$$\mathcal{F} = K_{px}J(\Theta)\delta\Theta \tag{4-60}$$

考虑到静态力，有

$$\tau = J^T(\Theta)\mathcal{F} \tag{4-61}$$

将式 (4-60) 代入式 (4-61)，得到

$$\tau = J^T(\Theta)K_{px}J(\Theta)\delta\Theta \tag{4-62}$$

其中的雅可比阵通常是在工具坐标系中写的．式 (4-62) 表明了关节角度的微小变化 $\delta\Theta$ 与应该产生的关节力矩 τ 之间的函数关系．

由于简单的关节位置控制法则一般为

$$\tau = K_pE + K_v\dot{E} \tag{4-63}$$

其中，K_p、K_v 为常值对角型增益矩阵，E 为伺服误差，定义为 $\Theta_d - \Theta$，因而，结合考虑式 (4-62) 和式 (4-63)，可以提出如下控制法则：

$$\tau = J^T(\Theta)K_{px}J(\Theta)E + K_v\dot{E} \qquad (4\text{-}64)$$

到此，我们通过使用雅可比矩阵，已经把直角坐标刚度转换为关节空间的刚度，也就是说，根据式(4-64)，只要把期望的手爪在直角坐标中表现出来的刚度值对应地代入位置增益矩阵 K_{px}，就可产生相应的关节力矩，实现机械手的柔顺控制。

第五章　机器人计算机控制系统结构

现有机器人计算机控制系统一般采用三种类型的控制结构.

集中控制,即利用一台微型计算机实现系统的全部控制功能.由于是单机控制,因而构造较为简单,也比较经济.但由于控制过程中需要进行坐标变换,特别在需要高次插补时,显然计算量很大,因此这种控制结构速度较慢.

主从控制,系统用主、从两个 CPU 进行控制,主 CPU 用于计算坐标变换、轨迹生成以及系统自诊断等,从 CPU 用于机械手所有关节的动作控制.

分级控制,采用多个微机分上下两级共同完成机器人的控制功能.上面一级主控计算机负责整个系统的管理以及坐标变换和生成轨迹段的运算.下面一级由若干微处理器组成.分管机器人各个关节在轨迹段内的插补运算以及伺服控制处理.由于机器人的不同功能可由不同的计算机并行地完成,因而提高了工作速度和处理能力.

PUMA 系列机器人就具有上述的分级控制结构.在前面的有关章节中对 PUMA 560 曾有所介绍,这一章以美国 Unimation 公司的产品为例,具体说明典型的机器人计算机控制系统结构.

PUMA 560 虽然属于生产线上实用的工业机器人,但也是研制智能机器人系统值得参考的一个重要机型,这一方面在于它的本体——六关节机械手灵活、精密;另一方面在于它的控制语言 VAL II 具有控制命令与位置数据相互独立的特点,因而常常被

用来开发机器人的智能例如视觉、触觉等传感器。只要给定位置输出，即可使机械手作出适应环境的动作反应。

本章不打算深入剖析 PUMA 560 系统，只就系统结构方面给以基本介绍。

§5.1 硬件配置与结构

图 5.1 是 PUMA 560 控制器原理框图，图 5.2 为结构框图。除计算机外部设备和伺服电机以外，原理图中所画出的各部件均安置在控制器中。

一、控制器

在控制器中有如下 12 种部件：

DEC LSI-11 主控计算机。PUMA 560 的初型使用了 LSI-11/02 计算机，在改进过程中，Unimation 公司又先后采用了 LSI-11/23 和 LSI-11/73。

DLV 11-J 串行接口板。对于“VAL PLUS”只用一块，“VAL II”用两块。

存储板 (CMOS)。VAL II 需要 64 kW，而 VAL，VAL PLUS 分别需要 32kW，48kW。

“A”并行接口板。带有 VAL II 语言的引导程序存储器 (EPROM)。

“B”并行接口板。带有时钟。

数字伺服板。共有六块，每块对应于一个关节，且可互换。

功率放大器板。

功率放大器控制板。

非标准输入/输出接口板。

电源板。

大功率功能板。

机械手电缆接插板。

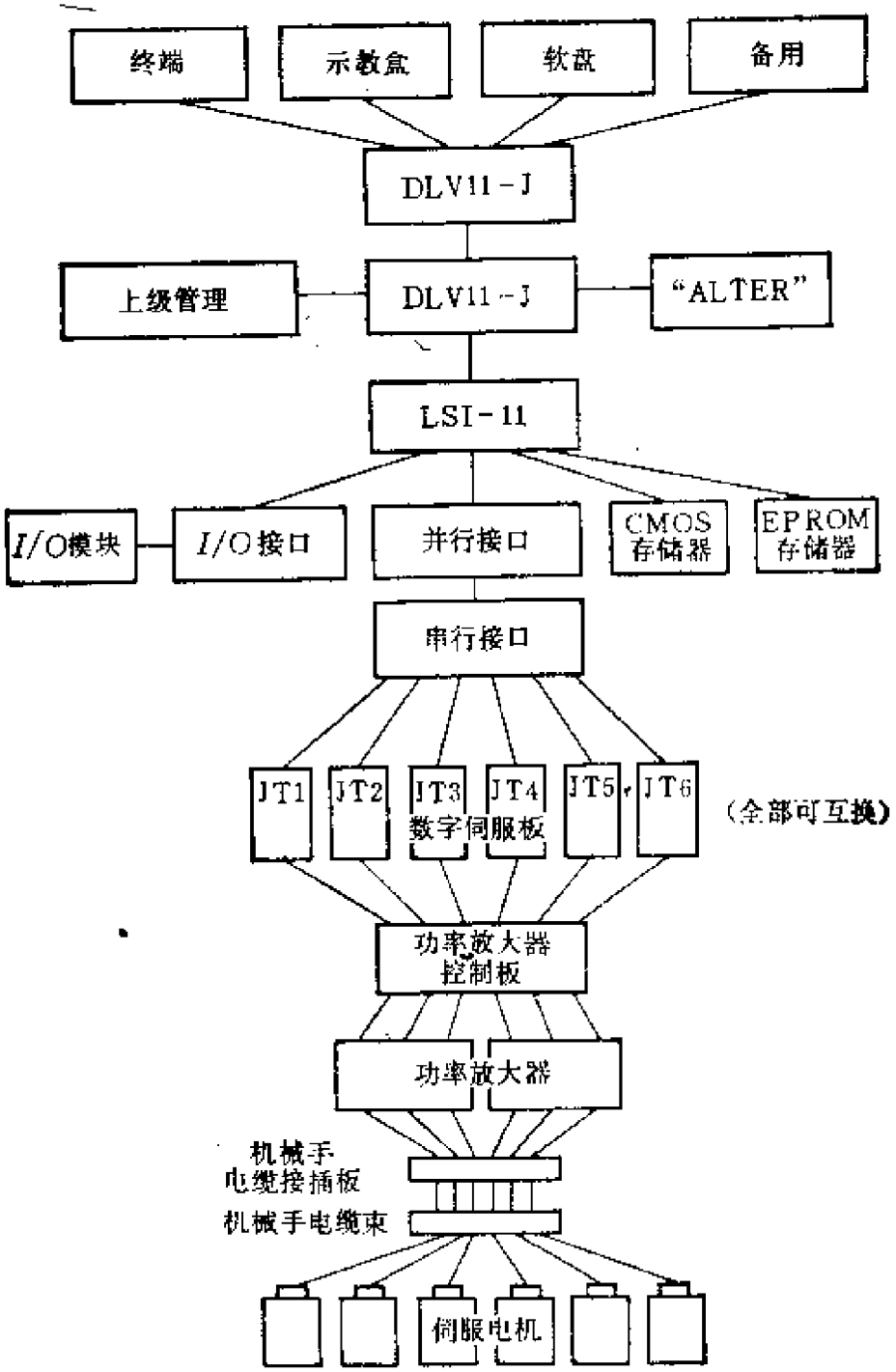


图 5.1 PUMA 560 控制器原理框图

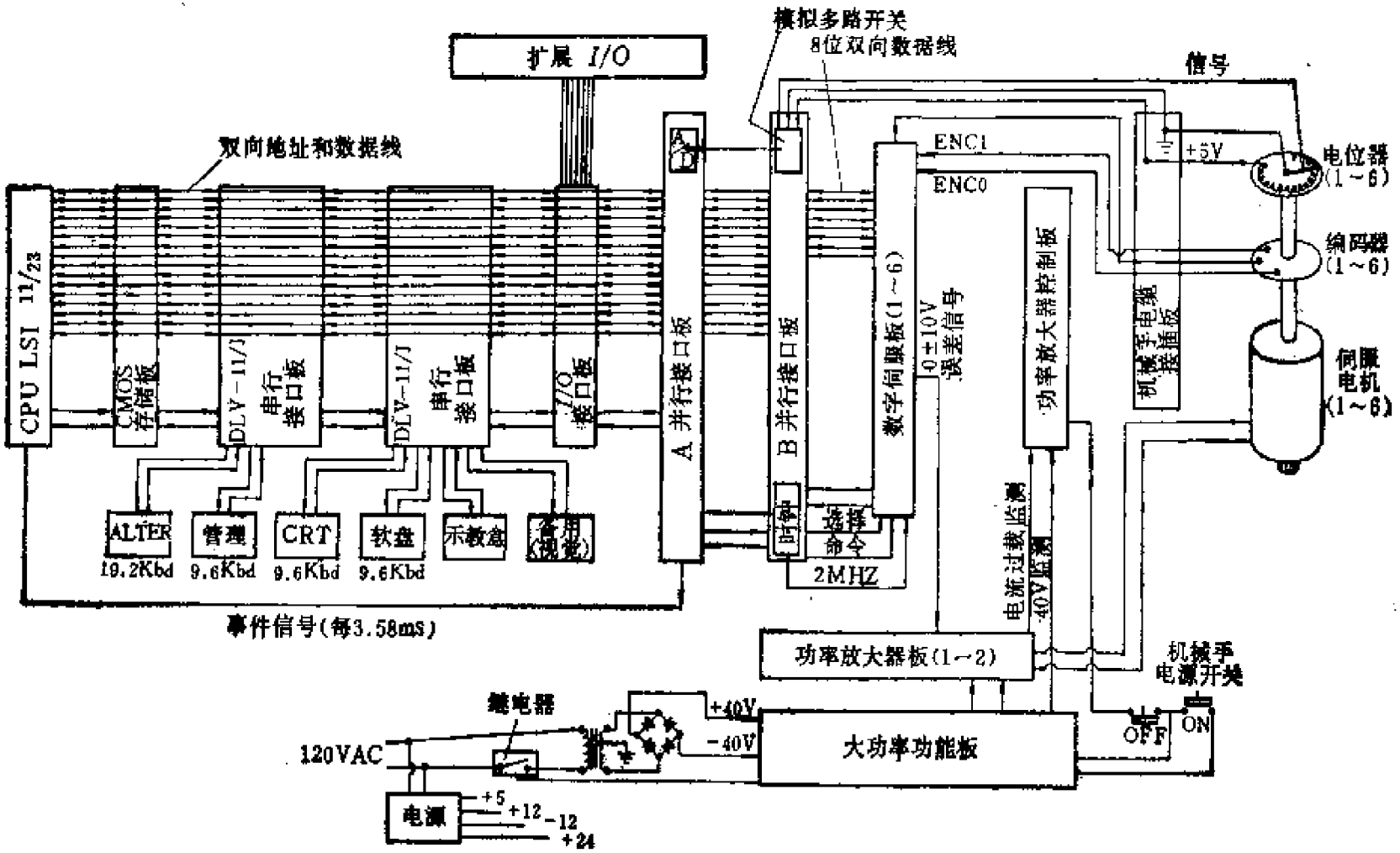


图 5.2 PUMA 560 控制器结构框图

下面具体介绍主要部件的功能。

1. LSI-11 计算机　LSI-11 是一个标准的 DEC 系统，其中包括处理器、存储器和通讯板。系统软件和用户程序存储在 CMOS 存储器中，处理器与外部设备的通讯是通过下述接口完成的。

DLV 11-J 有四个异步串行通讯接口，其中三个分别用作处理器与终端、示教盒和高密度磁盘的通讯。另外一个接口备用，或者与高一级计算机系统联接，接受有关的管理命令或者与其它的计算机联接起来，以便从外部对机械手的运行轨迹进行实时修正按照所谓的“ALTER”模式运行。

LSI-11 通过“A”并行接口板把命令和数据传送到“B”并行接口板，这些信息再由“B”板传送给控制系统的伺服驱动方面，一旦命令和数据得到接受并运行，就以相反的顺序传送给 LSI-11。

2. 数字伺服板　对应于六个关节的六块数字伺服板上都各有一个微处理器，用以对关节进行控制。LSI-11 每隔 28ms 向各数字伺服板送一次位置信息，微处理器把这些数字信息送到数模转换器生成驱动直流伺服电机的模拟信号，数字伺服板使用中断服务子程序把回答信息送到 LSI-11。

3. 功率放大板及其控制板　数字伺服板输出的直流模拟信号

5. 机械手电缆板　机械手的每个关节都装有增量码盘和矗测电位器，用来测量每个关节的位置．码盘和电位器与控制器之间的信息交换通过这块板进行．板上还装有码盘的零线接收器和信号噪声抑制器．

二、外部设备

系统必备的外部设备有终端和示教盒．另外可选择配置软磁盘驱动器、*I/O* 模块和附加存储器．

1. 终端　系统可配备两种终端，屏幕显示终端（CRT）和打印终端（TTY）．通过标准的 RS-232C 接口与系统进行通讯．用户通过屏幕终端编辑用户程序，对手臂进行示教以及与系统交换信息．运行程序时终端可与系统断开．

2. 示教盒　示教盒实际上是用户对于机器人系统的手动控制设备，其通过操纵各个关节，来控制机械手的形态．示教盒有四种示教方式：关节（JOINT）方式，自由（FREE）方式，基坐标（WORLD）方式和工具坐标（TOOL）方式．在示教盒上 RECORD 按钮可以用来把机械手的当前位置、姿态存入存储器．速度旋钮用以设置、调节手动控制时机器人手爪的运行速度．CLAMP 按钮控制手爪的开闭．字母数字显示器的功能是显示当前的工作状态和系统发出的出错信息．

3. 软磁盘驱动器　PUMA 560 使用了 5 in 高密度双面软磁盘，标准的 RS-232C 串行接口把它与主机连接在一起．波特率为 9600 或 2400．

软磁盘存储的主要内容是 VAL II 操作系统以及用户程序和数据．

三、机械手

机械手常称为系统的本体，六个自由度全部为旋转关节．连

接手臂的各个构件的关节如图 5.3 所示，分别为腰（关节 1），肩（关节 2），肘（关节 3），腕 1（关节 4），腕 2（关节 5），腕 3（关节 6）。图中标出了每个旋转轴及其最大旋转范围。

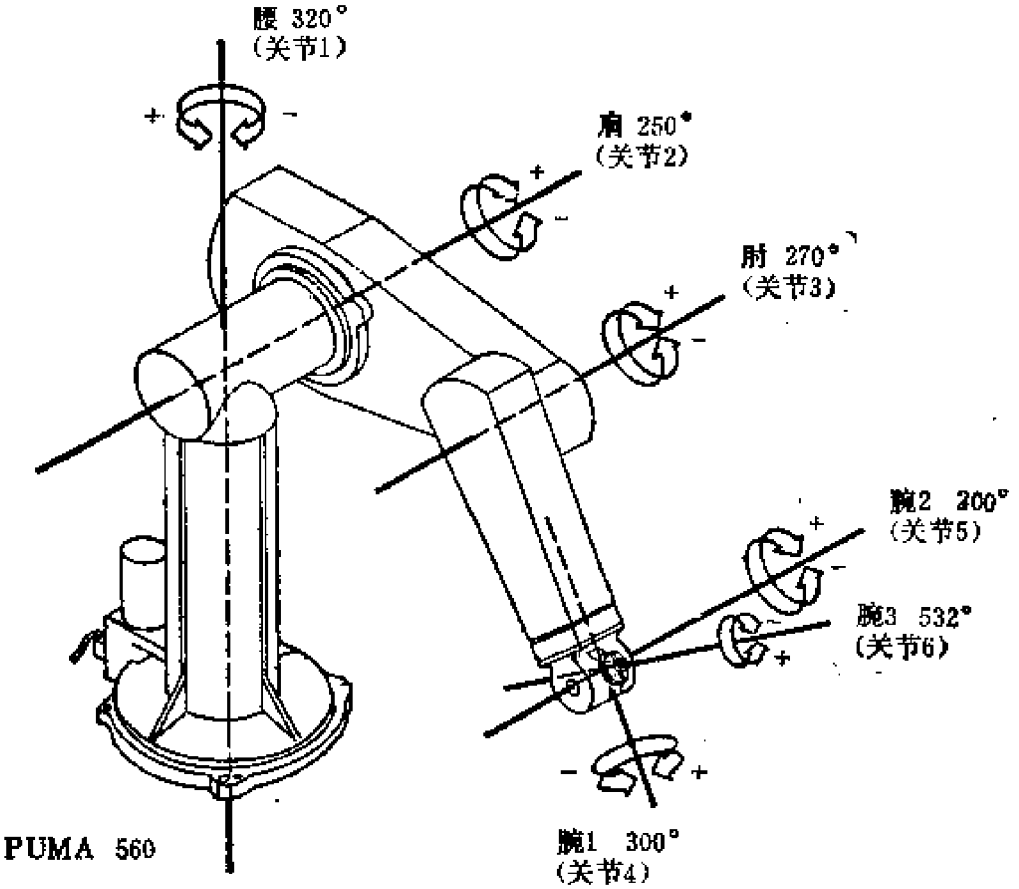


图 5.3 PUMA 560 的关节及其旋转范围

机械手的每个关节各由一个永磁式直流伺服电机以及相应的齿轮转动系统驱动，每个电机有一个光学码盘和一个测量电位器。电机通过 116:1 的齿轮减速器与电位器连接在一起。而码盘与电机同轴连接。

量测电位器的作用是在系统刚一启动时，确定机械手各关节的初始位置。每个电位器都与一套模数转换器 ADC 相连接。关节角度 θ_i 的计算公式可表示为：

$$\theta_i = \beta_i \cdot A_i + \phi_{0i} \quad i = 1, 2, 3, 4, 5, 6 \qquad (5\text{-}1)$$

式中，β_i 是 ADC 计数个数与关节角度的转换率，A_i 是 ADC 的读数，ϕ_{0i} 是 ADC 计数等于 0 时关节的角度。

电位器的精度不高，这一方面由于电位器的线性度不够好，另一方面是由于 ADC 只用 8 位二进制数计数，每计一个数相当于关节旋转 2 到 3 度，而且，ADC 的读数重复性也不高。为此，在实际系统中，往往采用对 ADC 多次重复读数的方法，用读数的总和计算关节角度。例如读 128 次，那么式 (5-1) 中的 β_i 就要取为原值的 1/128。

光学编码器的作用是量测每个关节位移的变化量，进而通过计算还可获得关节位移的速度，系统每 28 ms 从编码器读出关节实际位置，与位置的控制量比较 32 次，经数模变换获得伺服误差信息。编码器由光学增量码盘和读数器组成。关节 3 (肘)的码盘上刻有 200 条径向射线，其余关节的码盘刻有 250 条线，所有的读码器都采用四倍频电路对码盘上的射线进行扫描，并采用 16 位二进制数记录扫描读数。

根据已知的电机传动速比，又由于码盘与电机同轴，我们可

表 5.1 PUMA 560 的转换率 n_i 和 α_i

参数 / 关节 i	n(度/周)	α(度/周)
1	5.7557	5.7557×10^{-3}
2	3.3392	4.1741×10^{-3}
3	6.7098	6.7098×10^{-3}
4	4.7358	4.7358×10^{-3}
5	5.0062	5.0062×10^{-3}
6	4.6918	4.6918×10^{-3}

计算出电机旋转一周与关节角度的转换率 n_i 以及编码器每转一个数与关节角度的转换率 α_i，结果如表 5.1 所示。

由编码器测出的关节角度可根据下式计算：

$$\theta_i = \alpha_i \cdot cnc_i + \theta_{0i} \quad i = 1, 2, 3, 4, 5 \qquad (5\text{-}2)$$

式中，cnc_i 是编码器的计数值，θ_{0i} 是编码器计数为 0 时的关节角度。

对于关节 6（腕 3），由于它与关节 5（腕 2）机械上有耦合，当由编码器量测它的角度时，计算式应表示为

$$\theta_6 = \alpha_6 \cdot cnc_6 + \theta_{06} + \theta_5 \cdot c_f \qquad (5\text{-}3)$$

式中 c_f 为耦合系数，即关节 5 每转 1 度关节 6 跟着旋转的度数，实际系统中 $c_f = -0.180556$。

§5.2 系统功能和通讯

一、主控级功能

主控级由 LSI-11 计算机，通过机器上配置的 VAL II 语言，主要完成下述功能。

1. 系统初始化　当启动系统时，LSI-11 首先运行 VAL II 语言中的初始化程序；把所有的外部设备接口都置成启动状态（RESET）；把 VAL II 语言内部需要的数据字和标志字都赋上初始值；更为重要的是给关节控制级微处理器 6503 的部分内存赋上初始值（即为每个关节先送一组数据）。

初始化程序运行过程中，系统向用户询问

INITIALIZE (Y/N)?

回答“Y”后系统进一步询问

ARE YOU SURE (Y/N)?

若再回答“Y”，系统将把全部内存中原有用户程序和数据清除掉。

2. 标定机械手的初始位置　系统启动后，用户必须首先用 CALIBRATE 命令标定机械手的初始位置，然后才能进行各种操作，以便保证机械手的运行精度。

标定机械手的初始位置实际上就是对机械手的各个关节进行零位校准的过程。系统响应 CALIBRATE 命令后，驱动各个关节旋转一个很小的角度，使关节编码器中的读数器正对码盘的零线，这时通过量测电位器读出对应关节的角度，经过适当的精度修正，转换为编码器的计数值，赋于编码器，标定过程结束。

由于配有量测电位器，PUMA 560 启动时无论处于怎样的形态，都只需在很小的空间范围内稍作移动，就可完成标定工作，无需回归机械零位，这在现场使用中带来了很大的方便。

3. 处理用户命令　一旦用户输入一条命令，VAL II 立即给以分析，确认其合法性然后找出处理相应命令的子程序入口，转入解释执行过程。对于非法的命令，如非法的命令名或缺少变量等，系统回送出错误信息，对于合法命令但系统本身尚不具备运行条件的情况，它也回送出相应的信息。

对于最重要的机械手动作命令，VAL II 将根据命令说明的运动要求进行坐标变换轨迹规划的插补运算，其中轨迹段时间间隔都为 28ms。运算结果参数送往相应关节的数字伺服板，然后还将检测各个关节的运行情况，以确定它们是否正常工作。

4. 系统保护　VAL II 包含许多保护程序用来保护设备的安全运行。例如，当 VAL II 发现某个关节超出了软件允许的运动范围时，立即使机械手停止运动，并输出错误信息 * Fatal out of range * Jt (n)；再如，当量测电位器电路发生故障时，系统将输出 * Bad pot * Jt(n)。

二、关节控制级功能

数字伺服板模拟信号放大器以及功率放大器，构成系统的关

节控制级，其主要功能由数字伺服板上的微处理器 6503 体现：

（1）每个微处理器接收 28 ms 一次的来自 LSI-11 的新的位置信息，并检测、确认这一信息。然后对关节位置的新值（当前轨迹段的终止点）和当前值（当前轨迹段的起始点）进行轨迹段内的插值计算，其中微处理器把 28 ms 内关节应该运动的角度分成了 32 等分，于是轨迹段内每一个小间隔的时间为 0.875ms。微处理器每隔 0.875 ms 还从码盘的寄存器中读出关节的当前位置值，以便下一小间隔的插补计算使用。

（2）修正差值计算获得的误差信息以及相应的增量码盘的阈值。

（3）经 *D/A* 转换器把误差信息转换成电压模拟信号，将这个信号送到模拟信号放大器，驱动相应的关节运动。

三、主控级与关节控制级的通讯

1. 通讯约定　PUMA 560 系统一经启动，主控级计算机 LSI-11 就不停地与关节控制级的微处理器交换信息，直到系统停止运行。两级之间的通讯，LSI-11 是主动的。在交换信息之前，它都要按照通讯约定先送有关命令再进行操作。

"A" 并行接口和 "B" 并行接口为信息的通讯提供了有效的途径。"A" 并行接口在系统中的物理地址是 176770（对于 64 KB 内存）或 776770（对于 128KB 内存）。它有三个寄存器，控制状态寄存器 DRCSR，输出缓冲区 DROUT 和输入缓冲区 DRIN。

LSI-11 与关节控制级的通讯有三种类型：（1）往关节写数据，（2）从关节读数据，（3）读或写与关节无关的信息。

向微处理器送命令时需要说明这个命令是针对一个关节的还是针对一组关节的。若是针对一组关节的，则在命令的某一位置 "1"，于是从命令指出的某个关节一直到最后一个关节（关节 6）都执行这条命令。

当微处理器已经做好发送数据的准备或是已经接收到数据的时候，DRCSR 寄存器的某一位置“1”. LSI-11 无论读数据或是写数据都要等待、确认这个置“1”位.

2. 读、写关节数据

(1) 读操作

a. 针对一个关节，读一个数据：

```
procedure ReadSingle (int cmd, data);
begin "ReadSingle"
DRCSR ← 0;
DROUT ← cmd;
DRCSR ← 1;
while READYBIT = 1 do {READYBIT 表示 DRCSR
                       的二进制第七位}
        ;
data ← DRIN;
end "Read Single";
```

b. 针对一组关节，读一组数据：

```
procedure Read Vector (int cmd; int array data);
int j;
begin "ReadVector"
DRCSR ← 0;
DROUT ← cmd ∨ SEQBIT {SEQBIT 表示 DROUT 的
                      二进制第七位}
DRCSR ← 1;
for j ← 1 until 6 do
    begin
    while READYBIT = 1 do
          ;
```

```
        data [j] ← DRIN;
        end;
    end "ReadVector";
```

(2) "写"操作

a. 针对一个关节"写"一个数据:

```
    procedure WriteSingle (int cmd, data);
    begin "WriteSingle"
    DRCSR ← 0;
    DROUT ← cmd;
    DRCSR ← 1;
    DROUT ← data;
    while READYBIT = 1 do
            ;
    end "WriteSingle";
```

b. 针对一组关节"写"一组数据:

```
   procedure WriteVector (int cmd; int arraydata);
    int j;
    begin "WriteVector"
    DRCSR ← 0;
    DROUT ← cmd ∨ SEQBIT;
    DRCSR ← 1;
    for j ← 1 until 6 do
        begin
        DROUT ← data [j];
        while READYBIT = 1 do
              ;
        end;
   end "WriteVector";
```

表 5.2 读写关节数据命令表

命令码	命令名	功能
00X(20X)	POSMODE	位置模式. 送往微处理器的数据将被看作16位没有符号位的位置信息，伺服系统将把相应关节驱动到这个数据所表示的位置上.
01X(21X)	CURMODE	电流模式. 送往微处理器的数据将被看作为16位带有符号的数据,送往12位的数模转换器去控制驱动伺服电机的电流. 输往数模转换器的数据用二进制补码表示，正数产生正方向的力矩，负数产生反方向的力矩. 由于数模转换用了12位，所以八进制数3777将产生最大的正力矩,而4000产生最大的负力矩.
02X(22X)	SPOSTL	位置误差. 送入的数据将被存储起来作为位置误差. 微处理器驱动关节到达新的位置，其误差必须在这个数据所允许的范围之内.
03X(23X)	SETPOS	设定位置. 写进的数据将被存储起来作为当前关节的位置. 即再读关节的位置时将读出这个数据,该命令仅为编码器提供一个数据,用于标定机械手的绝对位置而不能用于驱动关节.
04X(24X)	CALIB	定标命令. 功能同于 SETPOS.
05X(25X)	SETDC	设置直流补偿值. 当关节以位置模式 (POSMODE) 工作时,输入的数据将被存起来并经伺服回路逐次把它加到 *D/A* 转换器 (VALII 没有使用这条命令).
06X(26X)	SETINT	设置积分域容限. 微处理器驱动关节运动时用到这个值. 如果关节当前的位置距终点的误差在积分容限的范围之内,而且容许积分,则执行位置积分.
07X(27X)	STDATA	存储数据. 把数据的低八位存储到微处理器的内存中. 数据的高八位就是存储数据的内存地址. 该命令一般用于设置关节的状态字，有时也用它设置伺服常数.
10X(30X)	STOPMDE	停止运行模式. 强迫指定关节中止运动.
14X(34X)	READPOS	读位置计数器. 读出关节当前的位置，输出16位数值.
15X(35X)	READSTAT	读状态字. 读出每个微处理器状态字的内容.
16X(36X)	READADC	读 ADC. 读每个关节的 ADC 值,即检测电位器的值. 读出数据的低八位是有用信息，高八位无用.

续表 5.2

命令码	命令名	功能
17X(37X)	DIAGREAD	诊断读.读微处理器内存任何单元的内容,用这条命令之前首先应该使用 STDATA 命令,把这个单元的地址存放到地址为 002 的单元中.

表 5.2 中列出了 LSI-11 送往关节微处理器的命令. 命令码一栏中的 X 是八进制数中的最低位,取值为 0 到 5 的整数,表示关节序号(关节序号的机内表示为 0～5),括号内是针对一组关节的命令码. 单关节命令与一组关节命令之间唯一的差别就是二进制数的第七位,对于前者,这一位为"0",对于后者,这一位为"1". 命令名栏中是相应命令的助记符,后面将用到它们.

3. 读写与关节无关的信息 LSI-11 有一条特殊的命令 SYSRW, 命令码是7,它既可以用作读命令,也可以用作写命令.它读入或写入的都是一个 16 位二进制数,表 5.3、表 5.4 分别说明了两种情况下各位数字的含义.

表 5.3 SYSRW 命令读入信息含义表

位数	标记位含义
bit 0～bit 7	从非标准 I/O 接口接插件读入遥控设备送给控制器的信息.
bit 8	机械手供电指示标记位. "0" 表示断电.
bit 9	用户程序运行状态指示标记位 (HOLD). "0" 表示执行用户程序当前一步后中断程序的运行.
bit 10	用户程序运行状态指示标记位 (RUN). "0" 表示正常执行用户程序. 如果用户程序中断了 (bit 9 为 0), 后又切换成 bit 10 为 0, 则从断点继续执行用户程序.
bit 11	用户程序运行状态指示标记位 (RESTART) 如果中断了用户程序 (bit 9 为 0), 后又切换成 bit 11 为 0, 则从第一步开始重新执行用户程序.

表 5.4 SYSRW 命令送出信息含义表

位　数	标 记 位 含 义
bit 0 ~bit 7	通过非标准 I/O 接口接插件，向外部遥控设备提供控制器运行情况的信息。
bit 8	允许手臂电源供电标记位。置"1"时允许启动手臂电源。
bit 9	手爪开关控制位。置"1"时手爪张开。
bit 10	手爪开关控制位。置"1"时手爪关闭。

第六章　机器人语言

§6.1　概述

机器人的一个重要特点就是它具有通用性。而为了使它具有通用性，就必须使它具有一种可编程序的机构，改变其程序便能实现不同的作业。“教”机器人，使它完成作业称为程序设计，这可以有以下三种方式：(1) 直接示教方式；(2) 离线数据程序设计方式；(3) 使用机器人语言的方式。

直接示教方式，也称为示教再现方式。即使用示教盒根据作业的需要把机器人的手爪送到作业所需要的位置上去，并处于所需要的姿态，然后把这一位置、姿态存储起来。对作业空间的各轨迹点重复上述操作，机器人就把整个作业程序记忆下来了。工作时，再现上述操作就能使机器人完成预定的作业，同时可以反复同样的作业过程。

直接示教法的优点是不需要预备知识，不需要复杂的计算机装置。所以被广泛使用，尤其适合单纯的重复性作业，例如搬运、喷漆、焊接等。

直接示教方式的缺点和不足的地方是：(1)示教时间太长、速度太慢，尤其是对于一些复杂的动作；(2)不同的机器人，或者即使同一个机器人，对于不同的任务都需要重新示教；(3) 无法接受感觉信息的反馈；(4)无法控制多台机器人的协调动作。

为了克服上述缺点，可以使用计算机辅助设计（CAD）软件设计数据，计算出为了完成某一作业，机器人手爪应该运动的位置和所处的姿态，即用 CAD 的方法产生用于示教的数据，这就是离线数据设计方式。对于一些复杂的作业，或需要给出连续的数据时，采用此方法是比较合适的。

第三种方式是使用编程语言，即机器人语言的方式。机器人语言是一种专用语言，即用符号来描述机器人的动作，类似于计算机的程序设计语言。这种方式的优点很多：（1）由于用计算机代替了手动示教，提高了编程效率；（2）语言编程与机器人型号无关，编好的程序可供多台机器人或不同型号的机器人使用；（3）可以接受感觉信息；（4）可以协调多台机器人工作；（5）可以引入逻辑判断、决策、规划功能以及人工智能的其他方法。

总之，智能机器人的研究离不开机器人语言的研究。

§6.2 机器人语言的发展历史

作为人工智能研究的一环，美国 Stanford 大学和 MIT、以及英国的一些大学在 1960 年后就开始发展机器人语言的研究。

1973 年，Stanford 人工智能实验室研究和开发了 WAVE 语言，这是一种符号语言，用于控制机器人机械手的动作，例如装配水泵、旋转曲柄等作业。WAVE 语言具有动作的描述、力和接触的控制、配合外部视觉系统进行手眼协调等功能。

1974 年，该实验室在 WAVE 语言的基础上开发了 AL 语言（Assambly Language），这是一种编译形式的语言，编译程序用 SAIL 写成，原语言具有 ALGOL 语言的结构，可以控制多台机器人联合协调工作。AL 语言对以后机器人语言的发展有很大的影响。

1979年，美国 Unimation 公司开发了 VAL 语言，首先配置

在 PUMA 和 Unimate 机器人上，成为第一个比较成功的机器人语言。 VAL 语言类似于 BASIC 语言，语句结构比较简单，易于编程，为工业机器人所实用。

美国 IBM 公司在机器人研究的初期研制了 ML 语言，这是一种比较单纯的语言，成功地用于各种装配任务。接着，该公司又研制了 AUTOPASS 语言，这是一种比较高级的机器人语言。AUTOPASS 可以对几何模型类任务进行半自动编程。IBM 公司后来还推出了 AML 语言。

1980 年，MIT 公布了 LAMA 语言，这是一种能用于自动装配的高水平机器人语言。

1981年，美国 Automatix 公司开发了 RAIL 语言，与视觉装置配合工作，用于零件检查。

其它的机器人语言还有：美国 Mcdonnell Douglas 公司开发的 MCL 语言；Edinburgh 大学研制的 RAPT 语言；法国 IMAG 人工智能小组研制的 LM 语言；意大利研制的 SIGLA 和 MAL 语言等。

§6.3 机器人语言的要素

一般所说的计算机语言，只指语言本身。而机器人语言，实际上是一个语言系统。机器人语言系统既包含语言本身——给出作业的指示和动作指示；又包含处理系统——根据上述指示来控制机器人系统；另外还包括了机器人的工作环境模型，如图6.1所示。

由于机器人语言系统的特殊构成，因而它具有与一般程序设计语言不同的功能要素，在研制过程中必须加以考虑(图 6.2)。这些要素如下所述。

(1) 作业环境与作业对象的描述，并且使其模型化，其中包括：

a 境环输入——一般由操作者通过与计算机的人机对话来进行。随着视觉功能的提高，一部分环境输入也可以自动生成；

b 环境描述——对工件的位置和姿态、工件之间的关系进行描述，并使之模型化；

c 环境模型的修改、更新——在作业进行中，工件的位置、姿态以及工件之间的关系可能随着作业而发生变化，语言系统对此要作相应的修改或更新环境模型。

(2) 作业的描述——可以用语句命令描述，也可以用自然语

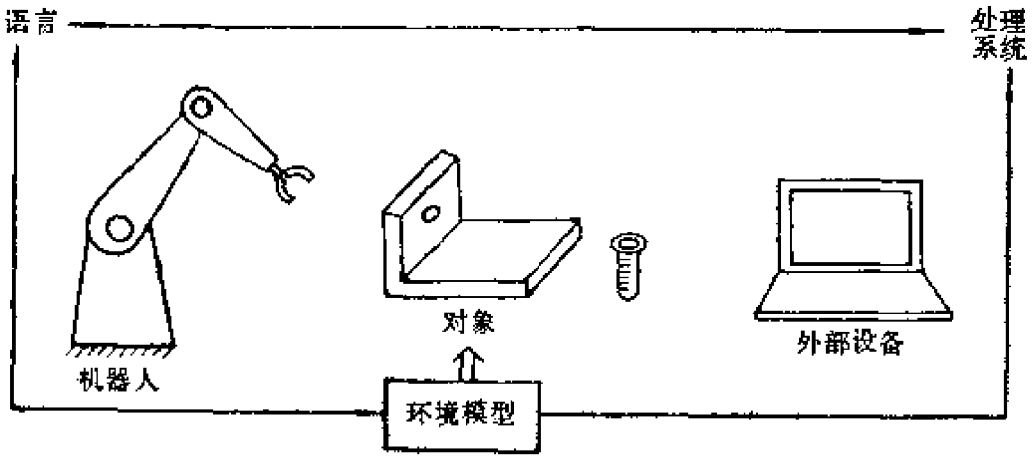


图 6.1 机器人语言系统

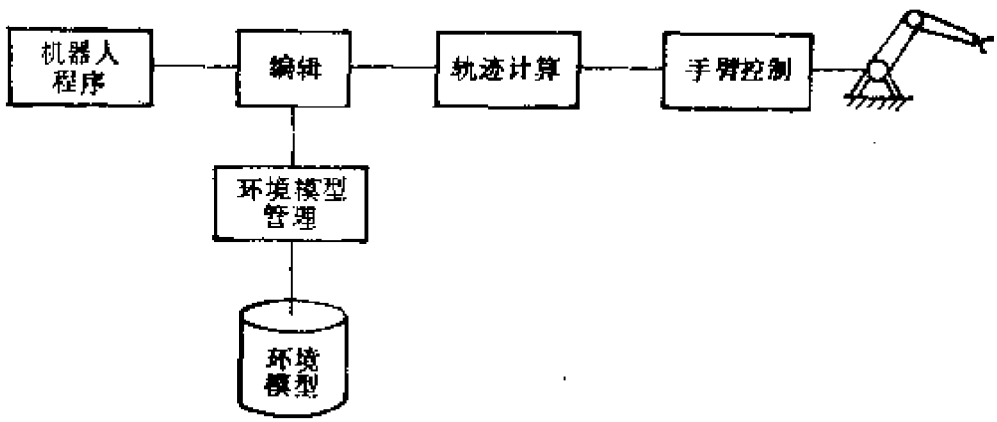


图 6.2 语言处理系统

言来描述。作业的描述与环境的模型有密切关系，而且描述水平决定了语言的水平。

(3) 程序设计的一般功能——与其它计算机语言相同。

(4) 人机接口及其后援功能——在编程和作业过程中，要便于人与机器人之间进行信息交换，而在运动出现故障时则能及时处理，保证安全。而且，随着作业环境和作业内容的复杂程度，需要有功能较强的相应接口。

§6.4 机器人语言的分类

根据作业描述水平的高低，机器人语言通常分为三级：(1)动作水平级；(2)对象物水平级；(3)作业目标水平级(也称任务级)。

动作水平级的语言是以机械手的运动作为作业描述的中心，由使手爪从一个位置到另一个位置的一系列命令组成。

对象物水平级的语言以部件之间的相互关系为中心来描述作业，与机器人的动作无关。

作业目标水平级的语言则以作业的最终目标状态和机器人动作的一般规则的形式描述作业。

这种分类办法较好地反映了语言的水平和功能，但在级与级之间还有些模糊或混乱，所以又有另一种分为五级的办法：(1)操作水平级；(2)原始动作水平级；(3)结构性动作水平级；(4)对象物状态水平级；(5)作业目标水平级。

操作水平级语言最简单，为了使机器人动作，设立了一些基本命令，解释程序对命令进行解释执行。它没有一般程序设计语言那样多的文法和句法，其水平与汇编语言相当。例如 IBM 公司初期制作的 ML 语言，它由两部分构成，一部分是为了完成机器人的基本动作和系统的操作而设置的子程序包，另一部分是可用带参数的命令起动那些子程序的解释程序。机器人系统构成时必须

通过这类语言来实现，所以几乎所有机器人都配备了这一水平级语言，而且许多系统并没有给这类语言命名。

原始动作水平级语言对于动作的描述仅限于手爪，动作的目的是移动某一物体，基本语句形式为

MOVE TO 〈destination〉

这一级语言的代表是 VAL 语言，它的句子结构比较简单，易于编程，语法类似于 BASIC。

结构性动作水平级语言在常用的程序语言的基础上加了有关机器人控制的句子结构，同时为了描述三维环境，语言中设立了各种数据及其相互运算的关系式。动作语句一般为

MOVE 〈object〉 TO 〈destination〉

AL 语言，以及后来把 AL 与 LISP 结合而成的 AL/L，还有 IBM 开发的 AML 都属于结构性动作水平级语言。

原始动作水平级语言和结构性动作水平级语言都以描述动作为中心。其核心是机器人手爪的各种运动语句，也可附加其它要求，如运动速度、运动各点之间的衔接等，这类语言还可接受感觉信号。

对象物状态水平级语言按照对象物之间的状态变化来进行程序设计，即以描写部件之间关系为中心的语言。用户仅给出相应的任务，不用给出明确的动作序列，至于如何去完成任务，则由系统根据环境的描述和作业的描述进行规划，自动生成相应的动作序列。这种语言一般都有处理感觉信息的能力，可利用感觉信息来修改和更新环境的描述和环境模型，也可利用感觉信息来进行控制、测试和监督。属于这一类的语言有 AUTOPASS，它带有自然语言的特色；LAMA 用 LISP 语言写成；还有如 RAPT、ROBEX 等。

作业目标水平级语言则以作业的最终目标来描述作业，为达到此目标而自动生成详细的动作、顺序和相应数据。这是一种最

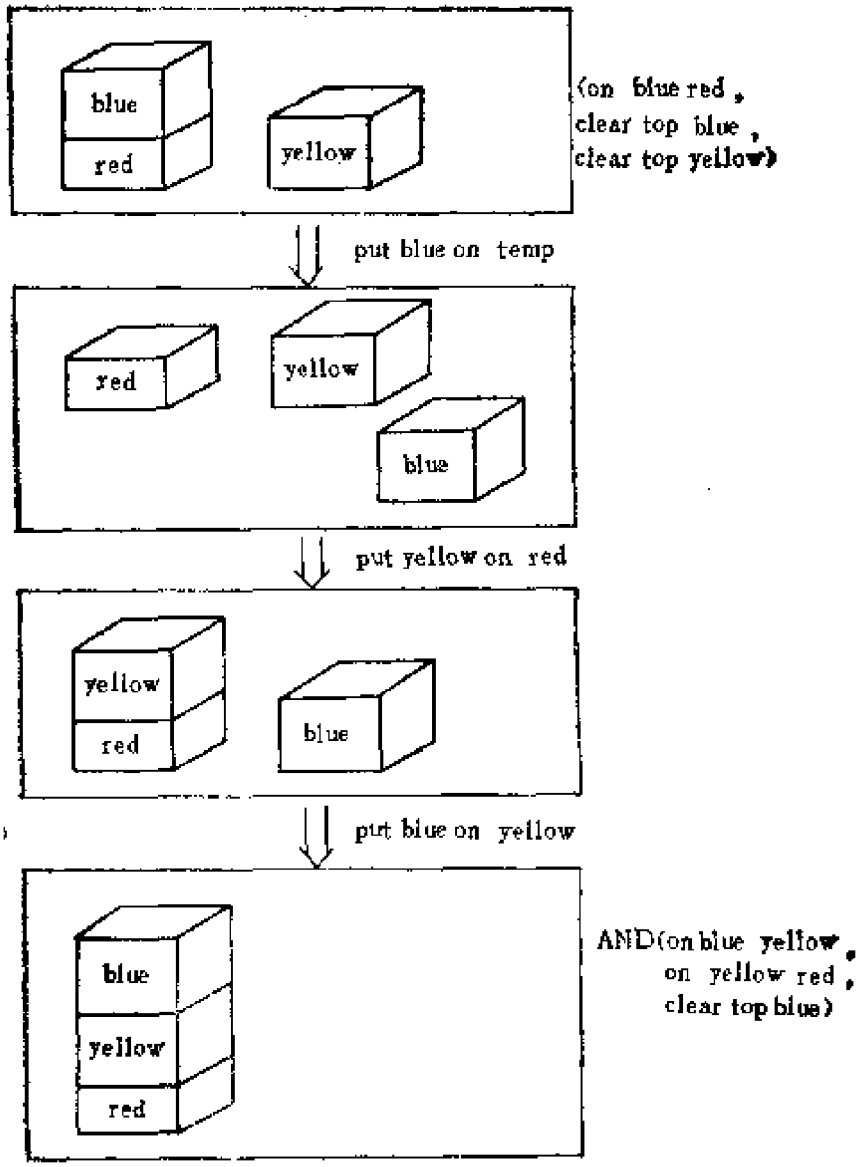


图 6.3　规划问题的子目标

理想的机器人语言，必须依靠较好的问题求解方法才能得以实现，这类语言具有判断环境、描述环境的能力，有规划任务的能力，它与人工智能的研究密切相关。这类语言有 MIT 人工智能研究所开发的 MICROPLANNER 和 CONNIVER，还有 PROLOG 等。前两种语言用 LISP 写成，可进行动作序列的规划，机器内部具有

模式匹配和追踪功能。为从初始状态到达目标状态，系统首先生成一系列子目标。图 6.3 例示了为把红、蓝、黄三个积木堆成目标状态所必须产生的子目标。以后的运行过程就与对象物状态水平的机器人语言相同了。

另外，机器人语言按表面形式可分为：(1)汇编型，如 VAL 语言；(2)编译型，如 AL、LM 语言；(3)自然语言型，如 AUTOPASS 等。

§6.5 VAL 语言

VAL 语言的主要特征为：(1) 能够在线、实时地使用；(2) 设有坐标变换的子程序，机器人的坐标可用直角坐标系来定义；(3) 对机器人动作顺序编制程序时，其中的定位参数必须给出实际的数据；(4)可以利用外部存储器(软盘)来记忆程序和数据。

VAL 语言的硬件支持系统如图 6.4 所示。

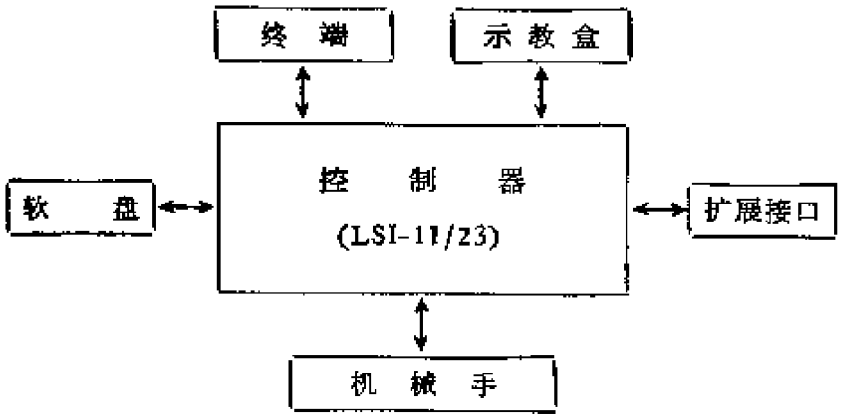


图 6.4 VAL 的硬件支持系统

VAL 的语言系统框图如图 6.5 所示，其中各主要组成部分介绍如下：

(1) 编辑：在编辑状态 (ED〈程序名〉) 下可用终端键盘输入文本程序，也可以通过示教盒按示教方式输入程序。编辑过程

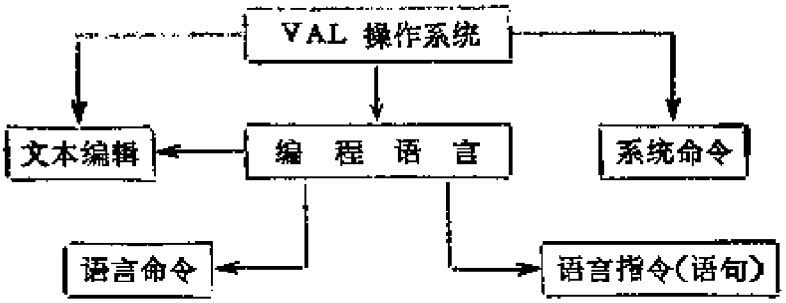


图 6.5 VAL 系统框图

包括对程序文件的生成和修改。编辑好的文件随即就存储在内存，也可以使用存盘命令（STORE〈文件名〉=〈程序名〉）存入软盘，需要时还可再调入（LORD〈文件名〉）。

（2）系统命令：包括位置定义、程序和数据列表、程序和数据存储、程序控制、系统状态的设置和控制、系统开关控制、系统诊断和修改等。

（3）编辑语言：语言指令由一条条语句组成，在运行命令（EXECUTE〈程序名〉）发出后，指令被逐条执行。在执行程序的同时，系统可以进行命令控制，也可以进行程序编辑。

一、VAL 的命令和指令简介

VAL 的命令和指令大致可分为三类：动作控制；程序控制和各种运算；坐标设置。下面简单介绍它们的功能。

（1）决定机械手位置的控制功能，其中包括：PTP 控制（Point To Point control，或 sequential positioning control），即在运动的路径上指定有限个必须通过的点；CP 控制（Continuous Path control），即整个运动路径都由插补点决定；两点之间为直线轨迹的 CP 控制；从某一点开始，手指的姿态一定的直线移动；从某一点开始，手指按原来的方向移动；移动时间一定的单坐标轴 PTP 控制。

（2）用示教盒进行的手动控制定位功能，其中包括：绕各个

关节轴的单独转动；在基坐标系（WORLD）中，沿 x、y、z 三轴的平移，以及围绕三轴的转动；在工具坐标系（TOOL）中，沿 x、y、z 三轴的平移，以及围绕三轴的转动；处于自由状态(FREE)，整个机械手的位置和姿态可任凭操作者随意摆布。

(3) 速度控制功能，其中包括：在手动控制时，用示教盒上的开关旋钮设定速度；在程序执行时，通过命令设定各程序步骤的速度(按标准速度的 0～100% 设定)。

(4) 位置精度的控制功能，其中包括：各轴完全达到最终目标位置时，才转到下一步骤；各轴在一定误差范围内达到最终目标位置时，转到下一步骤；与各轴偏差无关，插补指令值达到最终点时，就进入下一步骤；希望达到一定的轨迹精度时，可以使用偏差积分补偿。

(5) 程序设计功能，其中包括：无条件转移、逻辑转移语句以及根据外部信号来决定的条件转移语句；可以使用多达十次的子程序嵌套调用；有中断功能；有暂停、继续执行功能；有对整型数变量进行加、减、乘、除、求余数等五种算术运算。

(6) 程序的编辑功能，其中包括：删去或插入一步或多步指令；修改指令文字；修改位置数据；给指令内容加注释。

(7) 位置数据的定义(输入)方法。必要的位置已在程序设计时作为位置变量已给以文字命名，其实际数值可通过下述两种方法设置：用键盘键入机械手在基坐标系中各个轴的位置和姿态，键入机械手各关节轴的位置；用示教盒手动引导，确定机械手在基坐标系的位置和姿态。

(8) 复合坐标变换功能。如果机械手的位置、姿态数据用 x、y、z、α、β、γ 的 3×3 矩阵表示时，可以定义一个坐标变换，这样，当一个坐标系进行相对的位置变化后，所形成的新坐标系可以用原坐标系与坐标变换的积来计算。因此，如果机器人与任意作业空间的相对位置发生变化，只要变更作业空间的定位数据，程序

中所有的定位数据就可以原样使用；此外，不管选用什么形状、尺寸的手爪，都可以方便地参考工具坐标系定义手爪坐标系。

(9) 接近点和退避点的自动生成功能，例如

APPRO 〈loc〉〈dist〉

表示手爪从当前位置以关节插补方式移到与目标点〈loc〉在 z 方向相隔一定距离〈dist〉处；

APPROS 〈loc〉〈dist〉

含意同 APPRO 指令，但手爪的移动方式为直线运动；

DEPART 〈dist〉

表示手爪从当前位置以关节插补形式在 z 方向上移动一段距离〈dist〉；

DEPARTS 〈dist〉，

含意同 DEPART，但移动方式为直线运动。

(10) 软盘文件编排功能，其中包括：把程序和位置数据读出和存入软盘；软盘初始化；文件压缩和删除等。

(11) 手指的控制功能，其中包括：用 OPEN 和 CLOSE 语句控制手指的开闭，也可以随动作语句一起执行；手指的开度控制；确认手指抓住物体以后，进行条件转移。

下面列出 VAL 的主要语句结构。

动作语句：

MOVE 〈loc〉

MOVES 〈loc〉

MOVET 〈loc〉〈opening〉

MOVEST 〈loc〉〈opening〉

APPRO 〈loc〉[!]〈dist〉

APPROS 〈loc〉[!]〈dist〉

DEPART 〈dist〉

DEPARTS 〈dist〉

DRAW [〈dx〉] [〈dy〉] [〈dz〉] {在 x、y、z 方向上运动}

DRIVE 〈joint〉 〈change〉 〈speed〉 {某一关节以 〈speed〉 速度转动 〈change〉 角度}

手指控制语句:

OPEN [〈opening〉]

OPENI [〈opening〉]

CLOSE [〈opening〉]

CLOSEI [〈opening〉]

GRASP 〈opening〉 [〈label〉]

整数运算语句:

SETI 〈i. var 1〉 = 〈i. var 2〉 [〈operation〉 〈i. var 3〉]

TYPEI 〈i. var〉

数据语句:

HERE 〈loc〉 {把当前的位置赋给定位变量}

TOOL [〈trans〉]

SET 〈trans 1〉 = 〈trans 2〉 {把变量 2 的值赋给变量 1}

SHIFT 〈trans〉 BY [〈dx〉][〈dy〉] [〈dz〉]

INVERSE 〈trans〉 = 〈trans 2〉 {变量 2 为变量 1 的逆}

FRAME 〈trans〉 = 〈trans 2〉 〈trans 3〉 〈trans 4〉

控制语句:

GOTO 〈label〉

GOSUB 〈programe〉 RETURN

IF 〈i. var〉 〈relationship〉 〈i. var〉

THEN 〈label〉

IFSIG 〈ch〉 [〈ch〉] [〈ch〉] [〈ch〉]

THEN 〈label〉 {测试与外界联系的通道 〈ch〉 的信号,当 〈ch〉 为高电平时,转向标号为 〈label〉 的语句}

SIGNAL 〈ch〉 [〈ch〉···〈ch〉] {设置输出通道的值}

REACT 〈ch〉 [〈prog〉] [ALWAYS] {起动指定通道的信号监测器，当输入信号符合指定条件时，等待当前执行指令结束，一结束就转入〈prog〉指定的子程序}

REACTI 〈ch〉 [〈prog〉] [ALWAYS] {条件成立时，不等当前指令结束就马上转入〈prog〉}

WAIT 〈ch〉 {进入循环，等待外部条件成立}

IGNORE 〈ch〉 [ALWAYS] {关闭已被起动的信号监测器}

控制方式语句：

SPEED 〈val〉 [ALWAYS]

COARSE [ALWAYS]

FIND [ALWAYS]

NONULL [ALWAYS]

NULL [ALWAYS]

INTON [ALWAYS]

INTOFF [ALWAYS]

二、VAL 程序设计举例

例 6.1 机械手到 PICK 点抓住物体，放到 PLACE 的位置上去。

程序名：DEMO 1

程序编辑：

· EDIT DEMO 1 {____表示用键盘输入的部分}

· PROGRAM DEMO 1

1. ? OPEN {打开手指}

2. ? APPRO PICK 50 {向 PICK 的接近点 50mm 处移动}

3. ? SPEED 30 {下面的速度为 30%}

4. ? MOVE PICK　{向 PICK 点移动}

5. ? CLOSEI　{手指闭合抓住物体，"I" 表示闭合完毕以后才转入下一步}

6. ? DEPART 70　{离开 PICK 点 70mm}

7. ? APPROS PLACE 75　{以直线轨迹方式向 PLACE 的接近点 75mm 处移动}

8. ? SPEED 20　{下面的速度为 20%}

9. ? MOVES PLACE　{以直线轨迹方式移动到 PLACE 点}

10. ? OPENI　{打开手指，"I" 表示打开终了以后才转入下一步}

11. ? DEPAT 50　{离开 PLACE 点 50mm}

12. ? E　{编辑结束}

程序中定位数据的给定(用示教盒手动引导):

· HERE PLACE

x/JT1	y/JT2	z/JT3	o/JT4	a/JT5	t/JT6
225.66	618.84	—131.94	154.86	89.187	—131.347

CHANGE ?　{问数据是否要变更}

程序的执行:

· SPEED 30　{为了试验程序是否合适，所以程序用 30% 的速度进行}

· EXEC DEMO 13　{三次执行 DEMO 1 程序}

· STATUS　{为了监视程序执行，显示机械手状态}

⋮

PROGRAM COMPLETED

HALTED AT STEP 12　{程序执行完毕，终止在第 12 步}

·EXEC −10　{用负数表示无限止地执行 DEMO 1 程序，默认程序名与前一次相同}

例 6.2　使用机械手整齐排列桌上的十个零件。

程序清单：

```
    SETI N. PARTS = 0
100 VPICTURE
    VLOCATE PART 100
    APPRO CAMERA:PART:PICK VP 50
    MOVES CAMERA:PART:PICK VP
    GRASP 25
    DEPART 50
    APPRO PALLET 50
    MOVES PALLET
    OPENI
    SHIFT PALLET BY 5.2 25 4.0
    SETI N. PARTS = N.PARTS + 1
    IF N. PARTS NE 10 THEN 100
    STOP
```

§6.6　AL 语言

AL 语言 (Stanford 大学研制)能控制多台具有感觉信号的机器人，它由 WAVE 发展而来。

AL 的硬件支持系统如图 6.6 所示。

图 6.7 说明了 AL 的软件系统及其运行过程。用户写的源程序 FOO.AL 送交分析程序分析，如果无错误就生成一个 FOO.SEX 文

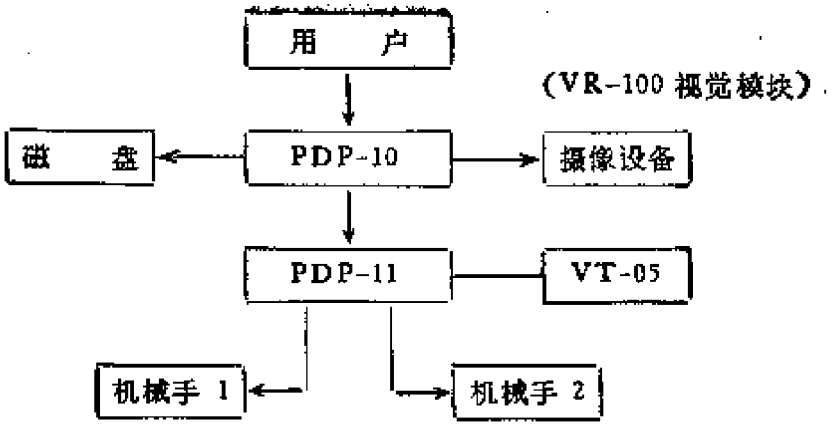


图 6.6　AL 的硬件支持系统

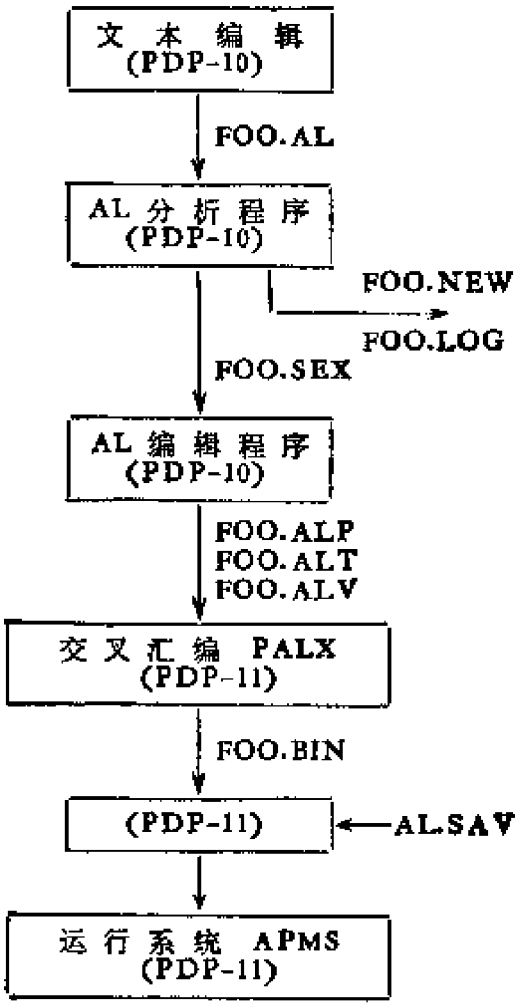


图 6.7　AL 的软件系统

件(中间码文件)送交编辑程序，如果有错误就输出一个带有错误信息的文件 FOO.LOG 和一个拷贝文件 FOO.NEW. 编辑程序对 FOO.SEX 进行编辑并进行模拟和轨迹计算，最后生成三个类型的文件 ALP、ALT、ALV，它们包括了程序码、常数和运动轨迹，然后送交叉汇编 PALX，生成 FOO.BIN 二进制文件. FOO.BIN 连同 AL.SAV (程序码编辑程序)送交 11TTY，最后产生执行文件.

AL 类似 ALGOL 语言，每个变量需事先说明，而且也是一种块结构语言.

一、变量的表示和类型

变量由一串英文字母、数字和下划线符号"-"构成. 例如: pump-base, handle, screw-hole-2 等. 所有的变量都必须由英文字母打头，大写小写具有同等意义.

变量可以用赋值语句进行赋值，变量与数值用"←"来连接，即把箭头右边的数值赋给左边的变量.

下面分别介绍几种重要的变量类型.

(1) 基本变量 AL 中最基本的数据类型为 SCALAR. 对于此种类型的变量可以进行五种运算，加(+)、减(—)、乘(*)、除(/)、指数(↑). 可以使用函数 SQRT、SIN、COS、TAN、ASIN、ACOS、ATAN，自然对数 LOG 和指数变换 EXP 等. 例如

```
SCALAR s1 s2;          {变量类型说明}
s1 ← 2;                {赋值}
s2 ← 3.50;             {赋值}
s1 ← S2* (s1—3.2);     {s1的值为—4.2}
```

SCALAR 型变量可以表示时间(TIME)、距离(DISTANCE)、角度(ANGLE)、力(FORCE)或者它们的组合. AL 还能处理这些变量的量纲，即秒(sec)、英寸(inch)、度(deg)、或盎司(ounce)

等。具有量纲的变量进行加、减、乘、除以后，其结果是否具有合适的量纲要进行检查。例如

```
SCALAR s1 s2;
DISTANCE SCALAR ds1;
TIME SCALAR tm1 tm2;
FORCE SCALAR fs1;
ANGLE SCALAR theta phi;
ds1 ← 1.0 * inch;
tm1 ← 3 * sec;
fs1 ← 2.2 * ounce;
tm2 ← tm1 + 4.5 * sec;
theta ← 90 * deg;
phi ← theta * 4 * deg;{右边单位为 deg², 单位量纲有错}
s1 ← SIN (30 * deg);
theta ← ACOS (0.7);
ds1 ← SQRT (ds1 * 3 * inch);
phi ← ATAN (s1/s2);
s1 ← LOG (33.0);
s2 ← s1 ↑ 3;
```

速度、角速度、力距定义的基本单位，加速度这样的单位不包含在基本单位内，必须作如下说明：

```
DEFINE feet = ⊂(12 * inches)⊃; {英尺的定义}
DIMENSION acceleration = VOLOCITE/TIME; {加速度单位的定义}
acceleration SCALAR as1;
as1 ← 6.7 * feet/(sec * sec);
```

AL 语言中有几个事先定义过的 SCALAR，例如：

```
SCALAR p1; {p1 = 3.14159}
```

DISTANCE SCALAR bhand yhand; {bhand 表示"蓝臂"、yhand 表示"黄臂"}

(2) 向量

```
VECTOR xhat yhat zhat nilvect; {向量说明}
xhat ← VECTOR (1, 0, 0); {向量赋值}
yhat ← VECTOR (0, 1, 0);
zhat ← VECTOR (0, 0, 1);
nilvect ← VECTOR( 0, 0, 0);
```

下面举例说明向量的使用和运算:

```
VECTOR v;
DISTANCE VECTOR dv1 dv2;
DISTANCE SCALAR ds;
v ← VECTOR (2, 1, 3);
dv1 ← VECTOR (4 * inch, 2 * inch, b * inch);
dv2 ← VECTOR(3, 0, 4) * inch;
ds ← |dv2|;      {ds = 5 inch}
v ← UNIT(v); {v = VECTOR(2/3, 1/3, 2/3)}
```

(3) 旋转 旋转 (ROT) 型变量用来描述围绕某一个轴的旋转,以表示姿态。任一旋转变量可表示为 ROT 函数,它有两个参数,一个对应旋转轴,用向量来表示;另一个是旋转角度。旋转规定按右手法则进行。此外,对于旋转 x, 函数 AXIS(x) 表示求取 x 的旋转轴,而 $|x|$ 则表示取 x 的旋转角。例如

```
ROT r1 r2 r3 r4; {旋转变量说明}
ANGLE SCALAR alpha beta gamma;
VECTOR v;
r1 ← ROT (xhat, 90 * deg); {绕 x 轴转 90°}
v ← r1 * zhat; {向量 zhat 旋转 r1, 赋值于向量 v, 因而 v = (0, −1, 0)}
```

r2 ← ROT (yhat, 45 * deg);

r3 ← r2 * r1; {r3 表示先绕 x 轴转 90°，再绕原来的 y 轴转 45°；或者可表示先绕 y 轴转 45°，再绕新的 x 轴转 90°}

v ← AXIS(r2); {把 v 赋值为 r2 的旋转角 yhat}

alpha ← |r2|; {把 alpha 赋值为 r2 的旋转角，即 alpha = 45°}

beta ← 15 * deg;

gamma ← 60 * deg;

r1 ← ROT (xhat, alpha);

r2 ← ROT (yhat, beta);

r3 ← ROT (zhat, gamma);

r4 ← r3 * r2 * r1; {r4 表示先绕 x 轴转 alpha 度，再绕原来的 y 轴转 beta 度，最后绕原来的 z 轴转 gamma 度；或表示依次绕 z 轴、新的 y 轴、新的 x 轴转 gamma 度、beta 度和 alpha 度}

(4) 坐标系 FRAME 型坐标系变量用来描述作业空间中物体的姿态和位置，变量的值表示对象物体的固定坐标系与作业空间坐标系之间的相对位置关系和姿态关系。FRAME 型变量有两个参数，即旋转和向量。函数 ORIENT、POS 可分别用来定义 FRAME 坐标系的姿态和位置。

对于在某一个坐标系中描述的向量，可以用"向量 WRT 坐标系"的形式来表示 (WRT: With Respect To)。例如

FRAME f1 f2; {坐标系变量说明}

f1 ← FRAME (ROT(zhat, 90 * deg), 2 * xhat * inch); {坐标系 f1 的原点位于基坐标系的 x 轴方向上 2in 处，它是围绕基坐标系的 z 轴旋转 90° 而形成的坐标系}

v1 ← xhat WRT f1; {v1 被赋值为基坐标系中表示的 f1 的 x 轴的单位向量，即 (2, 1, 0)}

f2 ← f1 + v1 * inch; {f2 的原点位于 (2, 1, 0)，姿态

与 f1 相同}

(5) 变换 TRANS 型变量用来进行坐标变换.它和FRAME一样具有旋转和向量两个参数,在执行时,先相对于作业空间的基坐标系旋转,然后进行平移操作。

TRANS 的旋转参数和向量参数具有不相同的单位，后者具有距离的量纲,所以在进行乘法运算时,首先对旋转参数进行乘法运算,然后对向量参数进行加法运算.

两个坐标系用符号"→"相联接时，意指使左边坐标系的原点和各轴经过平移和旋转,与右边坐标系的原点和各轴完全重合.例如

```
TRANS t1 t2 t3 t4;  {变换型变量说明}
t1 ← TRANS (ROT(xhat, 30 * deg), 2 * zhat * inch);
```

{绕 x 轴旋转 30°, 然后再沿 z 轴平移 2in}

```
v1 ← t1 * yhat * inch;  {y 轴经过 t1 变换后赋于 v1}
t2 ← f1 → f2;  {f1 × t2 = f2}
v2 ← t2 * (xhat * inch);  {v2 为 f2 中描述的 f2 的 x
```

轴}

```
t3 ← t2 * t1;      {变换的积}
t4 ← INV(t1);  {t1 的逆变换}
```

二、主要语句命令及有关功能

AL 程序一般包含了描述机械手应执行作业的一连串语句,各语句之间用";"分隔,开始和结尾以 BEGIN 和 END 为标记,例如

```
BEGIN
s1;
s2;
:
```

sn;

END

其中 s1，…，sn 为各种语句，也可以是一些程序块 。

(1) MOVE 语句 MOVE 语句用来表示机械手由起始位置和姿态到目标位置和姿态的运动。例如

MOVE 〈hand〉 TO 〈destination〉 VIA f1 f2 f3;

表示把某个机械手 〈hand〉 经过中间点 f1、f2、f3 移动到目标坐标系 〈destination〉

MOVE 〈hand〉 TO 〈block〉 WITH APPROCH = 3 * zhat * inch;

表示把 〈hand〉 移动到在 z 轴方向上离 〈block〉 3in 的地方，如果用 DEPARTURE 代替 APPROCH，则表示离开 〈block〉。

MOVE 〈hand〉 TO 〈hand〉 * FRAME(ROT(zhat, 90 * deg), nilvect * inch) ON TOROUE (zhat) ≥ 50 * ounce * inch DO STOP 〈hand〉;

句中 ON…DO… 为条件语句，整个句子表示 〈hand〉 绕 z 轴转 90° 而位置不变 (nilvect * inch)，动作过程中若力矩超过 50 ounce * inch 时则停止。

(2) 手爪控制语句 机械手一般具有两个手指，使用OPEN、CLOSE 两个语句来控制其开闭，对于开度可控的手指，还可以在语句中说明手指的开度。语句的一般形式为：

OPEN 〈hand〉 TO 〈scalar_exp〉;

CLOSE 〈hand〉 TO 〈scalar-exp〉;

其中 〈scalar-exp〉 是表示开度距离的 scalar 变量值，可事先定义，如 2.5 * inch。

执行 OPEN、CLOSE 时，两个手指是以同一速度动作。对于手指内侧设有触觉的情况，如果手指在达到关闭量之前，触觉传感器已处于“ON”的状态，则停止动作，并发出误差信号。

CENTER 语句是使手指闭合到与被抓对象接触为止，即只要有任何一个手指的接触传感器达到“ON”状态就停止动作。如果一个指头的触觉达到“ON”时，就在保持其状态的前提下使整个手腕移动，直到第二个指头的触觉达到“ON”为止。CENTEN 命令并非一定把对象物夹持在手爪的中央，而是要使对象物不能活动，确实把持了物体。在 CENTER 命令中把腕的名称作为自变量，例如 CENTER 〈arm〉。

(3) 中间点语句　中间点语句有三种。其中接近语句为

WITH APPROACH = 〈appr〉;

其中〈appr〉为目标点所设定的坐标系中的一个向量。语句表示机械手必须经过由〈appr〉所规定的接近点再移动到某个目标点。

退避语句为

WITH DEPARTURE = 〈depr〉

类似于接近语句，〈depr〉表示退离目标点必须经过的位置。

如果没有给出〈appr〉的值，接近点就被默认为在基坐标系 z 轴方向 3in 处；而在没有指定〈depr〉的情况下，则沿用前面所进行的运动的接近点作为默认值。语句

WITH DEPARTURE = NILDEPROACH

表示不设退避点，则从当前的位置，直接使腕离开。同样，语句

WITH APPROACH = NILDEPROACH

表示不设置接近点，将腕直接移动到目标点。

VIA 语句用于迴避障碍物，从而必须经过某些指定点的情况。

上述三种中间点语句实际上都与 MOVE 语句合并使用。下面的例子是要机械手把一块砖头拿起来，并把它放到与作业台同一高度的烤炉里去的程序，机械手必须通过烤炉的入口而进行移动。

BEGIN

```
FRAME brick, oven oven_door;
brick ← FRAME (ROT (yhat, 90 * deg), VECTOR(10,
30,3) * inch); {规定砖的起始位置和姿态}
oven ← FRAME (ROT (yhat, 90 * deg), VECTOR(10,
40, 3) * inch); {规定砖的终止位置和姿态}
oven_door ← FRAME (ROT (yhat, 90 * deg), VECTOR
(15, 40, 4) * inch); {规定烤炉入口位置和姿态}
OPEN bhand TO 3 * inch {保证手爪开度足够大}
MOVE bram TO brick
  WITH APPROACH = 3 * zhat * inch;
{把机械手向处于水平位置的砖移动,砖的 z 轴与基坐标系的 x 轴
平行}
CLOSE bhand to 1.7 * inch; {抓砖}
MOVE barm TO oven VIA oven_door
  WITH APPROACH = -3 * zhat * inch;
{通过烤炉入口把砖移入炉中}
OPEN bhand TO 3.0 * inch; {放下砖}
MOVE barm TO bpark VIA oven_door; {机械手移动
到待命位置}
END
```

(4) AFFIX 与 UNFIX 语句　在装配作业中,往往出现把某一个物体"附着"于另一个物体,使二者结合在一起的操作。语句

```
AFFIX pump TO pump_base;
```

执行后,即表明 pump_base 今后的运动也将引起 pump 作同样的运动,即两者将相伴运动(虽然没有实际连结在一起)。

把操作对象的坐标系"附着"于手腕的坐标系,使二者连接在一起的情况特别重要。如当 pump 与 barm 相伴结合以后,用户就可以不考虑手腕,只注意 pump 如何运动、运动到何处就行了。

语句

UNFIX pump FROM barm;

表示"附着"关系的解除。

(5) 力觉的处理 为了完成某个动作，希望使用感觉信息的阈值，此时可在 MOVE 语句中加入条件：

ON 〈condition〉 DO 〈action〉;

例如，当机械手的旋转负荷力矩超过 50oz·in 时，要求停止旋转，对此可采用下列语句：

MOVE barm TO barm * FRAME (ROT(zhat, 90 * deg), nilvect * inch) ON TORQUE (zhat) ≥ 50 ounce * inch DO STOP barm;

其中 STOP 是当达到某负荷时，机械手的运动停止的一个命令，zhat是指明力矩检测的方向，50 * ounce * inch 是指力矩的阈值。

(6) 力的控制与稳定性(抗偏离能力)的控制 把力和力矩加到被夹持的物体上，控制其外观上的稳定性，这种功能很有必要，其典型的应用例子就是把销钉插入孔中的作业。

稳定性控制，首先是设定所移动物体的轨迹，然后保持手腕具有一定的稳定性，按规定的轨迹移动。

插入销钉例子的程序如下：

```
{假定销钉已在手爪中，而且两者已经相伴结合}
BEGIN
  MOVE peg TO hole_entrance;
  MOVE peg TO hole_buttom;
  DIRECTLY
    WITH STIFFNESS=(VECTOR(10,10,90) * oz/inch,
VECTOR(200, 200, 200) * oz * inch/radius)
    WITH DURATION = 2 * sec;
END
```

在上例中，在 x、y 方向上的稳定性是比较低的，z 轴方向比较高，即旋转轴方向的稳定性比较高。如果夹持物体的各个主轴方向稳定性都能控制的话，那么用户可以按照给定的手指位置，使稳定的中心取任意值。

为了在所希望的方向的稳定性等于零，可以利用 FORCE 或者 TORQUE 语句加上偏置力来表示。下面的例子表示在 z 方向上在 3 秒钟内把一定的力加上。

```
MOVE barm TO barm 12 * zhat * inch
DIRECTLY
WITH STIFFNESS = (VECTOR (0, 90, 90) *oz/inch,
VECTOR(200, 200, 200) * oz * inch/radius)
WITH FORCE (xhat) = 40 * ounce
WITH DURATION = 3 * sec;
```

(7) OPRATE 语句　OPRATE 语句用来控制与 AL 系统相连续的各种装置。例如：

```
OPERATE driver
  WITH ANGULAR_VELOCITY = n * rpm 〈direction〉;
OPERATE driver
  WITH TORQUE = n * oz * inch 〈direction〉;
OPERATE driver
  WITH DURATION = n * sec;
```

前面两句不能同时使用，因为角速度和力矩不可能同时指定。〈direction〉是指旋转方向，必须是顺时针 (CLOCKWISE) 或者逆时针 (COUNTER CLOCKWISE) 中的一个。

(8) VAL 控制命令　对于 PUMA 机械手，它虽然可以由 AL 系统直接控制，但有时需要与 PUMA 控制装置中的 VAL 语言系统进行信息交换，这时可使用下列命令：

```
VAL ("string", WAIT);
```

VAL ("srring", NOWAIT);

当一串信息送往 VAL 时，AL 可以一直等待 VAL 执行完毕 (WAIT)，或者立即执行下一个语句(NOWAIT)，这两种情况分别用上面的第一句和第二句表示。

(9) 控制语句 AL 中有许多控制语句，与 ALGOL 语言非常相似。如:

```
IF···THEN···ELSE···;
FOR···loop;
WHILE···DO loop;
CASE statement;
DO···UNTIL···statement;
```

等等，但没有 JUMP 和 GOTO 语句。

(10) 并列控制语句 AL 语言可以同时控制若干台机械手。当使用 COBEGIN···COEND 模块时，模块内的语句是并行执行的。例如:

```
COBEGIN
  MOVE barm TO bpark;
  MOVE yarm TO ypark;
  MOVE garm TO gpark;
  MOVE rarm TO rpark;
COEND
```

为了使多个机械手同步地运动，可以使用EVENT、SIGNAL、WAIT 语句。例如:

```
EVENT e1 e2 e3;
```

EVENT 语句定义了事件，并给每个事件一个信号计数器，计数器的初值为零。又如:

```
SIGNAL e1;
```

当程序执行时，e1 计数器的内容加1，如果其结果为零或为负，则

允许执行等待这一事件的程序。若遇到

```
WAIT e1;
```

则事件 e1 的计算器内容减 1，当计数器的值变负时，进入等待状态；而当计数器的值为零或正时，则继续执行下条语句。

下面是使用这三种语句描述两个机械手交接积木的例子。

```
BEGIN
  EVENT passed  caught ready_pass;  {定义事件，使事件计数器为零}
    FRAME block pass catch;  {block 为积木坐标系，pass、catch 表示递交或接受积木时，机械手位置和姿态的坐标系}
    COBEGIN
      BEGIN    {蓝色机械手递交过程}
      MOVE barm TO block;
      CENTER barm;
      AFFIX block TO barm;
      MOVE block TO pass;  {把积木移到递交处}
      SIGNAL ready_pass;  {准备递交}
      WAIT caught;  {等待另一手接受}
      OPEN bhand TO 3.0*inch;  {放开积木}
      UNFIX block FROM barm;
      SIGNAL passed;  {递交完毕}
      END
    BEGIN
      OPEN yhand TO 3*inch;
      MOVE yarm  TO catch;  {向接受积木处移动}
      WAIT ready_pass;  {等待积木到来}
      CENTER yarm;  {抓住积木}
      SIGNAL caught;  {表明已抓住}
```

```
    WAIT passed;  {等待释放}
    MOVE yarm TO plate;  {把积木放到盘子里}
    END;
COEND;
END
```

(11) PROCEDURE 和 ARRAY 语句　在程序中有时在几个地方执行的是相同的一个作业，为此希望利用子程序的形式，这由 PROCEDURE 语句实现。

例如在装配作业中有几个螺钉要拧进螺孔内，此时可由下列程序实现。

```
PROCEDURE inset_screw(FRAM hole_location)
BEGIN
  screw ← screw_dispenser;
  MOVE driver_tip TO screw;
  AFFIX screw TO driver;
  MOVE screw_tip TO hole_location;
  COBEGIN
  MOVE screw TO — 0.75 * zhat * inch
    WITH FORCE = 20 * ounce
      ALONG zhat OF screw
    WITH DURATION = 2.5 * sec;
  OPERATE driver
    WITH ANGULAR_VELOCITY = 200 * rpm
    WITH DURATION = 3 * sec;
  COEND;
  UNFIX  screw FROM driver;
END
```

如果把这个 PROCEDURE 语句的变量设为 h，那么，上面的

程序就表示在位置 h 上进行拧螺钉作业。

AL 也能使用数组，例如某个台上有三个孔，分别用 hole(1)、hole(2)、hole(3) 来定义其坐标系，也可以写成 hole[1:3]，用 FRAME ARRAY hole[1:3] 来定义。

(12) 宏指令　宏指令的使用可以简化源程序，例如：

```
DEFINE grasp (object) =
  ⊂MOVE barm TO object; CENTER barm;
  AFFIX object TO barm RIGIPLY⊃
```

其中 grasp 定义为宏指令的名称，object 是参数，可以被更换。

§6.7 机器人语言程序设计举例

对机器人的作业进行程序设计时，需要两个部分，即作业环境的描述和作业描述。

1. 作业环境的描述

已有的机器人语言大部分都有作业环境模型，利用模型可以决定作业时所需的具体参数(零件的方向、位置、移动规则)，可以检查作业描述是否妥当，是否会发生冲突。另外希望在作业进行时，模型的内容可以自动修改。

用 AL 语言可作出各种环境描述程序，这里再举一些例子。

例 6.3

```
FRAME beam beam_hole;
FRAME bracket bracket_hole bracket_grasp;
FRAME bolt;
beam ← FRAME(ROT(z, −90 * deg), VECTOR(10, 6, 0));
beam-hole ← beam * TRANS(ROT(x,90 * deg),VECTOR(3, 0, 7));
```

其中，前三句是设定柱体、柱体孔、支架、支架孔、支架抓手、螺栓的坐标系。第四句决定柱体的坐标系，即相对于基坐标系来说，其原点为 (10cm, 6cm, 0cm)，而且绕基坐标系的 z 轴转 −90°. 第五句决定坐标孔的坐标系，其原点位于柱体坐标系的 (3cm, 0cm, 7cm) 处，且绕其 z 轴转 90°。另外，语句

```
AFFIX beam_hole TO beam;
```

可表示柱体孔是附着于柱体上。

下面也是一些描述环境的 AL 语句:

```
ASSERT FORM (DEPROACH, beam_hole, TRANS
            (NILROT, VECTOR(0, 0, −3));
bracket_hole ← bracket * TRANS (ROT (x, 180 * deg),
                VECTOR(3, 3, 0));
AFFIX bracket_hole TO bracket;
bracket_grasp ← bracket * TRANS (ROT (x, 180 * deg),
                VECTOR(0, 3, 3));
AFFIX bracket_grasp TO bracket RIGIDLY;
balt ← FRAME(ROT(z, 90 * deg) * ROT(x, 180 * deg),
                VECTOR(16, 30, 0));
```

例 6.4 用 RAPT 作出环境描述程序，用几何模型来描述零件(由点、直线、圆、平面组成)。

```
BLOCK = MACRO/B x y z R{用宏指令定义物体 BLOCK}
BODY/B
{REMARK THE RELEVANT FEATUERS OF THIS
STANDARD BLOCK BE THE TOP THE BOTTOM, THE
BACK, TWO SIDES AND TWO HOLES DRILLED
THROUGH THEM 注释行}
P1 = POINT/0, 0, 0;
P2 = POINT/x, 0, 0;
```

```
P3 = POINT/0, y, 0;
P4 = POINT/0, 0, z;  {生成物体角的四个点}
P5 = POINT/x/4, y/4, 0;
P6 = POINT/x — x/4, y/2, 0;  {P5, P6 为孔的中心点}
C1 = CIRCLE/CENTER, P5, RADIUS, R;
C2 = CIRCLE/CENTER, P6, RADIUS, R;  {定义两个圆孔}
L1 = LINE/P1, P2;
L2 = LINE/P1, P3;
L3 = LINE/P3, PARLEL, L1;
L4 = LINE/P2, PARLEL, L2;  {定义四条棱}
BACK1 = FACE/L2, xSMALL;
            {REMARK THE BACK OF THE BLOCK}
BOT1 = FACE/HORIZONTAL, 0, zSMALL;
            {REMARK THE BOTTOM}
TOP1 = FACE/HORIZONTAL, z, zLARGE;
            {REMARK THE TOP}
            {定义了三个面}
```

2. 作业描述

具体的作业描述程序是与各语言的水平有关的，也与构成此语言的通用语言(如 FORTRAN、ALGOL, LISP 等)有关。

下面举一个用 VAL 编制一个作业程序的例子。该例子要求机器人抓起送料器送来的部件，并运到检查站，检查站判断部件是A类还是B类，然后根据判断转入适当的处理程序。在这个程序中，要用到一些不同意义的外部信号：

传感器 1，置位表示送料器正在提供部件；

传感器 2，置位表示部件已送到检查站；

传感器 3、4、5，判断部件所需的特征信号；

传感器 6，置位表示检查完毕。

作业程序如下：

```
    REMARK VAL PROGRAM EXAMPLE
    SIGNAL —2                          {初值给定}
    OPENI 100.00
10  REACTI 7, ALWAYS                   {紧急信号监控开始}
    WAIT 1                             {等待供给的部件}
    SPEED 200.00
    APPRO PART, 50.00                  {部件的位置通过接近
                                       PART 的 50mm 处}
    MOVES PART
    CLOSEI 0.00                        {抓住部件}
    DEPARTS 50.00                      {垂直抬起 50mm}
    APPRO TEST, 75.00                  {向检查站移动}
    MOVE TEST
    IGNORE 7, ALWAYS                   {紧急信号监控停止}
    SIGNAL 2                           {部件准备完}
    WAIT 6                             {等待检查完}
    DEPART 100.00                      {取出部件}
    SIGNAL —2                          {复位信号 2}
    IF SIG —3,4,—5, THEN 20            {部件为 A 型转到 20}
    IF SIG 3, —4, —5 THEN 30           {部件为 B 转到 30}
    GOSUB REJECT                       {若不是 A，也不是 B 则
                                       取消此程序}
    GOTO 40
20  REMARK PROCESS PART"A"
    GOSUB PART A
    GOTO 40
```

```
30  REMARK PROCESS PART"B"
    GOSUB PART B
    GOTO 40
40  REMARK PART PROCESSING COMPLETE
    GET ANOTHER PART
    GOTO 10
```

例 6.5 用 AL 写的作业程序。

```
DIFINE OZ = "72.007789*DYNES";
OPERATE YFINGERS WITH opening = 3*cm;
MOVE YELLOW TO bracket_grasp;
CENTER YELLOW;
bracket_grasp ← YELLOW;
AFFIX bracket TO YELLOW;
MOVE bracket_hole TO beam_hole;
OPERATE BFINGERS WITH opening = 3*cm;
MOVE BLUE TO hole;
CENTER BLUE;
bolt ← BLUE;
AFFIX bolt TO BLUE;
MOVE bolt TO beam_hole⊕+VECTOR(0, 0,−5.3) WRT
     beam_hole;
MOVE BLUE TO⊕+VECTOR (0,0,5) WRT beam_hole;
```

{最后两句表示，首先把夹持螺栓的蓝色手臂移动到柱体小孔坐标系 z 轴方向 −5.3cm 处，然后是把蓝色手臂再从现在位置向柱体小孔 FRAME 的 z 轴方向移动 5cm，⊕ 表示手臂现在的位置}

```
WITH FORCE = 0 ALONG X,Y OF BLUE ON
     FORCE (Z WRT BLUE) > 60*OZ DO
```

```
    STOP BLUE;
```
{此句为一条件语句,蓝色手臂 z 轴方向的力超过 60 oz 时就停止移动}

```
    OPERATE YFINGERS WITH opening ← 3 *cm;
    UNFIX bracket FROM YELLOW;
    AFFIX bracket TO beam;
    MOVE YELLOW TO ypark;      {ypark 为放置场所}
    OPERATE BFINGERS WITH opening ← 3 *cm;
    UNFIX bolt FROM BLUE;
    AFFIX bolt TO beam;
    MOVE BLUE TO bpark;
    WRITE ("FINISHED")
```

例 6.6 用对象物状态水平语言 AUTOPASS 写的作业程序.

```
    OPERATE nutfeeder WITH car_ret_tab_nut AT fixture.
        nest
    PLACE bracket IN fixture SUCH THAT bracket bottom
        CONTACTS car_ret_tab_nut top AND bracket hole
        IS ALIGNED WITH fixture. nest
    PLACE interlock ON bracket SUCH THAT interlock
        hole IS ALIGNED WITH bracket hole AND inter-
        lock base CONTACTS bracket. top
    DRIVE IN car_ret_intlk_stud INTO car_ret_tab_nut AT
        interlock hole SUCH THAT TOROUE IS EG 12.0
        IN LBS USING air_driver ATTACHING bracket
        AND interlock
    NAME bracket interlock car_ret_intlk_stad car_ret_tab_nut
        ASSEMBLY support_bracket
```

第七章　计算机视觉概论

§ 7.1　概述

人工智能的目标之一就是使计算机具有处理感官输入信息的能力.

智能机器人为了具有人的一部分智能,像在前面叙述的那样,必须了解周围的环境,获取机器人周围世界的信息. 人们为了从外界环境获取信息,一般是通过视觉、触觉、听觉、嗅觉等感觉器官来进行的. 其中 80% 的信息是由人的眼睛,即视觉来获取的,即人们用视觉从自己周围收集大量的信息,并且进行处理. 然后根据处理结果来采取行动,所以对于机器人来说,视觉是非常重要,它对于机器人的功能有着很大影响. 尤其是智能机器人必须具有视觉.

人的视觉通常是识别环境对象的位置坐标,物体之间的相对位置,物体的形状颜色等;由于人们生活在一个三维的空间里,所以机器人的视觉也必须能够理解三维空间的信息. 但是这个三维世界在人的眼球网膜上成的像肯定是一个二维的图像. 人的脑子必须从这个二维图像出发,在脑子里形成一个三维世界的模型.仿照人脑这个功能从一个二维的图像中理解和构造出一个三维世界的模型,称为计算机视觉,或叫图像理解.

计算机视觉要从图像构造出对物体的明确而有意义的描述.

图像理解完全不同于图像处理，图像处理指的是图像到图像的转换，它不是一种清楚的描述结构，而描述是对物体进行识别、操纵、研究的必要条件。视觉系统是从输入中可靠地提取固有信息，并且在处理的各个阶段使用了许多前提和已有的知识。

计算机视觉是一个相当新的而且发展很迅速的领域，它的第一批实验是在 50 年代后期通过的。但许多基本概念是近五年内才形成的。从 60 年代开始首先处理积木世界，后来处理桌子、椅子、台灯等室内景物，进而处理室外的现实世界。70 年代以后有些实用性的视觉系统出现了。另外，随着这门新学科的发展，一些先进的思想，在人工智能、心理学、计算机制图学、图形处理等领域产生出来。

图像理解的过程如图 7.1 所示。

计算机视觉与通常的以二维世界作为对象的图像处理相比较有以下几个特征：

(1) 物体的形状是与观察者的观看方向有关系的，同一个物体从不同角度去观察，能得出不同的画面。

(2) 两个以上的物体互相有遮盖的现象发生，被遮盖的部分是看不见的。

(3) 物体表面图像亮度的分布是与照明方向有关的。

(4) 离开摄像机的距离不同，物体的大小也不同。

(5) 从形状的类似，再前进一步，着眼于识别物体本身具有的功能(例如不同形状的椅子，我们都把它看成椅子)。

图 7.2 为一图像理解系统的构成。

所以说计算机视觉是具有比较浓的人工智能的色彩，是人工智能的重要课题。

在图像理解中摄像机所拍摄的三维世界我们称为景物，这与成为系统的输入的二维图象有严格区别。输入图像可以是一幅图像，也可以是像录像以及航空摄影那样拍摄的时间不同、角度不同

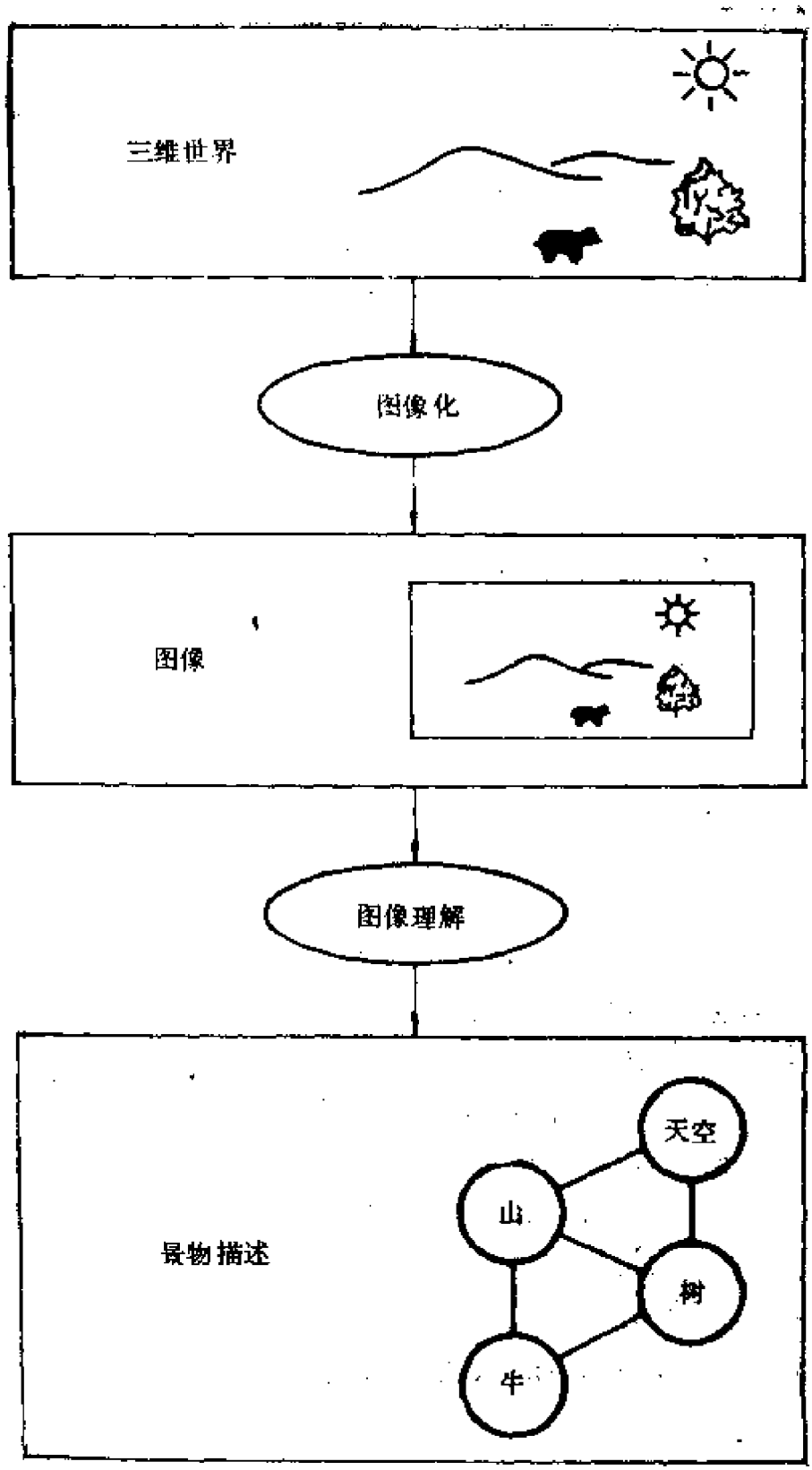


图 7.1　图像理解的过程

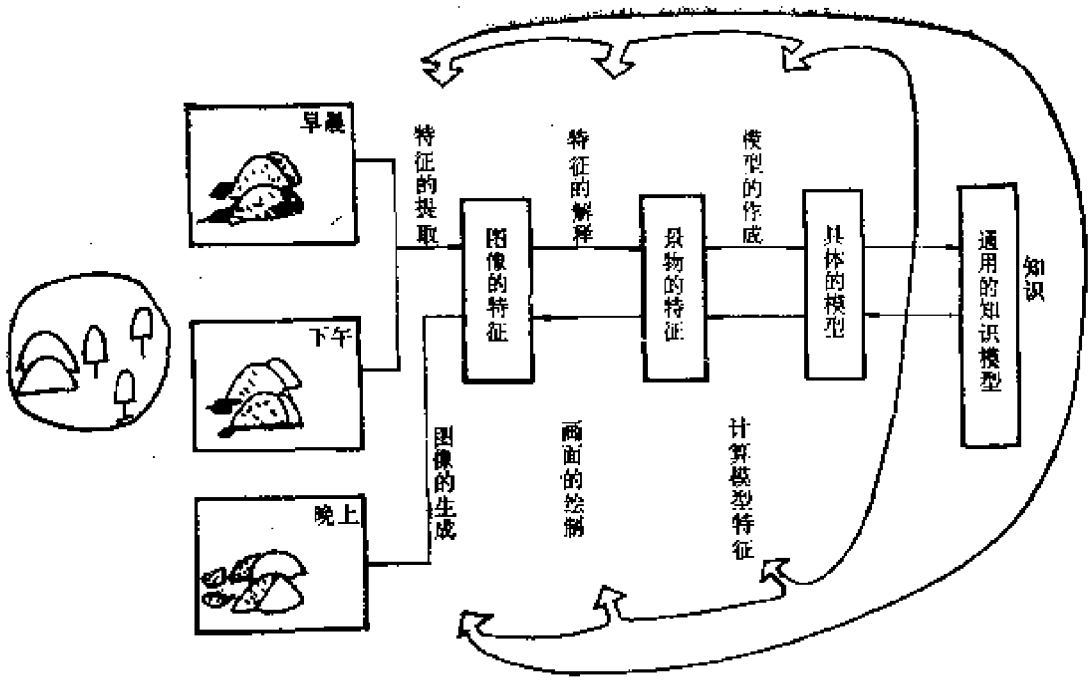


图 7.2　图像理解系统

的一批图像.

特征的提取是一种低水平的处理,是一个一直在研究的领域,与一般的图像处理是相同的.

在图像理解中还需把所得到的特征翻译成景物内的特征. 因为人们在观看物体时,不仅考虑亮度急剧变化的特点,同时要看物体的各个面的方向距离、颜色、反射率, 也就是要根据上述特征来构成物体的三维模型. 这些特征跟照明的情况、观看者的位置无关,是物体固有的特征,叫景物特征. 这些特征的提取叫中级水平的提取. 我们把物体的认识或图像的理解叫高级水平的处理. 图像的特征是二维的,物体的认识是三维的,所以有时把景物的特征叫 $2\frac{1}{2}$ 维.

§7.2 机器人的视觉硬件系统

人眼处理的信息无非是视野范围内的亮度、颜色，以及距离等．人眼的视觉系统由下列部分组成：(1)光电变换(视网膜的一部分)，(2)光学系统(焦点与栏的调节)，(3)眼球运动系统(水平、垂直、旋转运动)，(4) 信息处理系统(从视网膜到大脑的神经系统)．

为了实现此功能的计算机硬件系统如图 7.3 所示．

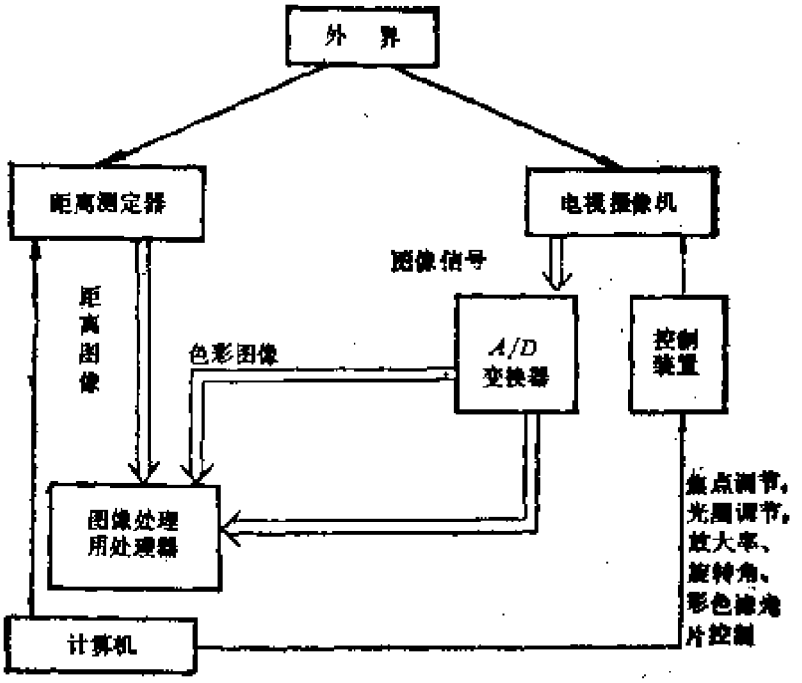


图 7.3 机器人的视觉硬件系统

输入图像经 A/D 变换器转换成数字量，从而变成数字化图形．通常一幅图像划分为 512 × 512 或者 256 × 256．各点亮度用 8 位二进制表示即可表示 256 个灰度．图像输入以后进行各种各样的处理，识别以及理解，另外通过距离测定器得到距离信息，

经过计算机处理得到物体的空间位置和方位．通过彩色滤光片得到颜色信息．上述信息经图像处理器进行处理，提取特征，处理的结果再输出到机器人以控制它进行动作．用通用计算机来进行处理，要达到实时控制目的是比较困难的．因为处理花时间，尤其是边界的提取，为了达到高速化的目的，正在开发图像处理器．这些处理器的开发的一个方向，就是使求取边缘等的局部处理使用专用的处理器，另一个方向是可以根据用户的要求来编出程序，成为具有一定柔性的图像处理器．

图像处理器所得到的特征，用计算机来进行分析，从而得到物体的种类和位置的关系，在这方面用得比较多的计算机语言是LISP．

另外，作为机器人的眼睛不但要对所得到的图像进行静止地处理，而且要积极地扩大视野，根据所观察的对象改变眼睛的焦距和光圈．因此，在系统中应能：(1)为了适应所要看对象的距离调节焦距；(2)根据其亮度调节光圈；(3)适应其大小调节放大率；(4)为了选择视野，使 TV 摄像机旋转．

§7.3 图像理解系统

其实上面已粗略介绍了图像理解系统的大致构成，这一节详细介绍一下．

图像理解系统在设计上应该考虑的有以下几个重要问题：(1)从图像中提取哪些信息？(2)这些信息在图像中怎样表示？(3)在理解过程中计算机必须具有哪些知识？(4)最合适的图像处理程序是什么？(5)知识以及各个阶段图像处理的结果如何表示？

为了稍详细地加以说明，我们可以用图7.4来表示一个图像理解系统．当然不是所有的图像理解系统，都得有像图7.4所示的那样一些过程和内容．实际上，不同系统的过程和内容可以有很大

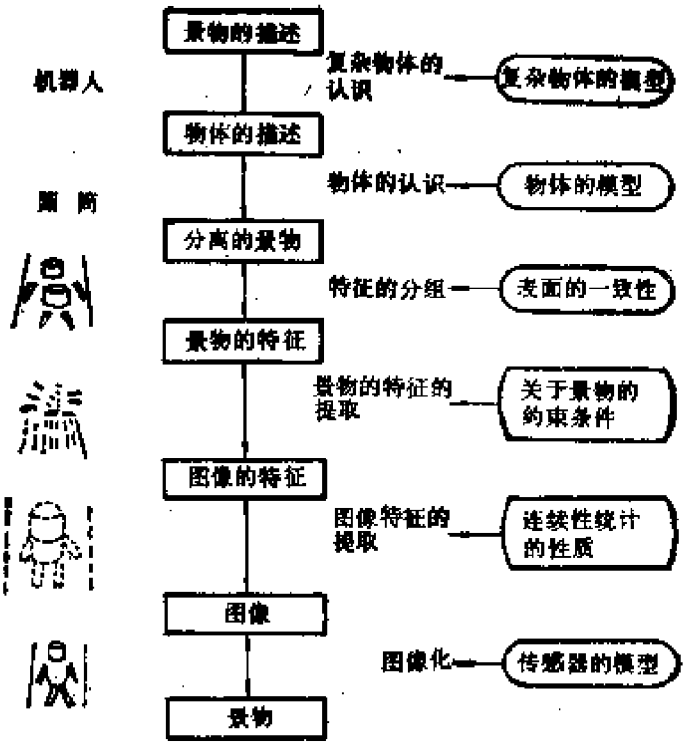


图 7.4　典型的图像理解系统

差异。

1. 传感器引起的失真的补偿

输入的图像是用灰度的二维阵列表示的，通常由于传感器的特性会引起几何学上或者光学上的失真，因此须进行补偿。

2. 图像特征的提取

图像凡是亮度发生急剧变化的地方，都是对应物体面与面之间的边缘，这是一个重要特征，所以把图像中亮度急剧变化的点提取出来并对其性质进行描述(如边缘的方向、幅度、直方图等)；对这些特征点统计的分布状况进行研究，可认识到物体的构造。如果把相邻的特征点连接起来则可以构成输入图像的线画。这个阶段的处理上面已说过是低水平的处理。

3. 景物特征的提取

上面处理的只是图像上的特征。人在观看物体时看到的不仅

是亮度急剧变化的点，同时还有各个面的方向、距离、颜色、反射率，这些特征与照明的情况和视点的位置无关，是物体所固有的特征。为了与图像上的特征相区别，一般称为景物特征。

图 7.5 表示了景物的特征

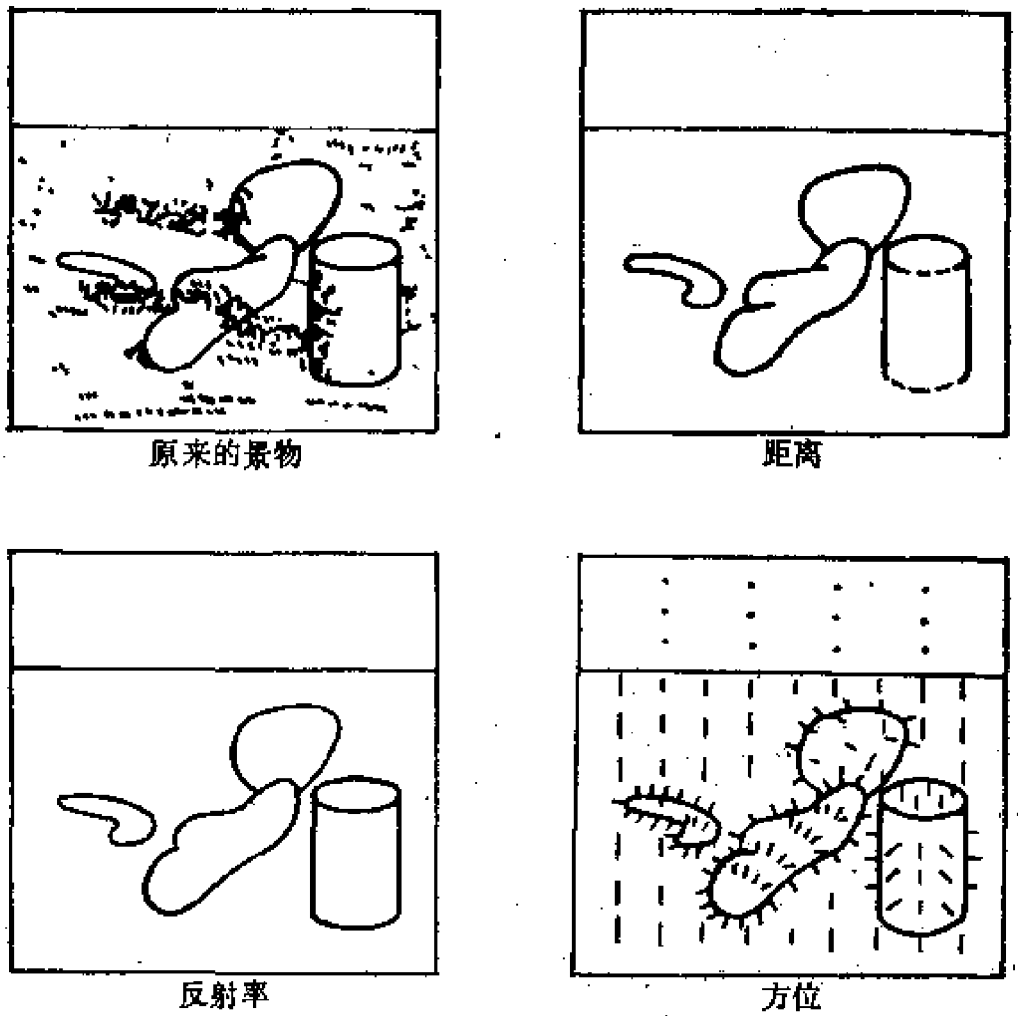


图 7.5　景物特征

图上只用线条加以表示，实际上各点都标有数值。距离是从视点到景物的各点的距离，面的方向是在面上各点与面垂直的方向。距离可以用两台摄像机，或窄细状光通过三角测量原理或者用激光测取反射回来的时间等方法求得，其它景物的特征通过灰

度图像不容易得到.

图像各点的亮度是由照明条件，面的反射率和面的方向决定的;如考虑理想的乱反射，当照明光的强度设为 I，面的反射率为 R，照明光的入射角为 i 时（图 7.6）图像各点的亮度为

$$L = IR\cos i$$

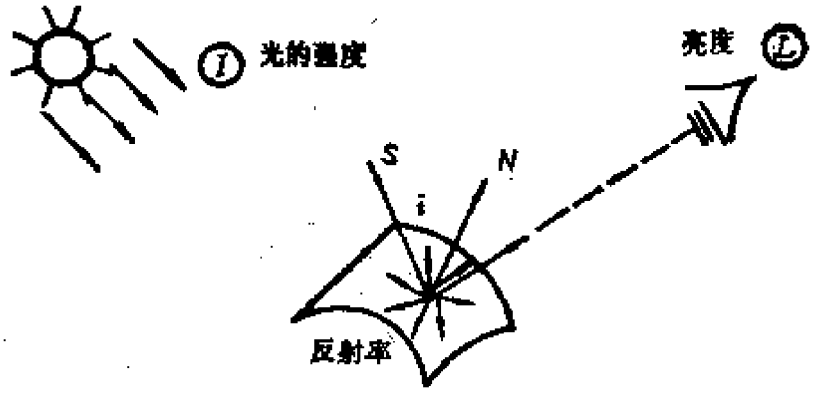


图 7.6　理想的乱反射

只根据上式并仅当 L 为已知，要想求出三个未知数 I、R、i 是很困难的,所以为了求景物的特征,必须知道关于景物的一些其它知识(或约束). 使用这些约束条件，求取景物的特征的方法有很多,其中主要的方法有如下两种.

(1) 由亮度决定形状的方法　Horn 使用远的点光源来照明(也就是对面来说有一样的照明)并且假定面的反射率一定，这可以决定面在三维空间中的形状:

$$L = IR\cos i$$

进一步有

$$dL/L = dI/I + dR/R + d(\cos i)/\cos i$$

根据假定 I 和 R 是一样的,则 dI, dR 为零,因而

$$dL/L = d(\cos i)/\cos i$$

上式表明，面的方向的变化率与图像上点的亮度的变化率是相等

的，所以在给定了适当的边界条件以后，对上式进行积分，可以决定面的方向。

对于边界条件，Horn 利用了顶部边界线和终端边界线，如图 7.7 所示。顶部边界线是曲面与背景的分界线，在此，面的方向能够唯一地决定。也就是边界上各点的面的方向是与视线成直角的，而且与图像上的轮廓垂直。

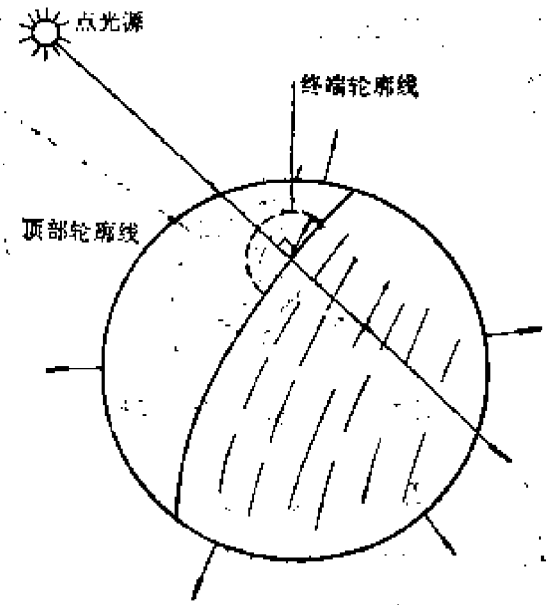


图 7.7　边界条件

终端边界线是光照区域与影子区域的分界线，此时面的方向与光源垂直。由上述这些已知条件以及公式

$$dL/L = d(\cos i)/\cos i$$

采用反复计算法可以决定其他部分面的方向，上式不适合影子部分，可以由边界条件根据内插法来推定。

另外 Horn 还去掉反射率一定的假定，对上述方法进行了扩展。

（2）由轮廓决定形状　人们一看到图 7.8 那样的线画就不难推测各个面的方向。Barrow 等人就是研究如何从这样的线画出

发决定各个面的方向。

图中那样的线画上有两类边界线：一类是上面已叙述过的前端边界线；另一类是两个相连接的面的边界线，称为不连续边界线。前端边界线，如前面所述那样，面的方向是唯一的决定的；而在不连续边界线的地方，面的方向不能唯一地决定，但是存在跟边界线相接的线相垂直这样一个约束条件。

为了决定这个不连接边界线处面的方向，Barrow 使用了两个假定，一个是面的"平滑"，另一个是"一般的视点"。当假定了面是"平滑"的时，其边界线当然也平滑，而且它在图像上的投影曲线也平滑，可是反过来在图像上的曲线"平滑"，在原来景物上的不一定是平滑曲线，为了解决此问题，Barrow 作了一个"一般的视点"的假定。

例如在图7.9的图像上，对应于椭圆的景物上的曲线可以有许多种。对于其中的不连续曲线，由于它们在图像上的投影为椭圆的情况是非常少见的，所以在现实中可能性是非常小的。如作"一

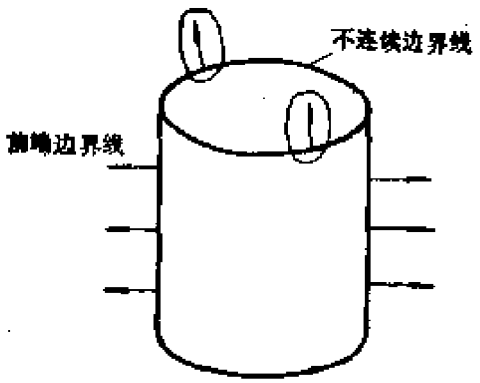


图 7.8　边界线分类

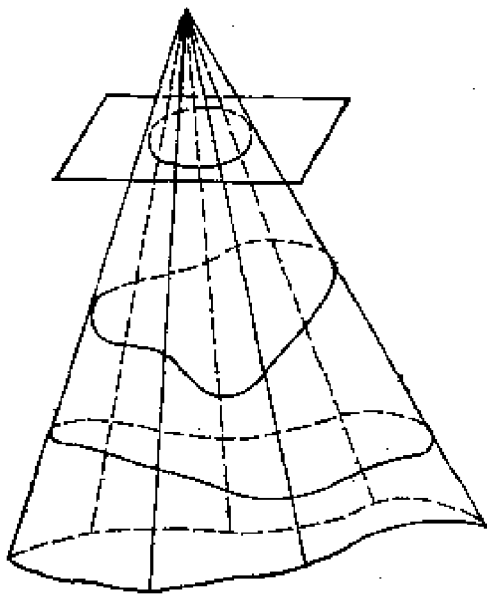


图 7.9　与图像上的椭圆对应的景物曲线

般的视点"这样一个假定，那末就可认为图像上平滑的曲线，在景物中也是平滑的。

剩下还有一个问题，即当景物上有许多曲线对应图像上的一根曲线时，从其中选择哪一条呢？在现实世界中，物体的表面是光滑的，所以多数是取能量最小的构成，由此，Barrow 等人主张选取曲率变化小而且挠率小的曲线，即，根据各曲线的曲率 K 和单位副法线向量 $\boldsymbol{b}$（图 7.10），分别求取如下式所示的曲线积分：

$$\int (dK \cdot \dot{\boldsymbol{b}}/ds)^2 ds$$

最后选择积分值最小的曲线。

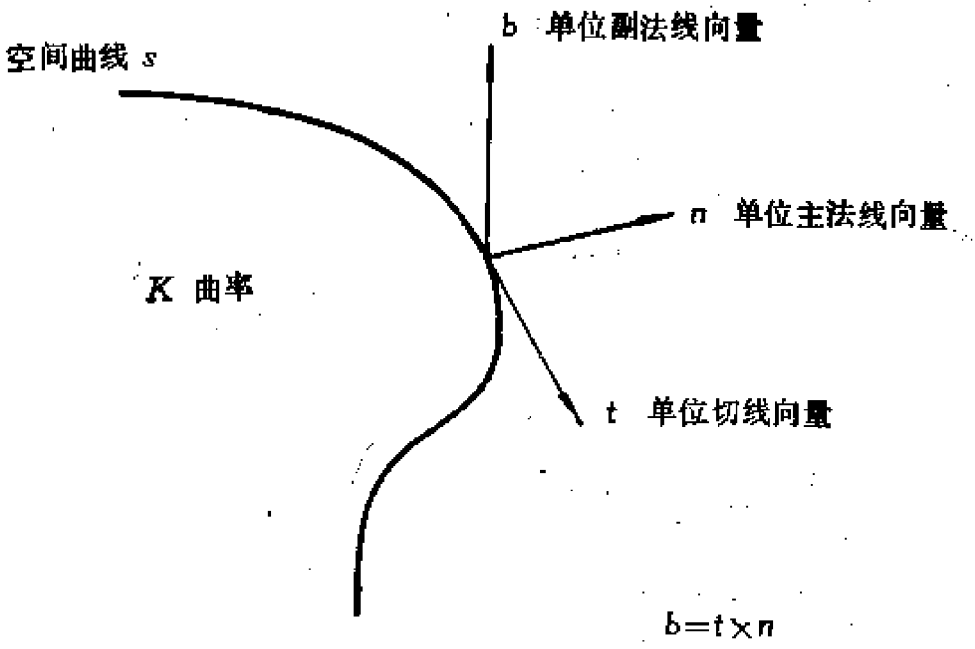


图 7.10 $\boldsymbol{b}$ 的定义

显然，在 K 一定，而 $\boldsymbol{b}$ 的变化为零也就是曲线为圆时上式积分为零。

在上述两个假定下，对线画进行三维空间的解释，Barrow 等人又从线画的三维解释出发确定面上各点面的方向，现在对于对称与圆筒等简单的曲面已得到良好的结果，但更复杂的物体还没

有解决.

除了这些研究以外，由物体结构要素的大小的变化来确定面的方向以及由亮度来求面的反射率等研究正在进行.

另外，不管哪一种方法都是对景物有一些假定，所以单独使用都使适用范围受到一定限制. 为此，把所有方法综合起来，研究出更一般的方法的工作也正在进行.

在智能机器人视觉系统中，最近研究的重点是景物特征的提取方法.

4. 景物的分割与物体的发现

利用景物的特征，把其性质大致相同的领域分割开来，这与图像处理中把亮度相同的领域分割出来的领域法相同. 但是亮度是与照明的情况、面的倾斜等许多因素有关，而在景物中是使用固有特征进行领域分割，所以可靠性高. 另外用亮度分割的领域分割法，其结果物理变化不明确，而用景物特征的分割结果，物体的各面在物理意义上是对应的. 所以可以根据景物检测出的面，由各个面的接续关系来发现物体. 面的性质和位置的描述当然也是用记号来描述，在面的发现阶段，其处理结果的描述是由图像的描述改为用记号来描述.

5. 物体的识别

从景物所发现的各物体再参照物体的模型来识别物体，与此同时确定各物体的三维空间位置和方向.

6. 景物的识别

研究各物体之间的连接关系，由这个连接关系和模型来认识更复杂的物体(例如由手脚等各部分物体的性质和连接关系可以识别人)，从而可以对景物的全体进行描述.

物体模型的描述方法有好多种. 图 7.11 为一个由于物体复杂，所以分层次描述的模型. 各模型由圆筒组成，圆筒的断面与轴是可变的.

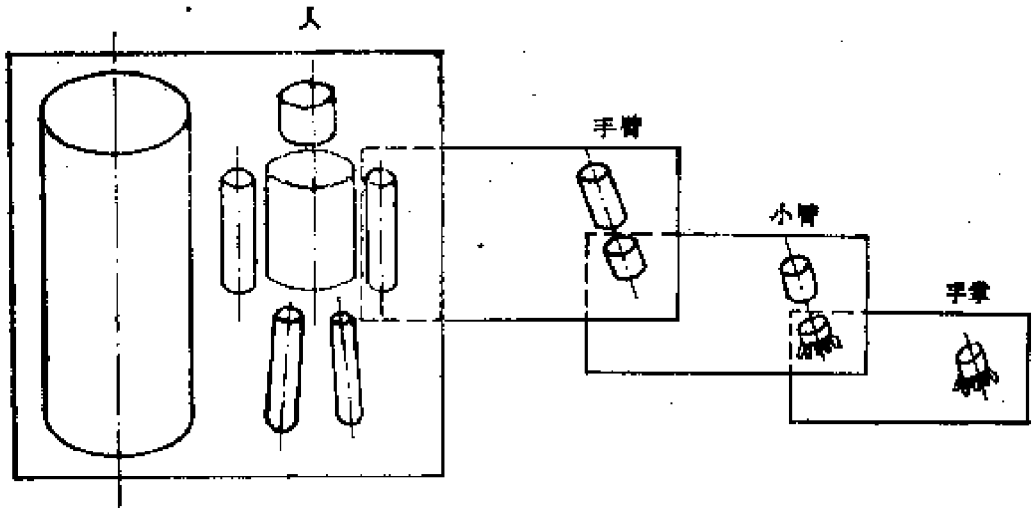


图 7.11　人的分层描述模型

7. 具有反馈的处理

上面所述的处理是从低水平到高水平的处理. 这种处理方法计算量很大,计算结果还有不确定的地方,如从图像的灰度信息要得到完整的线画就很困难. 从图像开始到得出模型这样一个处理过程叫作"由下往上",或称"数据驱动";从模型开始向下的处理过程叫作"由上往下",或称"模型驱动",只使用其中一种方法总是不充分的,通常使用由下往上,再由上往下称为"反馈"的处理方法.

首先从图像开始使用"由下往上"方法, 提取图像和景物的特征,发现物体,然后根据这些物体和物体之间的接续关系, 假定一个模型,再对这个模型进行验证,如此反复进行.

§ 7.4　图像特征的提取

上面已经叙述了景物特征的提取, 这一节再回过头来对图像特征的提取作一概述.

我们用眼观察物体时, 一般是根据光的影子来判断物体的形状, 即根据不同的亮度来判断物体的轮廓. 也就是说根据图像浓

淡变化，则可以提取物体的轮廓。所以，计算机视觉或景物分析首先是分析图像的灰度信息。一般找出亮度急剧变化的分点，并把这些点连成连线，即使图像变成了线画(线条图像)、或者采用区域划分的办法，求出亮度大体相同的区域。

1．空间微分法

提取图像亮度变化的特征的方法称为空间微分法，其中又有一次微分法和二次微分法。

$$\nabla W(x, y) = \frac{\partial W(x, y)}{\partial x} u_x + \frac{\partial W(x, y)}{\partial y} u_y$$

u_x、u_y 是 x、y 方向上的单位向量。

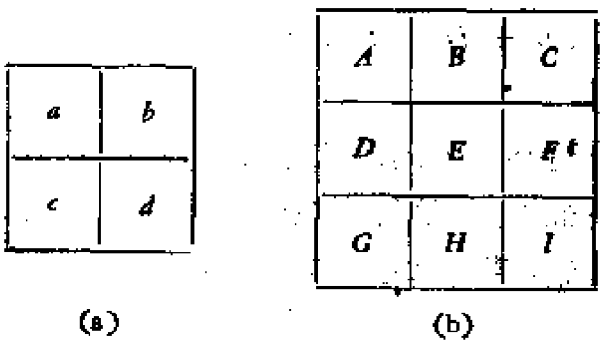
$W(x,y)$ 为 (x, y) 点的亮度，$\nabla W(x, y)$ 为亮度的微分，使用上式叫一次微分。这个方法一般用于多面体的情况。

二次微分为

$$\nabla^2 W(x, y) = \frac{\partial^2 W(x, y)}{\partial x^2} + \frac{\partial^2 W(x,y)}{\partial y^2}$$

这种方法一般在立方体的图像情况下使用较多。

在使用中一般是用差分的办法，在图像内各点设想一个图 7.12 所示窗口，这叫微分运算窗口。



(a)　　(b)

图 7.12　微分运算窗口

窗口里面的数值为亮度的数值，采用各种计算公式计算出窗口的空间微分值，当它超过某个阈值时就作为特征点提取。

对于图 7.12(a) 采用 Roberts 的方法，其微分值为

$$\sqrt{(a-d)^2+(b-c)^2} \qquad \text{(一次微分法)}$$

对于图 7.12(b) 3 × 3 的窗口可采用 Pingle 的方法，即其微分值为

$$\sqrt{Dx^2+Dy^2}$$

其中

$$Dx=(C+2F+I-A-2D-G)/8$$
$$Dy=(A+2B+C-G-2H-I)/8$$

各点的方向为

$$\alpha=\tan^{-1}(Dy/Dx)$$

上述方法都有其局限性，如第一种方法，在处理三角柱的边界时，因为三角柱斜面与背景之间亮度相差较小，所以很难提取边界。为此就有许多边界检测的方法。例如

Forsen: $|a-d|+|b-c|$

Fjiri: $(|a+c-b-d|+|a+b-c-d|)/2$

Rosenfeld: $[(A+B+C-G-H-I)^2+(A+D+G-C-F-I)^2]^{\frac{1}{2}}$,
$|A+B+C-G-H-I|+|A+D+G-C-F-I|$

Hawkins: $(B+D+F+H-4E)/4$

2. 直线的检测

把相邻的具有同一方向的边缘部分作为核心，然后一个一个地追踪相邻的特征点，先估计直线的方向，在估计出现下一个特征点的附近进行搜索，若找到适当的点就向前推进，若是没有就认为是线的端点。

此外，还有利用图像内全局的信息来寻找直线的方法。

3. 区域划分

把输入图像划分为性质相同的区域，如为了把背景与处理对象分离开，最简单的办法是设一个阈值，把亮度超过此阈值的区域

检测出来.

§7.5 线画的解释

计算机在分析积木世界的图像时，提出了一种线画的解释方法，这种方法奠定了图像理解的基础. 为了考虑问题的本质，作了下面的假定，先考虑一种简单的情况，即三面顶点且不带影子的情况：(1)图像中所有的棱(亮度急剧变化的点)可全部检测出来，所得到的线画是准确无误的；(2)景物中不存在影子；(3)在景物中的存在的顶点，是由三个面构成的，这样的顶点叫三面顶点(对于图7.13所示的非三面顶点，暂不在研究之列)；(4)摄像机的角度、位置即使稍微变化一下，线画的结构也不变化(图7.14中的例子则不满足这些条件).

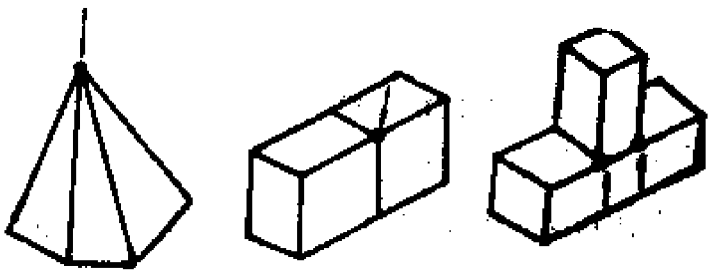


图7.13 非三面顶点的例子

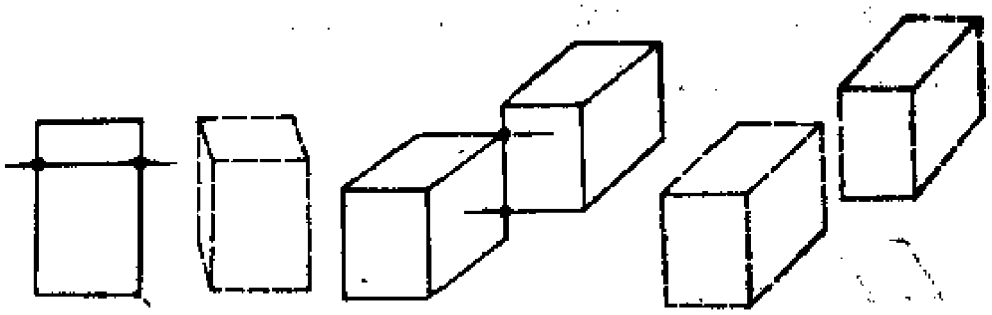


图7.14 摄像机的位置变化引起线画构造的急剧变化

1. 线画的标记

作为图像理解的第一阶段，先研究输入图像中各条线在三维空间中对应什么，也就是对图像内的线条的意义进行标记.

两平面相交所决定的积，在画面内是一根线，其中如果有一个平面隐藏起来看不见时叫边界线(境界线)，两平面都看得见的线叫内线. 境界线用箭头表示，即绕境界线走使看得见的面在朝箭头前进方向的右侧，这样来标记箭头. 内线又分凸的棱和凹的棱两种，凸的内线用"+"号标注，凹的内线用"—"标注. 如图 7.15 所示.

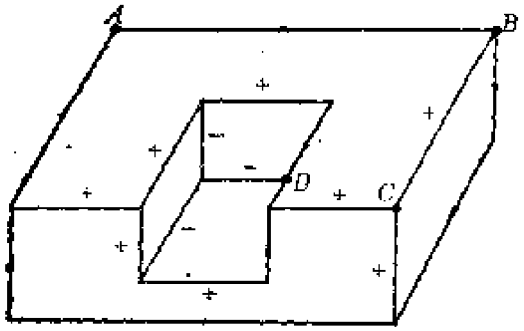


图 7.15 线画的标记

图形内有四种类型交点：L 型(如 A 点)，A 型即箭头型(如 B 点)，Y 型(如 C 点)，T 型(如 D 点). 根据不同的组合，形成交点的直线有四种记号：+，—，→ 和 ←.

L 型有 16 种组合，A 型有 64 种组合，Y、T 型也有 64 种组合. 总共有 208 种交点，可是在现实世界中，只有其中的 18 种，因为在现实世界中只有图 7.16 所示的四类三面顶点.

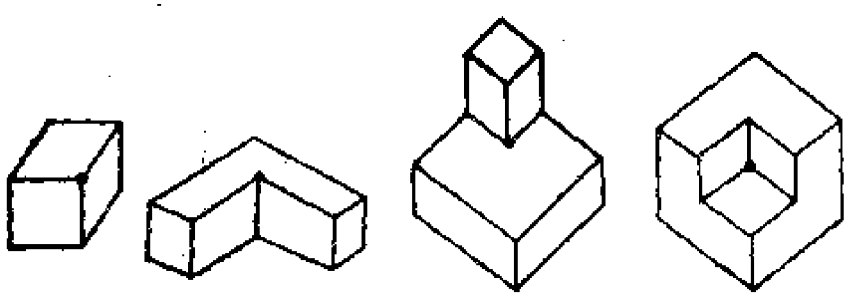


图 7.16 四类实际存在的三面顶点

这样，实际存在的顶点的形式共有 18 种，如图 7.17 所示。

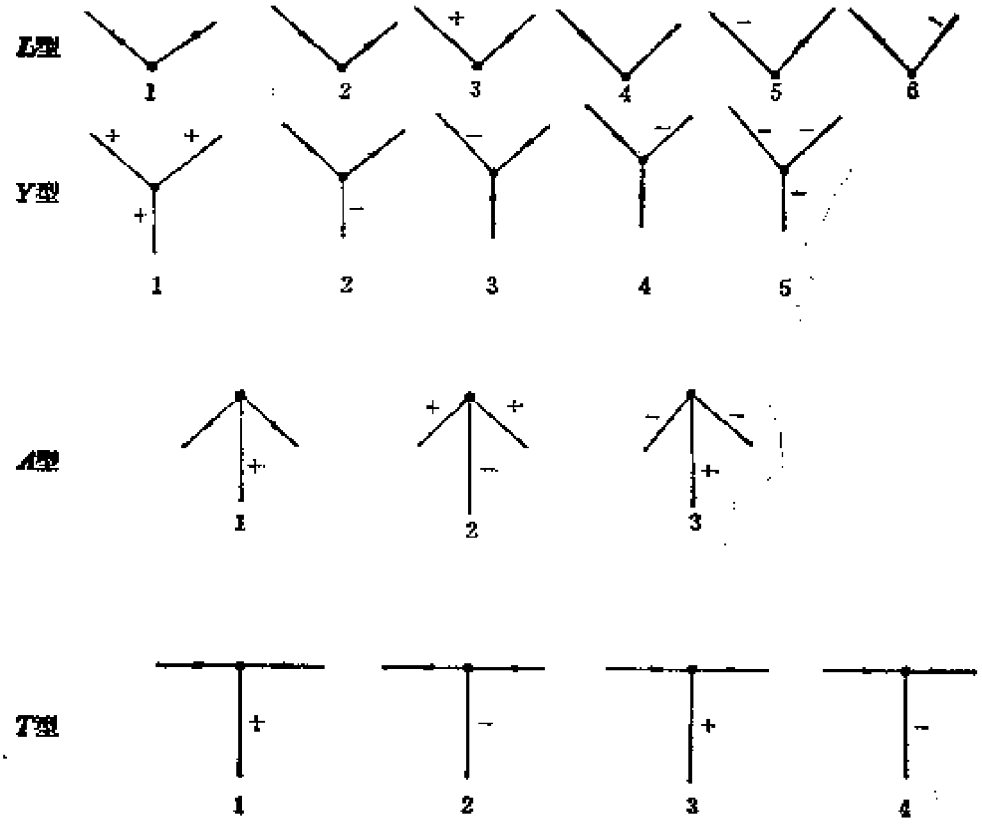


图 7.17　实际存在的 18 种顶点

T 型是交点境界线与一部分被隐藏起来的棱所构成的，不是对应顶点。

在线画的标记中往往采用人工智能的一个基本手法——探索传播法。

下面以图 7.18 的线画标注为例进行说明。

首先把与背景的境界线按顺时针方向标上箭头，则如图中的 (b) 所示。然后对各交点进行标记，边标记边搜索，使其没有矛盾。

先看 *A* 点，它属于 *A* 型，由 *HA*、*AB* 的箭头方向，再与上述 18 种顶点相对照，可知应该为 A_1 型，所以 *AD* 线应标上“+”号，同样，*CD*、*ED* 也应标上“+”。

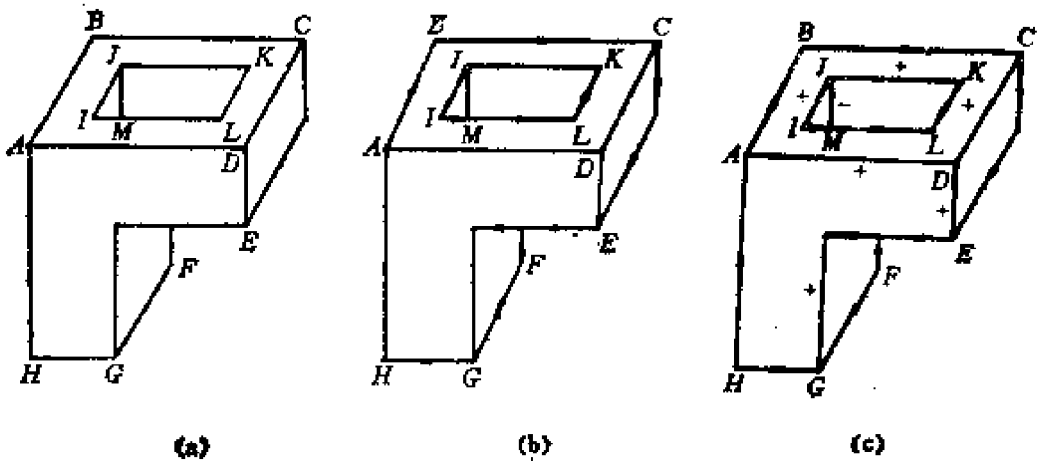


图 7.18　线画标记的例

再看 I 点，它为 L 型，IJ 有两种可能，一种为“+”，一种为“→”. 为了确定究竟是哪一种，再考虑 J 点，因 J 点为 A 型，如果 IJ 为“→”，J 点与 A_1、A_2、A_3 相矛盾，所以否定了 IJ 为“→”的可能，如果 IJ 为“+”，则 J 点可以为 A_2 型顶点，所以可以确定 IJ 标记为“+”号，同时 JK 为“+” JM 为“−”.

又例如，一个外侧线不在画面上的例子(从画面溢出的情况)，符号标记更复杂(图 7.19).

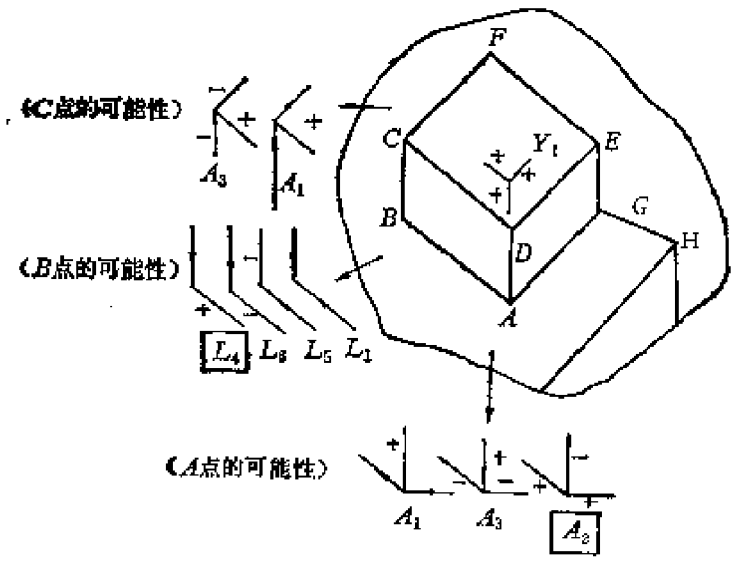


图 7.19　外侧线不在画面的线画标注(一)

一般是从种数最少的箭头型开始进行探索，即从 A 点开始到 H 点为止。

A 有三种可能，B 有四种可能，传到 C 则只有两种可能，即除掉了 B 点的 L_4，从 C 再传到 D，则 D 应为 Y_1 型交点。从 D 再返回到 A，看出 A 的 A_2 型可以去掉。

又继续传播到 H，利用同样方法进行探索。

由 H 点只可能有 A_1，A_3 型交点，所以 G 点应排除 Y_4 的可能。

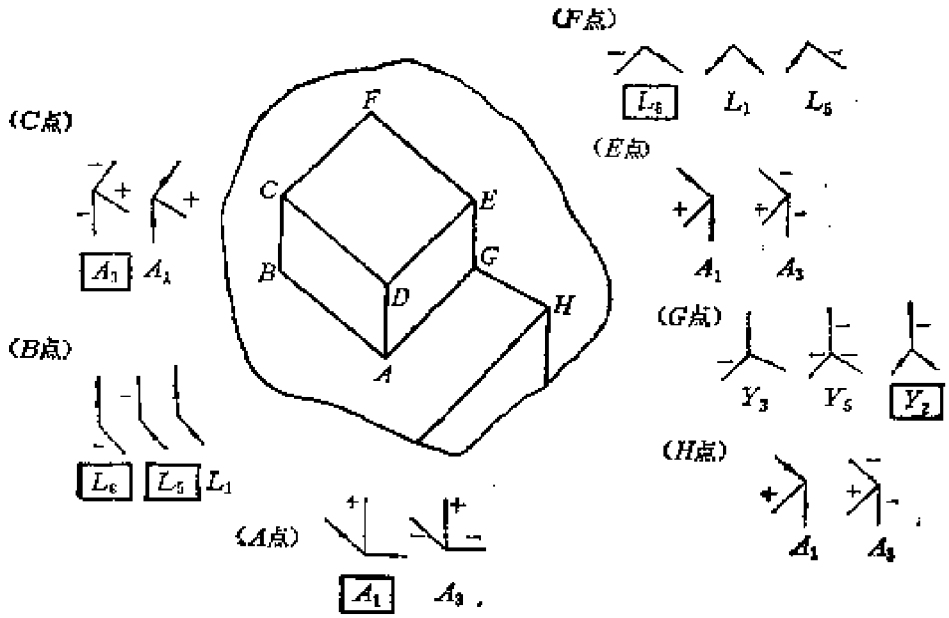


图 7.20　外侧线不在画面的线画标注（二）

再从 $H \rightarrow G \rightarrow A \rightarrow B \rightarrow C$ 传播，A 点应去掉 A_1，B 点应去掉 L_1，L_5，C 点应去掉 A_4。

再传到 F、E、G、F 点应去掉 L_6，最后得到两种结果，这个结果在 A、B、C、D 点只有一种标记，而 E、F、G、H 点则有两种标记(图 7.21)。

上述那样利用局部的约束条件，能够对全局进行解释，这种手法不仅适用于线画的解释，也适合于人工智能的其它方面。

这种符号标记也可以判断所给的线画是否表示的是一个实在

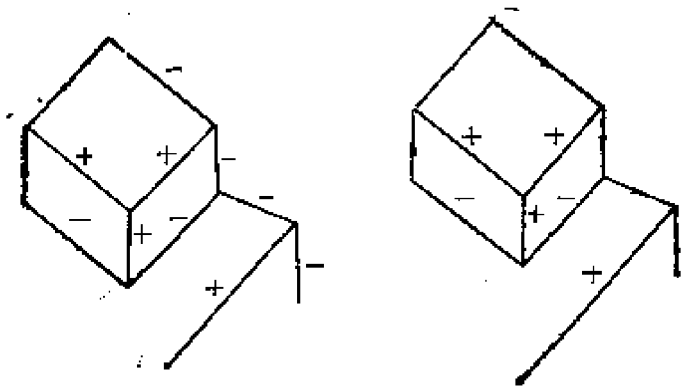


图 7.21　外侧线不在画面的线画标注（三）

多面体，例如图 7.22 线画，从境界线开始，依次对相邻的交点进行符号标记，但是图中"?"表示的交点不在前述 18 种交点范围内. 所以可以得出结论，图示多面体实际上是不存在的例子.

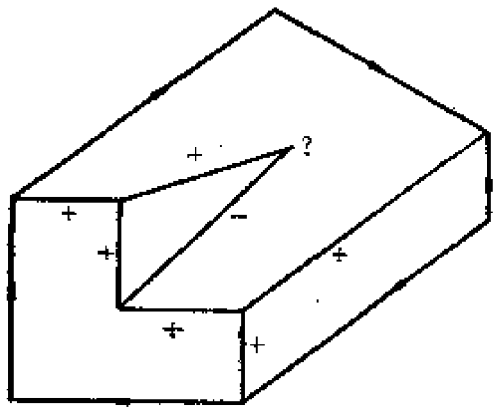


图 7.22　实际不存在的多面体

这种方法也可推广到具有非三面顶点的物体. Walt 把这种思想推广到带有阴影的输入图像的线画解释，他除了考虑凸凹之外，又考虑了裂纹、前方边缘、可以分离的凹线、影线等 11 种符号，并且弄清了它们所对应的法则，显然，由于符号种类的增多，乍一看好象是天文数字那样多的组合，但是由于把现实世界现存的一些法则作为约束条件，所以问题能比较容易地得到解决. 人们一般认为这是确立人工智能的一种方法的范例.

2. 不完整线画的解释

上面讨论的是线画解释，即假设输入图像的边缘能全部地正确地检查出来. 但是在一些复杂的情况下，就不能指望得到完整的线画，1972 年 Falk 提出了一种分析不完整线画的方法，这种

方法有一个先决条件，即构成对象物体的种类有一定的限制，而且这些物体的几何参数是已知的。

如图 7.23 所示的由特征提取的线画大概缺了一部分，我们称为不完全的线画。

可以设想如图中（b）那样存在一些线条，并把它延长，即大体估计那里应有一根线存在。然后加以延长，再根据面的形状以及

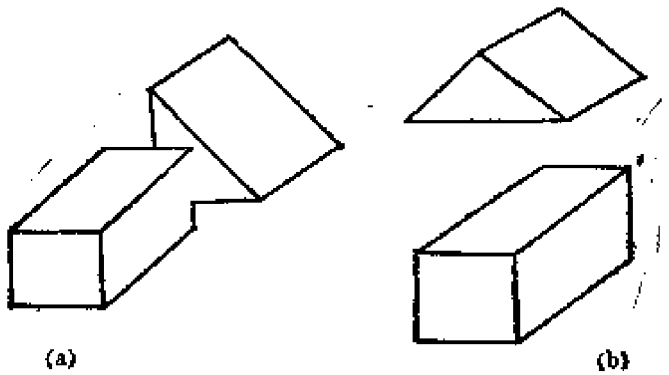


图 7.23　不完整线画

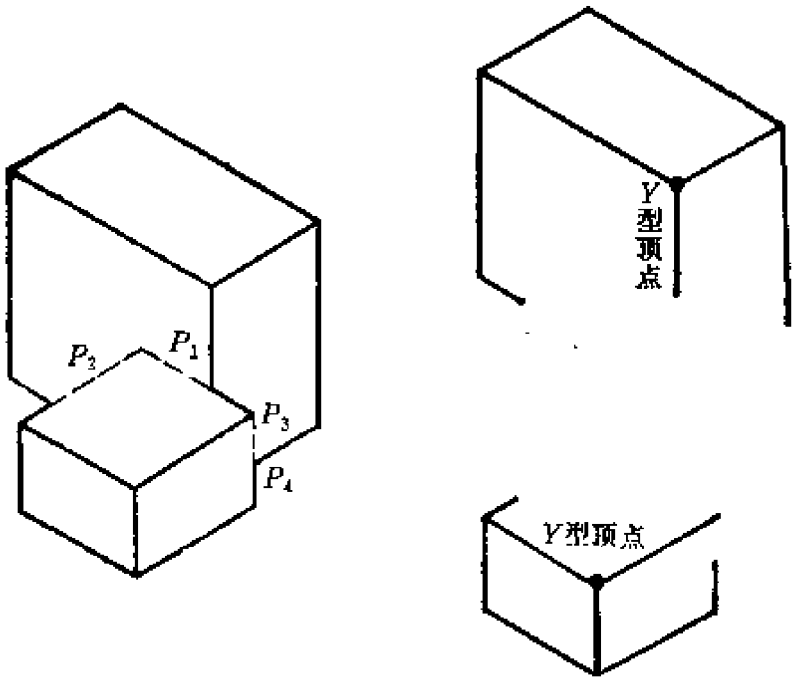
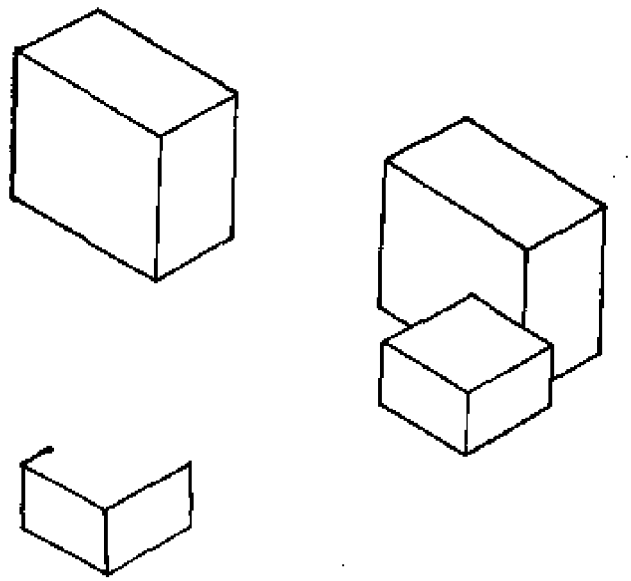


图 7.24　不完整线画的解释（一）

线的长度等特征推断出物体的模型，用此模型再在画面上进行匹配。

下面我们举个例子来说明这一过程(图 7.24)。

寻找 Y 型顶点，并且以此作为物体的核心。 往这些顶点进行联系，并且从逻辑上判断哪些线属于哪个物体，这样得到图 7.24 (b) 那样两组，按上面所述的方法，根据预测法，把所缺的线在一定程度上予以补缺[图 7.25(a)]。



(a)根据预测法补缺　　(b)与模型对照后得到的线画

图 7.25　不完整线画的解释(二)

填上未解决的线 P_1P_2，补齐前面物体所缺的部分，然后与模型进行对照。

把识别的结果构成的线画再与原来的线画进行比较，如果没有什么矛盾，则认为成功了(图 7.25(b))。

日本白井研究了一个处理由许多个多面体组成的线画方法，就是在从输入图像中提取线画的过程中灵活运用积木世界的知识。他不像 Falk 的方法那样，只是推断线条的欠缺情况，而是改变特征的提取方法，对认为有可能有线条的部分进行深入的探索，

这是一种具有反馈的处理方法(图7.26(a)),可以应用于复杂图像的特征提取.

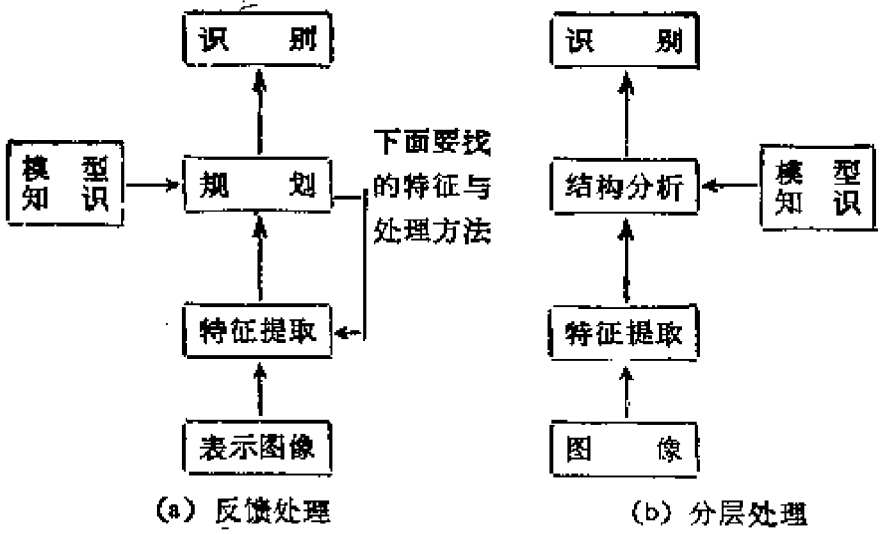


图 7.26　反馈处理与分层处理

3. 三维模型与学习

计算机为了把输入图像作为三维世界来解释，有必要利用三维模型. 下面介绍一下如何积极地使用三维模型来进行处理，并且介绍一下一般的模型构成和学习的方法.

(1) 三维模型的利用　上面说到在积木世界的解释中，我们假定从输入图像中能全部正确地检出其线画，我们研究与景物中的棱和面相对应的线和领域，研究结果可以找到一个一个的物体，如再与长方体三角柱等多面体进行对照的话，则就可以知道物体的形状及其配置结构. 这种处理的顺序叫"由下往上"法，或称"数据驱动"法. 但是对于复杂的图像，要从中抽取出准确无误的线画，几乎不可能. 所以采用上述处理流程正确地解释景物是困难的.

假如图像内所存在的对象已知的话，图像处理就变得容易. 在处理正面的人脸的照片时，如果首先发现了脸的轮廓，那么眼睛、

鼻子等的大体位置就能够预测。寻找在预测位置的近旁能与眼、鼻、口等的特征相吻合的模型，就可能分析脸的照片。这样一种使用脸的模型来研究图像特征的手法称为"由上往下"法，或称"模型驱动"法。脸的处理是用二维模型。对于三维世界的处理，首先由通用的物体模型生成当前看到的具体的景物模型，然后再预测用图像表示的特征，视线和物体所处的方向。例如在分析某一特定区域的航空摄影的时候，有可能预先给出所有的三维模型（如道路、海岸线）在三维空间中的位置，根据这个模型依次地在图像内寻找预想的特征。使用这个结果，能够分析复杂的图像，当然对于一般的图像理解问题，使用什么样的物体模型好，是不知道的，所以可用下面的方法。

首先从图像中尽可能多地提取可靠的特征，然后利用有关对象（景物）的知识，把图像的特征翻译成景物的特征。因为在系统中预先存储了许多物体的模型，所以能够根据景物的特征，选择有可能的物体模型。从可能性大的模型开始，假设在景物中存在此物体，并且在图面上加以检验，也就是用"由上往下"的方法。如果在预测位置上发现了预测的特征，我们认为假定是成功的，如果与预测产生了矛盾，那么认为假定是不对的，再选择下一个模型反复地进行处理。

作为一个简单的例子，考虑一个机械零件的识别。机械零件整体的形状是复杂的，但是大多数情况下可以用圆柱面、平面来局部地近似。此时一般使用图 7.27(a) 所表示的圆柱坐标系中的三维模型。旋转轴取 Z 轴，水平断面用圆弧和直线来表示边界，垂直面是用连接它的柱面来表示。当物体所处的方向一旦确定，计算机很容易从三维模型算出物体的线画，如图 7.27(b) 所示。

使用空间微分运算求出边缘，把边缘点相连提取直线和椭圆等特征，图像内的椭圆弧是对应景物中物体模型的圆弧。假定是垂直投影，所提取的椭圆的长直径则对应于圆的直径，据此从所存

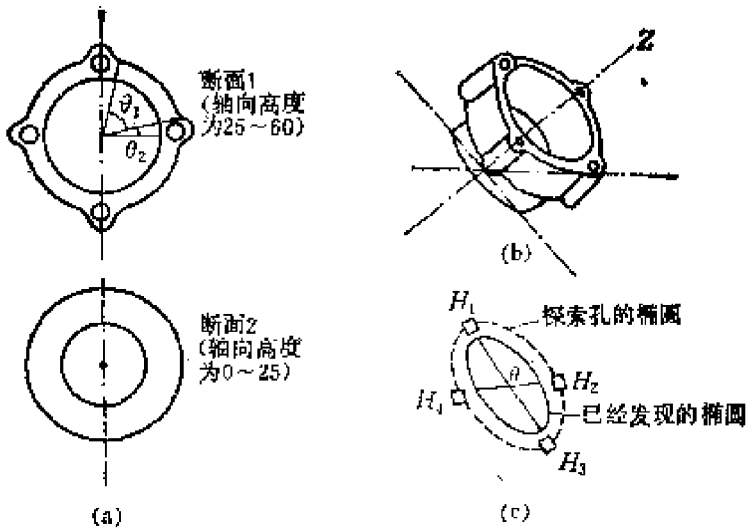


图 7.27　利用简单的三维模型进行图像理解

储的物体模型中选择具有这样圆的模型，如果具有相同中心而且相似的椭圆（或者椭圆弧），则所选择的模型必须具有同心圆(或圆弧)．根据模型决定图面上物体的像需要 5 个参数，从椭圆的常数决定其中的 4 个，即由中心位置决定包含圆的平面和旋转轴的交点，由椭圆的长径和短径的比以及长轴的倾斜度决定旋转轴的方向，绕旋转轴物体的转动角，只要发现了其他特征(如小的孔)，它的位置就确定了．接着由物体的模型预测图像内的特征，如果在输入图像内在预测位置上发现了这些特征，则认为所选择的模型是正确的．这个位置和形状由于是由模型给出的，所以可靠性较高．例如，在寻找小孔时，用“由下往上”方法就可能与图像内的噪声混同，而用现在这个方法，因为能够利用在如图 7.27(c) 所示的指定椭圆上有 4 个小孔这一条件，所以误认为别的物体的可能性是很小的．

(2) 一般的三维模型　一般把物体考虑成轴，物体的外形能够用圆柱来近似．D. Marr 提出了一种分层构造的模型．例如人的身体用一个圆柱来近似，它由头、躯体、手、足所组成，而头、躯体、手、足也用圆柱来近似，具有一种顺序细化的构造．一般的物

体也可用同样的模型来描述.

由于各个部分都用圆柱面来近似造成的误差较大，所以一般用锥面来近似，如图 7.28 所示.

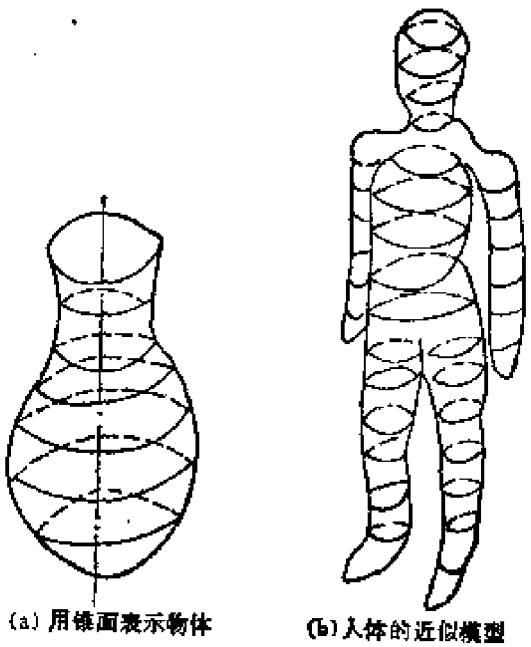


图 7.28　用锥面近似表示物体

(3) 模型的学习　到现在为止，图像理解仅着眼于物体的形状，根据其类似性进行理解. 然而为了把例如各种各样的椅子理解为抽象的"椅子"，并与其他的物体相区别，就必须建立"椅子"的概念. P. Winston 提出了学习概念的方法，即把知识教给机器人的方法.

图 7.29(a) 是一个房子的图像，计算机用图 7.29(b) 那样的网络表示出房子的概念.

图 7.29(b) 中只是描述组成"房子"的各个物体之间的关系，但并不反映"房子"概念的本质，因而必须进一步学习.

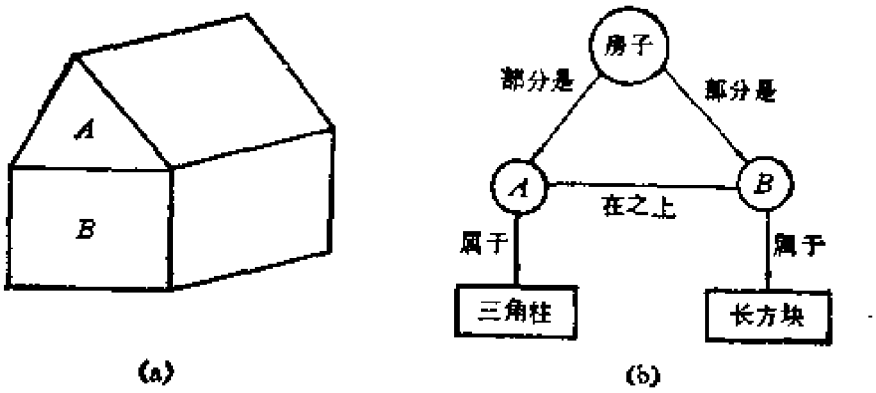


图 7.29　"房子"的图像和概念

图 7.30(a) 是一个"房子"的反例，它的关系网络与标准的"房子"网络相比较，结果表明，"单个物体 A 必须在物体 B 之上"这个条件对于"房子"概念来说是必不可少的，于是这一关系记作"必须是"。

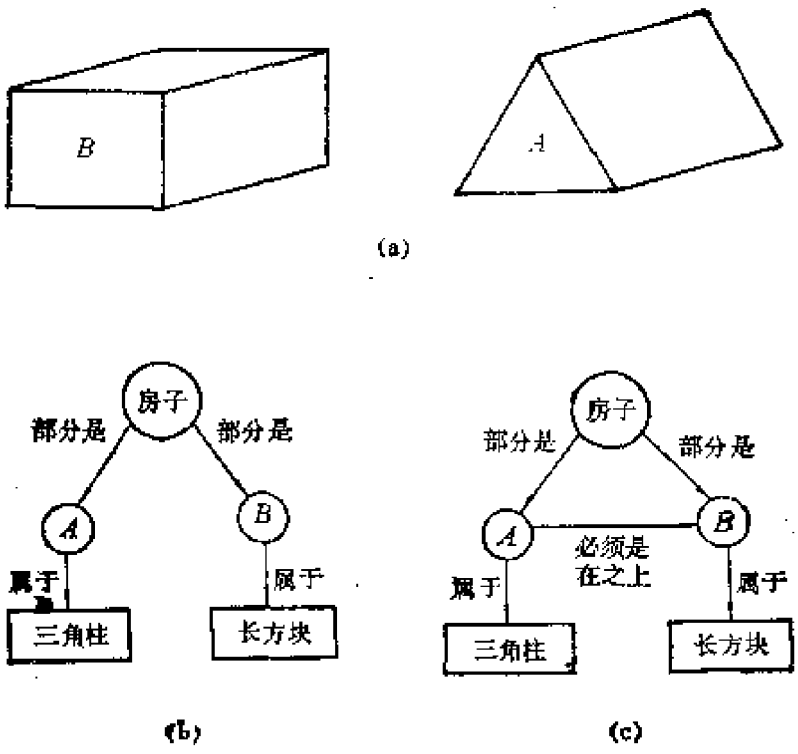


图 7.30 "房子"的反例和概念学习(一)

再通过一个反例进行学习。这是两个堆迭在一起的长方体(图 7.31)，通过比较学习的结果说明在"房子"上部的物体必须是三角柱。

再通过一个反例进行学习。这是两个堆迭在一起的三角柱[图 7.32(a)]，比较结果表明"房子"下部的物体必须是一个长方体。

通过几次反例的学习，就完整地建立了一个"房子"的准确概念[图 7.32(b)]。

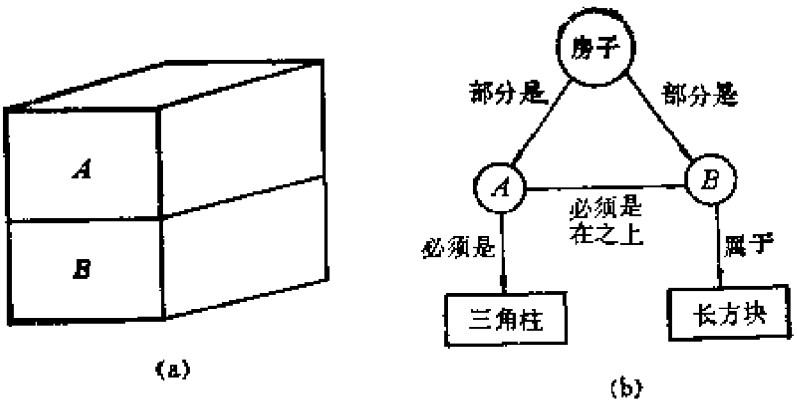


图 7.31 "房子"的反例和概念学习(二)

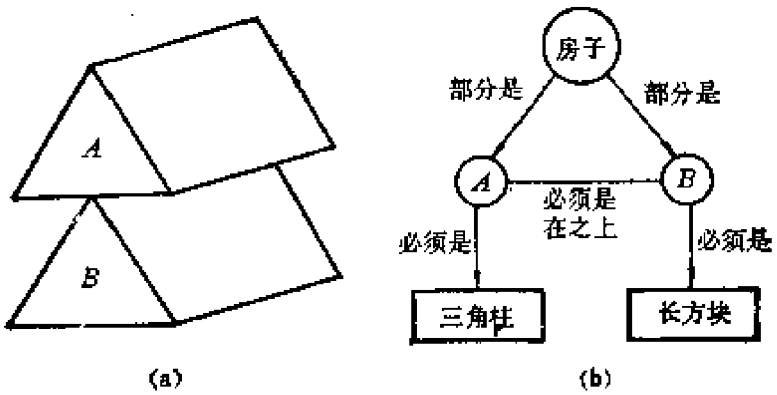


图 7.32 "房子"的反例和概念学习(三)

第八章　景物的图像表示

早在十七世纪人们就认识到，人的视觉是由眼和脑两部分组成。眼的作用是进行转换，它把视见的景物转换成视网膜上的图像，产生对视神经的刺激，然后由有关的神经和脑对视网膜上的图像进行传输、加工、分析和解释，产生人对景物的视见的感觉。因此，Martin D. Levine 把人的视觉系统看成是由生物光学转换装置(眼)和其后的"计算机"(脑)组成。脑实现对视网膜上的图像逐级进行加工、解释和理解，因此，被称为"机体计算机" (Meat Machine)，它的功能正是机器人视觉要实现的内容。十分遗憾，至今，人们对脑在这方面的工作过程知道得甚少，而且，对愈是低层的经验性的处理过程知道得愈少，甚至可以说一无所知。这正是用人工方法实现视觉功能的难点所在。

模仿人的视觉系统，机器人视觉是一个序贯的对外部世界的描述和转换的过程，最后给出对所观察到的景物的高度概括性的描述，从而使机器人具有环境理解的功能。因此，在研究机器人视觉前首先论述景物的初始描述——图像的形成过程，也就是图像和景物之间的关系。

§ 8.1　图像及其表示

一、图像函数

在数学上，一幅图像用一个图像函数表示。图像函数是定义

在 n 维空间的实函数 $f(x_1, x_2, \cdots, x_n)$。或者把自变量写成向量的形式，把图像函数表示成 $f(\boldsymbol{x})$。自变量的个数称为图像的维数，图像函数的值表示某种性质。例如一张照片是一幅二维图像，图像函数的值表示该点的灰度，因此，它的图像函数为

$$f(x, y) \geqslant 0 \qquad x, y, f(x, y) \in R$$

或者

$$f(\boldsymbol{x}) \geqslant 0 \qquad f(\boldsymbol{x}) \in R, \quad \boldsymbol{x} \in R^2$$

实际图像的大小总是有限的，因此，$f(\boldsymbol{x})$ 的定义域是有限域。但是，为了在数学处理时方便，规定：当 $\boldsymbol{x}$ 在原定义域外时，令 $f(\boldsymbol{x}) = 0$。因此，把 $f(\boldsymbol{x})$ 统一看成是定义在整个空间上的函数。

有的图像需要用多个图像函数表示，例如多光谱扫描图像和彩色图像就是这样的情况。这样的图像用如下函数矢量表示：

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})]^T$$

例如，一幅彩色图像表示成：

$$\boldsymbol{f}(\boldsymbol{x}) = [f_r(\boldsymbol{x}), f_g(\boldsymbol{x}), f_b(\boldsymbol{x})]^T$$

其中 $f_r(\boldsymbol{x})$，$f_g(\boldsymbol{x})$，$f_b(\boldsymbol{x})$ 分别是图像的红、绿、蓝三种颜色的灰度。

以上图像函数都是非时变的。如果景物或机器人处于运动过程中，机器人视觉系统采集的图像是时变的。从时变图像分析景物(或机器人)的运动，或推测景物的三维信息是机器人视觉的一个十分重要的研究方面，正引起计算机视觉科学工作者们越来越大的兴趣。时变图像的图像函数为 $\boldsymbol{f}(\boldsymbol{x}, t)$。与非时变的图像函数比较，它增加了一个时间变量 t。如果把变量 t 离散化(等时间间隔摄象)，时变图像是一个非时变图像序列。

二、图像的频域表示

一幅图像用一个空间函数(图像函数)表示。因此，分析图像

的空间变化特性的一种有效方法是把图像函数分解成正交函数的和．其中，最常使用的是对 $f(x)$ 进行傅里叶 (Fourier) 变换．

首先讨论一维连续图像的情况．这样作完全是为了论述的方便，并考虑到所得的结果很容易推广到多维的情况．

一维连续图像 $f(x)$ 的傅里叶变换为

$$F(u)=\int_{-\infty}^{\infty}f(x)e^{-j2\pi ux}dx \tag{8-1}$$

其中 $j=\sqrt{-1}$，参数 u 为空间频率．由于 $f(x)$ 的定义域和值都是有界的，所以 $f(x)$ 绝对可积，根据傅里叶变换的存在定理，式 (8-1) 的积分存在．

傅里叶反变换为

$$f(x)=\int_{-\infty}^{\infty}F(u)e^{j2\pi ux}du \tag{8-2}$$

在式(8-2)中 du 是频率增量，$F(u)du$ 是指数函数 $\exp(j2\pi ux)$ 的系数，因此，式 (8-2) 表明，傅里叶变换 (8-1) 是把 $f(x)$ 分解成指数函数．$F(u)$ 是 $f(x)$ 的频谱密度，简称为频谱．

比较式 (8-1) 和 (8-2)，立即可以看到，傅里叶正变换和反变换除参数的符号相反外，其数学形式完全相同，因此，傅里叶正变换所具有的性质傅里叶反变换亦具备，反之亦然．

由于图像函数由其傅里叶变换 $F(u)$ 唯一确定，因此，可以用图像函数的傅里叶变换 $F(u)$ 来表示图像．用 $F(u)$ 表示图像，使我们从空间域 (x) 转到频域 (u) 考察图像，这往往带来方便．例如，对图像进行滤波，设线性非时变滤波器的冲击响应是 $g(x)$，如果在空间域考虑以上滤波问题，滤波处理后的图像为

$$h(x)=\int_{-\infty}^{\infty}f(\tau)g(x-\tau)d\tau \tag{8-3}$$

以上积分称为求 $f(x)$ 和 $g(x)$ 的褶积，一般把它表示成

$$h(x)=f(x)*g(x) \tag{8-4}$$

只要进行简单的变量代换即可证明褶积具有如下性质：

$$f(x) * g(x) = g(x) * f(x) \tag{8-5}$$

$$H(u) = \mathscr{F}[f(x) * g(x)] = F(u)G(u) \tag{8-6}$$

其中 $\mathscr{F}[\cdot]$ 表示对“$\cdot$”进行傅里叶变换，$F(u)$，$G(u)$，$H(u)$ 分别是 $f(x)$，$g(x)$，$h(x)$ 的傅里叶变换。式(8-6)表明，在空间域求褶积在频域变成相乘。由于傅里叶正变换和傅里叶反变换数学形式相同(除参数符号相反外)，因此必定有：空间域的相乘运算在频域变成求褶积。

以上褶积运算和相乘运算的相互转换只是傅里叶变换的性质之一。傅里叶变换的所有基本性质如下表所示。

续表 8.1

空　间　域	频　域
(8) 空间域微分定理 $\frac{d^n f(x)}{dx^n}$	$(j2\pi u)^n F(u)$
(9) 频域微分定理 $(-j2\pi x)^n f(x)$	$\frac{d^n F(u)}{du^n}$
(10) 帕塞法尔 (Parseval) 定理[1] $\int_{-\infty}^{\infty} \|f(x)\|^2 dx = \int_{-\infty}^{\infty} \|F(u)\|^2 du$ $\int_{-\infty}^{\infty} f(x)\bar{g}(x)dx = \int_{-\infty}^{\infty} F(u)\bar{G}(u)du$	

(1) 这里叙述的是傅里叶变换的一般性质，$f(x)$，$g(x)$ 不受值域为实数域的限制.

三、离散图像

以上论述了用连续的图像函数表示图像. 但是，要用计算机来采集、存储图像，以及对图像进行加工都必须把图像函数 $f(x)$ 离散化. 离散化指的是对 $f(x)$ 的值进行量化和对 $f(x)$ 进行抽样. 量化的结果使 $f(x)$ 的值只能是整数，称为灰度级. 对于一维图像，抽样是指用序列 $f(n\Delta x)$ $(n = \cdots, -2, -1, 0, 1, 2, \cdots)$ 来代替连续图像函数 $f(x)$，如图 8.1 所示. 图中的 Δx 称为抽样间隔，序列 $f(n\Delta x)$ 称为离散图像.

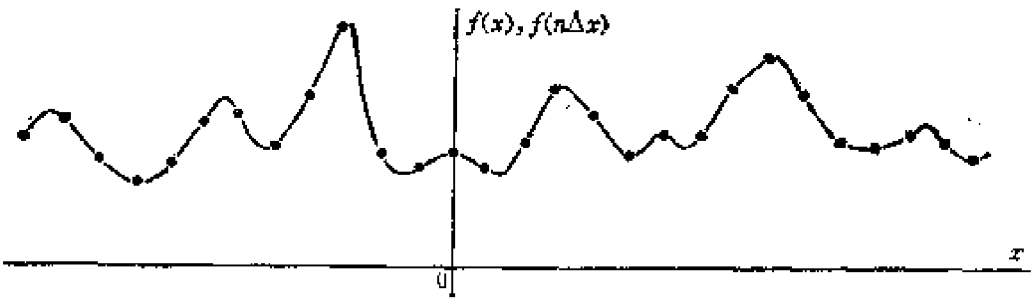


图 8.1　一维连续图像和离散图像

用离散图像代替连续图像，即是用图像的局部信息代替图像，自然存在这样的问题：这些局部信息能否反映整体呢？也就是，能否从 $f(n\Delta x)$ 唯一地重新恢复出 $f(x)$ 呢？如果对图像 $f(x)$ 不加任何限制，回答是否定的．只有当 $f(x)$ 满足一定的条件时，才能从离散图像唯一地恢复出原来的图像，这就是有名的抽样定理(Shannon)，这一定理奠定了用计算机处理信号的理论基础．由于其重要性，这里将对它作适当的说明．

为说明抽样定理，引入狄拉克函数 (Dirac) $\delta(x)$，并介绍它的一些基本性质．

定义函数

$$R_h(x)=\begin{cases}\dfrac{1}{h} & |x|\leqslant h\\ 0 & |x|>h\end{cases}$$

定义狄拉克函数为

$$\delta(x)\triangleq\lim_{h\to 0}R_h(x) \tag{8-7}$$

容易证明 $\delta(x)$ 具有如下性质：

$$1.\quad \delta(x)=\begin{cases}0 & x\neq 0\\ \infty & x=0\end{cases} \tag{8-8}$$

$$2.\quad \int_{-\infty}^{\infty}\delta(x)dx=1 \tag{8-9}$$

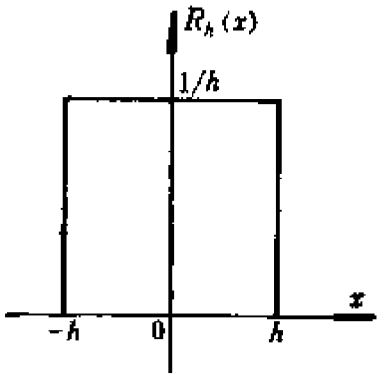


图 8.2　$R_h(x)$ 函数

3. 如果 $f(x_0)$ 存在

$$\int_{-\infty}^{\infty} f(x)\delta(x-x_0)dx = f(x_0) \tag{8-10}$$

4. $\delta(x)$ 的傅里叶变换为

$$\mathscr{F}[\delta(x)] = 1 \tag{8-11}$$

5. 由性质 4 可得

$$\int_{-\infty}^{\infty} e^{j2\pi ux}dx = \int_{-\infty}^{\infty} \cos(2\pi ux)dx = \delta(u) \tag{8-12}$$

6. $$\mathscr{F}[\cos 2\pi u_0 x] = \frac{1}{2}\int_{-\infty}^{\infty}(e^{j2\pi u_0 x} + e^{-j2\pi u_0 x})e^{-j2\pi ux}dx$$

$$= \frac{1}{2}(\delta(u-u_0) + \delta(u+u_0)) \tag{8-13}$$

7. $f(x) * \delta(x-x_0) = f(x-x_0)$

8. $C(x) = \sum_{m=-\infty}^{\infty} \delta(x - m\Delta x)$，称为梳状函数。

$$\mathscr{F}[C(x)] = \frac{1}{\Delta x}\sum_{m=-\infty}^{\infty} \delta(u - m/\Delta x) \tag{8-14}$$

证明：

将 $C(x)$ 展开成傅里叶级数：

$$c_m = \frac{1}{\Delta x}\int_{-\Delta x/2}^{\Delta x/2} C(x)e^{-j2\pi mx/\Delta x}dx$$

$$= \frac{1}{\Delta x}$$

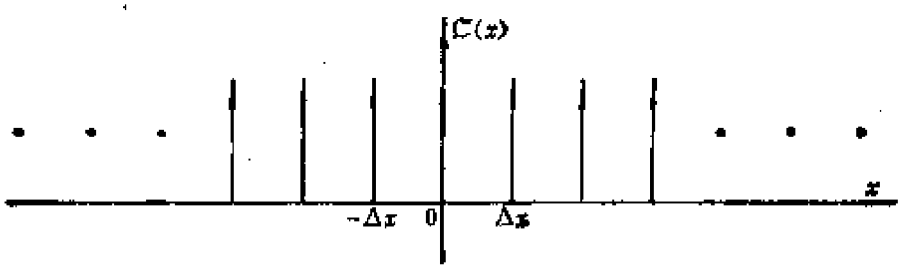


图 8.3 梳状函数

$$C(x) = \sum_{m=-\infty}^{\infty} c_m e^{j2\pi mx/\Delta x} = \frac{1}{\Delta x} \sum_{m=-\infty}^{\infty} e^{j2\pi mx/\Delta x}$$

因此，

$$\mathscr{F}[C(x)] = \frac{1}{\Delta x} \int_{-\infty}^{\infty} \left(\sum_{m=-\infty}^{\infty} e^{j2\pi mx/\Delta x} \right) e^{-j2\pi ux} dx$$

$$= \frac{1}{\Delta x} \sum_{m=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-j2\pi x(u-m/\Delta x)} dx$$

$$= \frac{1}{\Delta x} \sum_{m=-\infty}^{\infty} \delta(u - m/\Delta x)$$

现在，我们来说明抽样定理.

对于一维图像 $f(x)$，构造如下的抽样函数：

$$f_d(x) = f(x)C(x) = \sum_{m=-\infty}^{\infty} f(m\Delta x)\delta(x - m\Delta x) \quad (8\text{-}15)$$

即用 $f(x)$ 调制梳状函数 $C(x)$. 由式 (8-15) 可以看到，抽样函数 $f_d(x)$ 只不过是离散序列 $f(m\Delta x)$ 的另一种数学表示.

对 $f_d(x)$ 进行傅里叶变换，并注意到 $\delta(x)$ 的性质 7、8，得

$$\mathscr{F}[f_d(x)] = F_d(u) = F(u) * \frac{1}{\Delta x} \sum_{m=-\infty}^{\infty} \delta(u - m/\Delta x)$$

$$= u_0 \sum_{m=-\infty}^{\infty} F(u - mu_0) \quad (8\text{-}16)$$

$$u_0 = 1/\Delta x$$

u_0 称为抽样频率. 由式 (8-16) 看出，离散图像函数 $f_d(x)$ 的频谱只不过是将 $F(u)$ 重复位移 u_0 然后再叠加的结果. 即连续图像的频谱和抽样频率唯一地确定了离散图像的频谱. 然而，我们感兴趣的是其逆命题，也就是在什么条件下才能从离散图像的频谱 $F_d(u)$ 唯一地确定连续图像的频谱.

设图像函数满足

$$\mathscr{F}[f(x)] = F(u) = 0 \qquad |u| \geqslant u_c$$

则称这图像是有限带宽图像，u_c 称为截止频率。实际图像(不考虑其中的白噪声)都是有限带宽的。

抽样定理：

如果连续图像的傅里叶变换 $F(u)$ 和抽样间隔 Δx 满足

$$F(u) = 0 \qquad |u| \geqslant u_c \tag{8-17}$$

$$\Delta x \leqslant \frac{1}{2u_c} \tag{8-18}$$

则可由离散图像 $f(m\Delta x)$ 唯一地确定连续图像的频谱和连续图像函数 $f(x)$：

$$F(u) = \Delta x \sum_{m=-\infty}^{\infty} f(m\Delta x) e^{-j2\pi u m \Delta x} \tag{8-19}$$

$$u \in \left[-\frac{1}{2\Delta x}, \frac{1}{2\Delta x}\right]$$

$$f(x) = \sum_{m=-\infty}^{\infty} f(m\Delta x) \frac{\sin\left(\frac{\pi}{\Delta x}(x - m\Delta x)\right)}{\frac{\pi}{\Delta x}(x - m\Delta x)} \tag{8-20}$$

证明：

不失一般性，设 $f(x)$ 是有限带宽图像，它的幅频谱如图 8.4(a)所示。由式(8-16)有

$$\Delta x F_d(u) = \sum_{m=-\infty}^{\infty} F(u - m/\Delta x) \tag{8-21}$$

其中 $F(u)$ 和 $F_d(u)$ 分别是连续图像函数和离散图像函数的傅里叶变换，Δx 是抽样间隔。由于 $\Delta x \leqslant 1/2u_c$(即 $1/\Delta x \geqslant 2u_c$)，因此，将 $F(u)$ 位移 $1/\Delta x$ 再叠加不会产生频谱重叠。于是按式(8-21)，$\Delta x F_d(u)$ 是 $F(u)$ 彼此分离地周期性地重复的结果，如图 8.4(b)所示。

以上论述表明，如果满足条件式(8-17)、(8-18)，则有

$$F(u) = \Delta x F_d(u) \qquad u \in \left[-\frac{1}{2\Delta x}, \frac{1}{2\Delta x}\right]$$

并由式(8-15)知

$$F(u) = \Delta x \sum_{m=-\infty}^{\infty} f(m\Delta x) e^{-j2\pi u m \Delta x}$$

$$u \in \left[-\frac{1}{2\Delta x}, \frac{1}{2\Delta x}\right]$$

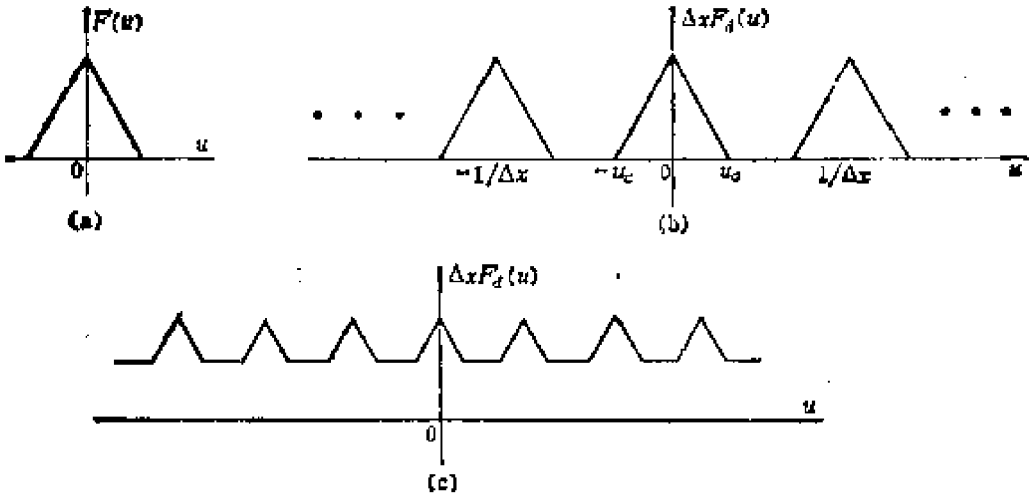


图8.4 (a) 连续图像的频谱，(b) $\Delta x < \frac{1}{2u_c}$ 情况下离散图像的频谱，(c) $\Delta x > \frac{1}{2u_c}$ 情况下离散图像的频谱

以下证明式(8-19)：

在区间 $[-1/2\Delta x, 1/2\Delta x]$ 上把 $F(u)$ 展开成傅里叶级数：

$$F(u) = \sum_{m=-\infty}^{\infty} C_m e^{j2\pi m \Delta x u}, \quad u \in \left[-\frac{1}{2\Delta x}, \frac{1}{2\Delta x}\right]$$

$$C_m = \Delta x \int_{-1/2\Delta x}^{1/2\Delta x} F(u) e^{-j2\pi m \Delta x u} du$$

$$= \Delta x \int_{-\infty}^{\infty} F(u) e^{j2\pi(-m\Delta x)u} du$$

$$= \Delta x f(-m\Delta x)$$

所以

$$F(u) = \sum_{m=-\infty}^{\infty} \Delta x f(-m\Delta x) e^{j2\pi m\Delta x u}$$

$$= \Delta x \sum_{m=-\infty}^{\infty} f(m\Delta x) e^{-j2\pi m\Delta x u}$$

$$u \in \left[-\frac{1}{2\Delta x}, \frac{1}{2\Delta x}\right]$$

式(8-19)得证。以下证明式(8-20):

对如上 $F(u)$ 进行傅里叶反变换:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi u x} du$$

$$= \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} \left(\Delta x \sum_{m=-\infty}^{\infty} f(m\Delta x) e^{-j2\pi m\Delta x u}\right) e^{j2\pi u x} du$$

$$= \Delta x \sum_{m=-\infty}^{\infty} f(m\Delta x) \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} e^{j2\pi u(x-m\Delta x)} du$$

$$= \sum_{m=-\infty}^{\infty} f(m\Delta x) \frac{\sin\left(\frac{\pi}{\Delta x}(x - m\Delta x)\right)}{\frac{\pi}{\Delta x}(x - m\Delta x)}$$

定理得证。

和连续图像一样,一维离散图像可以用它的傅里叶变换表示.式(8-19)已经给出了一维离散图像的傅里叶变换。一维离散图像的傅里叶变换对为

$$F_d(u) = \Delta x \sum_{m=-\infty}^{\infty} f(m\Delta x) e^{-j2\pi u m\Delta x} \tag{8-22}$$

$$f(m\Delta x) = \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} F_d(u) e^{j2\pi u m\Delta x} du \tag{8-23}$$

因为 $F_d(u)$ 和 $\exp(j2\pi un\Delta x)$ 都是周期为 $1/\Delta x$ 的周期函数，所以式 (8-23) 可以改写成

$$f(m\Delta x)=\int_0^{\frac{1}{\Delta x}} F_d(u)e^{j2\pi um\Delta x}du \tag{8-24}$$

注意到离散图像 $f(m\Delta x)$ 的频谱 $F_d(u)$ 仍然是连续(频域)函数. 为了用计算机实现图像的傅里叶变换，还必须在频域实行离散化，即对 $F_d(u)$ 进行抽样.

任何实际图像都是大小有限的有限带宽的图像. 一幅大小有限的一维图像用如下序列表示：

$$f(m\Delta x)=\begin{cases}0 & m<0\\ f(m\Delta x) & 0\leqslant m<M\\ 0 & m\geqslant M\end{cases} \tag{8-25}$$

即图象的宽度是 $M\Delta x$.

注意到傅里叶正变换和反变换的数学形式相同(除参数符号相反外)，空间域的抽样定理在频域仍然是适用的. 对于宽度为 $M\Delta x$ 的一维图象，频率抽样间隔应当满足

$$\Delta u\leqslant\frac{1}{M\Delta x}$$

因此，选择频率抽样间隔为

$$\Delta u=\frac{1}{M\Delta x} \tag{8-26}$$

由式 (8-22) 得有限离散频谱为

$$\begin{aligned}F_d(n\Delta u)&=\Delta x\sum_{m=0}^{M-1}f(m\Delta x)e^{-j2\pi n\Delta um\Delta x}\\&=\Delta x\sum_{m=0}^{M-1}f(m\Delta x)e^{-j2\pi nm\Delta x/M\Delta x}\\&=\Delta x\sum_{m=0}^{M-1}f(m\Delta x)e^{-jmn\frac{2\pi}{M}}\end{aligned} \tag{8-27}$$

把式 (8-24) 的积分表示成求和的形式：

$$f(m\Delta x) = \Delta u \sum_{n=0}^{M-1} F_d(n\Delta u) e^{j2\pi n\Delta u m\Delta x}$$

$$= \frac{1}{M\Delta x} \sum_{n=0}^{M-1} F_d(n\Delta u) e^{jmn\frac{2\pi}{M}} \tag{8-28}$$

$$m = 0, 1, 2, \cdots, M-1$$

不失一般性，令 $\Delta x = 1$，并把 $F_d(n\Delta u)$、$f(m\Delta x)$ 分别表示成 F_n 和 f_m，则由式 (8-27) 和 (8-28) 得离散傅里叶变换对为

$$F_n = \sum_{m=0}^{M-1} f_m e^{-jmn\frac{2\pi}{M}} \tag{8-29}$$

$$f_m = \frac{1}{M} \sum_{n=0}^{M-1} F_n e^{jmn\frac{2\pi}{M}} \tag{8-30}$$

$$m = 0, 1, \cdots, M-1$$

式 (8-29)、(8-30) 实现了一维有限图像的离散傅里叶变换，F_n 称为图像的有限离散谱。在证明抽样定理时我们已经看到，在空间域对有限带宽图像抽样，则在频域引起频谱周期性的延伸；周期为 M，只有当 $n \leqslant M-1$ 时才有 $F(n\Delta u) = F_d(n\Delta u)$。由于傅里叶正变换和反变换的数学形式相同，因此，在频域对有限大小图像的频谱抽样，反应在空间域是对图像周期性的延伸，周期为 M，因此，只有在 $m \leqslant M-1$ 时或 (8-24) 才成立。以后用 $\tilde{f}_m$（或 $\tilde{f}_{(m\Delta x)}$）表示由于频域抽样产生的周期性图像函数。

在用计算机完成式 (8-29)、(8-30) 所示的变换时必须使用快速算法，即快速傅里叶变换 (Fast Fourier Transformation) FFT，它使得运算次数大大减少（有关 FFT 的问题，可以参考数字信号处理或图像处理方面的书籍）。

前面论述了连续图像函数 $f(x)$和连续冲击响应 $g(x)$的褶积，以及空间域和频域间褶积运算和相乘运算间的转换。如果$f(x)$和

$g(x)$ 的频谱 $F(u)$ 和 $G(u)$ 都具有截止频率 u_c，那么，只要抽样间隔满足 $\Delta x \leqslant 1/2u_c$，按抽样定理，连续图像函数的褶积运算完全可以用离散图像的褶积运算来代替。

设 $f(m\Delta x)$ 和 $g(m\Delta x)$ 的连续谱分别是 $F_d(u)$ 和 $G_d(u)$，根据抽样定理，当 $u \in \left[-\frac{1}{2\Delta x}, \frac{1}{2\Delta x}\right]$ 时有

$$F(u) = F_d(u)$$
$$G(u) = G_d(u)$$

输出图像的连续谱为

$$\begin{aligned} H(u) &= G(u)F(u) \\ &= G_d(u)F_d(u) \quad u \in \left[-\frac{1}{2\Delta x}, \frac{1}{2\Delta x}\right] \end{aligned}$$

因此，输出离散图像为

$$\begin{aligned} h(m\Delta x) &= \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} G_d(u)F_d(u)e^{j2\pi um\Delta x}du \\ &= \int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} G_d(u)\left(\Delta x \sum_{\tau=-\infty}^{\infty} f(\tau\Delta x)e^{-j2\pi u\tau\Delta x}\right)e^{j2\pi um\Delta x}du \\ &= \Delta x \sum_{\tau=-\infty}^{\infty} f(\tau\Delta x)\int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}} G_d(u)e^{-j2\pi u(m-\tau)\Delta x}du \\ &= \Delta x \sum_{\tau=-\infty}^{\infty} f(\tau\Delta x)g((m-\tau)\Delta x) \end{aligned} \tag{8-31}$$

式 (8-31) 是求离散褶积的公式。与连续情况的褶积公式 $\int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau$ 比较，式 (8-31) 是显然的。

若图像和冲击响应都是大小有限的，即对于 f_m, g_m，有 $m=0, 1, \cdots, M-1$。设 f_m 和 g_m 的有限离散谱分别是 F_n 和 G_n，则输

出的离散频谱为

$$H_n = G_n F_n \tag{8-32}$$

对 H_n 进行傅里叶变换，输出图像为

$$\begin{aligned} h_m &= \frac{1}{M}\sum_{n=0}^{M-1} G_n F_n e^{jmn\frac{2\pi}{M}} \\ &= \frac{1}{M}\sum_{n=0}^{M-1} G_n\left(\sum_{\tau=0}^{M-1} f_\tau e^{-j\tau n\frac{2\pi}{M}}\right)e^{jmn\frac{2\pi}{M}} \\ &= \sum_{\tau=0}^{M-1} f_\tau \frac{1}{M}\sum_{n=0}^{M-1} G_n e^{j(m-\tau)n\frac{2\pi}{M}} \\ &= \sum_{\tau=0}^{M-1} f_\tau \tilde{g}_{m-\tau} \end{aligned}$$

即

$$h_m = \sum_{\tau=0}^{M-1} f_\tau \tilde{g}_{m-\tau} = \sum_{\tau=0}^{M-1} g_\tau \tilde{f}_{m-\tau} \tag{8-33}$$

上式中 $\tilde{f}_m$ 和 $\tilde{g}_m$ 是周期为M的离散函数，因此，称 h_m 为 f_m 和 $\tilde{g}_m$ 的循环褶积，并记作

$$h_m = f_m * g_m[M] = g_m * f_m[M] \tag{8-34}$$

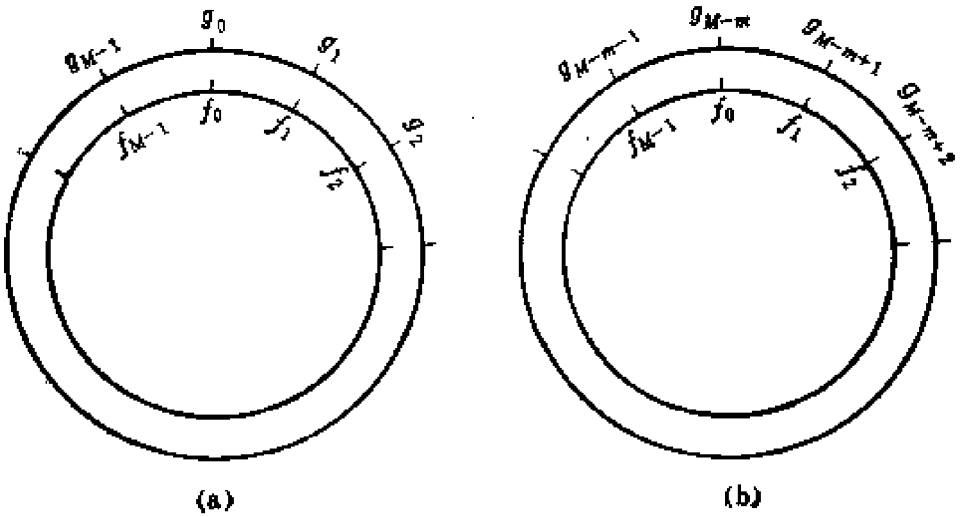


图 8 5　循环褶积

循环褶积可用图 8.5 形象地表示。图中同心圆的弧长代表自变量 m，并在相应的位置上标注出离散值，如图 8.5(a) 所示，褶积 h_m 是将外圆顺时针转 m 格，如图 8.5(b) 所示，再将两同心圆上对应的离散值相乘后求和。

四、二维图像

以上关于一维图像的结果很容易推广到多维情况。由于视觉是对二维图像进行分析和解释，因此，将只就二维情况进行讨论。

二维图像函数是定义在二维空间的实函数。如果函数的值表示所在点的灰度，则有

$$f(x, y) \geqslant 0, \quad x, y, f(x, y) \in R$$

二维图像的傅里叶变换为

$$F(u, v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)e^{-j2\pi(ux+vy)}dxdy \qquad (8\text{-}35)$$

$$f(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u, v)e^{j2\pi(ux+vy)}dudv \qquad (8\text{-}36)$$

其中 u、v 分别是和 x、y 对应的空间频率。$F(u, v)$ 称为二维图像 $f(x, y)$ 的频谱。

阵列 $f(m\Delta x, n\Delta y)$ 称为二维离散图像，Δx、Δy 称为抽样间隔，阵列的每一个点称为像素，一幅图像的像素总数称为分辨率。与一维情况一样，存在如下抽样定理。

二维抽样定理：

若二维图像 $f(x, y)$ 的频谱 $F(u, v)$ 且有截止频率 u_c, v_c，即

$$F(u, v) = 0 \qquad |u| \geqslant u_c, |v| \geqslant v_c$$

而且抽样间隔满足

$$\Delta x \leqslant \frac{1}{2u_c}$$

$$\Delta y \leqslant \frac{1}{2v_c}$$

则有

$$F(u,v)=\Delta x\Delta y\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}f(m\Delta x,n\Delta y)e^{-j2\pi(um\Delta x+vn\Delta y)}$$

$$|u|\leqslant\frac{1}{2\Delta x},\quad |v|\leqslant\frac{1}{2\Delta y} \tag{8-37}$$

而且，二维连续图像可由它的二维离散图像表示成

$$f(x,y)=\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}f(m\Delta x,n\Delta y)\frac{\sin\left(\frac{\pi}{\Delta x}(x-m\Delta x)\right)}{\frac{\pi}{\Delta x}(x-m\Delta x)}$$

$$\cdot\frac{\sin\left(\frac{\pi}{\Delta y}(y-n\Delta y)\right)}{\frac{\pi}{\Delta y}(y-n\Delta y)} \tag{8-38}$$

由以上二维傅里叶变换和二维抽样定理可以看到，多维情况是一维情况的逐维重复．有关结果的证明和一维情况的证明没有实质性的不同．

与一维情况类似，二维离散图像的傅里叶正变换和反变换分别是

$$F_d(u,v)=\Delta x\Delta y\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}f(m\Delta x,n\Delta y)e^{-j2\pi(um\Delta x+vn\Delta y)} \tag{8-39}$$

$$f(m\Delta x,n\Delta y)=\int_{-\frac{1}{2\Delta x}}^{\frac{1}{2\Delta x}}\int_{-\frac{1}{2\Delta y}}^{\frac{1}{2\Delta y}}F_d(u,v)e^{-j2\pi(um\Delta x+vn\Delta y)}dudv \tag{8-40}$$

其中 $F_d(u,v)$ 称为二维离散图像 $f(m\Delta x,n\Delta y)$ 的连续频谱．由二维抽样定理知，连续图像的频谱和离散图像的频谱间存在如下关系：

$$F(u,v)=F_d(u,v);|u|\leqslant\frac{1}{2\Delta x},\quad |v|\leqslant\frac{1}{2\Delta y} \tag{8-41}$$

如果把 $f(m\Delta x, n\Delta y)$ 表示成 $f(m, n)$，并令 $\Delta x=1$，$\Delta y=1$，对式 (8-39)、(8-40) 进行简单的变量代换，离散图像的傅里叶变换对可以表示成

$$F_d(u, v)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(m,n)e^{-j(mu+nv)} \tag{8-42}$$

$$f(m, n)=\frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} F_d(u,v)e^{j(mu+nv)}dudv \tag{8-43}$$

离散图像的褶积为

$$h(m, n)=f(m, n)*g(m, n)=g(m, n)*f(m, n)$$

$$=\sum_{\lambda=-\infty}^{\infty}\sum_{\tau=-\infty}^{\infty} g(\lambda, \tau)f(m-\lambda, n-\tau) \tag{8-44}$$

令 $H_d(u, v)$，$F_d(u, v)$，$G_d(u, v)$ 分别是 $h(m, n)$，$f(m,n)$，$g(m, n)$ 的连续谱，与一维情况一样，傅里叶变换和褶积运算间存在如下关系：

若　　$h(m, n)=f(m, n)*g(m, n)$

则　　$H_d(u, v)=F_d(u, v)G_d(u, v)$

如果 $f(m, n)$ 是有限离散图像 ($m=0,1,2,\cdots,M-1$; $n=0, 1, 2, \cdots, N-1$)，则对连续谱 $F(u, v)$ 进行抽样，并取抽样间隔为 $\Delta u=1/M\Delta x$，$\Delta v=1/N\Delta y$，得有限离散图像的傅里叶变换对为

$$\begin{cases} F_d(k, l)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m, n)e^{-j2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)} & (8\text{-}45) \\ \tilde{f}(m, n)=\frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F(k, l)e^{j2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)} & (8\text{-}46) \end{cases}$$

其中 $F_d(k, l)$ 是 $F_d(k\Delta u, l\Delta v)$ 的另一种表示形式。$\tilde{f}(m,n)$ 是对有限图像 $f(m, n)$ 周期性地延伸的结果。频率抽样间隔 Δu 和 Δv 是 $\tilde{f}(m, n)$ 的基频。

二维有限离散图像的循环褶积为

$$\tilde{h}(m, n) = f(m, n) * g(m, n)[M, N]$$
$$= g(m, n) * f(m, n)[M, N]$$
$$= \sum_{\lambda=0}^{M-1} \sum_{\tau=0}^{N-1} g(\lambda, \tau) f(m - \lambda, n - \tau) \quad (8\text{-}47)$$

而且，同样有

$$\mathscr{F}[f(m, n) * g(m, n)[M, N]] = F_d(k, l) G_d(k, l) \quad (8\text{-}48)$$

§8.2 颜色空间

一、三色原理

人们很早就注意到了虹的鲜艳色彩，在很多个世纪里，这一直是哲学家和科学家们感兴趣的事情。但是，关于说明光的波长和人的颜色感觉之间的理论最近三百年才建立起来，这应归功于牛顿和麦克斯韦。颜色是人眼对于不同波长的光刺激的一种主观反应。自然界中的物体对于各种波长的光有着各自的反射和吸收特性，因此呈现出不同的颜色，才使客观世界显得如此绚丽。颜色不仅使我们获得美的享受，同时，也是我们用来识别物体的重要特征，是我们用来生动表示景物的重要手段之一。因此，有必要就人对颜色的感觉，就如何表示颜色进行定量的研究。在这里将对色度学的基本知识，对主要的色坐标加以简要的说明。

可见光的波长从 380nm 一直到 780nm，人对颜色的感觉是不同波长的可见光刺激人的视觉器官的结果，在这一频段范围内，随波长减少，使人产生红、橙、黄、绿、青、蓝、紫的颜色感觉。

对人的视觉器官进行研究的结果表明，在人的视网膜上存在着两类感光细胞，称为柱状细胞和锥状细胞。柱状细胞灵敏度高，能感受很微弱的光；锥状细胞的灵敏度较低，但能很好地区分颜色。由于柱状细胞不能区分颜色，只能使人产生明暗的感觉，因此

在昏暗的环境里，所有的物体看起来都似乎是蓝灰色的。

锥状细胞是怎样区分颜色的呢？ 人们对此作过许多实验，提出了视觉三色原理的假说，这种假说能解释人的辨色原理，而且符合客观实际。但是，至今未能被解剖生理学所证实。

视觉三色假说假设存在三种锥状细胞，它们有不同的光谱灵敏度，其灵敏度的最大值分别在红、绿、蓝光的区域里，如图8.6所示。这三条光谱灵敏度曲线的和就是人眼的相对光谱灵敏度，或称光谱效率，或视见函数 $V(\lambda)$。 当光照射到人眼的视网膜上时，红、绿、蓝三种接收器分别产生各自频段内的光刺激，并在神经系统中把这三种刺激混合起来，使我们产生各种颜色的感觉。

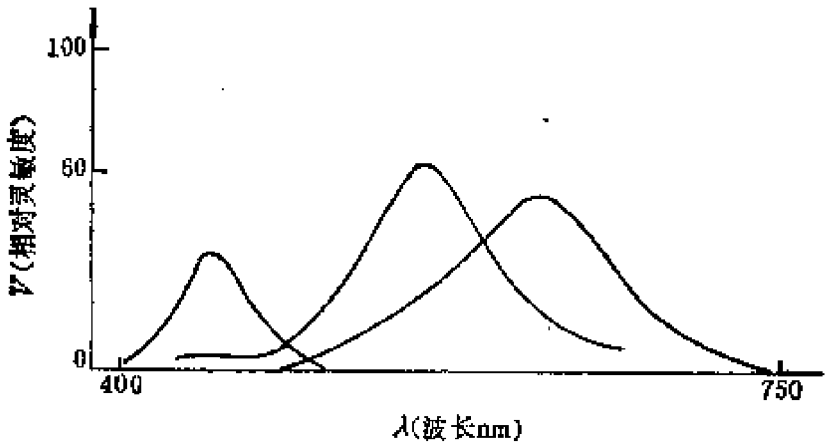


图 8.6　三种接收器的相对灵敏度

有趣的是，人对任何一种颜色的感觉，都可以用红、绿、蓝三种单色加权混合产生。因此，红、绿、蓝三种单色称为三基色。 以上三色原理可以用图 8.7 所示的实验证明。

图中 C 和 M 是两块白色的屏幕，待配色光投射到屏幕 C 上，M 上射入红 (R)、绿 (G)、蓝 (B) 三基色光，三种基色的通量可以用 c_r、c_g、c_b 调节。人在 V 处观察，调节 c_r、c_g、c_b 改变三基色光的混合比，直到两块屏幕上呈现出相同的颜色为止。对于某些颜色 (ϕ)，为了使两块屏幕上呈现相同的颜色，必须把三基色中的某

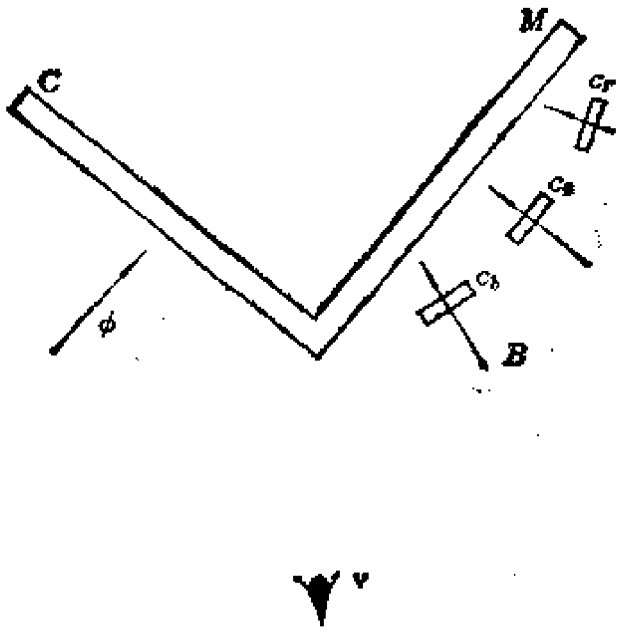


图 8.7　三色原理实验

一种(例如红色)投射到 C 上，然后调节 c_r、c_g、c_b 才能实现，这表示为了配出所需的颜色，在混色时这种基色(例如红色)的加权系数必须是负数.

根据三色原理，以下介绍国际照明委员会 (C. I. E) 规定的 RGB 表色系统和 XYZ 表色系统，以及彩色电视技术中使用的 YIQ 表色系统.

二、RGB 表色系统

根据三色原理，各种颜色的光都可以由红绿蓝三种基色光加权混合而成，因此，彩色空间是三维的线性空间，任何一种具有一定亮度的颜色光是空间中的一个点(或向量). 因此，我们可以选择具有确定光通量的红绿蓝三基色光作为这三维空间的基，这样组成的表色系统称为 RGB 表色系统.

按国际公认的 RGB 表色系统，三基色光的波长为

基　色　光	波　长 (nm)
R	700.0
G	546.1
B	435.8

在 RGB 表色系统中，标准白光是三基色的光通量 ϕ_r、ϕ_g、ϕ_b 按如下比例混合而成：

$$\phi_r:\phi_g:\phi_b = 1:4.5907:0.0601 \tag{8-49}$$

因此，一般把光通量为 1 流明的红光，4.5907 流明的绿光，0.0601 流明的蓝光作为三基色的“单位量”，简称为“基色量”，并分别用 (R)、(G)、(B) 表示．(R)、(G)、(B) 即是 RGB 表色系统的坐标基，任何一种具有一定亮度的彩色光都可以表示成

$$C = R(R) + G(G) + B(B) \tag{8-50}$$

彩色光 c 由矢量 $[R、G、B]^T$ 唯一的确定．c 的光通量为

$$\phi = R + 4.5907G + 0.0601B \tag{8-51}$$

ϕ 代表彩色光 c 的明亮程度．

显然，光的色度只决定于 R、G、B 间的比例关系．如果我们不考虑光的亮度，只对色度感兴趣，则只要知道 R、G、B 的相对值就可以了．因此令

$$\begin{cases} r = \dfrac{R}{R+G+B} \\ g = \dfrac{G}{R+G+B} \\ b = \dfrac{B}{R+G+B} \end{cases} \tag{8-52}$$

r、g、b 称为色度坐标．由于 $r + g + b = 1$，因此，只有两个色度坐标是独立的．这表明色度空间是二维的，我们可以选择 r、g 色度坐标给出 RGB 表色系统的色度图，如图 8.8 所示．图中舌形

曲线上的各点是波长从 380nm～780nm 的单谱色光，标准白光在 $r = g = 1/3$ 的位置。

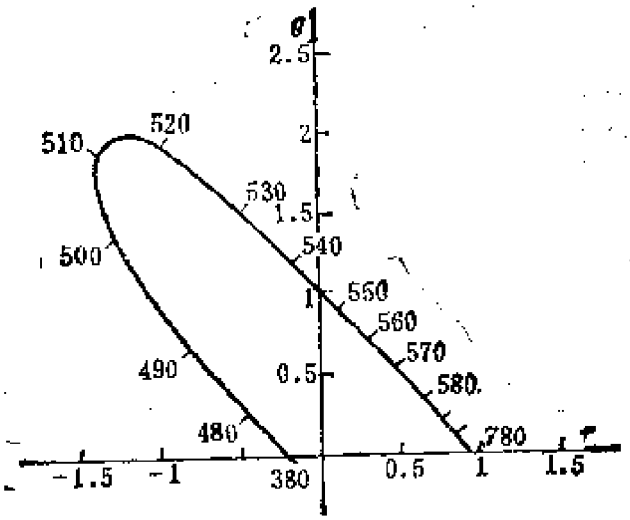


图 8.8 r-g 色度图

三、XYZ 表色系统

由图 8.8 可以看到，如果使用 RGB 表色系统，在混色时往往要求色系数为负数，这给计算带来不便。1931 年，国际上又规定了一种新的表色系统——XYZ 表色系统。这种表色系统是对 RGB 表色系统进行坐标变换产生的。RGB 表色系统和 XYZ 表色系统都是国际照明委员会 [C. I. E] 所采纳的，所以称为 CIE 表色系统，而把 XYZ 表色系统称为 CIE 标准表色系统。由于现在广泛使用的是 XYZ 表色系统，因此，常常把它简称为 CIE 表色系统。

XYZ 表色系统把彩色光表示为

$$C = X(X) + Y(Y) + Z(Z)$$

其中 (X)、(Y)、(Z) 是 XYZ 表色系统的坐标基——基色量。

XYZ 表色系统除了要满足混色时三色系数不为负的要求外，

还要满足另外两个条件，它们是：(1) Y 的数值正好是彩色的光通量；(2) 当 $X = Y = Z$ 时仍然代表标准白光。由这三个条件可以导出 XYZ 表色系统的三个基色量在 RGB 表色系统中的坐标为

$$\begin{bmatrix}(X)\\(Y)\\(Z)\end{bmatrix} = \begin{bmatrix}0.4185, & -0.0912, & 0.0009\\ -0.1587, & 0.2524, & -0.0025\\ -0.0828, & 0.0157, & 0.1786\end{bmatrix}\begin{bmatrix}(R)\\(G)\\(B)\end{bmatrix}$$

$$= T\begin{bmatrix}(R)\\(G)\\(B)\end{bmatrix} \tag{8-53}$$

由线性空间的基本知识可知，RGB 表色系统的三色系数和 XYZ 表色系统的三色系间的转换关系为

$$\begin{bmatrix}X\\Y\\Z\end{bmatrix} = T^{-1}\begin{bmatrix}R\\G\\B\end{bmatrix} = \begin{bmatrix}2.7689, & 1.7517, & 1.1302\\ 1.0000, & 4.5907, & 0.0601\\ 0.0000, & 0.0565, & 5.5943\end{bmatrix}\begin{bmatrix}R\\G\\B\end{bmatrix} \tag{8-54}$$

由上式可以看到，Y 的值正好是彩色光的光通量。

在 XYZ 表色系统中，引进色度坐标：

$$\begin{cases} x = \dfrac{X}{X+Y+Z} \\ y = \dfrac{Y}{X+Y+Z} \\ z = \dfrac{Z}{X+Y+Z} \end{cases} \tag{8-55}$$

$$x + y + z = 1$$

因此，用 x、y 为自变量得二维的色度图，如图 8.9 所示。任何一种彩色光，对人眼所引起的视觉作用可以用“亮度”、“色调”和“色饱和度”三个量来描述。这三个量称为彩色三要素。

“亮度”指的是明暗程度。“色调”指的是彩色光的颜色，即彩

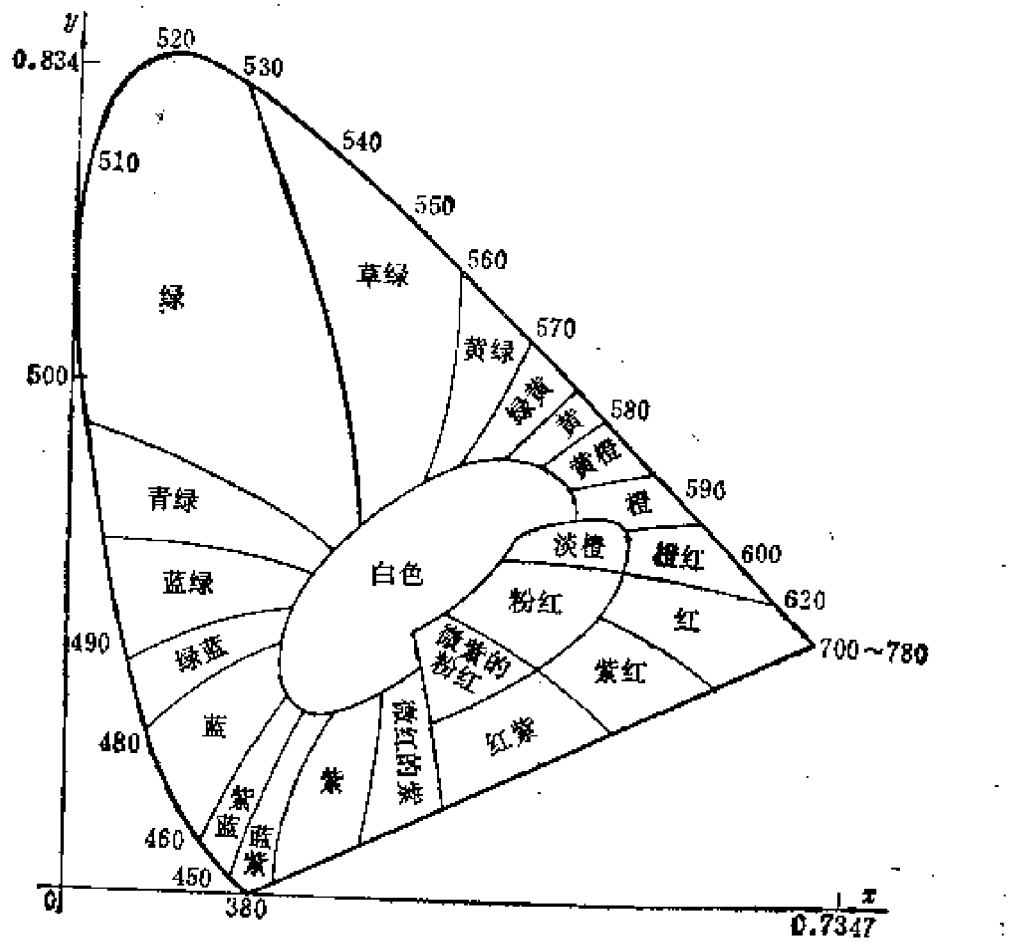


图 8.9　x-y 色度图

色光是由什么基色和白色混合成的。“色饱和度”是对颜色浓淡的度量，即颜色中白色的缺少程度。亮度用 Y 的值表示，色调和色饱和度用色度图表示。

在图 8.10 所示的 x-y 色度图中，舌形曲线上的点是波长从 380nm 到 780nm 之间的单谱色光。在舌形曲线所包围的面积中任意一点代表某种浓淡的颜色。如果两种颜色混色，例如用图中 A、B 两点所代表的颜色混色，则可配出线段 AB 上各点的颜色。例如 AB 线段上 D 点的颜色可由 A、B 两点混色得到，加权系数分别是 BD/AB 和 AD/AB。因此，联结舌形曲线两端点的直线段 FG 上的各点是波长为 780nm 的红色和波长为 380nm 的紫色的

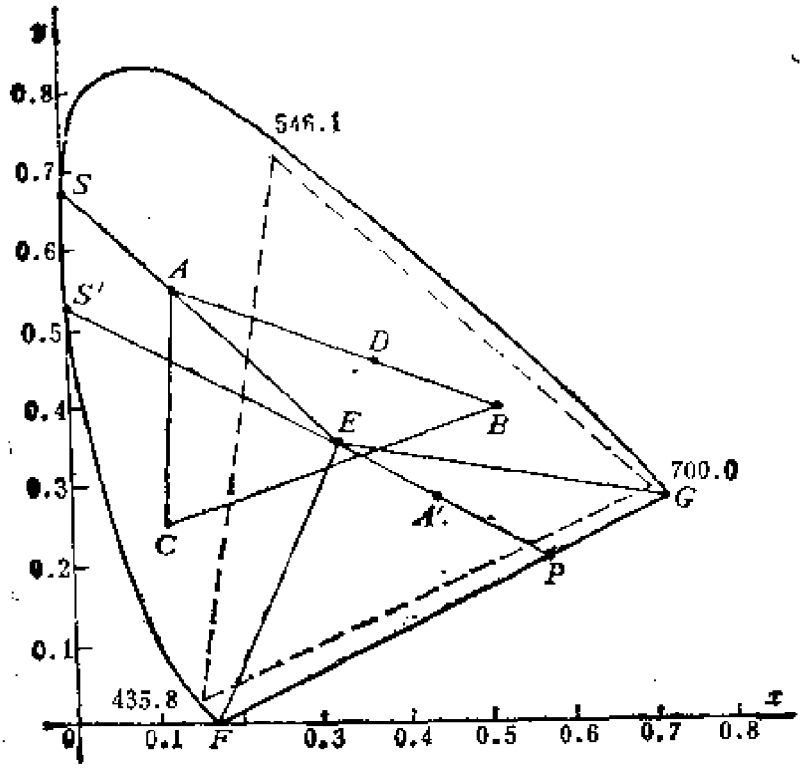


图 8.10　混色示意图

混合色．如果用三种颜色混色，例如用图 8.10 中 A、B、C 三点所代表的颜色混色，则可配制出 $\triangle ABC$ 中所有点的颜色．由波长分别是 700.0nm、546.1nm、435.8nm 的谱色组成的三角形称为三基色三角形，如图 8.10 中虚线组成的三角形所示．

在图 8.10 中，$x = y = 1/3$ 的点 E 代表标准白色．从 E 点出发经 A 点画射线，射线和舌形曲线交于 S 点，这表明 A 点的颜色是由 S 点的谱色和白色（E）混色产生的．称 S 点的谱色为 A 点的色调，或称为 A 点的主色，对应的波长称为 A 点的主色波长，它决定了 A 点的颜色种类．显然，A 点距 S 点的距离越小，A 点的颜色就越浓．因此，A 点的色饱和度定义为

$$S_A \triangleq \frac{EA}{ES} = \frac{x_A - x_E}{x_S - x_E} = \frac{y_A - y_E}{y_S - y_E} \tag{8-56}$$

其中 (x_A, y_A), (x_E, y_E), (x_S, y_S) 分别是 A、E、S 点的色坐标. 因此，色饱和度 S_A 是 A 点颜色中白色的缺少量的度量. 饱和度越大表示颜色越浓，越是接近于基色，因此越纯. 见图 8.10，对于三角形 $\triangle EFG$ 中的点 A'，我们同样可以定义色饱和度为

$$S_{A'} = \frac{EA'}{EP} = \frac{x'_A - x_E}{x_P - x_E} = \frac{y'_A - y_E}{y_P - y_E} \qquad (8\text{-}57)$$

由图 8.10 可以看到，A' 点的颜色和 S' 点的颜色按一定的比例混合可得标准白色，这说明 A' 和 S' 互为补色，S' 点的单色光波长 λ' 称为 A' 的主补色波长.

四、YIQ 表色系统

YIQ 表色系统首先由美国"国家电视系统委员会" (National Television Systems Committee) 所采用，以实现黑白电视和彩色电视的兼容，即可以用彩色电视机收看黑白电视节目.

为了得到足够的亮度和性能稳定性，并考虑到制造工艺上的困难，显象管的荧光粉不是按 RGB 标准基色光谱配制的. 标准的 NTSC 制式和 PAL 制式的 RGB 色度坐标如下：

NTSC 制式

R: $x = 0.670$, $y = 0.323$;

G: $x = 0.214$, $y = 0.710$;

B: $x = 0.140$, $y = 0.084$.

PAL 制式

R: $x = 0.640$, $y = 0.330$;

G: $x = 0.290$, $y = 0.600$;

B: $x = 0.150$, $y = 0.060$.

图 8.11 示出了以上两种制式的色度三角形. 在以下论述过程中所提到的 R、G、B 指的是以上两种制式实际使用的 R、G、B.

由于人眼对于颜色的相对视见度不同，所以选择三色的基色

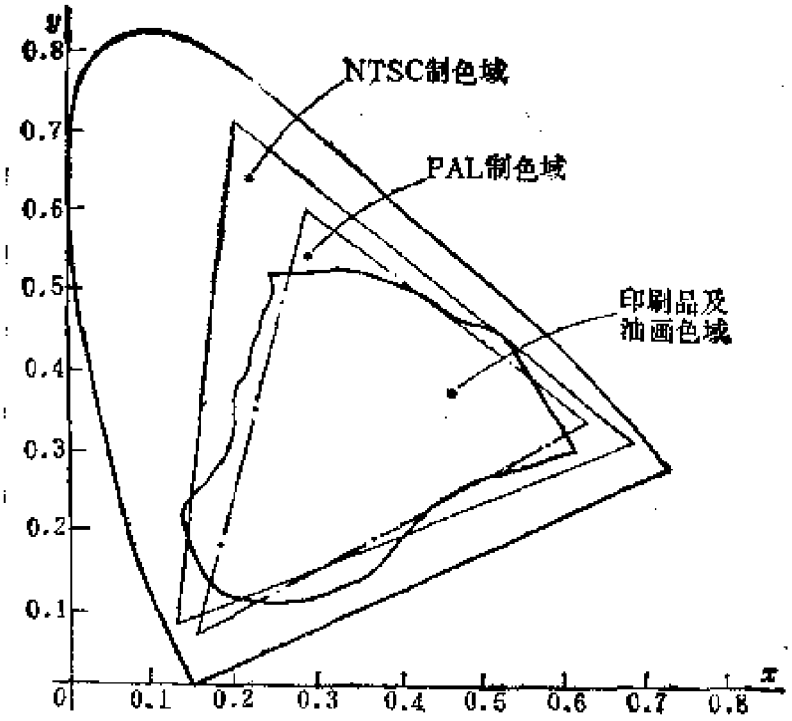


图 8.11　色度三角形

量为

$$(R) = 0.299,\quad (G) = 0.587,\quad (B) = 0.1143$$

因此，亮度信号 Y 为

$$Y = 0.30R + 0.59G + 0.11B \tag{8-58}$$

规定如下色差信号：

$$R - Y = 0.70R - 0.59G - 0.11B;$$

$$G - Y = -0.30R + 0.41G - 0.11B;$$

$$B - Y = -0.30R - 0.59G + 0.89B.$$

色差信号是色度的另外一种表示形式。以上三个色差信号是线性相关的，因为

$$Y - Y = 0.30R + 0.59G + 0.11B - 0.30Y$$
$$- 0.59Y - 0.11Y$$

$$= 0.30(R - Y) + 0.59(G - Y) + 0.11(B - Y)$$
$$= 0$$

所以

$$G - Y = -\frac{0.30}{0.59}(R - Y) - \frac{0.11}{0.59}(B - Y) \qquad (8\text{-}59)$$

因此，彩色电视所传送的三个信号是

$$Y = 0.30R + 0.59G + 0.11B; \qquad (8\text{-}60)$$

$$R - Y = 0.7R - 0.59G - 0.11B; \qquad (8\text{-}61)$$

$$B - Y = -0.30R - 0.59G + 0.89B. \qquad (8\text{-}62)$$

实际上，为了进一步压缩色度信号的带宽，彩色电视实际传送的是 Y、I、Q 三个信号，I、Q 和色差信号间的关系是

$$I = 0.74(R - Y) - 0.27(B - Y) \qquad (8\text{-}63)$$

$$Q = 0.48(R - Y) + 0.41(B - Y) \qquad (8\text{-}64)$$

把式(8-61)和(8-62)代入以上两式，得 Y、I、Q 和 R、G、B 间的转换关系为

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.30, & 0.59, & 0.11 \\ 0.60, & -0.28, & -0.32 \\ 0.21, & -0.53, & 0.31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (8\text{-}65)$$

§8.3 摄像机的图像生成模型

前面两节论述了有关图像的一些基本问题. 应当指出，机器人视觉与一般的图像处理有本质的区别. 图像处理指的是从图像到图像的转换，例如图像滤波、图像增强等等，它不涉及图像的含义，不涉及对图像的理解. 机器人视觉则是研究怎样用人工的方法，实现对外部世界的描述和理解. 它的输入是外部景物，输出是对所观察到的景物的高度概括性的描述——理解. 从摄像机获得的图像只是对景物的初始表示.

一、摄像机的几何模型

摄像机作为机器人的眼睛，实现从景物到图像的转换，实现从三维景物空间到二维图像空间的转换．人们首先感兴趣的是转换的几何关系，也就是三维空间中的点与它在二维空间中的像间的对应关系．显然，这种关系是我们从图像(二维)推断景物的空间信息(三维)的基本出发点．

任何摄像机，无论它配备什么样的镜头，都可以抽象为图 8.12 所示的几何模型．

图中 (x_c, y_c, z_c) 坐标系称为视觉坐标系，它固结在摄像机上，"O_c" 点为摄像机的镜头中心，z 轴和摄像机的光轴重合而且指向摄像机，f 是镜头的焦距．图中 X-Y 平面称为图像平面．设在三维空间中有一点 V，它的坐标为 (x_c, y_c, z_c)，$-z_c$ 称为 V 点的深度，连接 V 点和镜头中心点 O_c 的线段 VO_c 称为投影线，投影线 VO_c 和图像平面的交点 (X, Y) 就是 V 点的投影，或者称为 V 点的像．这样的投影关系称为中心投影或透射投影 (Perspective Projection)．当焦距 f 无限大时，以上投影转变成正投影

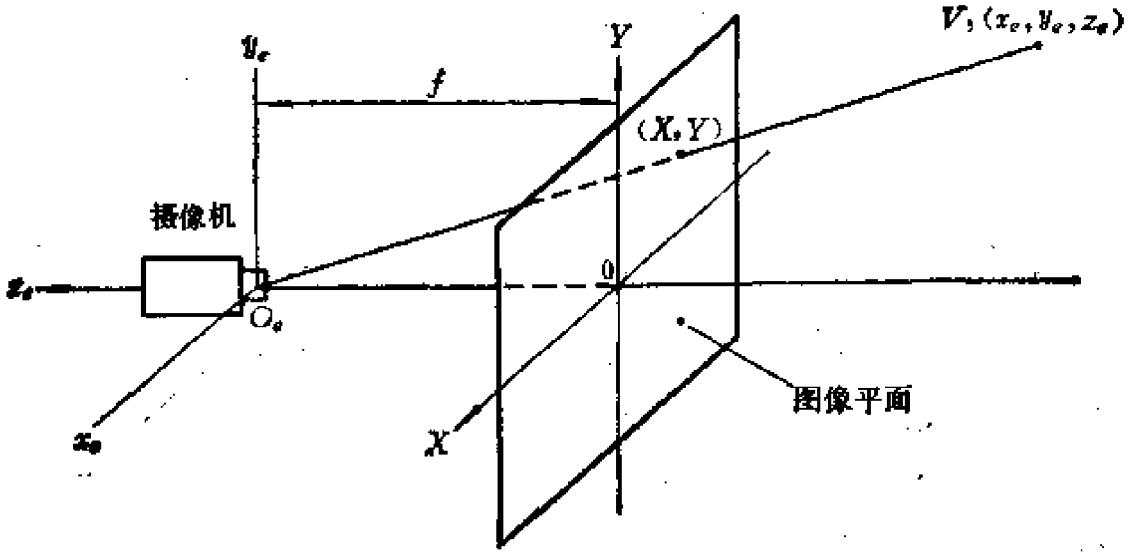


图 8.12　摄像机的几何模型

(Orthographic Projection).

由相似三角形不难得出三维空间的 V 点和它的像之间的关系为

$$\begin{cases} X = \dfrac{f}{-z_c} x_c \\ Y = \dfrac{f}{-z_c} y_c \end{cases} \tag{8-66}$$

其中，$-z_c$ 称为深度，$-f/z_c$ 称为缩小比 (Minification Ratio)。

或者，将上式改写为

$$\begin{cases} x_c = \dfrac{-z_c}{f} X \\ y_c = \dfrac{-z_c}{f} Y \\ z_c = z_c \end{cases} \tag{8-67}$$

其中 $-z_c/f$ 称为放大比(Magnification Ratio)。如果把式(8-67)中的 z_c 看成是参变量，式(8-67)实际上就是投影线的参数方程。因此图像平面上的每一个点和一条空间投影线对应，这条投影线的方向余弦为

$$N = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = \frac{1}{\sqrt{f^2 + X^2 + Y^2}} \begin{bmatrix} X \\ Y \\ -f \end{bmatrix} \tag{8-68}$$

因此，透射投影只能保证图像点与投影线间的一一对应，一条投影线上的所有点对应同一个图像点，这表明在投影过程中丢失了空间点的深度信息。这是实现机器人视觉的主要难点之一。现在，关于计算机视觉的研究工作，相当一部分是在围绕着怎样从所摄取的二维图像恢复三维信息的问题进行，在这方面有了不少进展，但是，至今仍然没有令人满意的方法。

由式(8-66)可以看到，投影变换是非线性的。为了把以上变

换线性化，使得考虑问题和计算方便，我们引进齐次坐标系（Homogenous Coordinate System）。它实质上是对笛卡尔坐标系的维数扩充。设有三维空间点 $\boldsymbol{V}=[x_c, y_c, z_c]^T$，它的齐次坐标定义为

$$\tilde{\boldsymbol{V}}=[wx_c,\ wy_c,\ wz_c,\ w]^T$$

相反，设某点的齐次坐标为 $\tilde{\boldsymbol{V}}=[x'_c, y'_c, z'_c, w]^T$，它对应的笛卡尔坐标为

$$\boldsymbol{V}=\left[\frac{x'_c}{w},\ \frac{y'_c}{w},\ \frac{z'_c}{w}\right]^T$$

在齐次坐标系中，投影变换可表示成如下的线性形式：

$$\tilde{\boldsymbol{V}}_P=\begin{bmatrix}WX\\WY\\WZ\\W\end{bmatrix}=P\tilde{\boldsymbol{V}}$$

$$=\begin{bmatrix}1, & 0, & 0, & 0\\0, & 1, & 0, & 0\\0, & 0, & 1, & f\\0, & 0, & -\frac{1}{f} & 0\end{bmatrix}\begin{bmatrix}x_c\\y_c\\z_c\\1\end{bmatrix}=\begin{bmatrix}x_c\\y_c\\z_c+f\\-\frac{z_c}{f}\end{bmatrix}\qquad(8\text{-}69)$$

其中 P 称为投影变换矩阵。由式(8-69)得对应的笛卡尔坐标为

$$\boldsymbol{V}_P=\left[\frac{f}{-z_c}x_c,\ \ \frac{f}{-z_c}y_c\ \ \ -f-\frac{f^2}{z_c}\right]^T\qquad(8\text{-}70)$$

$\boldsymbol{V}_P$的第一、二个分量正好是 $\boldsymbol{V}$ 点的像。

由齐次坐标表示的逆投影变换为

$$\tilde{\boldsymbol{V}}=P^{-1}\tilde{\boldsymbol{V}}_P\qquad(8\text{-}71)$$

其中

$$P^{-1}=\begin{bmatrix}1, & 0, & 0, & 0\\ 0, & 1, & 0, & 0\\ 0, & 0, & 0, & -f\\ 0, & 0, & \frac{1}{f}, & 1\end{bmatrix} \tag{8-72}$$

将这结果代入式(8-71)得

$$\dot{\mathbf{V}}=\begin{bmatrix}1, & 0, & 0, & 0\\ 0, & 1, & 0, & 0\\ 0, & 0, & 0, & -f\\ 0, & 0, & \frac{1}{f}, & 1\end{bmatrix}\begin{bmatrix}X\\ Y\\ Z\\ 1\end{bmatrix}=\begin{bmatrix}X\\ Y\\ -f\\ (f+Z)/f\end{bmatrix}$$

所以

$$\begin{cases}x_c=\dfrac{f}{f+Z}X\\ y_c=\dfrac{f}{f+Z}Y\\ z_c=\dfrac{-f^2}{f+Z}\end{cases} \tag{8-73}$$

消去上式中的 Z，解得

$$\begin{cases}x_c=\dfrac{-z_c}{f}X\\ y_c=\dfrac{-z_c}{f}Y\\ z_c=z_c\end{cases} \tag{8-74}$$

把 z_c 看成是自由参数，式(8-74)是投影线的方程．因此逆投影变换 P^{-1} 实现从图像点到投影线的变换．

以上在论述摄像机的几何模型时，我们选择了视觉坐标系 (x_c, y_c, z_c) 作为三维空间的参考坐标系．实际上，机器人和它的

眼睛(摄像机)具有各自的固有坐标系，如图 8.13 所示。摄像机作为机器人的眼睛它提供给机器人的关于景物的信息必须以机器人坐标系 (x_a, y_a, z_a) 为参考坐标系，只有这样，手眼才能协调地配合工作。

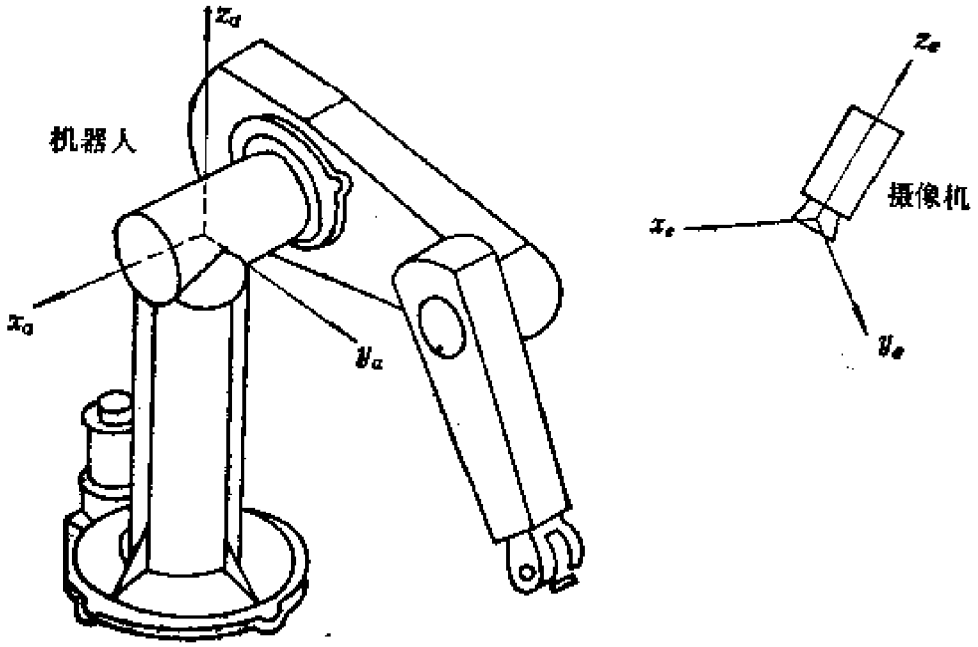


图 8.13　视觉坐标系和机器人坐标系

设由机器人坐标系到视觉坐标系间的齐次变换为

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}, & a_{12}, & a_{13}, & -x_0 \\ a_{21}, & a_{22}, & a_{23}, & -y_0 \\ a_{31}, & a_{32}, & a_{33}, & -z_0 \\ 0, & 0, & 0, & 1 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ z_a \\ 1 \end{bmatrix} = T\tilde{V}' \qquad (8\text{-}75)$$

其中 $a_{11}, a_{12}, \cdots, a_{33}$ 代表旋转变换；x_0, y_0, z_0 代表旋转后的平移量。将式 (8-75) 代入式 (8-69)

$$\tilde{V}_P = PT\tilde{V}' \qquad (8\text{-}76)$$

经过简单的计算得

$$X = -f\frac{a_{11}x_a + a_{12}y_a + a_{13}z_a - x_0}{a_{31}x_a + a_{32}y_a + a_{33}z_a - z_0} \tag{8-77}$$

$$Y = -f\frac{a_{21}x_a + a_{22}y_a + a_{23}z_a - y_0}{a_{31}x_a + a_{32}y_a + a_{33}z_a - z_0} \tag{8-78}$$

只要摄像机镜头的中心点不在机器人坐标系的 x_a-y_a 平面内，必定有 $z_0 \neq 0$。并考虑到焦距 f 是常数，因此，投影变换关系式(8-67)、(8-68) 可改写成

$$X = \frac{A_{11}x_a + A_{12}y_a + A_{13}z_a + A_{14}}{A_{31}x_a + A_{32}y_a + A_{33}z_a + 1} \tag{8-79}$$

$$Y = \frac{A_{21}x_a + A_{22}y_a + A_{23}z_a + A_{24}}{A_{31}x_a + A_{32}y_a + A_{33}z_a + 1} \tag{8-80}$$

其中 (x_a, y_a, z_a) 是空间点 V 在机器人坐标系中的坐标，(X, Y) 是 V 点在摄像机图像平面中的投影。$A_{11}, A_{12}, \cdots, A_{33}$ 是11个投影参数，它决定于摄像本身的光学系统，以及摄像机相对于机器人的安装情况。确定这11个参数的过程称为系统校准。

二、摄像机的光度学模型

上面论述了摄像机的图像和景物间的几何关系。现在来讨论图像的灰度和景物间的关系。

首先介绍几个基本的光学概念和术语。

(1) 光能量通量 ϕ：单位时间内照射在某面积上的光能量。

(2) 光源的发光强度 I：光源的发光强度 I 指的是光源发出的通过单位立体角内的光能量通量：

$$I = \frac{d\phi}{d\omega} \tag{8-81}$$

式(8-81)中 $d\omega$ 是立体角的微分，整个球面所张的立体角为 4π 立体弧度。

(3) 表面的照度 E：表面照度 E 指的是照射在单位面积上的

光能量通量：

$$E = \frac{d\phi}{dA} \tag{8-82}$$

(4) 表面发光强度 L：表面的发光强度 L 指的是在规定方向上的单位面积在单位立体角内的光能量通量。

$$L = \frac{d^2\phi}{\cos\theta dA d\omega} \tag{8-83}$$

如图 8.14 所示，图中 n 是 dA 的法线向量，v 是观察方向。

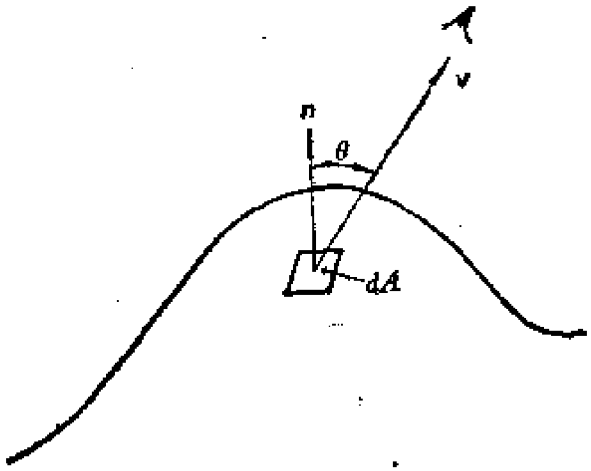


图 8.14　表面在 v 方向上的发光强度

为了说明摄像机的图像灰度与景物的关系，让我们来分析图 8.15 所示的成像系统。假设镜头的焦距已经调好，在图像平面上形成了清晰的图像。

设在物体的表面上有一块微小的面积 dA_0，它的发光强度是 L，它在图像平面上的像是 dA_F。参看图 8.15，由式 (8-83) 得 dA_0 在物镜所张的立体角内的光能通量为

$$d\phi = dA_0 \int L\cos\theta d\omega \tag{8-84}$$

由于 dA_F 是 dA_0 的像，所以式 (8-84) 所示的光能量通量 $d\phi$ 全

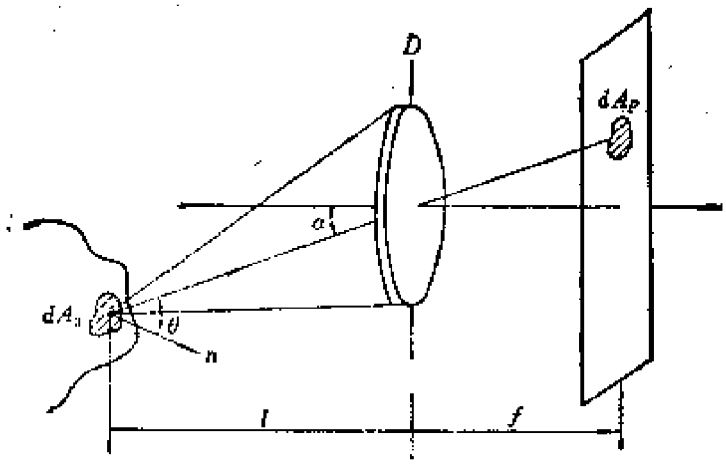


图 8.15　图像生成示意图

部投射到了 dA_P 上．于是，dA_P 的照度为

$$E_p = \frac{d\phi}{dA_P} = \frac{dA_0}{dA_P}\int L\cos\theta d\omega \tag{8-85}$$

见图 8.15，由简单的几何分析可知，dA_0 和 dA_P 间的关系为

$$dA_0\frac{\cos\theta}{l^2} = dA_P\frac{\cos\alpha}{f^2} \tag{8-86}$$

将式 (8-86) 代入式 (8-85) 得

$$E_p = \left(\frac{l}{f}\right)^2\cos\alpha\int L d\omega \tag{8-87}$$

设 L 是常数，这样我们就可以把 L 移到积分号外．而积分 $\int d\omega$ 则是以 dA_0 为中心镜头所张的立体角，这立体角的大小为

$$\int d\omega = \frac{\pi D^2}{4l^2}\cos^3\alpha \tag{8-88}$$

其中 D 是镜头的直径，l 是 dA_0 和镜头间的垂直距离，α 是偏转角度．

把式 (8-88) 代入式 (8-87) 得

$$E_p = \frac{\pi L}{4}\left(\frac{D}{f}\right)^2 \cos^4\alpha \qquad (8\text{-}89)$$

由上式所得到一些很有趣的结果。

(1) 图像平面的照射强度(图像的灰度)和偏角 α 的余弦的四次方成正比。为补偿由此而造成的图像灰度的非线性失真,因此,应对图像生成装置进行校准,使得其灵敏度和偏转角度 α 无关。

(2) 图像平面的照度 E_p 和景物的表面发光强度 L 成正比,物体的表面发光强度决定于照明的性质、物体表面的反射特性,以及观察的方位。这是一个十分复杂的问题,下面将对这问题作扼要的说明,这里只是简单地指出:照明越强摄像机输出的信号也越强。

(3) 图像平面的照度 E_p 和景物的远近 l 无关。只要物体的表面足够大,使得它在图像平面上的象足以覆盖一个像素,那么,这个图像的灰度值与它离开镜头的远近无关。这是由于物体的表面发光强度和距离的平方成反比,另一方面,一个像素对应的物体的表面积却与距离的平方成正比,两种作用相互抵消产生了这样的结果。正是由于这个原因,才使得我们能看到十分遥远的物体,使得我们看到月亮仍然是那样明亮。

(4) 最后,我们由式(8-89)看到,图像平面的照度 E_p 和镜头的直径焦比 (D/f) 的平方成正比,因此,当把摄像机的镜头从较短焦距的镜头换成较长焦距的镜头时(若镜头的直径 D 不变),摄像机的灵敏度将下降。

下面我们来讨论物体的反射特性和照明对图像的影响。如图 8.16 所示,dA 是一元表面,n 是 dA 的单位法向向量,v 是观察方向的单位向量,s 是光源方向的单位向量,i 为光线的入射角(n 和 s 的夹角),e 是观察角(n 和 v 间的夹角),g 是相角(v 和 s 间的夹角)。对于元表面 dA,照明效果和物体表面的反射特性表现为:dA 的发光强度是表面的方位、观察方向和照射方向的函数 $L(i, e, g)$。这个函数随物体表面性质不同而异。

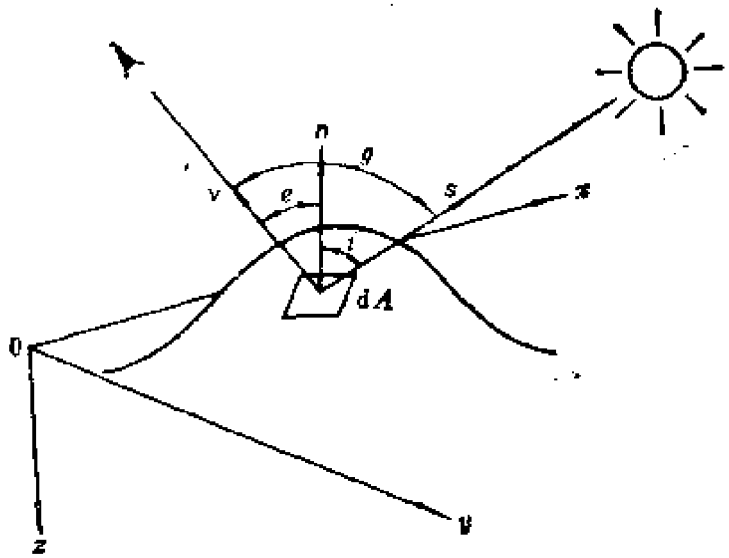


图 8.16　物体表面，入射角，观察角和相角的表示

朗伯表（Lambertian Surface）是一种具有简单反射特性的表面类型，常在一些文章和书籍中提到。这种表面的粗糙程度恰到好处，使得发光强度 L 和视角的余弦（$\cos e$）成正比。朗伯平面表现出的现象是，无论以什么样的角度对它进行观察，它都显得同样的明亮。这是因为表面的发光强度 L 和 $\cos e$ 成正比，而所观察到的表面的面积（例如与一个像素对应）则和观察角度的余弦成反比，因此产生了这样的总的观察效果。

另外还存在两种极端的情况，一种称为灰面（Dusty Surface），它的特点是表面的发光强度均匀分布，即在任何观察方向上表面的发光强度相同。月球的表面就是这种情况的一个例子，这说明为什么在满月时尽管月球表面的照度是不均匀的（与太阳光的入射角的余弦成正比），但是我们看到的却是一轮同样明亮的银盘（除环形山的阴影部分外）。

另外一种极端的情况是镜面（Specular Surface），这是一种人们十分熟悉的情况。这种表面的特点是反射光只在反射角的方

向上存在.

实际表面一般介于这两种极端的情况之间. 绝大多数表面都程度不同地存在着镜面效应. 对于实际的镜面效应，可以用观察向量 v 和反射角向量间夹角的余弦近似描述. 在空间定义一坐标系 (x, y, z)，不失一般性，设表面的单位法向向量为

$$n = [0,\ 0,\ 1]^T$$

设光源方向向量 s 在 y-z 平面内:

$$s = [0,\ \sin i,\ \cos i]^T \tag{8-90}$$

因此，镜面反射方向的单位向量为

$$r = [0,\ -\sin i,\ \cos i]^T \tag{8-91}$$

观察方向单位向量的一般形式为

$$v = [\cos\theta \sin e,\ \sin\theta \sin e,\ \cos e]^T \tag{8-92}$$

观察向量和反射向量间夹角 (α) 的余弦为

$$\cos\alpha = v \cdot r = -\sin i \sin\theta \sin e + \cos i \cos e \tag{8-93}$$

相角 g 的余弦为

$$\cos g = v \cdot s = \sin i \sin\theta \sin e + \cos i \cos e \tag{8-94}$$

所以

$$\cos\alpha + \cos g = 2\cos i \cos e$$

令

$$C = \cos\alpha = 2\cos i \cos e - \cos g \tag{8-95}$$

由此，把表面的发光强度函数表示成

$$L(i, e, g) = KC^l + (1 - K)\cos i \tag{8-96}$$

$$K \in [0, 1]$$

上式的第一、二项分别表示表面的镜面和灰面的效应，参数 K 是组合效应的调节参数，参数 l 是镜面反射分布尖锐度的调节参数.

应当指出，式 (8-96) 只是物体表面发光强度的近似描述，而且，所反映的仍然是简单情况. 它只适合于单个的点光源或单个的平行光源的情况，实际情况要复杂得多，即便原始光源是一个平行

光源这种最简单的情况，景物的多次反射形成的照明仍然是极其复杂的，很难作定量的描述。

图像的灰度是摄像机的光学特性、物体表面的反射特性、照明情况、景物中各物体的分布情况(产生重复反射照明)的综合结果，仅仅从摄得的图像分解出以上各种因素在这过程中所起的作用是不可能的，这是导致机器人视觉难以实现的另一个主要原因。

§8.4 机器人视觉设备

机器人视觉设备由照明设备、图像采集设备、主机和快速处理部件组成。

由于任务不同，机器人的工作环境千差万别。最一般的情况(也是要求最高的情况)是，机器人的照明情况完全是不可控的，例如室外自然景物就是这样的情况。但是，对于许多在工业中有专门用途的机器人，往往可以根据任务要求设计机器人视觉的照明。例如，我们可以根据不同的情况选择点光源照明、平行光照明、激光束照明、可移动的光点照明、片状光源或其它形式的结构光源的照明，等等。初学者往往对照明重视得不够。应当认识到，正确地选择照明方式往往决定了机器人视觉的实现方案，或者使问题大大简化，甚至对完成任务的成败起着关键性的作用。但是，照明随不同的任务，随选用的视觉方案而变，在这里不便专门就照明问题进行一般性的论述。

图像采集部分原则上由图8.17的几个部分组成。其中摄象机起转换器的作用，它把观察到的景物转换成全电视信号(模拟信号)。数字化仪用全电视信号中的同步信号实现和摄像机同步工作，并把摄像机输出的模拟图像信号逐行进行等时间间隔采样，而且把每个采样点的灰度转换成灰度级，顺序地存入缓冲寄存器。这样在缓冲寄存器中存储的是一幅(或多幅)数字化的图像，每一个

采样点称为一个像素，一幅图像的像素的数目称为这图像采集设备的分辨率。缓冲寄存器实质上是内存的一部分，因此，主机可对每一个像素进行访问。显示逻辑的功能是：把缓冲寄存器中的数字化图像重新转换成全电视信号，以便在监示器上显示缓冲寄存器中的内容。

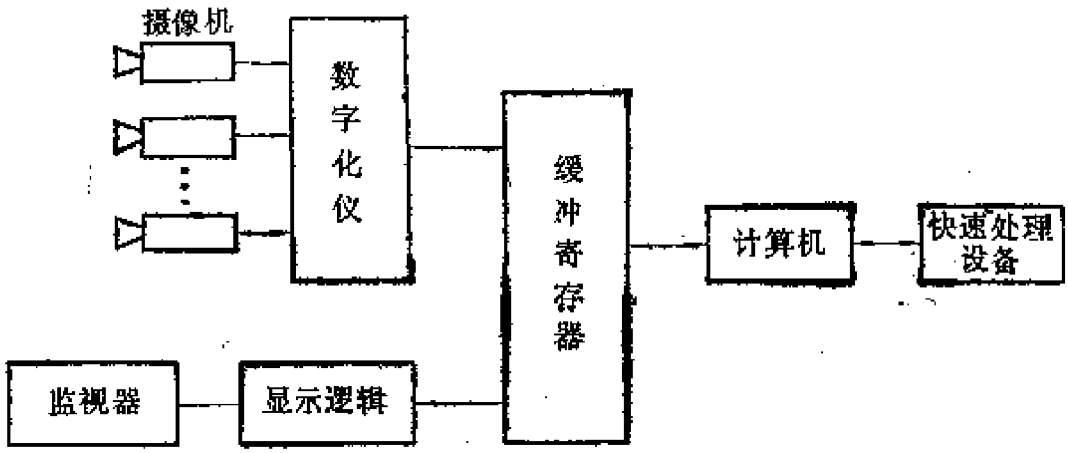


图 8.17　机器人视觉系统

能同时转换多个摄像机的输出信号，具有多个缓冲寄存器的实时图像采集设备已经商品化。除特殊要求的情况外，一般只要根据对功能、分辨率、灰度级、信噪比、软件及快速处理部件的配备等要求选购。

供选用的摄像机主要有两种。一种称为视像摄像机（Vidicon Camera），这种摄像机是由摄像管和相应的电子线路组成，它输出标准的全电视信号，也能输入外来的水平和垂直信号，工作在外同步状态。它的光谱敏感范围为 300～700nm（波长）。景物发射出的光聚焦在对光敏感的靶子上形成图像，用电子束对这靶子扫描，把图像转换成串行输出的电信号。光敏感靶子、电子枪以及若干电极装在一个真空管（摄像管）内，因此，视像摄像机具有体积大、

易碎怕振的缺点;而且,电子束的偏转存在时间漂移。此外,这种摄像机还存在使用寿命和平均无事故时间较短,以及内部存在高压(约900V)的问题。因此,人们现在越来越倾向于使用另外一种摄像机——固态摄像机(Solid-State Camera),其中最有代表性的是电荷耦合器件(Charge Couple Device)摄像机,简称CCD摄像机。

CCD摄像机可以是一维的(线阵的),也可以是二维的(面阵的)。CCD由一组离散的光敏感元件阵列组成,当光照射到这个阵列上时,每个光敏感元件上产生电荷,电荷的多少和光敏元件上的照度成正比。这些电荷聚集在每个光敏元件下面的电容里,然后用两相时钟脉冲把这些电荷顺序地传送到放大器输入端,于是,在放大器的输出端产生代表图像的电压时间信号。

CCD摄像机具有体积小、重量轻、寿命长、抗冲击等优越性。由于CCD是固态的,所以它工作起来特别可靠,不存在如视像摄像机那样的电子束偏转漂移、存在高压和使用寿命不长这样一些问题,而且CCD摄像机耗电极少,一般只需几十个毫瓦就可以启动。CCD的光谱敏感范围为420～1100nm(波长)。

但是,CCD存在灵敏度不一致的问题。它的光敏元件间的灵敏度的不一致可高达10%。虽然,由于光敏元件的灵敏度的稳定性很好,这种不一致性可以校正,但实现对每个像素逐一加以校正要耗费计算时间,所以这仍然是CCD摄像机的一个问题。即便这样,CCD摄像机仍然更适合于机器人视觉。

由于机器人视觉借用了电视技术的一些成就,用摄像机把景物转换成图像,因此,以下将对摄像机输出的电视信号作简要的说明。

美国电子工业协会EIA(Electronics Industries Association)规定了电视信号的标准形式,称为EIA RS-170电视信号形式。按照这种形式,一幅图像(一帧)被分成两场,每场240行,从上到

下逐行扫描。第一场的各行是一帧的奇数行，第二场的所有行是同一帧图像的偶数行，这种扫描方式称为隔行扫描。在场与场之间有相当于 22.5 行的时间从屏幕的下部返回到屏幕的上部（场回扫），因此，每帧有 525 行，其中的 45 行用作场回扫。每场耗时 1/60s，每帧的时间为 1/30s。

EIA RS-170 全电视信号由灰度信号、水平同步周期、垂直同步周期组成。图 8.18 是一扫描行的电视信号。灰度信息的电平在 0.7V（黑）和 1.5V（白）之间，每一扫描行结束后有大约 11μs 的间隔时间，称为水平同步周期，在这周期里有宽度为 4.7μs、电平为零的脉冲——水平同步脉冲。在每扫完一场（240 行）后有一宽度约为 68.25μs 的负脉冲，它标志一场的结束，这个脉冲就是垂直同步脉冲。

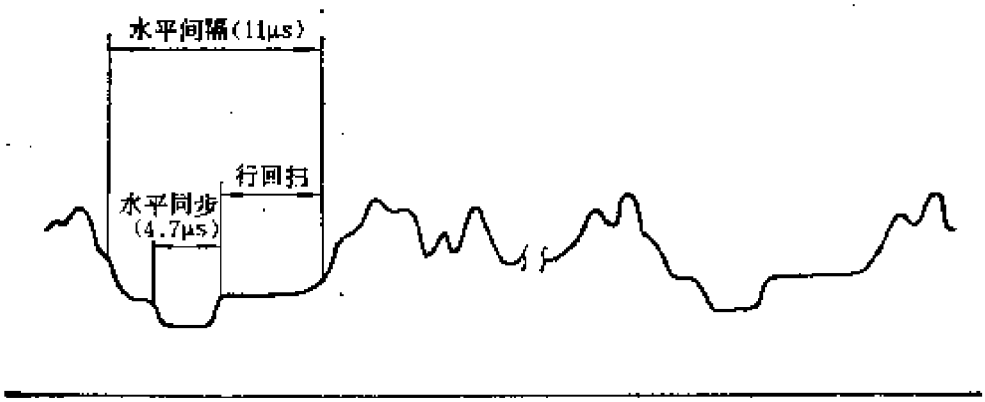


图 8.18　一行扫描周期的电视信号

最后，还有一个需要注意的问题：为了与电视的标准形式一致，除了一些工业用途的摄像机外，象素并不是正方形，而是长方形。这往往为人忽略而导至识别和定位产生困难。

第九章　视觉信号的初级处理

Tenenhaum 对人们的视觉的行为进行了仔细认真的观察和分析，指出，人在观察和理解自然景物的过程中，存在着许多层次的中间描述．根据这一特点，机器人视觉是一个多层次的对景物的描述和理解过程．　本章将论述对视觉信号(图像)的初级处理，主要从图像灰度的变化实现对图像的棱线表示和图像分割．

§9.1　图像的预处理

图像的预处理是对输入的视觉数据进行的一系列的加工中的第一步，其任务是对输入的图像进行改进，以获得能正确反映外部景物的高质量的图像，为以后的视觉处理创造条件．

一、几何失真的校正

上一章中的摄像机的几何模型，只是一种理想的情况．当用实际的摄像机摄取景物的图像时，由于镜头系统的精密度有限、电子扫描存在非线性、光敏元件阵列分布不可能绝对一致，因此不可避免地使图像存在几何畸变．

设 $g(x', y')$ 是实际摄得的图像，$f(x, y)$ 是理想情况下的图像．对图像的几何失真进行校正就是要寻求如下变换：

$$\begin{cases} x = T_1(x', y') \\ y = T_2(x', y') \end{cases} \tag{9-1}$$

对畸变图像 $g(x', y')$ 逐点进行校正，并把 $g(x', y')$ 的值(灰度)赋到正确的位置 (x, y) 上，得到经过几何校正的图像 $f(x, y)$。以上赋值的过程称为再采样。

为了求得校正函数 T_1 和 T_2，必须提供一定数量的控制点，这些点在畸变图像中的坐标 (x_i', y_i') 和在理想图像中的坐标 (x_i, y_i) 是已知的，而且，假设校正函数的形式是已知的。最常见的情况是假设校正函数是线性的：

$$\begin{cases} x = a_{11}x' + a_{12}y' + a_{13} \\ y = a_{21}x' + a_{22}y' + a_{23} \end{cases} \tag{9-2}$$

如下双线性畸变函数也是常用的校正函数形式：

$$\begin{cases} x = a_{11}x' + a_{12}y' + a_{13}x'y' + a_{14} \\ y = a_{21}x' + a_{22}y' + a_{23}x'y' + a_{24} \end{cases} \tag{9-3}$$

对于小范围内的局部畸变，以上校正函数足以进行准确的校正。对于大范围的复杂的畸变，或者选用高阶多项式函数进行校正，或者把图像划分成若干小的区域，在每一小区域里用以上线性校正函数或双线性校正函数进行校正。

无论选用哪种校正函数形式，校正函数的系数都采用平方误差最小方法确定。以双线性校正函数为例，把控制点代入校正函数 (9-3)，得如下方程组：

$$\begin{cases} x_i = a_{11}x_i' + a_{12}y_i' + a_{13}x_i'y_i' + a_{14} \\ y_i = a_{21}x_i' + a_{22}y_i' + a_{23}x_i'y_i' + a_{14} \end{cases} \tag{9-4}$$

$$i = 1, 2, \cdots, N$$

将方程组 (9-4) 写成矩阵形式：

$$QA = C \tag{9-5}$$

其中

$$A = [a_{11}, a_{12}, a_{13}, a_{14}, a_{21}, a_{22}, a_{23}, a_{24}]^T \tag{9-6}$$

$$C = [x_1, y_1, x_2, y_2, \cdots, x_N, y_N]^T \tag{9-7}$$

矩阵 Q 的 $2i-1$ 行和 $2i$ 行分别是

$$q_{2i-1}=[x_i', y_i', x_i'y_i', 1, 0, 0, 0, 0] \tag{9-8}$$

$$q_{2i}=[0, 0, 0, 0, x_i', y_i', x_i'y_i', 1] \tag{9-9}$$

平方误差最小意义下 A 的最优解为

$$A=(Q^TQ)^{-1}Q^TC \tag{9-10}$$

应当注意，经过校正后的值 x 和 y 一般不是整数，进行再采样时对此有两种处理方法。一种是把 x、y 取成最邻近的整数；另一种方法是进行内插，求出当 x、y 为整数时的灰度值。

二、灰度修正

存在两种灰度修正。一种叫灰度校正，另一种称为灰度变换。

灰度校正是对光敏元件阵列中敏感元件灵敏度的不一致性进行补偿。首先，用亮度均匀的发光源对所有位置上的光敏元件的灵敏度进行标定。在对灰度进行校正时，根据每一个像素对应的光敏元件的灵敏度，对图像的灰度逐点进行校正。

灰度变换是对每一个像素的灰度进行某种转换。最常见的情况是增强图像的对比度，即加大最亮点和最暗之间的动态范围。一般选择某个我们感兴趣的灰度区间实现对比度增强，其灰度转换曲线如图9.1所示。其中 p_i 是输入灰度，p_0 是输出灰度，$[a, b]$ 是感兴趣的灰度区间。

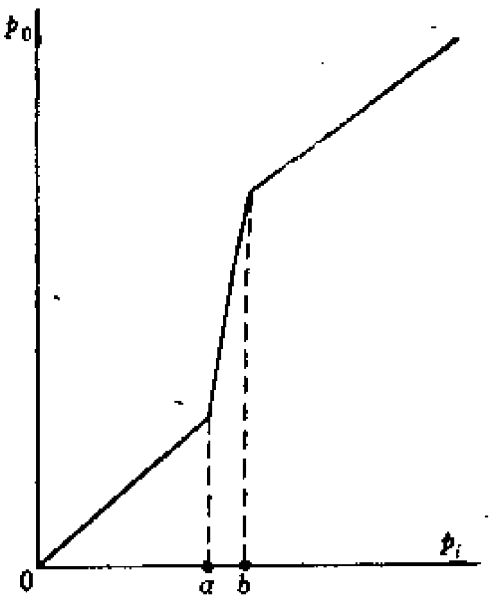
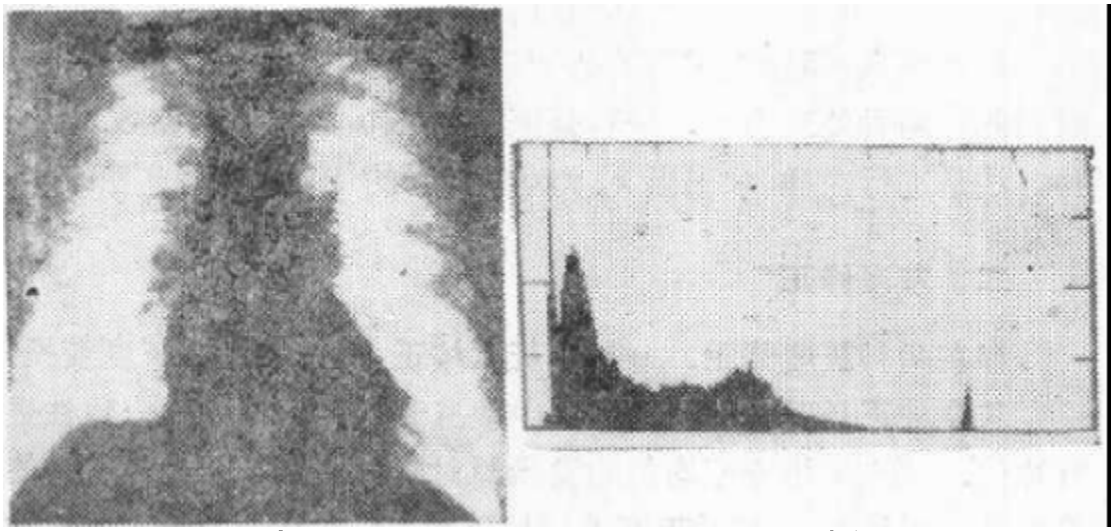


图 9.1　对比度增强的灰度转换曲线

另外一种常见的情况称为直方图变换。即通过对图像的灰度进行变换使得图像的灰度

直方图具有某种性质．图像的灰度直方图是灰度的函数 $h(p)$，它表示在图像中灰度级是 p 的像素的数目(或相对值)．因此直方图 $h(p)$ 反映了图像中某种灰度的出现频率．图 9.2(a)、(b) 分别是图像和其灰度直方图的一个实例．



(a)　　(b)

图 9.2　图像和其灰度直方图

直方图均衡 (Histogram Equalization) 是最常用的一种直方图变换，它通过对图像的灰度进行某种转换，将直方图值较大处的灰度伸长，把直方图值较小处的灰度压缩，使得图像的直方图为常数．

设 p 是输入灰度，q 是输出灰度，灰度转换关系为

$$q = f(p) \tag{9-11}$$

设 $h(p)$ 是原始图像的直方图，$g(q)$ 是经过灰度变换后图像的直方图，转换关系 $f(p)$ 的选择必须使得

$$g(q) = \frac{N}{M} \tag{9-12}$$

其中，M 是图像的灰度级数，N 是图像的像素的数目．

由式 (9-11) 得

$$dq = f'(p)dp \,. \tag{9-13}$$

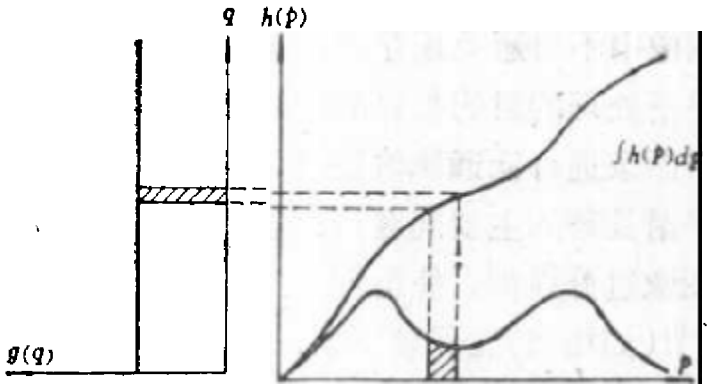
在对图像的灰度进行变换前和变换后，灰度区间 dp 和 dq 中的像素数目应当相等(见图 9.3(a))。

$$h(p)dp = g(q)dq \tag{9-14}$$

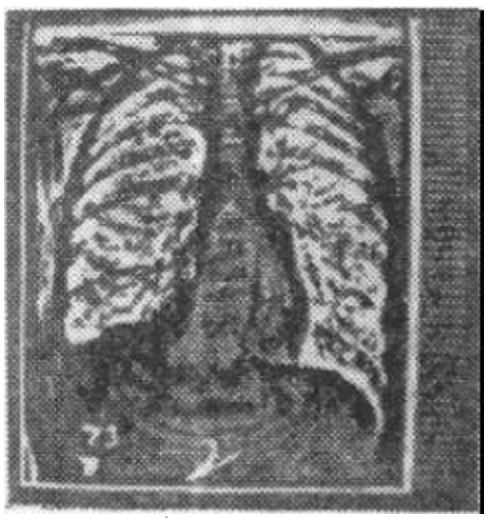
将式 (9-12) 和 (9-13) 代入上式，得

$$f'(p) = \frac{M}{N} h(p)$$

所以灰度变换关系为



(a)



(b)

图 9.3 (a) 直方图均衡变换原理示意图；
(b) 直方图均衡变换处理后的图像。

$$f(p)=\frac{M}{N}\int_0^p h(p)dp \qquad (9\text{-}15)$$

直方图均衡化变换使得对于多数像素对比度得到了增强，因此，改善了图像特征的可检测性．对于实现两幅相似景物的图像比较，直方图均衡化变换是一种很好的预处理，以克服摄像时由照明条件不同造成的影响．图 9.3(a) 是直方图均衡变换原理示意图；图 9.3(b) 是经过直方图均衡变换处理后的图像．

三、图像平滑处理

摄得的图像中不可避免地存在着噪声干扰，多数情况下干扰是加性的．平滑处理的目的是要减少图像中的加性噪声干扰．平滑本质上是对图像进行低通滤波处理，程度不同地总是要使图像模糊，因此，平滑处理的主要问题是：怎样才能较好地去除噪声而又不至于使图像过分模糊，使得图像中的一些变化迅速（频率较高）的重要细节（如棱线）能保留下来．

如果干扰只发生在一些孤立的点上，或发生在一些很细的窄条上，这时，我们可以识别出受干扰的像素，用邻近点的灰度的内插值来取代受干扰的像素的灰度．即只对图像的局部实行平滑处理．这种平滑处理给图像的质量造成的不良影响很小．

一种能减少随机噪声干扰而又对图像的质量不产生任何不良影响的方法是，对同一景物摄取多幅图像，然后把它们的对应像素灰度值求平均值．这种方法称为集平均（Ensemble Avevage）法，它的作用相当于加长曝光时间．集平均法只适合于静止的景物．

一般的平滑方法是局部平均（Local Average）法．作法是以某像素为中心，在图像上开一个窗口，把这窗口内的所有像素的灰度值加权求和，并以这个值来取代这个中心像素的灰度值．加权系数的大小随离开中心像素的距离而变，一般，离中心像素越远，加权系数越小．以上平滑原理可以表示成线性算子形式．图 9.4

是一个近似的 3 × 3 高斯 (Gauss) 分布平滑算子. 它表示了在图像上开的窗口，算子中的系数是对应像素的灰度加权系数. 用平滑算子作用于图像的每一个像素，即实现了对整幅图像的平滑处理.

0.0625	0.125	0.0625
0.125	0.250	0.125
0.0625	0.125	0.0625

图 9.4　3×3 近似高斯分布平滑算子

§9.2　提取二值化图像的边界

提取边界是图像分割的重要组成部分，通过提取出物体的图像的边界，实现把物体的图像和背景分割开来.

实践证明，二维图形的形状信息主要包含在它的边界中，根据 Hubel 和 Wiesel 的工作，人的视觉系统首先在视网膜上实现边界线和棱线的提取，然后再把所得的视觉信号提供给大脑. 因此，我们用提取出的边界(一条封闭曲线)来表示二维图形(物体的像). 这种做法，实质上是把一个二维的问题表示成一维的情况，大大地节省了处理的时间，为实现物体识别和物体定位带来了很大的方便.

首先论述二值化图像的情况.

二值化图像指的是只有两个灰度级的图像. 不失一般性，设这两个灰度值为“0”和“1”，而且假设背景的灰度值是 1 (白)，物体图像的灰度值是 0 (黑). 相当多工业用途的机器人视觉系统使

用的是二值化图像，完成分类、检验、定位、轨迹跟踪等任务。

设有一二值化图像如图 9.5(a) 所示。图中 P_0 在物体(的图像) O 的边界线上，我们称 P_0 为起始点，它的坐标为 (x_0, y_0)。起始点 P_0 由对画面进行扫描求得。

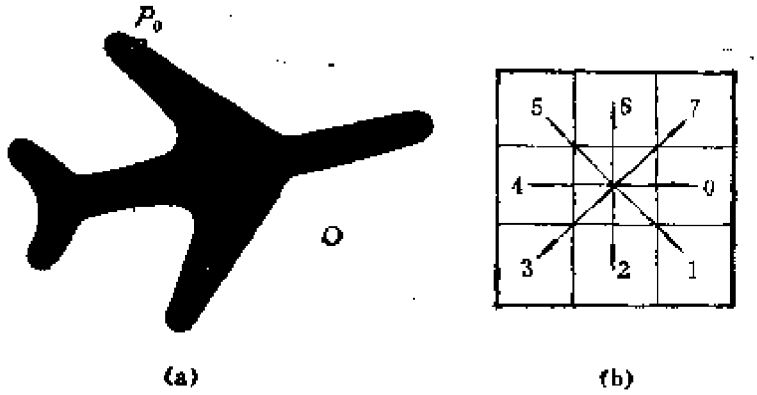


图 9.5 (a) 物体的象和起始点 P_0；
(b) 弗里曼码。

以起始点 P_0 为中心开一个 3 × 3 的窗口如图 9.5(b) 所示，沿物体边界顺时针方向的下一个边界点 P_1 只能是 P_0 周围的八个相邻像素之一，即从一个边界点向下一个边界点转换只能沿图 9.5(b) 所示的 1～8 八个方向进行，我们称这八个方向码为弗里曼(Freeman)码。以下是提取二值化图像的边界的算法：

(1) 给定起始点 P_0，任意给定弗里曼码 C_{-1}，并执行

$$f(x_0, y_0) := 2;$$

(2) Loop: P_i 是边界点，$C_i := C_{i-1}$；

(3) 由 C_i 求出 $x_{i+1} = x_i + \Delta x_i$，$y_{i+1} = y_i + \Delta y_i$

如果 $f(x_{i+1}, y_{i+1}) = 2$，存储 C_i，退出；

如果 $f(x_{i+1}, y_{i+1}) = 1$，执行 (4)；否则执行 (6)；

(4) Loop: $C_i := (C_i + 1)$ MOD8，由 C_i 求出 $x_{i+1} = x_i + \Delta x_i, y_{i+1} = y_i + \Delta y_i$；

(5) 如果 $f(x_{i+1}, y_{i+1}) = 2$，存储 C_i，退出；

如果 $f(x_{i+1}, y_{i+1}) = 1$，执行 (4)；否则存储 C_i，x_{i+1}，y_{i+1}，执行 (2)；

(6) Loop：$C_i := (C_i - 1)\text{MOD}8$，由 C_i 求出 $x_{i+1} = x_i + \Delta x_i$，$y_{i+1} = y_i + \Delta y_i$；

(7) 如果 $f(x_{i+1}, y_{i+1}) = 2$，存储 C_i，退出；

如果 $f(x_{i+1}, y_{i+1}) = 0$，执行(6)；否则 $C_i := (C_i + 1)\text{MOD}8$；存储 C_i，x_{i+1}，y_{i+1}，并执行 (2)。

在以上提取边界的算法中，$f(x_i, y_i)$ 是第 i 个边界点 P_i 的灰度值；C_i 是从 P_i 到 P_{i+1} 的弗里曼码。提取出的边界被表示成边界点序列 $P_0, P_1, \cdots, P_{N-1}$ 和边界的弗里曼码链 $C_0, C_1, C_2, \cdots C_{N-1}$。对于已提取过边界的物体，在其边界上用第三种灰度值(2)作出标记，这样，当在屏幕上存在多个物体的像时才能作到对每个物体的像只提取一次边界。

§9.3 用动态规划法提取灰度图像的边界

对于灰度图像，提取边界线和棱线的困难主要是图像中存在的噪声造成的。一种很容易想到的办法是，先对图像进行平滑处理(或别的滤波处理)，然后再对经过处理后的图像提取边界线。这确实是一种提高信噪比的办法，但是，平滑处理不可避免地将使图像在灰度变化剧烈的地方出现一定程度的模糊，对提取边界和棱线产生不良的影响。

本节论述提取边界的动态规划法，这种方法适用于加性干扰低信噪比的图像，它使用具有马尔科夫 (Marcov) 性质的对数似然函数为评价函数，用动态规划的方法提取最优局部边界线段，然后用启法式图搜索方法将这些边界线段组合成完整的边界线。

图像的第 (i, j) 个像素的灰度值用 g_{ij} 表示。设 g_{ij} 是由图像

信号 f_{ij} 和噪声干扰信号 n_{ij} 的和组成，即

$$g_{ij} = f_{ij} + n_{ij}$$

其中，n_{ij} 是期望值为零，方差为 σ^2 按正态分布的白噪声，n_{ij} 的分布密度函数为 $N(0, \sigma^2)$。图像信号的灰度值有两种情况，它们分别是背景的图像灰度 r_{out} 和物体的图像灰度 r_{in}。

一、用动态规划法提取局部边界线段

如图9.6所示，物体图像的边界用向量序列 $\{t_m\}m=1, 2, \cdots, M$ 表示，其中向量 t_m 是像素间的分界向量。对于物体的图像，t_m 的方向总是顺时针方向。见图 9.6，图中每一个方格代表一个像素，画斜线的部分表示物体的像。

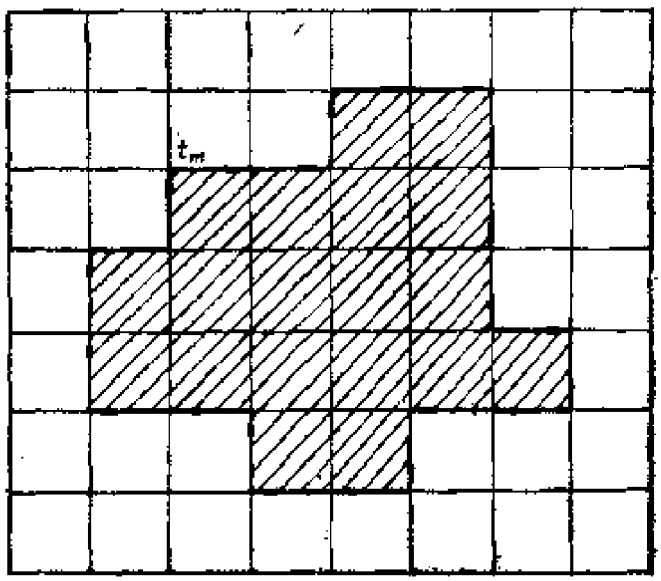


图 9.6　边界的表示

设边界的起始向量 t_1 已知，沿 t_1 的方向，规定由像素组成的长方形区域为 D_1、D_2、D_3、D_4、D_5、D_6、D_7，如图 9.7 所示。图中每一个区域由相应的虚线所包围的像素组成。这样的长方形域是进行动态规划的区域，也就是用动态规划的方法求这区域内的局部

边界线．规划区域的大小应事先选定，一般不超过 D_7。由图 9.2 可以看到

$$D_{K+1} = D_K \cup d_{K+1}$$

其中 d_{K+1} 是包围 D_K 的长方形环状域．D_K 的大小由其所包含的像素的数目来度量，值为

$$\rho_K = K(2K - 2) \tag{9-16}$$

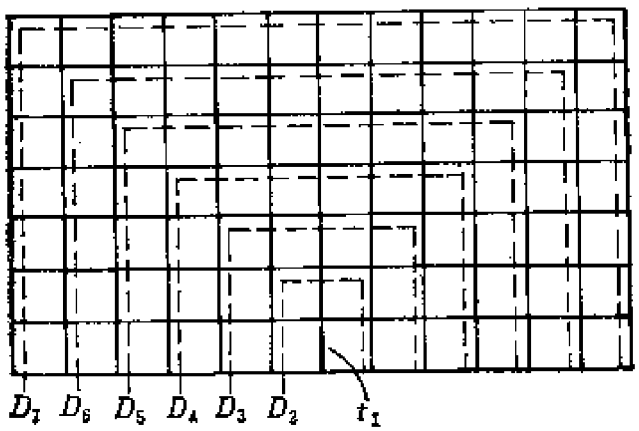


图 9.7　规划区域

对局部边界线的唯一限制是：如果局部边界线离开了区域 D_K，则不再允许它重新进入 D_K。图 9.8(a) 所示的局部边界线满足以上限制，是合法的；图 9.8(b) 所示的局部边界线是非法的，因为它离开了 D_3 之后又重新进入 D_3。由于这个限制，本方法只适用于低曲率边界的提取．

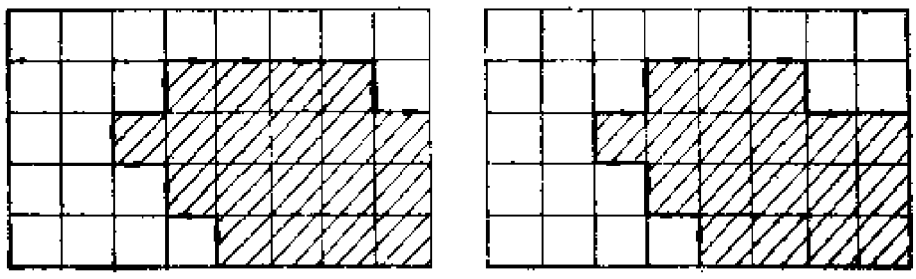


图 9.8　对局部边界线的限制

以上关于局部边界线的限制，实际上是规定局部边界线是由沿着D域边界上的向量和过渡向量 t_i 形成，如图 9.9 所示。图中t_{iK}是过渡向量，它实现从 D_{K-1} 的边界到 D_K 边界的过渡。 如果把过渡向量 t_{iK} 的端点标注成 x_K^i，x_K^l 就是在 D_K 中的局部边界线的终结点。如图 9.10 所示，D_2 有三个终结点，当 $K \geqslant 3$ 时，D_K的终节点数为

$$N = 4K - 3$$

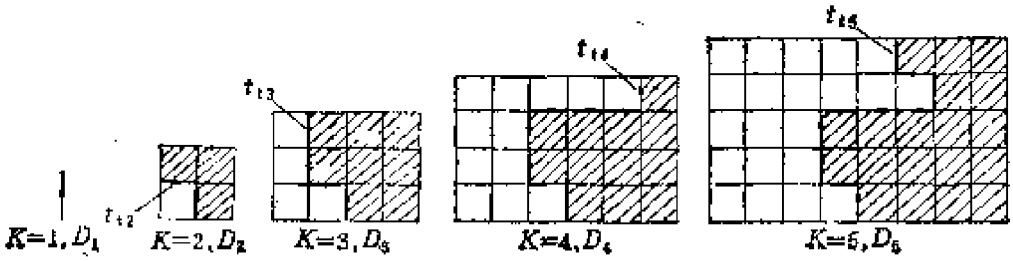


图 9.9　局部边界线的组成

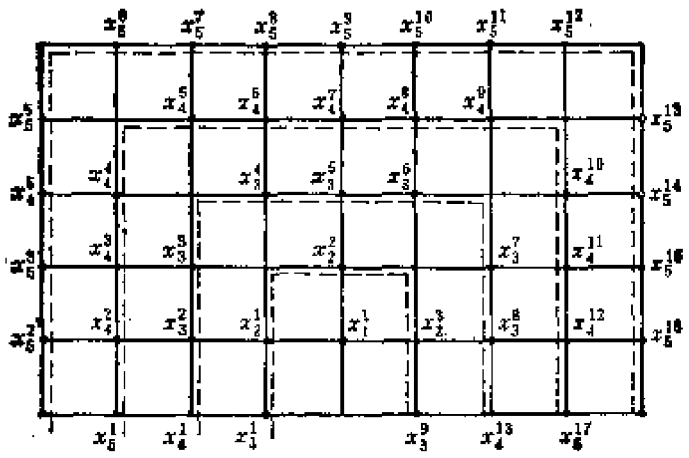


图 9.10　状态图

由于对局部边界线存在如上所示的限制，不难看出，从 D_K 上的任一终节点 x_K^i 到 D_{K+1} 上的任一终结点间的路径是唯一的。因

此，我们把K的值称为级，把 D_K 上的终结点 x_k^i 称为第K级上的状态，而把 D_K 上的局部边界线用状态链 $x_1^{i_1} x_2^{i_2} x_3^{i_3} \cdots x_K^{i_K}$ 来表示。

定义

$$B_K = \{g_{ij} | P_{ij} \subset D_K\} \tag{9-17}$$

$$b_K = \{g_{ij} | P_{ij} \subset d_K\} \tag{9-18}$$

式中 P_{ij} 表示象素 (i, j)，g_{ij} 是 P_{ij} 的灰度值，因此 B_K 表示了区域 D_K 中的某种灰度分布，b_K 表示 d_K 中灰度的某种分布。显然存在 $B_K = B_{K-1} \cup b_K$。设 D_K 中一状态链为 S_K（即 D_K 中的局部边界线），则出现 B_K 和 S_K 的对数似然函数为

$$l(B_K, S_K) = l(B_K | S_K) + l(S_K) \tag{9-19}$$

其中 $l(S_K)$ 是状态链 S_K 的对数似然函数，$l(B_K | S_K)$ 是给定状态链 S_K 的条件下，出现灰度分布 B_K 的对数似然函数。由于对局部边界线的限制，可将对数似然函数 $l(B_K | S_K)$ 写成如下递归的形式：

$$l(B_K | S_K) = l(B_{K-1} | S_{K-1}) + l(b_K | x_K) \tag{9-20}$$

$$l(B_1 | S_1) = 0 \tag{9-21}$$

其中 x_K 是 S_K 的最后一个状态，把 x_K 合并到 S_{K-1} 的尾端即是 S_K。和 x_K 对应的过渡向量 t_{iK} 把区域 d_K 分成了两个部分，分别跟背景和物体图像所在的区域对应。

$l(S_K)$ 可写成如下递归的形式：

$$l(S_K) = l(S_{K-1}) + \ln P_T(x_K | x_{K-1}) \tag{9-22}$$

$$l(S_1) = l(x_1) = \ln P_S(x_1) \tag{9-23}$$

其中 $P_T(x_K | x_{K-1})$ 称为状态转移概率，它表示给定当前状态 x_{K-1} 的情况下，下一个状态是 x_K 的概率，而与以前的状态无关，因此，S_K 是一个马尔科夫状态链。状态转移概率 $P_T(x_K | x_{K-1})$ 由人事先给定。$P_S(x_1)$ 是先验概率，它表示起始状态在局部边界上的概率，一般取 $P_S(x_1) = 1$。

将式 (9-20) 和 (9-22) 代入式 (9-19)，得

$$l(B_K, S_K) = l(B_{K-1} | S_{K-1}) + l(S_{K-1}) + \ln P_T(x_K | x_{K-1})$$

$$+ l(b_K|x_K)$$
$$= l(B_{K-1},S_{K-1}) + \ln P_T(x_K|x_{K-1})$$
$$+ l(b_K|x_K) \tag{9-24}$$

考虑到噪声正态分布的假设，因此，

$$l(b_K|x_K) = \sum_{P_{ij}\in I_{OK}} \ln G(g_{ij} - r_{in}) + \sum_{P_{ij}\in I_{BK}} \ln G(g_{ij} - r_{out})$$

其中

$$G(y) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp(-y^2/2\sigma^2)$$

$I_{OK} = \{P_{ij}|P_{ij}\in d_K$ 而且 P_{ij} 在物体的像上$\}$

$I_{BK} = \{P_{ij}|P_{ij}\in d_K$，而且 P_{ij} 在背景上$\}$

经过化简得

$$l(b_K|x_K) = C - \sum_{P_{ij}\in I_{OK}} (g_{ij} - r_{in})^2/2\sigma^2$$
$$- \sum_{P_{ij}\in I_{BK}} (g_{ij} - r_{out})^2/2\sigma^2 \tag{9-25}$$

其中 C 是与 x_K 无关的常数。

公式 (9-24) 中的状态转移函数一般可这样来选择，使得它满足：(1) $P_T(x_K^i|x_{K-1}^j)$ 和 x_K^i 与 x_{K-1}^j 间的距离成反比；(2) 对于任一给定的 j，

$$\sum_i P_T(x_K^i|x_{K-1}^j) = 1 \tag{9-26}$$

我们把矩阵 $P_{K-1,K}$ 称为状态转移概率矩阵，它的第 (i,j) 个元素为 $P_T(x_K^i|x_{K-1}^j)$。因此，$P_{K-1,K}$ 矩阵表示了第 $K-1$ 级的所有状态到 K 级的所有状态间的状态转移概率。例如选择状态转移矩阵为

$$P_{12} = \begin{bmatrix} 1^a \\ 2^a \\ 1^a \end{bmatrix} [\alpha]^{-1}$$

$$P_{23}=\begin{bmatrix}\varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha}\\ 4^{\alpha} & 2^{\alpha} & 2^{\alpha}\\ 5^{\alpha} & 3^{\alpha} & 3^{\alpha}\\ 6^{\alpha} & 4^{\alpha} & 4^{\alpha}\\ 5^{\alpha} & 5^{\alpha} & 5^{\alpha}\\ 4^{\alpha} & 4^{\alpha} & 6^{\alpha}\\ 3^{\alpha} & 3^{\alpha} & 5^{\alpha}\\ 2^{\alpha} & 2^{\alpha} & 4^{\alpha}\\ \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha}\end{bmatrix}\begin{bmatrix}\beta_1 & 0 & 0\\ 0 & \beta_2 & 0\\ 0 & 0 & \beta_3\end{bmatrix}^{-\alpha}$$

$$P_{34}=\begin{Bmatrix}\varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 11^{\alpha} & 9^{\alpha} & 4^{\alpha} & 2^{\alpha} & 2^{\alpha} & 2^{\alpha} & 2^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 10^{\alpha} & 10^{\alpha} & 5^{\alpha} & 3^{\alpha} & 3^{\alpha} & 3^{\alpha} & 3^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 9^{\alpha} & 9^{\alpha} & 6^{\alpha} & 4^{\alpha} & 4^{\alpha} & 4^{\alpha} & 4^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 9^{\alpha} & 9^{\alpha} & 7^{\alpha} & 5^{\alpha} & 5^{\alpha} & 5^{\alpha} & 5^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 8^{\alpha} & 8^{\alpha} & 8^{\alpha} & 6^{\alpha} & 6^{\alpha} & 6^{\alpha} & 6^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 7^{\alpha} & 7^{\alpha} & 7^{\alpha} & 7^{\alpha} & 7^{\alpha} & 7^{\alpha} & 7^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 6^{\alpha} & 6^{\alpha} & 6^{\alpha} & 6^{\alpha} & 8^{\alpha} & 8^{\alpha} & 8^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 5^{\alpha} & 5^{\alpha} & 5^{\alpha} & 5^{\alpha} & 7^{\alpha} & 9^{\alpha} & 9^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 4^{\alpha} & 4^{\alpha} & 4^{\alpha} & 4^{\alpha} & 6^{\alpha} & 9^{\alpha} & 9^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 3^{\alpha} & 3^{\alpha} & 3^{\alpha} & 3^{\alpha} & 5^{\alpha} & 10^{\alpha} & 10^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & 2^{\alpha} & 2^{\alpha} & 2^{\alpha} & 2^{\alpha} & 4^{\alpha} & 9^{\alpha} & 11^{\alpha} & \varepsilon^{\alpha}\\ \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha} & \varepsilon^{\alpha}\end{Bmatrix}$$

$$\times\begin{bmatrix}r_1 & & 0\\ & r_2 & \\ 0 & & \ddots\\ \vdots & & & r_9\end{bmatrix}^{-\alpha}$$

其中 ε 是很小的数，使得相应的状态转移几乎不可能，$\alpha,\beta_i,\gamma_i,\cdots$ 的选择使得 $P_{K-1,K}$ 的任一列的元素的和为 1。 正实数 α 称为刚性

系数，a 的值选择得越大，用动态规划法求得的局部边界趋于图 9.11 所示的直线的倾向性越强。

由计算对数似然函数的公式(9-24)可以看到，$l(B_K, S_K)$ 具有马尔科夫性质，因此可作为正向动态规划的收益函数。由此，求 D_K 中的局部边界线在数学上归结为：在 D_K 中选择状态链，使得 $l(B_K, S_K)$ 取极大值。这是一个典型的正向单自由端动态规划问题。

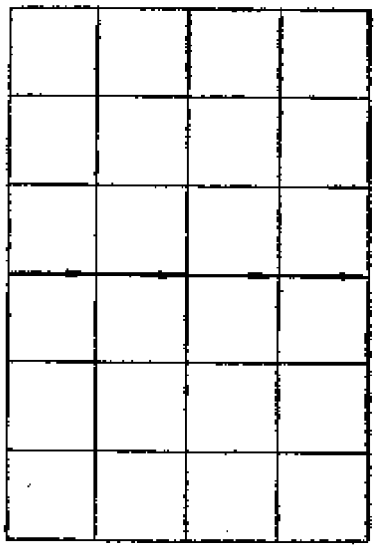


图 9.11 倾向性直线

二、正确概率下界

以下就动态规划法抽取的局部的边界的正确性进行简单讨论，即讨论局部界的正确概率的下界。

考虑这样一个状态链生成过程，它利用计算对数似然函数的递归公式(9-24)，每级只保留使 $l(B_K, \hat{S}_K)$ 取极大值的状态 $\hat{s}_K$，并把它作为状态链中的第 K 个状态，这样构成的状态链用 $\hat{S}_K$ 表示。显然，对于用动态规划得到的状态链 S_K，$\hat{S}_K$ 是次优的。设在区域 D_K 中的真实局部边界状态链是 S_K^*，由于 $\hat{S}_K$ 是次优的，所以必定有

$$P_r(\hat{S}_K = S_K^* | S_K^*) \leqslant P_r(S_K = S_K^* | S_K^*) \qquad (9\text{-}27)$$

其中 S_K 是用动态规划法得到的最优局部边界状态链，$P_r(S_K = S_K^* | S_K^*)$ 表示 S_K 正好是真实边界的概率；$P_r(\hat{S}_K = S_K^* | S_K^*)$ 表示次优状态链正好是真实边界的概率，为了今后书写方便，把它写成 P_{Kl}。由式(9-27)知，P_{Kl} 是用动态规划法得到的状态链是真实边界的概率的下界。因此，可用它来对动态规划法的精度进行评价，和对选择动态规划的参数提供原则性的指导。

为了论述方便，把每一级 d_K 区域中的灰度用图 9.12 中的符号

表示。

g^{44}	g^{43}	g^{36}	g^{47}	g^{48}	g^{49}
g^{43}	g^{33}	g^{34}	g^{35}	g^{36}	$g^{4,10}$
g^{42}	g^{32}	g^{22}	g^{23}	g^{37}	$g^{4,11}$
g^{41}	g^{31}	g^{21}	g^{24}	g^{38}	$g^{4,12}$

图 9.12　d_K 区域中的灰度表示符号

定义

$$\tilde{P}_K(v|u) \triangleq P_r(\hat{s}_K = x_K^v | \hat{s}_{K-1} = x_{K-1}^u) \tag{9-28}$$

则用次优算法得到的状态链 $\hat{S}_K$ 正好是真实边界的概率为

$$P_{KI} = P_S(x_1) \prod_{k=2}^{K} \tilde{P}_k(v^*|u^*) \tag{9-29}$$

其中 $P_r(x_1)$ 表示初始状态 x_1 在真边界上的概率，一般取 $P_S(x_1)=1$；$\tilde{P}_K(v^*|u^*)$ 为

$$\tilde{P}_K(v^*|u^*) = P_r(\hat{s}_K = x_K^{v^*} | \hat{s}_{K-1} = x_{K-1}^{u^*})$$

$x_{K-1}^{u^*}$ 和 $x_K^{v^*}$ 分别是第 $K-1$ 级和第 K 级真实边界上的状态。

由式 (9-25) 可得

$$\begin{aligned} \bar{I}_2(v|u^*) &= \ln \tilde{P}_2(v|u^*) \\ &= C_2 - \frac{1}{2\sigma^2}\left[\sum_{j=1}^{v}(g^{2j} - r_{out})^2 \right. \\ &\quad \left. + \sum_{j=v+1}^{4}(g^{2j} - r_{in})^2\right] + \ln P_T(x_2^v|x_1), \end{aligned} \tag{9-30}$$

$$v = 1, 2, 3$$

当 $K \geqslant 3$ 时

$$\bar{I}_K(v|u^*) = \ln \tilde{P}_K(v|u^*)$$

$$= C_K - \frac{1}{2\sigma^2}\left[\sum_{j=2}^{v}(g^{K,j-1} - r_{out})^2 + \sum_{j=v+2}^{4K-3}(g^{K,j-1} - r_{in})^2\right] + \ln P_T(x_K^v \mid x_{K-1}^{v'}) \quad (9\text{-}31)$$

$$v = 1, 2, \cdots, 4K-3$$

上两式中 C_K 与第 K 级所取的状态 x_K^v 无关. 按次优算法，第 K 级的状态在边界上的概率为

$$\tilde{P}_K(v^* \mid u^*) = P_r([\tilde{l}_K(v^* \mid u^*) - \tilde{l}_K(v \mid u^*)] > 0)$$

$$v = 1, 2, \cdots, v^*-1, v^*+1, \cdots, 4K-3$$

利用式 (9-31) 很容易得到

$$\tilde{l}_K(v^* \mid u^*) - \tilde{l}_K(v \mid u^*) = -\frac{1}{2\sigma^2}\sum_{j=v+1}^{v^*}[r_{out}^j - r_{in}^j - 2g^{K,j-1}(r_{out} - r_{in})] + \ln\frac{P_T(x_K^{v^*} \mid x_{K-1}^{u^*})}{P_T(x_K^v \mid x_{K-1}^{u^*})} \quad \text{当 } v < v^* \text{ 时}$$

$$= -\frac{1}{2\sigma^2}\sum_{j=v^*+1}^{v}[r_{in}^j - r_{out}^j - 2g^{K,j-1}(r_{in} - r_{out})] + \ln\frac{P_T(x_K^{v^*} \mid x_{K-1}^{u^*})}{P_T(x_K^v \mid x_{K-1}^{u^*})} \quad \text{当 } v > v^* \text{ 时} \quad (9\text{-}32)$$

构造如下向量 $z_K(v^*, u^*)$

$$z_K(v^*, u^*) = \begin{Bmatrix} \tilde{l}_K(v^* \mid u^*) - \tilde{l}_K(1 \mid u^*) \\ \tilde{l}_K(v^* \mid u^*) - \tilde{l}_K(2 \mid u^*) \\ \vdots \\ \tilde{l}_K(v^* \mid u^*) - \tilde{l}_K(v^*-1 \mid u^*) \\ \tilde{l}_K(v^* \mid u^*) - \tilde{l}_K(4K-3 \mid u^*) \\ \vdots \\ \tilde{l}_K(v^* \mid u^*) - \tilde{l}_K(v^*+1 \mid u^*) \end{Bmatrix} \quad (9\text{-}33)$$

利用 $z_K(v^*, u^*)$ 我们可以把式 (9-32) 写成紧凑的形式

$$P(v^*|u^*) = P_r(z_K > 0) \tag{9-34}$$

其中 0 是所有元素都是零的向量，$z_K > 0$ 表示向量 z_K 的元素都大于零. 由式 (9-32) 可以看到，z_K 的每一个元素都是随机变量 g^{Ki} 的线性函数，所以 z_K 是 $4K-4$ 维空间的正态分布的随机向量*，它的分布密度函数的期望向量和协方差矩阵分别为

$$\mu_K = \frac{(r_{in} - r_{out})^2}{2\sigma^2}\begin{bmatrix} h_{v^*-1} \\ \hline h_{4K-4-v^*} \end{bmatrix} + P_T$$

$$Q_K = \frac{(r_{in} - r_{out})^2}{\sigma^2}\left[\begin{array}{c|c} J_{v^*-1} & 0 \\ \hline 0 & J_{4K-4-v^*} \end{array}\right]$$

其中 P_T 是一向量，它的元素是式(9-32)中和转移概率有关的项. h_n 和 J_n 分别是

$$h_n = \begin{bmatrix} n \\ n-1 \\ \vdots \\ 1 \end{bmatrix}; \quad J_n = \begin{bmatrix} n & n-1 & n-2 & \cdots & 1 \\ n-1 & n-1 & n-2 & \cdots & 1 \\ n-2 & n-2 & n-2 & & \vdots \\ \vdots & \vdots & \vdots & & \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

所以，

$$P(v^*|u^*) = \int_0^\infty \cdots \int_0^\infty N(x, \mu_K, Q_K)dx_1 dx_2 \cdots dx_{4K-4}$$

这个积分可以用蒙特卡罗 (Monte Carlo) 方法完成，并由式 (9-29)，得正确概率的下界 P_{Kl} 的结果如下表所示：

表 (9-1) 中，S/N 是信噪比，它的定义是

$$S/N \triangleq |r_{in} - r_{out}|/\sigma$$

σ 是刚性系数，K 是选取的最大级数，P_{Kl} 是局部边界(用动态规划法得)是真实边界 S^* 的概率下界——正确概率下界.

---

* 正态随机变量的线性组合仍然是正态的，随机向量的各分量是正态的，向量必是正态的.

表 9-1　正确概率下界

S/N	a	K	$P_{K,l}$
2	6	3	0.904
		4	0.770
		5	0.620
		6	0.486
		7	0.372
	8	3	0.942
		4	0.843
		5	0.723
		6	0.608
		7	0.485
1	6	3	0.937
		4	0.880
		5	0.670
		6	0.470
		7	0.299
	8	3	0.974
		4	0.896
		5	0.758
		6	0.591
		7	0.429

三、边界追踪

在用动态规划法提取物体图像的全部边界时，首先应确定提取局部边界的区域 D_K 的大小(即 K 值)和刚性系数 a，用动态规划法提取出局部边界．然后用某种方法并把它们连接起来，组成全部边界线，这一过程称为边界追踪 (Boundary following)。

以下介绍用启发式图搜索方法实现边界追踪。

对于给定的起始状态 x_1 和起始边界向量 t_1 用动态规划法对区域 D_K^1 中的局部边界进行规划 (K 值已事先选定)，把 D_K^1 的第 K 级

上的状态 x_K^i（见图 9.10）称为这次规划的输出状态，每一个输出状态对应 D_K^1 中的一条局部边界，而且有一个收益值 $l(D_K^1, S_K^i)$ 和这个状态相联系．按 $l(D_K^1, S_K^i)$ 从大到小的顺序，选出前三个输出状态．然后以这三状态（和其对应的转移向量）为第二次扩展的初始状态，分别在各自的规划区域 $D_K^{2,1}$、$D_K^{2,2}$、$D_K^{2,3}$ 中用动态规划法产生三个新的输出状态（三个分支），而且规定每个新状态的收益值为本次规划的收益与其父结点的收益的和．逐层重复以上过程，构成一棵三叉树，树上的每一个节点都有一个收益值与之对应．因此，对于每一个初始状态隐含了一棵三叉树，边界追踪的任务就是对这棵树进行搜索，使得从初始状态出发，在这棵树上搜索出一条路径，使它重新回到初始状态所对应的像素位置，这条路径上的状态链就是边界．

实现边界跟踪并不要求把这棵树完全构造出来，我们可以选择每一个节点对应的收益和这个结点的深度的商为启发信息，每

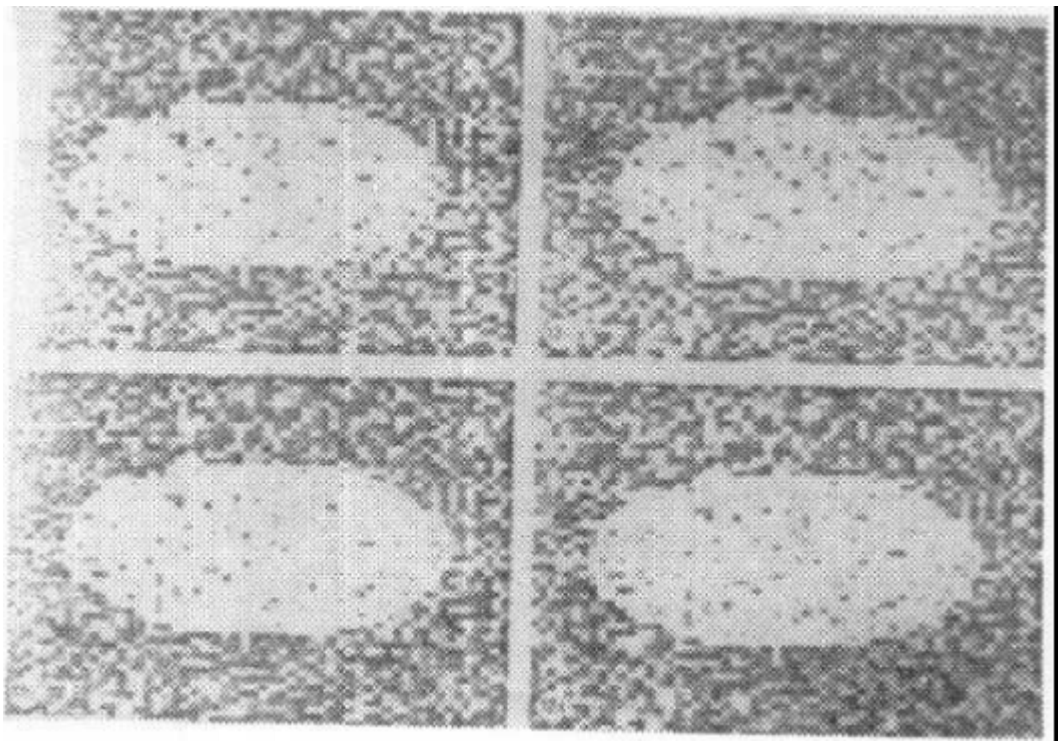


图 9.13　用动态规划法提取边界的实验结果

次进行扩展时，选择启发信息最大的节点进行扩展，这样的过程称为启发式收搜索。

图 9.13 是用以上方法提取边界的例子。图 9.13(a) 是一幅 64 × 64 信噪比 $S/N = |r_{in} - r_{out}|/\sigma = 2$ 的原始图像，图 9.13 (b) ~ (d) 是选择 $K = 5$，刚性系数分别是 $a = 5, 6, 8$ 所得的结果。

图 9.14 是用动态规划法提取 CAT 图像的边界的结果。原图的分辨率是 256 × 256，$r_{in} = 175$，$r_{out} = 85$，$\sigma = 20$。使用的动态规划参数为：$K = 5$，$a = 8$。

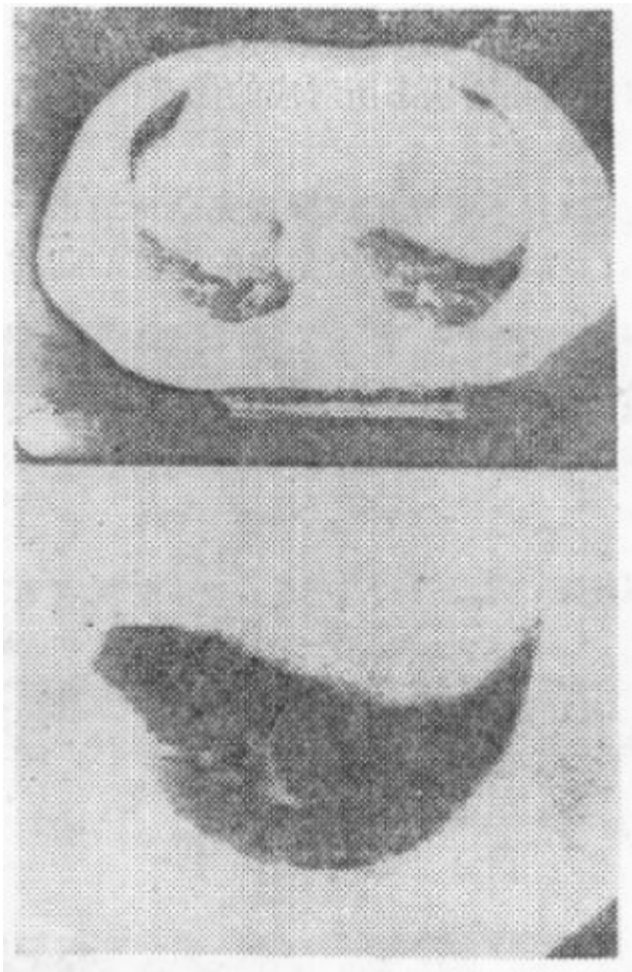


图 9.14 对 CAT 图像进行处理的实例

动态规划法适合于对低信噪比的图像进行处理，但是，计算复

杂，初始状态需要用别的方法提供。

§9.4 棱元素和局部棱线段的提取

在第八章已经论述过，图像的灰度是物体表面的光反射特性、表面的朝向、照明情况、景物分布状况(产生多次反射照明)、观察角度、摄像机光学性质等因素的综合效果。即便是照明没有任何变化，对于同一景物在不同角度摄得的图像，其灰度可以差异很大。景物的信息主要包含在灰度的变化情况中，因此，作为视觉处理的第一步，提取出图像中灰度发生剧烈变化的区域，然后把它们细化成相应的线条，这一过程称为提取棱线。

在图像的某些地方灰度发生迅速的变化是由于如下几种情况造成的：(1)在物体表面的棱线处，表面的朝向发生突变而使图像灰度发生突变。对于这种情况，图像的棱线和物体表面的棱线对应；(2)具有不同光反射特性的物体相互遮挡，使灰度在物体图像的交界处发生突变。这时，图像棱线即是物体图像的边界线；(3)同一物体的表面上不同的区域具有不同的光反射特性，例如物体表面上具有某种图案；(4)由于光被遮挡在景物中引成的影子，其中包括人为地设计的通过光源投射的某种图案在内。无论是以上哪一种情况造成的灰度突变都包含了关于景物的重要信息。Marr 和 Nishihara 在他们提出的视觉信息处理原理中，把提取棱线作为基本的出发点，称由棱线组成的图为初始结构 (Primal Sketch)，这种想法与 Hubel 和 Wissel 关于人的视觉系统在视网膜上提取棱线的观点是一致的。

提取棱线一般分成两步进行，第一步提取出棱线上的像素—棱元素 (Edge Element)，或提取局部棱线段 (Edge-bar)，然后用棱线追踪把它们组成棱线。本节论述棱元素和局部棱线段的提取。

提取棱元素和局部棱线段一般说来有三种方法，它们是(1)空间导数法；(2)样板匹配法；(3)参数模型法．以下论述这三种方法．

一、空间导数法

由于我们只是对图像中灰度变化剧烈的区域感兴趣，因此很容易想到用图像函数 $f(x, y)$ 的空间导数来探测棱元素．其中普遍采用的是用图像函数的梯度探测棱元素．图像函数 $f(x, y)$ 的梯度是一向量，它的幅值和方位角分别是

$$G(x, y) = [(\partial f/\partial x)^2 + (\partial f/\partial y)^2]^{1/2} \tag{9-35}$$

$$\theta(x, y) = \mathrm{tg}^{-1}[(\partial f/\partial y)/(\partial f/\partial x)] \tag{9-36}$$

幅值 $G(x, y)$ 表示图像的灰度在 (x, y) 处的变化率，因此，可以根据它的值的大小来判断 (x, y) 点是否在棱线上．最普遍的作法是，根据待处理的图像规定一个阈值 T，如果 $G(x, y) \geqslant T$，则认为 (x, y) 在棱线上．方位角 $\theta(x, y)$ 指示了棱线的方向，如果 (x, y) 在棱线上，$\theta(x, y)$ 是这条棱线的法线和 x 轴的夹角．因此方位角为实现棱线追踪提供了十分有用的信息．

为了论述方便，令

$$\Delta_1 \triangleq \frac{\partial f(x, y)}{\partial x}$$

$$\Delta_2 \triangleq \frac{\partial f(x, y)}{\partial y}$$

对于离散二维图像，存在多种计算 Δ_1，Δ_2 的近似算法．最早使用的是罗伯兹 (Roberts) 交叉算子 $R(i, j)$，其算法为

$$\Delta_1 = f(i+1, j+1) - f(i, j) \tag{9-37}$$

$$\Delta_2 = f(i, j+1) - f(i+1, j) \tag{9-38}$$

$$R(i, j) = (\Delta_1^2 + \Delta_2^2)^{\frac{1}{2}} \tag{9-39}$$

罗伯兹交叉算子计算偏导数 Δ_1、Δ_2 的算法可用如下线性算子表示。

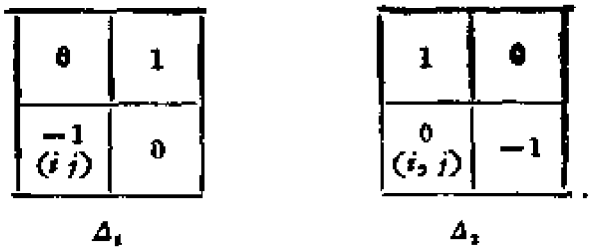


图 9.15　Roberts 交叉算子

以上正方形窗的左下角元素是像素 (i, j)，窗中标注的系数是对应元素的加权系数，Δ_1，Δ_2 分别是窗中像素的加权和.

罗伯兹交叉算子开的窗口较小，为了提高抗干扰能力，一般使用3 × 3的窗口，这就是有名的 Prewitt 和 Sobel 算子，如图 9.16 所示.

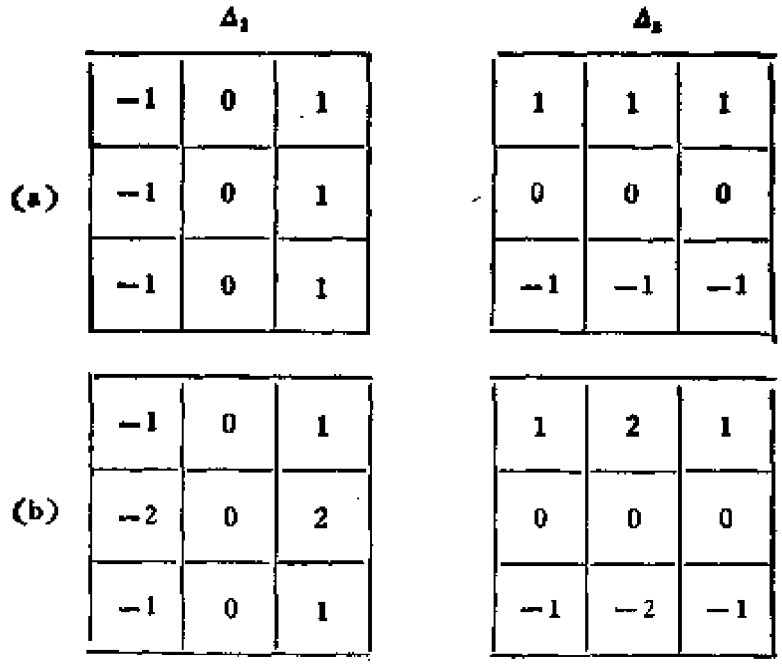


图 9.16　(a) Prewitt 算子；
(b) Sobel 算子.

图中，窗中心的像素是像素 (i, j). 在以上两种算子中 Sobel 算子用得更多.

用梯度求图像的棱元素由如下两步组成：

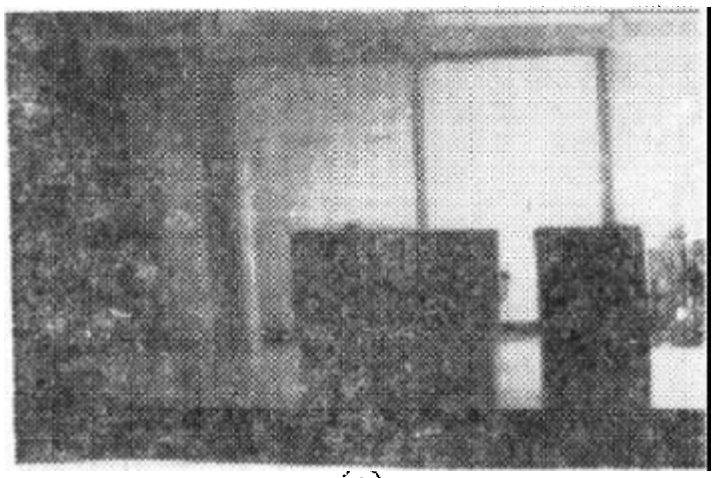
(1) 用以上算子作用于图像上的每一个像素，并构造中间图像 $G_m(i,j)$，构造方法如下，

$$G_m(i,j)=\begin{cases}(\Delta_1^2(i,j)+\Delta_2^2(i,j))^{\frac{1}{2}}; & (\Delta_1^2(i,j)+\Delta_2^2(i,j))^{\frac{1}{2}}\geqslant T\\ 0; & (\Delta_1^2(i,j)+\Delta_2^2(i,j))^{\frac{1}{2}}<T\end{cases}$$

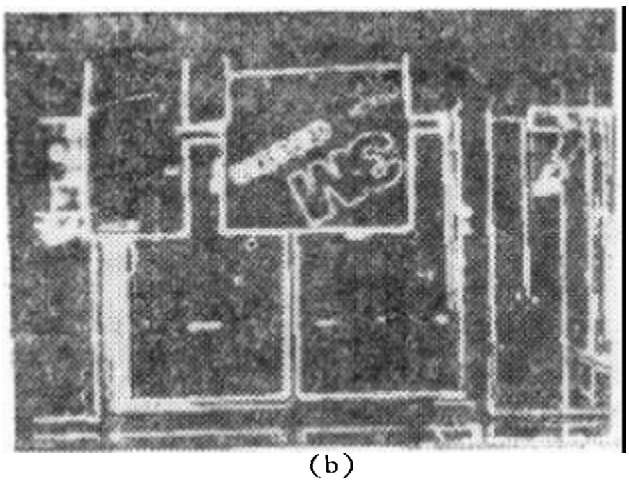
其中 T 是规定的阈值；

(2) 构造图像 $G_s(i,j)$，方法如下：

$$G_s(i,j)=\begin{cases}B_m; & (i,j)\text{ 是 }G_m(i,j)\text{ 的局部极大点;}\\ 0; & \text{其它.}\end{cases}$$



(a)



(b)

图 9.17 (a) 原始图像；
(b) 经 Sobel 算子和阈值处理后的图像。

其中 B_m 是最大灰度级。 $G_r(i, j)$ 中灰度值为 B_m 的像素为棱线元素。

图 9.17 是用 Sobel 提取棱线元素的例子。

另外一种用空间导数求棱元素的方法使用拉普拉斯 (Laplace) 算子:

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

如图 (9.18) 所示的一维情况那样，以 $\nabla^2 f$ 过零为条件来探测棱线元素。

对于离散图像，不难求得拉普拉斯算子如图 9.19 所示。

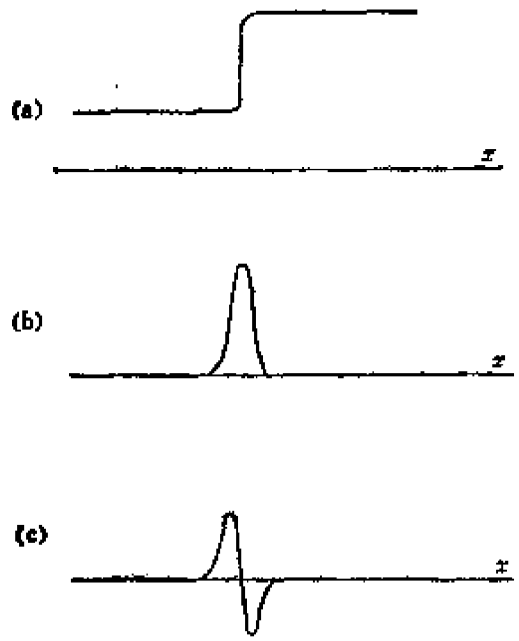


图 9.18 (a) 图像函数 $f(x)$;
(b) $\partial f/\partial x$; (c) $\partial^2 f/\partial x^2$.

0	1	0
1	−4	1
0	1	0

图 9.19 拉普拉斯算子

由于拉普拉斯算子法需求图像的二阶导数，使得高频噪声得到增强，因此拉普拉斯算子法的抗干扰能力差。一般，在使用这种方法时需先对图像进行低通滤波处理。 例如 Marr 和 Hildreth (1980)在使用这种方法时，先用高斯低通滤波器对图像进行处理，然后才把拉普拉斯算子作用于图像，并进行过零检测。

拉普拉斯算子法的另外一个缺点是，它不能提供棱线的方向信息。由于存在以上两个问题，近来拉普拉斯算子法用得较少。

二、Kirsch 样板匹配法

样板匹配法是建立一组小区域的不同方向的棱线的样板，当探测图像中某一像素是否在棱线上时，以这个像素为中心，把这组样板逐个地与图像进行匹配。Kirsch 算子法就是这种方法，它所使用的 3 × 3 的样板如 9.20 所示。这四个样板代表棱线的四种方向，实质上是四个带加权系数的算子。利用这四个算子构造中间图像如下：

$$G_m(i,j)=\begin{cases}\max\limits_l K_l[f(i,j)]; & \max\limits_l K_l[f(i,j)]\geqslant T\\ 0; & \max\limits_l K_l[f(i,j)]<T\end{cases}$$

其中 T 是规定的阈值，$K_l[f(i,j)]$ 表示以像素 (i,j) 为中心，用第 l 个 Kirsch 样板和图像匹配。因此，$\max\limits_l K_l[f(i,j)]\geqslant T$，表示图像能和某一样板匹配，这个样板的编号给出了棱线的方向信息。和以前一样，令 $G_m(i,j)$ 局部极大值点的灰度为最大灰度值 B_m，将其余点的灰度置为零，得图像 $G_s(i,j)$。$G_s(i,j)$ 的亮点即是棱线上的点——棱元素。

K_1

−1	0	1
−1	0	1
−1	0	1

K_2

1	1	1
0	0	0
−1	−1	−1

K_3

0	1	1
−1	0	1
−1	−1	0

K_4

1	1	0
1	0	−1
0	−1	−1

图 9.20　Kirsch 样板

三、Hueckel 参数模型法

Hueckel 法属于模型参数法。

如图 9.21(a) 所示，D 是以原点为圆心的一个圆形的区域. 图 9.21 是组成区域 D 的像素及每个像素的灰度，对于这个实例，区域 D 由 52 个像素组成. 一般可把 D 的像素的数目选成 32，52，69，88，173，D 的相应半径为

$$R = (S_d/\pi)^{\frac{1}{2}}$$

其中 S_d 是区域 D 的面积.

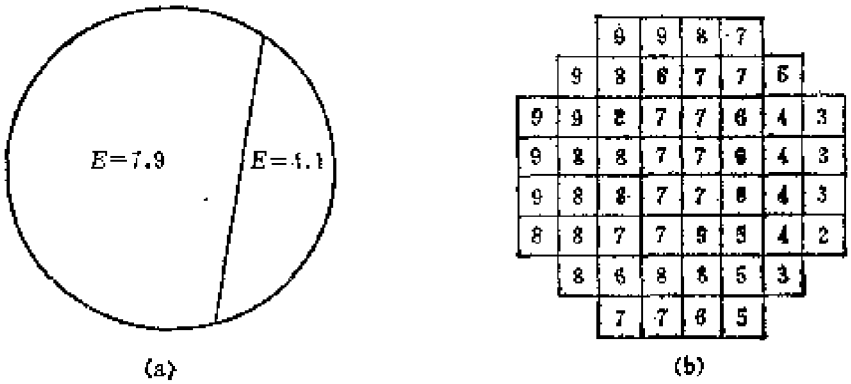


图 9.21 (a)区域 D；(b) 组成区域 D 的像素

设在 D 中有一条棱线. 棱线的方程为

$$cx + sy = \rho \tag{9-40}$$

其中 c，s 是棱线的方向余弦，因此，

$$c^2 + s^2 = 1 \tag{9-41}$$

设在棱线两侧的灰度分别是 b 和 $b + d$，则 D 中灰度的分布为

$$E(x, y, c, s, \rho, d, b) = \begin{cases} b; & cx + sy \leqslant \rho \\ b + d; & cx + sy > \rho \end{cases} \tag{9-42}$$

设图像函数为 $f(x,y)$，令 $F(x, y) = f(x + x_i, y + y_i)$. $F(x,y)$ 和参数模型 $E(x, y, c, s, \rho, d, b)$ 间的接近度用如下希尔伯特 (Hilbert) 距离度量：

$$\Lambda(c, s, \rho, d, b) = \int_D [F(x, y)$$

$$- E(x, y, c, s, \rho, d, b)]^2 dxdy \qquad (9\text{-}43)$$

现在的任务是寻求一种有效的算法，以确定参数 c, s, ρ, d, b，使得 $\Lambda(c, s, \rho, d, b)$ 最小，进而确定以 (x_i, y_i) 为中心的圆形邻域里是否存在棱线段，及棱线段的位置和方向 (c, s, ρ)。

不失一般性，令 D 的半径

$$R = 1 \qquad (9\text{-}44)$$

可数的无限集 $\{H_i(x, y) | i = 0, 1, 2, \cdots, \infty\}$ 是 D 上的希尔伯特函数空间的完备的归一化正交基。$H_i(x, y)$ 为

$$r^2 \triangleq x^2 + y^2$$

$$Q(r) \triangleq (1 - r^2)^{\frac{1}{2}}$$

$$h_0(x, y) \triangleq (5/6\pi)Q(r)(1 + 2r^2)$$

$$h_1(x, y) \triangleq (8/27\pi)^{\frac{1}{2}}Q(r)(5r^2 - 2)$$

$$h_2(x, y) \triangleq (3/\pi)^{\frac{1}{2}}Q(r)x(4r^2 - 1)$$

$$h_3(x, y) \triangleq (3/\pi)^{\frac{1}{2}}Q(r)y(4r^2 - 1)$$

$$h_4(x, y) \triangleq (3/\pi)^{\frac{1}{2}}Q(r)x(3 - 4r^2)$$

$$h_5(x, y) \triangleq (3/\pi)^{\frac{1}{2}}Q(r)y(3 - 4r^2)$$

$$h_6(x, y) \triangleq (32/3\pi)^{\frac{1}{2}}Q(r)(x^2 - y^2)$$

$$h_7(x, y) \triangleq (32/3\pi)^{\frac{1}{2}}Q(r) \cdot 2xy$$

$$H_0(x, y) \triangleq (2\pi/25)^{\frac{1}{2}}h_0(x, y)$$

$$H_1(x, y) \triangleq (9/2)^{1/2}h_1(x, y)$$

$$H_2(x, y) \triangleq h_2(x, y) + h_4(x, y)$$

$$H_3(x, y) \triangleq h_3(x, y) + h_5(x, y)$$

$$H_4(x, y) \triangleq (3/2)h_6(x, y)$$

$$H_5(x, y) \triangleq (3/2)h_7(x, y)$$

$$H_6(x, y) \triangleq (5/4)^{\frac{1}{2}}(h_2(x, y) - h_4(x, y))$$

$$H_7(x, y) \triangleq (5/4)^{\frac{1}{2}}(h_3(x, y) - h_6(x, y))$$

$$\vdots$$

图 9.22 是前八个归一化正交基的(值的)过零线。

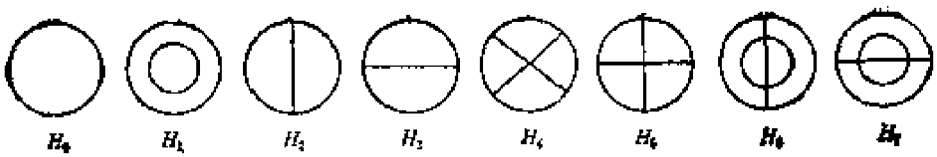


图 9.22　$H_0 \sim H_7$ 的值的过零线

由于元素归一性，即此有

$$A(c,s)=e_0(c,s)+U(c,s) \tag{9-53}$$

Hueckel 证明了：对于参数 c,s，$A(c,s,\rho,d,b)$ 的全局和局部极小点分别和 $A(c,s)$ 的全局和局部极大点一致。因此，我们可以从求 $A(c,s)$ 的极大点，求得参数 c,s，然后用如下公式求得其余的模型参数 (ρ,d,b)：

$$\rho=e_2(c,s)/\{2^{\frac{1}{2}}[U(c,s)+e_1(c,s)]\} \tag{9-54}$$

$$d=4A(c,s)/[(3\pi)^{\frac{1}{2}}(1-\rho^2)^2(1+2\rho^2)] \tag{9-55}$$

$$b=a_0-d\{4+\rho[3+\rho(2+\rho)]\}(1-\rho^2)/8 \tag{9-56}$$

限于篇幅，这里不再证明以上原理。

求出最优参数 c,s,ρ,d,b 并不意味着图像 $F(x,y)$ 确实存在相应的棱线。以下论述 Hueckel 提出的棱线存在的判断准则。

令向量 $\boldsymbol{E}'$，$\boldsymbol{F}'$ 分别是希尔伯特向量在 $\{H_i|i=0,1,\cdots,7\}$ 所张的子空间中的投影，可以证明，$\boldsymbol{E}'$，$\boldsymbol{F}'$ 的夹角的余弦为

$$K=\frac{\boldsymbol{E}'\cdot\boldsymbol{F}'}{\|\boldsymbol{E}'\|\|\boldsymbol{F}'\|}=\left|\frac{A(c,s)}{[6a_1^2+2(a_2^2+a_3^2+a_4^2)+3(a_0^2+a_7^2)]^{1/2}}\right| \tag{9-57}$$

如果在 $F(x,y)$ 中存在理想的棱线，则 $K=1$。如果 $K<0.9$，则表明 $F(x,y)$ 中干扰太强。例如，$F(x,y)$ 为图 9.21(b) 所示，用上述算法求得

$$(c,s,\rho,d,b)=(-0.99,0.14,-0.28,3.80,4.08),$$

$$K=0.95.$$

求得的棱线是否可以接受，除 K 值的大小外，还要考虑灰度阶跃量 d 的大小。Hueckel 提出如下可接受准则：

$$1-K<(1-\mathrm{conf})d^2/(\mathrm{diff}+d^2) \tag{9-58}$$

其中 conf 和 diff 是两个阈值，对于灰度是 4bit (16 级)的图象，它们的选择范围为

$$0.86 < \text{conf} < 0.9; \quad 1.50 < \text{diff} < 4.0$$

四、几种提取棱元素的方法的比较

企图不加任何限制地对各种方法进行排队，指出哪种方法最好，哪种方法次之，是不科学的．对一种提取棱元素方法进行评价，主要看它的抗干扰能力、漏测率的大小、提取出的棱元素的位置准确度、提供的关于棱线的信息、处理一幅图象所需的时间等．一种方法可能在一些方面表现出明显的优越性，但是，在另一些方面则不如别的方法．以上几种方法都有各自的存在理由，关键在于用户是否能根据具体的任务合理地选择．

1978 年 Pratt 提出了一种评价方法，他采用的评价函数为

$$F = \frac{1}{\max[N_A, N_I]} \sum_{i=1}^{N_A} \frac{1}{1 + ad_i^2} \tag{9-59}$$

其中 N_A 是实际提取出的棱线点数，N_I 是理想的棱线点数，d_i 是实际提取出的棱线点和棱线间的距离，a 是一常数，因此，ad_i^2 反映了提取出的棱线点的位置准确度．按如上评价函数，在正态分布的白噪声干扰情况下几种主要方法的品质曲线如图 9.23 所示．

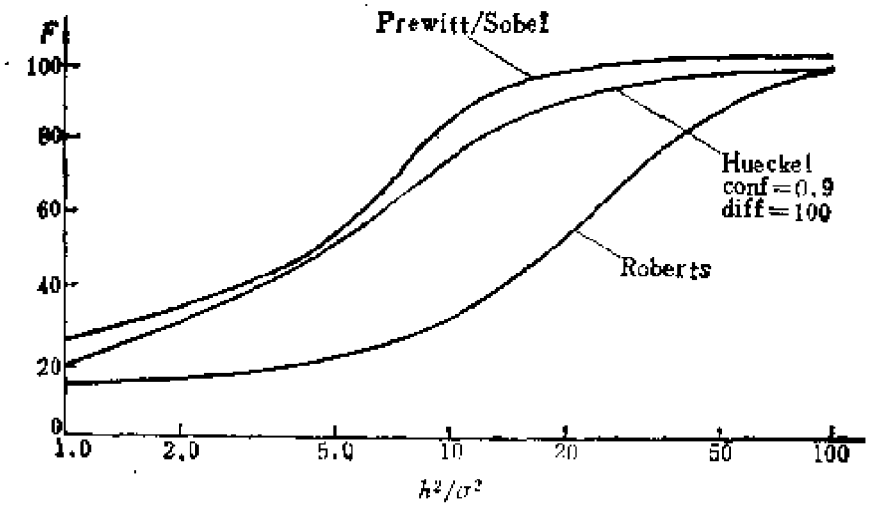


图 9.23　几种提取棱元素方法的品质曲线

图中σ^2是白噪声分布的方差，h是跃变型棱线两侧的灰度阶跃量，因此，h^2/σ^2表示信噪比.

Pratt所得的如上结果可供参考，正如我们在一开始指出的那样，选择什么样的方法应根据任务的要求而定.

最后要指出，人们一般不会低估提取棱线的重要性，但对提取棱线的困难往往认识不足. 由于实际图像中常存在相当强的噪声干扰，使得棱元素的提取不能得到令人满意的结果. D.H. Ballard和C. M. Broom指出，考虑到计算机视觉的任务是建立对物体的描述，最好是努力去寻求这样的方法，使得能从不怎么可靠的棱元素得到改善的结果，而不是去寻求某种理想的提取棱元素的方法.

五、彩色图像的棱元素

以上论述了提取单色图像的棱元素的几种主要的方法，以下说明怎样将它推广到彩色图像的情况.

对于彩色图像，每一个像素有三个灰度值，分别表示红、绿、蓝三种颜色的灰度，表为$R(x, y)$, $G(x, y)$, $B(x, y)$. R、G、B称为彩色图像的三个分量. 对于多光谱图像，分量的数目是任意的.

设用D来表示微分算子或模板匹配算子，用H表示阈值算子. 对于彩色图像可用如下方法提取棱线元素:

$$H(DR + DG + DB) \tag{9-60}$$

上式表示先对彩色图像的三个分量分别进行D处理，然后对它们的和进行阈值处理，输出是二值化的图像. 也可用下式所示的方法提取彩色图像的棱元素:

$$HDR + HDG + HDB \tag{9-61}$$

采用上式所示的方法，输出的图像是四值化图像，灰度值是0, 1, 2或3. 这个数值的大小表示了该像素在棱线上的可信度.

1976年，Nevatia把Hueckel方法推广到彩色图像的情况.

它的基本思想是，如果在D区域中存在棱线，那么，无论对于哪一个分量，在参数模型$E(x, y, c, s, \rho, d, b)$中，表示棱线方向和位置的三个参数$(c, s, \rho)$应当是一样的．因此，他提出，分别对三个分量$(R、G、B)$使用 Hueckel 方法计算出$(c_r, s_r, \rho_r)$,$(c_g, s_g, \rho_g)$,$(c_b, s_b, \rho_b)$，然后令

$$c = \frac{1}{3}(c_r + c_g + c_b)$$

$$s = \frac{1}{3}(s_r + s_g + s_b)$$

$$\rho = \frac{1}{3}(\rho_r + \rho_g + \rho_b)$$

用(c, s, ρ)表示彩色图像的棱线的位置和方向．模型中的其它参数则由使下式取极小值确定：

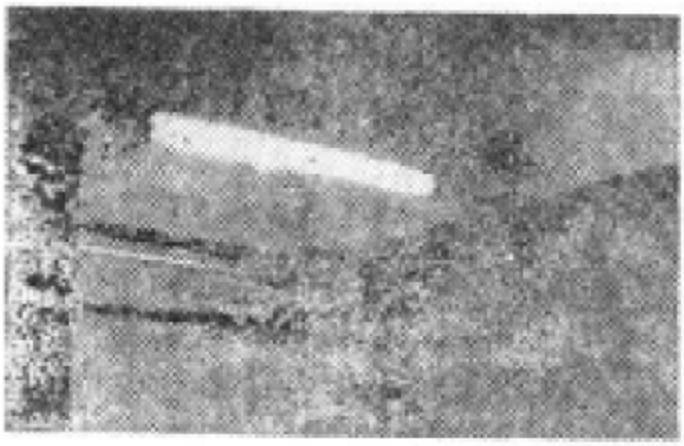
$$\Lambda = \Lambda_r + \Lambda_g + \Lambda_b \qquad (9\text{-}62)$$

其中Λ_r、Λ_g、Λ_b分别是三种灰度分布和模型间的希尔伯特距离，见式(9-43)．

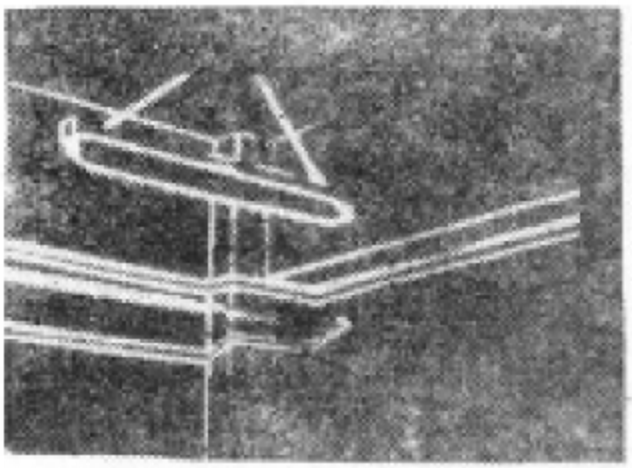
§9.5 棱线追踪

一、局部信息法

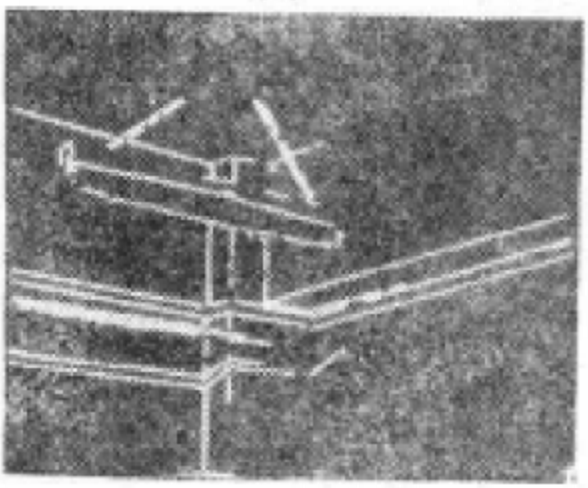
棱线追踪指的是把棱元素或局部的棱线段连接起来，得到较长的棱线或边界线．典型的方法是对棱元素图像(上节中的$G_s(i, j)$]进行从左到右，从上到下扫描，当发现某一像素是棱元素时，将它加以标记，并把它所在的位置加进表中．然后对它的八个相邻的像素进行检验，若发现其中一个是棱元素(未加标记)时，移到这个新发现的像素上来，并重复如上的过程，直到不能发现新的像素为止，这样在表中得到了一条棱线．程序恢复对棱元素图像进行扫描，由于对已追踪过的棱线已注上了标记，因此不会对同一条棱线进行多次追踪．在追踪的过程中，若发现八个相邻元素中有



(a)



(b)



(c)

图 9.24 (a) 灰度图像；(b) 用 Sobel 算子对 (a) 进行处理后的图像；(c) 实现跟线追踪后的图像。

多个是棱元素时，这表明棱线在这里可能有分叉，这时选择其中之一进行追踪，把其余的棱元素存入栈中，作为以后回溯追踪的起点。

在采用以上算法应当注意这样几个问题。(1)在以上算法中，实际假设了棱线的宽度是一个像素，因此进行处理的棱元素图像必须是经过细化的图像。(2)由于在原始图像中存在噪声，因此，提取出的棱元素中可能出现虚假的棱元素，也可能漏测一些棱元素，由于漏测将使棱线产生间断。为了解决棱线间断的问题，可使追踪过程具有一定的"惯性"，当遇到断点时，继续沿着已生成的棱线方向继续往前搜索一段距离，如果发现新的棱元素，追踪一段距离，如果在断点前后的棱线方向相同，则表明它们是同一条棱线，用内插的方法将两段棱线连接起来。对于虚假的棱线，可根据棱线的长度，棱线上各棱元素的方向信息的分散情况作出判断。

图 9.24 是用局部信息法实现棱线追踪的实例。

二、动态规划法

上面论述的追踪过程只使用了图像元素提供的局部信息，一种好的追踪方法还应当考虑全局信息和使用有关问题的知识。例如，对于积木世界的问题，利用线画的知识，可以正确地连接间断的棱线，添补遗漏的棱线。利用全局信息实现棱线跟踪的方法有动态规划法，图搜索法，Hough 变换法。本章第三节论述的提取边界的方法同时使用了动态规划和图搜索两种方法，它用动态规划法得到 D 域中的三条最优边界段，然后用启发式图搜索法把这些边界线段连接起来，构成完整的边界。

这里论述 Ballard 提出的动态规划法。他利用一种次优的算法产生局部的棱线段，然后用动态规划法求出由这些棱线段组成的最优棱线。

动态规划是解最优化问题的一种有效的快速算法，它适用于

当评价函数具有马尔科夫性质的情况。例如，求解如下的最优化问题：

$$\max_{x_i} h(x_1, x_2, x_3, x_4) \tag{9-63}$$

其中，每个变量 x_i 只能取有限个离散值，$h(\cdot)$ 是已知函数，解以上最优化问题的直接方法是穷举四个变量的所有可能的取值，并计算出相应的函数值，对这些函数值进行比较，最后求出最优解。如果有 N 个变量，每个变量有 K 个离散的可取值，穷举算法需要进行 K^N 次求函数值的计算。

如果函数 $h(\cdot)$ 的形式为

$$h(\cdot) = h_1(x_1, x_2) + h_2(x_2, x_3) + h_3(x_3, x_4) \tag{9-64}$$

则可采用分级优化的办法求 $h(\cdot)$ 的最优化解。我们称 x_n 的每一个可取值为第 n 级的状态，令

$$f_2(x_2) = \max_{x_1} h_1(x_1, x_2) \tag{9-65}$$

由式(9-65)，第 2 级的每一个状态在第一级中都有一个状态 x_1^* 与之对应。

由于在 h_2，h_3 中不存在变量 x_1，因此，可令

$$f_3(x_3) = \max_{x_2} [f_2(x_2) + h_2(x_2, x_3)] \tag{9-66}$$

同样，第三级中的每一个状态在第二级中有一个状态 x_2^* 与之对应。

类似地，令

$$f_4(x_4) = \max_{x_3} [f_3(x_3) + h_3(x_3, x_4)] \tag{9-67}$$

$$\max_{x_i} h(x_1, x_2, x_3, x_4) = \max_{x_4} f_4(x_4) \tag{9-68}$$

由式 (9-68) 求得 $h(\cdot)$ 的最优值，并由式 (9-68) 的最优解 x_4^*，反向递归求出各级的对应状态 x_3^*，x_2^*，x_1^*。$(x_1^*, x_2^*, x_3^*, x_4^*)$ 即是所求的最优解。

以上原理可以推广到任意级 (N 级) 的情况，递归算法为

$$f_1(x_1) = 0 \tag{9-69}$$

$$f_n(x_n) = \max_{x_{n-1}} [f_{n-1}(x_{n-1}) + h_{n-1}(x_{n-1}, x_n)] \tag{9-70}$$

$$\max_{x_i} h(x_1, x_2, \cdots, x_N) = \max_{x_N} f_N(x_N) \tag{9-71}$$

以上算法称为双自由端前向动态规划算法. 由于在这种分级优化算法中,每级只对一个变量求最优,因此,使得运算次数大大减少. 同样,设级数为 N,每级的状态数是 K,采用动态规划算法,求函数值的次数为 $(N-1)K^2+K$.

应当特别注意的是,只有当评价函数具有式(9-64)所示的形式时才可能使用动态规划算法. 因为只有这样才使第 n 级的状态的优化值只与第 $n-1$ 级状态的优化值有关(马尔科夫性质),从而才能把全局优化问题分解成分级优化问题来处理.

Ballard 使用的评价函数为

$$h(\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_N) = \sum_{n=1}^{N} S(\boldsymbol{X}_n) - \alpha \sum_{n=1}^{N-1} q(\boldsymbol{X}_n, \boldsymbol{X}_{n+1}) \tag{9-72}$$

其中 $\boldsymbol{X}_n$ 表示棱线上的像素 $\boldsymbol{X}_n = [x_n, y_n]^T$. $S(\boldsymbol{X}_n)$ 表示在 $\boldsymbol{X}_n$ 点图像灰度的变化强度(用 §9.4 的算子法求得), $q(\boldsymbol{X}_n, \boldsymbol{X}_{n+1})$ 代表棱元素 $\boldsymbol{X}_n$ 和 $\boldsymbol{X}_{n+1}$ 的方向的夹角.

$$q(\boldsymbol{X}_n, \boldsymbol{X}_{n+1}) = \mathrm{diff}[\phi(\boldsymbol{X}_n), \phi(\boldsymbol{X}_{n+1})] \tag{9-73}$$

$\alpha > 0$ 是加权系数. 约束条件为

$$S(\boldsymbol{X}_i) \geqslant T; \quad \|\boldsymbol{X}_n - \boldsymbol{X}_{n+1}\| \leqslant \sqrt{2}$$

按评价函数式(9-72),高强度低曲率为评价棱线优劣的准则. 由于评价函数式(9-72)具有式(9-64)所示的形式,因此,可以用如下递归算法求各级状态的评价值:

$$f_1(\boldsymbol{X}_1) = 0$$

$$f_n(\boldsymbol{X}_n) = \max_{x_{n-1}} [S(\boldsymbol{X}_{n-1}) - \alpha q(\boldsymbol{X}_{n-1}, \boldsymbol{X}_n) + f_{n-1}(\boldsymbol{X}_{n-1})] \tag{9-74}$$

采用如下优化算法产生从 X_m^t 出发的曲线段：

(1) 令 $n=1$，执行 $x_S:=x_m^t$；

(2) 只考虑满足 $S(x)\geqslant T$ 的像素，在 x_S 的 8 个相邻元素中选择满足条件 $\mathrm{diff}[\phi(x_S),\phi(x_{Si})]<\frac{\pi}{2}$ 的三个相邻像素（x_{Si} 是 x_S 的相邻像素）；

(3) 在这三个像素中选择 x_{Si}^*，使得递归公式 (9-74) 的左端取最大值，连接 x_R^t，x_{Si}^*，并执行 $x_S:=x_{Si}^*$；

(4) 如果 $m=N$，记录 f_N 并停止，否则，执行 $m:=m+1$，执行第二步。

给定一个起始像素 x_m^t 调用如上算法，得到一条长度为 N 从 x_m^t 到 x_m^h 的曲线段，以及关于这曲线段的评价函数值 f_m。x_m^t 称为这曲线段的尾，x_m^h 称为它的头。在 x_m^h 的八个相邻像素 $x_{m+1,i}^t$ 中，选择满足如下条件的三个前向像素：

$$S(x_{m+1,i}^t)\geqslant T \tag{9-75}$$

$$\mathrm{diff}[\phi(x_m^h),\phi(x_{m+1,i}^t)]<\frac{\pi}{2} \tag{9-76}$$

不失一般性，令这三个相邻像素为 $x_{m+1,i}$，$i=1,2,3$。以这三个

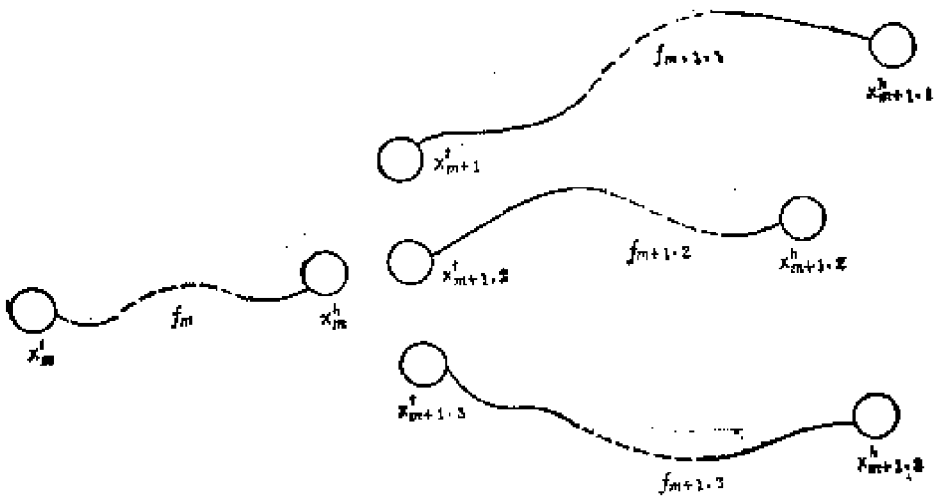
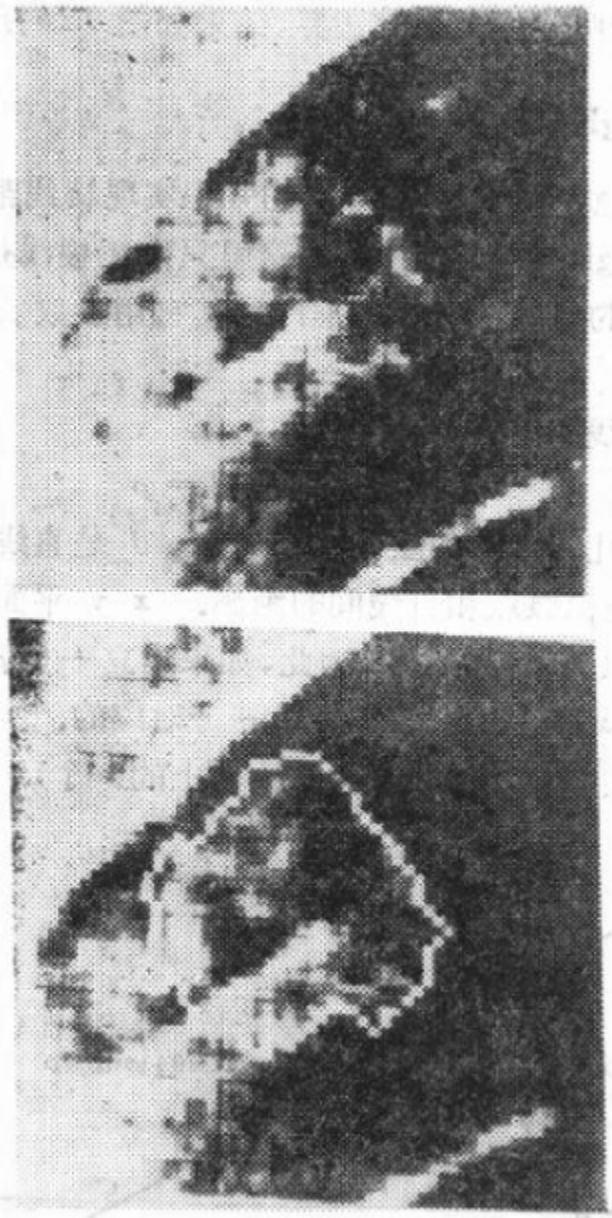


图 9.25 分段生成的曲线段及其评价值

像素为新的出发点，得到三条长度为N的曲线段，如图 9.25 所示。图中 $f_{m+1,i}$ 是新生成的三条曲线段的评价值。在这三条曲线段尾中，选择 $x^t_{m+1,i}$，使得

$$f_{m+1,i} - \beta q(x^h_m, x_{m+1,i}) \tag{9-77}$$

最大，并连接 x^h_m，和 $x^t_{m+1,i}$。重复以上过程，使棱线逐段向前延



(a)

(b)

图 9.26 (a) 经高频强调后的 X 光照片
(b) 提取出的肿瘤边界

伸．每段的长度一般为段数 m 的函数 $N(m)$．

从以上论述可以看到，Ballard 提出的这一方法并不是严格的动态规划法，它的准确度不如本章第三节论述的方法，但是算法较简单，运算次数较少．

Ballard 利用如上方法，并在肿瘤为圆形的先验知识的引导下，提取了X光照片中肿瘤的边界，结果如图 9.26 所示．

三、Hough 变换法

Hough 变换法利用棱线形状的知识实现棱线跟踪．为了简明地说明 Hough 变换法的原理，这里只论述棱线是直线的情况，但是，所说明的原理可以直接推广到其它用参数表示的曲线的情况．

设直线棱线的方程为

$$x\cos\theta + y\sin\theta = \rho \tag{9-78}$$

其中 θ 和 ρ 的几何含义如图 9.27 所示．θ 是直线的垂线与 x 轴的夹角，ρ 是坐标原点和直线间的距离．x-y 平面上的一条直线完全由其参数 (θ, ρ) 所代表，因此，如果建立一 θ-ρ 二维空间，这空间里的点和 x-y 平面上的直线是一一对应的．

相反，由式 (9-78) 可知，x-y 平面上的一个点 (x, y)，变换为

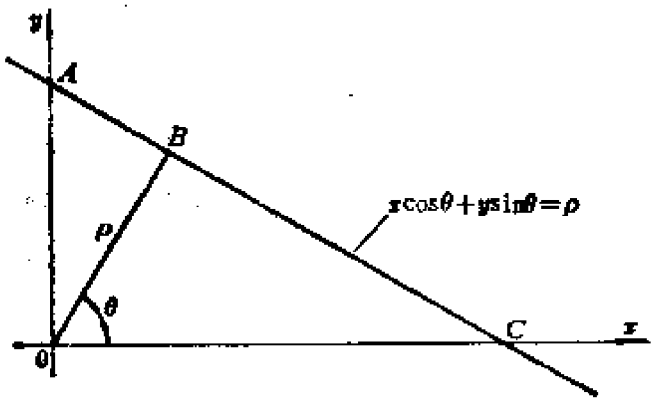


图 9.27　直线和其参数

θ-ρ 平面上的一段正弦曲线.

$$A(x, y)\cos(\theta - \theta_0(x, y)) = \rho \tag{9-79}$$

$$A(x, y) = (x^2 + y^2)^{\frac{1}{2}} \tag{9-80}$$

$$\theta_0(x, y) = \mathrm{tg}^{-1}\frac{y}{x} \tag{9-81}$$

正弦曲线上的每一个点代表在 x-y 平面上通过 (x, y) 点的一条一定斜率的直线. 图 9.28 上的三条正弦曲线和图 9.27 上的 A、B、C 三点对应,由于 A、B、C 三点在一条直线上,所以这三条正弦曲线相交于一点 (θ_l, ρ_l).

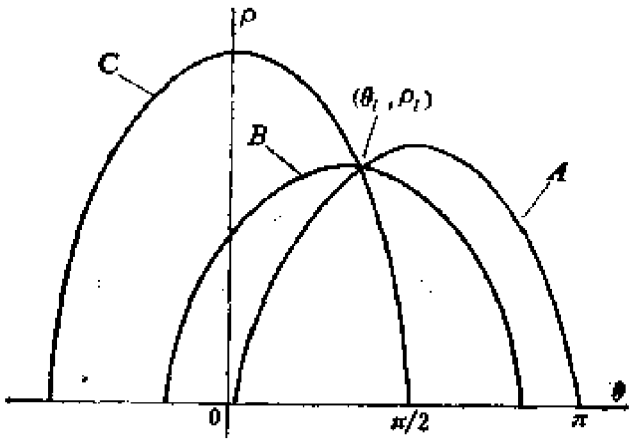


图 9.28 θ-ρ 平面上的正弦曲线

设有一组棱元素点 (x_i, y_i), $i = 1, 2, \cdots, N$, 把它们变换成 θ-ρ 平面上的正弦曲线,若一些正弦曲线相交于一点,则表明它们对应的棱元素点在一条直线上, 这条直线的参数就是正弦曲线交点处的 θ、ρ 坐标. 这就是用 Hough 变换探测直线的基本原理.

实际作法是,用陈列 (θ_m, ρ_m) 来表示 θ-ρ 平面. 对于 x-y 平面上的一个棱线点 (x_i, y_i), 由式 (9-80)、(9-81) 求出它在平面上对应的正弦曲线的参数,设棱元素点 (x_i, y_i) 的方位角为 θ_i [见式 (9-36)], 在 θ-ρ 平面上画正弦曲线段

$$\begin{cases} A(x_i, y_i)\cos(\theta - \theta_0(x_i, y_i)) = \rho \\ \theta_i - \Delta\theta \leqslant \theta \leqslant \theta_i + \Delta\theta \end{cases} \tag{9-82}$$

如果阵列点 (θ_K, ρ_K) 在(9-82)所示的正弦曲线段上，则将 $H(\theta_K, \rho_K)$ 加 1 [$H(\theta_m, \rho_m)$ 一开始被初始为零]。对于所有的棱线元素实行如上变换，得一幅新的图像 $H(i, j)$。由 $H(i, j)$ 的灰度值的分布，可得出 x-y 平面中的直线棱线(或近似的直线棱线)。

以上方法可以推广到别的可用参数表示棱线形状的情况。它

$$= \frac{1}{2} d_i L_i \tag{9-83}$$

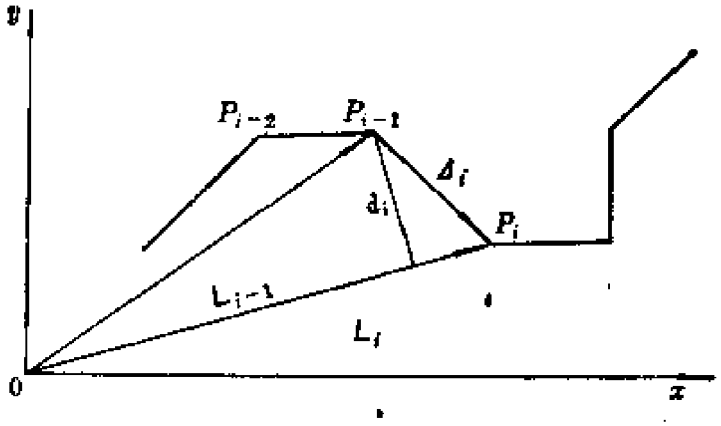


图 9.29　面积增量

其中 $\Delta_i = [\Delta x_i, \Delta y_i]^T$;

$$L_i = (x_i^2 + y_i^2)^{\frac{1}{2}}$$

因此，如图 9.30 所示，用算法

$$f_0 = 0$$

$$\Delta f_i = x_i \Delta y_i - y_i \Delta x_i$$

$$f_i = f_{i-1} + \Delta f_i$$

得到的 f_K 为直线 L_K 上面的面积和 L_K 下面的面积差的 2 倍，即：

$$S_K = 2(S_a + S_c - S_b)$$

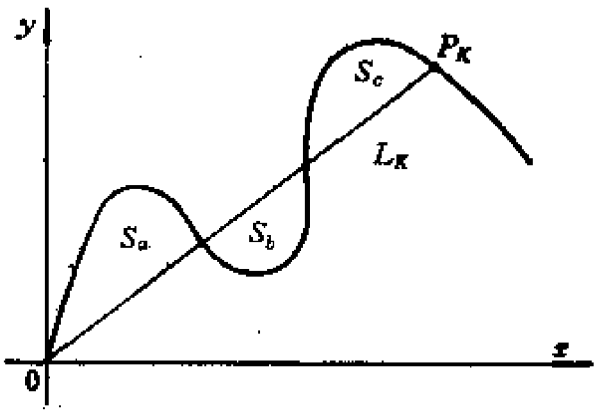


图 9.30　曲线和 L_K 包围的面积

由以上原理，得棱线的折线逼近算法为

$$f_0 = 0 \tag{9-84}$$

$$\Delta f_i = x_i \Delta y_i - y_i \Delta x_i \tag{9-85}$$

$$f_i = f_{i-1} + \Delta f_i \tag{9-86}$$

$$L_i = (x_i^2 + y_i^2)^{\frac{1}{2}} \tag{9-87}$$

当检验

$$|f_K| > T L_K \tag{9-88}$$

得到满足时，停止递归，输出点 P_{K-1}。然后把坐标原点移到 P_{K-1}，重复以上的运算和检验过程。其中 T 是一个控制参数，L_K 称为线段。检验条件式 (9-88) 表明，单位线段的面积差不超过规定的阈值 T 是以上逼近算法的准则。

如果给定的棱线是物体图像的边界线，则有

$$|S_a - S_p| \leqslant \frac{T}{2} \sum_{i=1}^{n} L_i \tag{9-89}$$

其中 S_p 是多边形的面积，S_a 是边界线所包围的实际面积，L_i 是多边形的第 i 条边的长度。

如果棱线是凸的[1]，则用折线逼近曲线的最大偏差 ε_m 为

$$\varepsilon_m L_K = |f_K| \leqslant T_K L_K$$

所以

$$\varepsilon_m \leqslant T_K \tag{9-90}$$

对于任意曲线，如图 9.31 所示，由于在递归运算中 Q、R 点不满足检验条件式 (9-88)，因此，有

$$\frac{T}{2} L_Q \geqslant S_a + S_b \tag{9-91}$$

$$\frac{T}{2} L_R \geqslant S_c - S_a \tag{9-92}$$

两式相加，得

(1) 指棱线上任意两点的连线不和棱线相交。

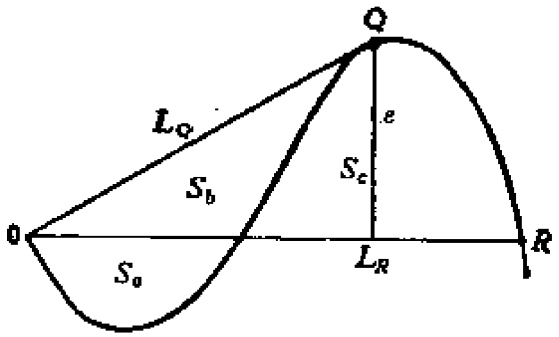


图 9.31　偏差 e 的估计

$$\frac{T}{2}(L_Q + L_R) \geqslant S_b + S_c \geqslant \frac{1}{2}\varepsilon L_R$$

所以

$$\varepsilon \leqslant (1 + L_Q/L_R)T \qquad (9\text{-}93)$$

如果在曲线中不存在窄的尖峰，一般有 $L_R > L_Q$，因此，由式 (9-93) 可作如下估计：

图 9.32　曲线的折线逼近实例
（不考虑窄尖峰的影响）

$$\varepsilon_m \leqslant 2T$$

由于以上逼近算法以单位线段的面积偏差不超过给定的阈值为准则，因此，对于曲线中窄的尖峰不敏感，从抗干扰的角度考虑，这种特性是如上逼近法的一个优点。如果需要考虑曲线中窄的尖峰的影响，可对算法作如下改进：如果在递归的过程中，L 持续减小，则测量当前点 P_K 和 Q 点（见图 9.31）间的距离 D，如果 $D > T$，则输出 P_{K-1} 点。

图 9.32 是用如上折线逼近算法的几个实例。图 9.33 是考虑曲线中的窄尖峰影响的两个逼近实例。

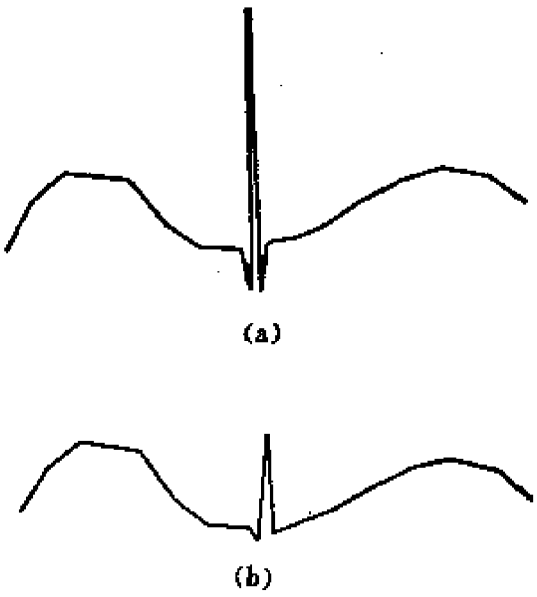


图 9.33 曲线的折线逼近
（考虑窄尖峰的影响）

二、B-Spline 仿样逼近

B-Spline 仿样逼近实质上是一种平滑和内插技术，它实现对折线的平滑和内插，如图 9.34 所示，图中 V_i 是折线的顶点，X_i 是 V_i 的对偶。

令：

$$C_0(t) = \frac{t^3}{6}$$

$$C_1(t) = \frac{-3t^3 + 3t^2 + 3t + 1}{6}$$

$$C_2(t) = \frac{3t^3 - 6t^2 + 4}{6}$$

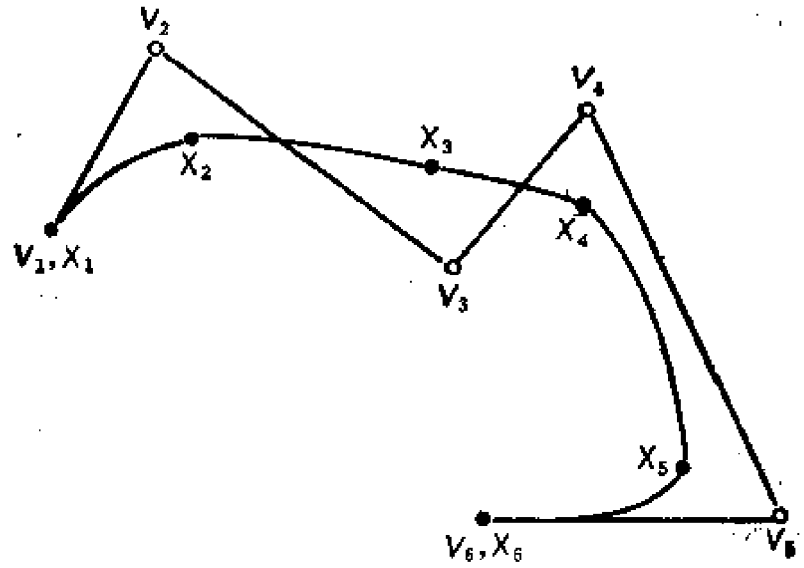


图 9.34 B-Spline

$$C_3(t) = \frac{-t^3 + 3t^2 - 3t + 1}{6}$$

$$t \in [0, 1]$$

以上四个函数满足

$$\sum_{i=0}^{3} C_i(t) = 1$$

的条件。

把以上 $C_i(t)$ 看成是平滑因子，$\boldsymbol{X}_i$、$\boldsymbol{X}_{i+1}$ 和它们两者之间的内插值为

$$\boldsymbol{X}_i(t) = C_3(t)\boldsymbol{V}_{i-1} + C_2(t)\boldsymbol{V}_i + C_1(t)\boldsymbol{V}_{i+1} + C_0(t)\boldsymbol{V}_{i+2} \tag{9-94}$$

$$t \in [0, 1]$$

$$\boldsymbol{X}_i = \boldsymbol{X}_i(0) \tag{9-95}$$

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i(1) \tag{9-96}$$

式 (9-94) 可以写成如下矩阵形式：

$$\boldsymbol{X}_i(t) = [t^3, t^2, t, 1]C[\boldsymbol{V}_{i-1}, \boldsymbol{V}_i, \boldsymbol{V}_{i+1}, \boldsymbol{V}_{i+2}]^T \tag{9-97}$$

其中，方阵 C 为

$$C=\begin{bmatrix}-1, & 3, & -3, & 1\\ 3, & -6, & 3, & 0\\ -3, & 0, & 3, & 0\\ 1, & 4, & 1, & 0\end{bmatrix} \tag{9-98}$$

设 $V_i, i=1,2,\cdots,n$ 是给定的 n 个顶点．两个附加的顶点 V_0 和 V_{n+1} 决定于折线的类型和对端点的要求．对于封闭折线，

$$V_0=V_n;\quad V_{n+1}=V_1 \tag{9-99}$$

对于非封闭折线，如果要求 $X_1=V_1$，$X_n=V_n$，而且端点处的曲率为零，则

$$V_0=2V_1-V_2 \tag{9-100}$$

$$V_{n+1}=2V_n-V_{n-1} \tag{9-101}$$

因此，V_i 和 X_i 间的关系可以表示成如下矩阵形式：

对于封闭折线

$$6\begin{bmatrix}X_0\\X_1\\\vdots\\X_n\\X_{n+1}\end{bmatrix}=\begin{bmatrix}4, & 1 & & & & & 1\\ 1, & 4, & 1 & & & & \\ & & \ddots & & & & \\ & & & 1, & 4, & 1 & \\ 1, & & & & 1, & 4 & \end{bmatrix}\begin{bmatrix}V_0\\V_1\\\vdots\\V_{n-1}\\V_n\end{bmatrix} \tag{9-102}$$

对于非封闭折线

$$6\begin{bmatrix}X_0\\X_1\\\vdots\\X_n\\X_{n+1}\end{bmatrix}=\begin{bmatrix}6, & 0 & & & & & \\ 1, & 4, & 1 & & & & \\ & & \ddots & & & & \\ & & & 1, & 4, & 1 & \\ & & & & 0, & 6 & \end{bmatrix}\begin{bmatrix}V_0\\V_1\\\vdots\\V_{n-1}\\V_n\end{bmatrix} \tag{9-103}$$

将参数 t 离散化，设 t 的采样间隔为 Δt，则内插的抽样点为

$$X_i(K\Delta t)=[(K\Delta t)^3,(K\Delta t)^2,K\Delta t,1]C[V_{i-1},V_i,V_{i+1},V_{i+2}]^T \tag{9-104}$$

可以证明，上式可以分解成

$$\boldsymbol{X}_i(K\Delta t)=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}^T\begin{bmatrix}1&&&\\1,&1&&\\1,&1,&1&\\1,&1,&1,&1\end{bmatrix}^K\begin{bmatrix}6&&&\\-6,&2&&\\1,&-1,&1&\\0,&0,&0,&1\end{bmatrix}\times\begin{bmatrix}\Delta t^3\\\Delta t^2\\\Delta t\\1\end{bmatrix}C\begin{bmatrix}\boldsymbol{V}_{i-1}\\\boldsymbol{V}_i\\\boldsymbol{V}_{i+1}\\\boldsymbol{V}_{i+2}\end{bmatrix}\tag{9-105}$$

曲线的导数为

$$\boldsymbol{X}_i(K\Delta t)=\begin{bmatrix}0\\0\\0\end{bmatrix}^T\begin{bmatrix}1&&\\1,&1&\\1,&1,&1\end{bmatrix}^K\begin{bmatrix}2,&0,&0\\-1,&1,&0\\0,&0,&1\end{bmatrix}\begin{bmatrix}3\Delta t^2\\2\Delta t\\1\end{bmatrix}\times\begin{bmatrix}-1,&3,&-3,&1\\3,&-6,&3,&0\\-3,&0,&3,&0\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_{i-1}\\\boldsymbol{V}_i\\\boldsymbol{V}_{i+1}\\\boldsymbol{V}_{i+2}\end{bmatrix}\tag{9-106}$$

以上论述的是三次 B-Spline。不同阶次的 B-Spline 基本原理是一样的，只不过基函数 $C_i(t)$ 不一样而已。由于三次多项式是曲率可以改变符号的最低次多项式，所以三阶 B-Spline 应用最广泛。

§9.7 图像分割

图像分割又称为区域生成 (Region Growing)，指的是把图像划分成若干个区域(即若干个像素集)，使得同一区域里的像素具有一致的特征。这样作的根据是，对于颜色、反射特性、文理结构等特征，物体的同一表面的各微小部分特性相同。因此，以像素的

特征一致为准则，把图像划分成区域，使得相邻区域的特征有明显的差异，势必导致每一区域和景物中的一定的物体或某一表面对应.

从以上论述可以看到，图像分割实质上是聚群问题，即把相邻的具有相同特征的元素聚集为群. 它之所以引起人们的普遍关注，主要是图像分割之后，一幅图像可以用少数几个区域来表示，从这个意义上讲，这是一种数据压缩；此外，只有在经过分割之后，才能实现物体识别和图像理解.

提取边界无疑是一种强有力的图像分割方法，常常被用来把物体的图像从背景中分割出来. 关于这方面的问题，前几节已作了详细的论述. 下面将主要从聚群的角度论述图像分割的问题.

和一般的聚群法一样，存在着两种图像分割方法，分别称为凝聚法（Agglomerative）和分裂法（Divisive).

凝聚法是一种自下而上的方法，它从许许多多微小的原子区域（Atomic Region）开始. 原子区域可以是单个的像素，或者是相邻的具有相同灰度或相同颜色的像素的集合. 凝聚法首先要确定两区域的一致性度量（例如用两区域的灰度均值的差作为区域间的一致性度量），然后对相邻区域进行检验，结果一致，将其合并为一个区域. 重复这样的合并过程，直到不存在特征一致的相邻区域为止.

存在着两种通行的凝聚算法. 一种算法从左上角的原子区域开始，检验这个原子区域和它的相邻原子区域，如果一致，将其合并；然后检验这个新生成的区域及与它相邻的原子区域，如果一致，将其合并. 重复以上的过程，直到不存在相邻的原子区域可合并为止. 这时才转到右边的一个原子区域，以这个原子区域为核，重复以上的凝聚过程. 这是一种串行算法. 另外一种算法是并行算法，它计算出所有相邻区域的合并价值，然后选择两个最有合并价值的区域进行合并，称为第一级合并. 在进行第二级合并时，计

算出上级合并成的区域和其邻域的合并价值，然后选择所有的邻域对中合并价值最大的那对邻域进行合并.逐级重复以上的过程，直到将所有的有合并价值的邻域对合并完为止.

实践中用得更多的是分裂法. 分裂法是一种自上而下法，一开始，它把全幅图像看成是一个区域，并根据某种原则将其分成几个区域，这称为第一级划分. 第二级划分是对上一级区域再进行划分. 重复以上的过程，直到达到进一步划分已无意义为止.

判断一个区域是否需要进行划分，以及怎样进行划分的通常办法是灰度直方图法. 灰度直方图的横坐标是图像的灰度级，纵坐标是灰度为某一灰度级的像素的数目或出现率(像素数目被图像的全部像素数目除). 如果灰度直方图中出现明显的尖峰，表明相当一部分像素的灰度聚集在峰值附近的一个小区间里，即表明这些像素的灰度一致. 因此，我们把灰度直方图的谷点灰度作为灰度阈值，按像素灰度的大小把图像分成几个区域. 在对区域进行检验和划分时，构造区域的灰度直方图，并按同样的办法对区域进行再划分. 直到每一个区域的灰度直方图无明显的尖峰为止.

以上方法很容易推广到多光谱扫描图像和彩色图像. 这时，图像函数是一函数向量，因此，对一个区域进行划分使用多直方图法. 对函数向量的每一个分量构造直方图，在这多个直方图中，选择尖峰明显而且峰间距离较大的一个直方图，确定阈值，并用这个直方图对应的光谱灰度对区域进行划分. 重复这一过程，直到每一区域的所有直方图中都不存在明显的尖峰为止.

第十章　工业视觉系统

§10.1　概述

通过前面三章的介绍，我们已经知道，计算机视觉的总目标是要把从电视摄像机等传感器输进来的图像信息，用计算机进行分析，求出周围物体在三维空间内的形状、大小、位置等理解性描述。可是对于这些研究，由于注意通用性、一般性，因而处理方式相当复杂，即使是大型计算机也要花相当长的计算时间，因而无论从经济观点还是从所花的时间来看，都不能原原本本地照搬到生产线上去。但是在具体的工业应用上，即使是机器人应该具备的视觉系统，也往往只要求专用性、实用性，如果我们把照明条件、背景，以及拍摄物体的角度适当地加以控制的话，就会使问题简单化。

这一章所要介绍的工业视觉系统，就是一种简化了的计算机视觉系统，它着眼于二维图像的处理，注重视觉原理的工程实现技术，并考虑到实际系统的应用效率。实际上可以说，工业视觉系统的研制和应用过程就是计算机视觉研究赖以发展的实践基础，而一般所说的机器人视觉，大多数即指与机器人配合操作的工业视觉系统。

工业视觉系统的发展情况如表 10.1 所示。其中，区分了当时的试制系统(以"▲"标注)和已成为产品的实用系统(以"●"标注)，另外，成果发表的顺序按年代时间编排。

表 10.1　工业视觉系统的发展

	1970 年	1975 年	1980 年
检查	●印刷电路板伤残的检查	●印刷电路板，IC及其掩膜图形的自动检查 ●混合集成电路的安装检查	●药片，胶囊的检查 ●食品(鱼、黄瓜)等级的标记 ▲发动机阀门的安装检查
定位	▲汽车的车轮嵌进车轮箍	●晶体管装配 ●IC 装配 ●IC 连线自动化	●泵与管的安装 ▲焊接的位置确定 ▲钢板的运输
识别	▲智能机器人的研究　▲ PIPS　▲微处理机 ▲图像理解	▲简单的机械零件识别 ▲多种的机械零件识别 ▲重叠机械零件的识别	●机械人普及 ●微型机普及

视觉系统在工业应用中的尝试，从 70 年代初期就开始了。美国的 GM（General Motor）公司研究了把车轮嵌进轮箍这样一个作业自动化系统，它把从电视摄像机输入的车轮图像进行处理，找到在同心圆上分布的螺栓孔，用机械手使其位置重合，从而在轮箍上进行装配。这样复杂的处理是由计算机来完成的，而且机器人手臂动作也很费时间，因而不适合实际应用。同一时期，日本目漱等人进行了用视觉自动检查印刷电路板伤痕的研究，并试制了能够实用的装置，这在当时被认为是计算机视觉研究领域中的划时代事件。其成功的原因主要在于利用了印刷电路板的导线和其它部分本来就可以用二值表示的固有特点，一开始就把图像二值化，这样就简化了信息处理工作；另外，又采用了专用的二值化处理硬件，提供了实时性。后来，在日本又陆续研制成功了集成电路（IC）及其掩膜图形等更加复杂的电路图像的目视检查自动化，其中虽然出现了一些新的方法，但上面所说的两点仍然是关键所在。

到了70年代后半期，随着 LSI 技术的迅速发展，存储器、运算单元等的性能价格比有所提高，用它们所制成的实用视觉系统也多了起来。首先应用到晶体管 IC、LSI 等半导体器件的装配线上，通过显微镜，把对象物体用电视摄像机拍摄下来，再把它们二值化，用专门的硬件进行图像处理。为了使它们具有一定程度的适应性，使用了电子计算机，此外，为了提高系统的性能价格比，把电视摄像机、专用硬件，电子计算机作成层次结构，让一台计算机接收几台电视摄像机的输入，以提高电子计算机的效率。LSI 等的装配工序曾经是一种劳动密集的操作，又要使用显微镜，操作人员过度疲劳，成品率受到影响。显然，目测检查的自动化，有助于半导体产品的成批生产和质量的提高，反之，LSI 的发展又促进了工业视觉系统的实用化。在此期间，从表 10.1 中还可看出，诸如泵的给排水软管的安装、螺栓的联结、及部件的检查等多种视觉系统也已经得到了应用。

在美国，比起计算机视觉的基础研究，实用化系统的发展比较迟缓. GM 公司在70年代后半期开始研制了一个检查汽车发动机点火用的 IC 芯片是否安装正常的系统. 它用电视摄像机输入浓淡图像，但因为用微处理机来进行处理，速度受影响，差不多要化 1s 左右才能完成. GM 公司后来又试制了识别传送带上的机械零件，并将它们进行分类的系统.

到了 80 年代，由于微处理机的普及，视觉系统的性能价格比大大提高，视觉系统进入了真正的实用期. 视觉系统的研究、试制以往是以计算机制造厂为中心进行的，但随着微处理机的普及，其它各类企业对视觉系统的研究也热心起来. 在美国，以它作为提高生产率的一根支柱. 以 NST (美国国家科学基金委员会)为中心，实用的视觉系统的研制也在飞速地发展. 此外，美国通用电气公司、西门子、法其尔特、罗克特等公司的制造厂商和 MIT、斯坦斯福大学、SRI、卡内基梅龙大学、贝尔研究所等研究机关也都对视觉系统进行了研究. 在英、西德、法、瑞士、荷兰等欧洲国家也开始了这方面的研究工作. 其中，引人注目的是通用性视觉系统的研究和开发. 以前往往是针对孤立的应用场合研制系统，使得试制成本很高，而通用性系统的发展为工业视觉系统开阔了实用前景.

§10.2 工业视觉系统的功能和性能要求

一、视觉系统的功能

在生产线上的目视检查和装配自动化方面的视觉系统虽有各种各样，但其作用大致可分为如下三种类型：

(1) 分类，指对被操作零件的识别，即按照预先知道的零件标准，将零件盒中或传送带上的混杂在一起的零件进行区别分类.

(2) 定位，指对被操作零件的位置、姿态的测定，以便为装配

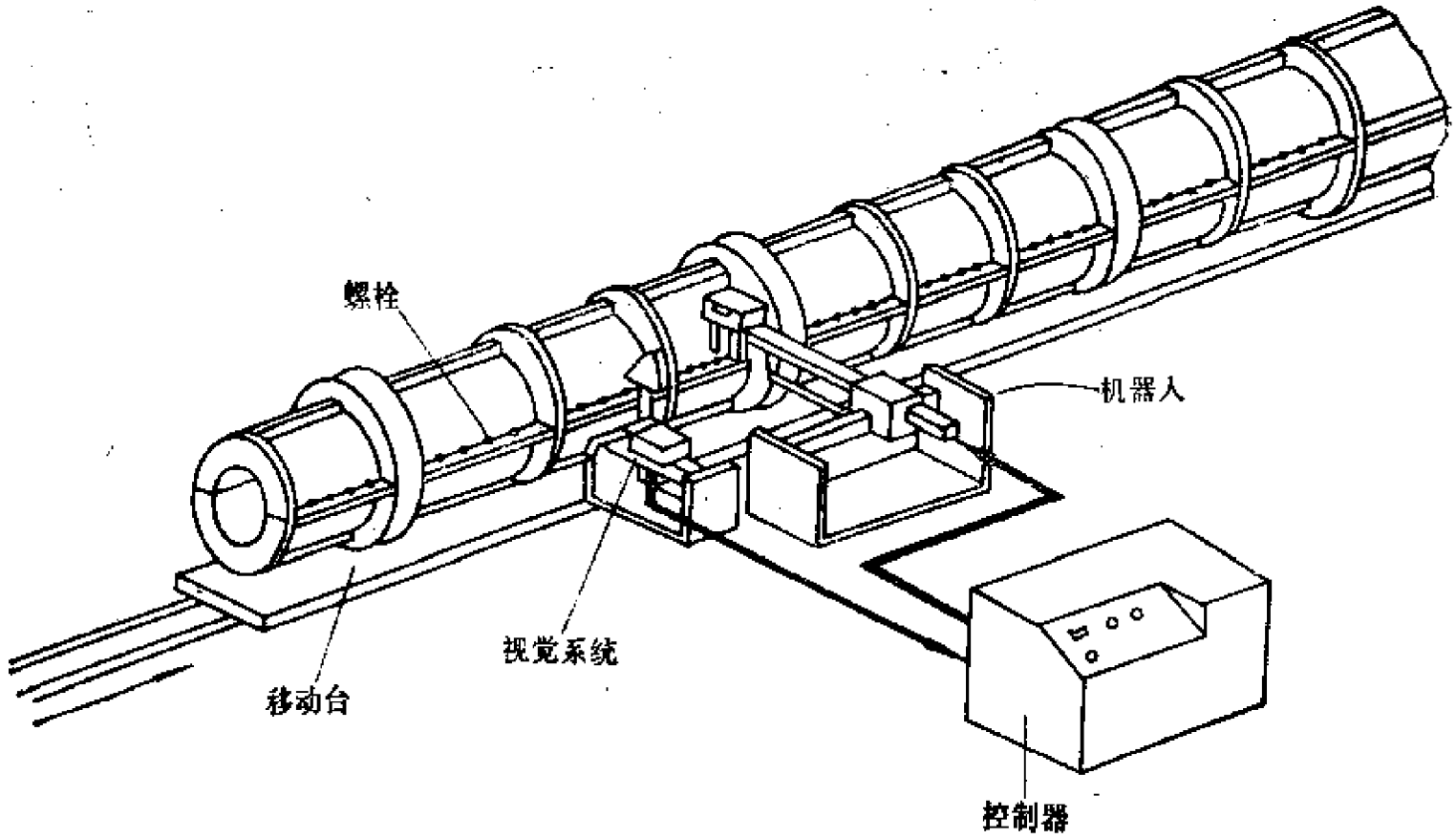


图 10.1　在混凝土柱上固定螺栓

表 10.2　视觉系统功用

功用		应用实例	图像传感器	图像处理方法	
				二值化	处理方法
检查	多幅图像	印刷电路板 IC 掩膜板	线性传感器，光导摄像管 激光析像管	浮动阈值	局部特征提取，利用模型特征提取，激光衍射
	单体	药品的胶囊，片剂	电视摄像机	其他	属性测定
	分等级	食品(鱼、黄瓜等)	CCD 线性传感器	其他	属性测定
	半成品	发动机阀门，混合集成电路 IC 的安装	CCD 线性传感器		边缘检出
定位	联线	晶体管，IC 的引线联接	光导摄像管，CCD 线性传感器	浮动阈值 其他	模板对照投影法
	组装	拧螺钉，导线的安装	同上		模板对照
	焊接	电弧焊的信号决定	同上	其他	属性测定，边缘检出
	装卸、搬运	钢板卷搬运	光导摄像管		边缘检出
分类	机械零件	机械零件的分类	光导摄像管 CCD 线性传感器	适应	边缘检出 窄条光

等操作提供必要的信息。图10.1给出了在混凝土柱上固定螺栓的例子，首先用视觉系统测出螺栓的位置，把这个信息送给机器人，然后用机械手来固紧螺栓。又如晶体管集成电路的装配，泵软管的安装、电弧焊接等都要首先用视觉系统来定位。

（3）检查，即用视觉系统代替人来进行目视检查。例如查找印刷电路板和掩膜板的伤痕，药片和胶囊的检查，标记食品的等级，及成品装配和半成品的检验，都可用视觉系统来完成。

表 10.2 说明了工业视觉系统的三种功能及有关情况。

二、工业视觉系统的性能

实用的工业视觉系统必须满足实际生产线所要求的几个条件。它们除了要求价格低廉外，还要求高速、可靠、具有一定的通用性等。为了满足这些要求，在工业视觉系统中作了以下种种努力。

（1）环境安排。视觉系统的环境易于观察，就能保证得到易于处理的图像。例如，如果从倾斜方向去观察三维空间的物体时，随着观察的角度不同，其形状就会有明显的差异。这会使后面的处理很费事。为此，在工业视觉系统中，几乎都是从垂直方向来观察对象，这样会使对象物体二维图像的处理简化。而且在实际应用中，往往一开始就把输入的图像作二值化处理。所以摄像时，要精心安排照明光源的位置、方向或背景，使物体和背景有足够的亮度反差，容易相互区别。安排一个易于处理的环境，不仅会使设备的成本下降，而且对下面所要说的高速性和可靠性也是很重要的。

（2）实时性。与流水作业类似，摄像装置所要观察的对象也是一个一个地出现的，机器人或其他自动化设备常常要等待视觉系统的定位和检查结果来实时地决定动作。所以视觉系统一般都把输入图像二值化，使处理简单，同时，还适当考虑处理方式的硬件化，或使用专用处理器，另外，系统只考虑必要的分辨能力，研究

处理必要的部分，尽量缩小处理范围。这些都是实现高速性的必要措施。

(3) 高可靠性。工业视觉系统应当比操作工人的肉眼具有更高的可靠性。因此，很重要的一点就是从观察对象的许多特征中，选择最能可靠地判别对象的特征作为识别特征，而且，在系统的设计过程中要考虑这样的识别方法：即使部分特征提取失败了，也不影响最后结果。

(4) 通用性。小批量或中批量生产的特点就是产品经常变更，视觉系统要能适应这种变更，具有一定的通用性。为此，可以把产品的模型，以人-机对话的形式送给计算机，把这些信息以文件方式传送给各个作业位置的摄像装置，也可利用 CAD 产生的各个产品的信息来作为对象的模型。在此基础上进行图像处理。

三、工业视觉系统的硬件组成

工业视觉系统的硬件主要有摄像装置即图像传感器、图像处理器和电子计算机。它们的组成方式如图 10.2 所示，(a) 和 (b) 针对单台图像传感器，(c) 是用一台计算机接收几台图像传感器的输入，这种群控方式提高了计算机的使用效率。

图像传感器通常使用由光导摄像管或卡尔诺摄象管等制作的电视摄像机。后来出现了所谓 CCD 和 MOS 固体图像传感器。与过去的摄像管相比，它们体积小、重量轻，便于安装在机器人的手指上，它还具有图像残像特征小、空间畸变小的优点。所以在近来的视觉系统中，往往采用固体图像传感器。在固体图像传感器中，有一维线性传感器和二维面传感器之分。因为线性传感器可以具有 2000 个以上的像素，所以常常使用在需要高分辨率的场合。面式传感器的分辨率也在不断提高，当前市场上已经陆续推出了各种像素与标准电视摄像机相同的商品。

如果用微型机来处理图像，要高速地进行三维微分、模型对照

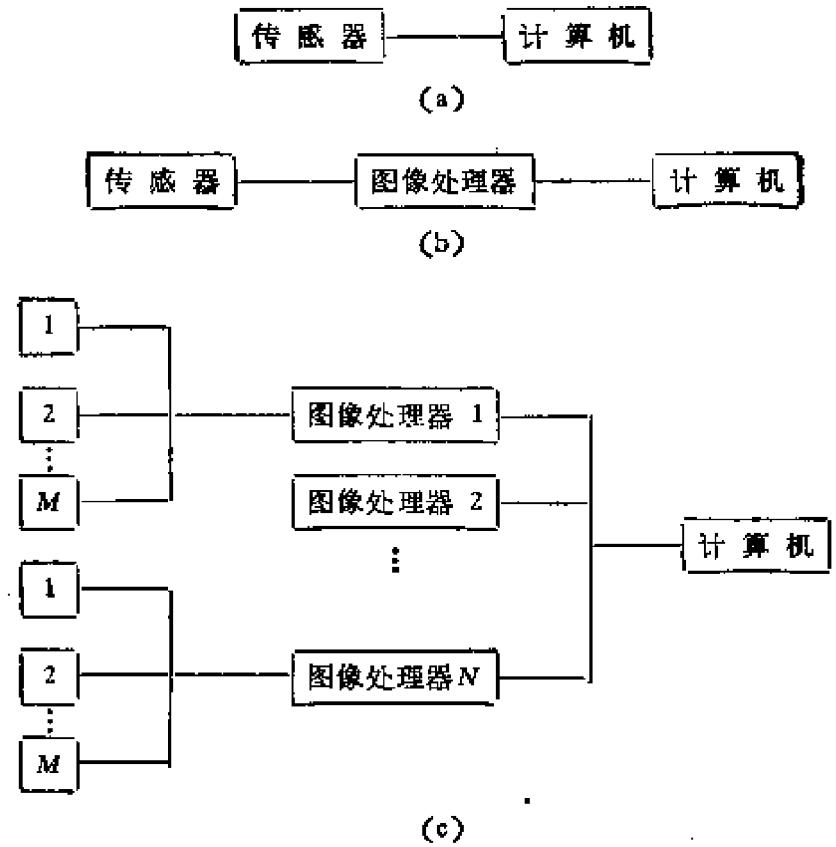


图 10.2　工业视觉系统的硬件组成

往往就不适合了，所以必须有专门进行这些处理的图像处理器．在三维微分过程中，一般先对图像中的局部区域进行处理，然后再对整个图像进行这种处理．这种串行处理的方式极费时间．因此最好有一个高速的局部并行图像处理器，如图 10.3 所示．

图中表示了自动查找印刷板伤痕的处理方法．首先把来自电视摄像机的信号二值化，送到移位寄存器进行存储，然后把 7×7 个像素的局部区域的图像信号进行并行采样，再用简单的逻辑电路对它进行局部处理，用各点的周围点的值（“0”或“1”）的逻辑和来把摄像图形加以放大，又按其逻辑加以缩小（在后面详述）．因为这些工作是在电视摄像管的扫描时间内结束，所以实时处理是可能的．

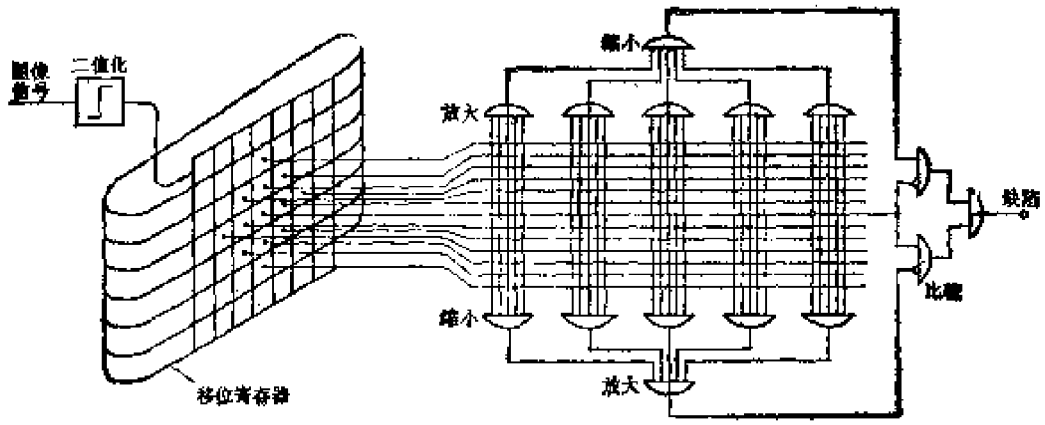


图 10.3　并行图像处理器

四、二值图像与灰度图像

工业视觉系统的输入图像可以是二值图像，也可以是灰度图像，表 10.3 给出了使用这两种图像的对比。在计算机处理之前，一般把图像从 64 × 64 分割成 512 × 512 个点的像素，各像素点的亮度用“数”来表示。

表 10.3　二值图像与灰度图像的比较

	二值图像	灰度图像
亮　　度	“0”或“1”	16～64 级
数字化硬件	比 较 器	A/D变换器
存 储 器	少	大
运 算 器	少	大
性　　能	需要有良好的反差	通用

在二值图像中，各点的亮度被分为或“黑”或“白”，再变换成数字“0”或“1”，而在灰度图像中，则把亮度变换成 64 个不同级别的值。为了得到二值图像，用比较器就足够了，而在灰度图像中，则要有 A/D 变换器。显然，后者所需的存储容量和计算量要比前者多得多。

另一方面，从性能要求来说，二值图像必须有良好的对比度，而灰度图像系统就没有这个限制了。不过，二值图像处理所需的硬件量少，花费的时间也少，加上图像处理的专用硬件化也容易，所以大多数系统都使用二值化图像。但因硬件的性能价格比逐年提高，估计今后使用更通用的灰度图像的系统会有所增加。

五、二值化方法

为了得到二值图像，要对来自电视摄像机的信号进行阈值处理，其中可分为固定阈值法和适应型阈值法两种。把阈值保持在一个固定值上，很少会得到满意的结果，这是因为照明等原因，背景和物体的亮度一般并不均匀，而是随场地而变动的。为此产生了将阈值作相应改变的适应型阈值法。如图 10.4 所示，适应型阈值法先把图像分割成几个小区域，对每个区域作亮度直方图，直方图上频度高峰相应于物体，而频度低峰相应于背景，这样，如果峰间的谷上设定阈值，就可以把物体和背景区分开来。

适应型阈值法需要花费很多的计算时间，因而常采用一种如

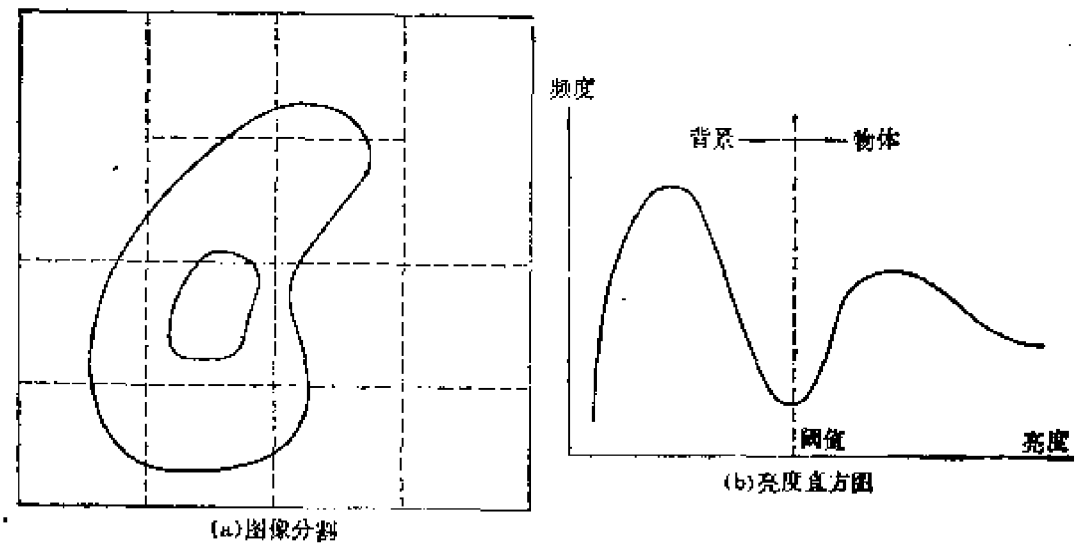


图 10.4　适应型阈值法

图 10.5(a) 所示的浮动式阈值电路，图中 (b) 和 (c) 说明，由于阈值可以上下变动，从而能够检测出固定阈值法所不能检测的信号.

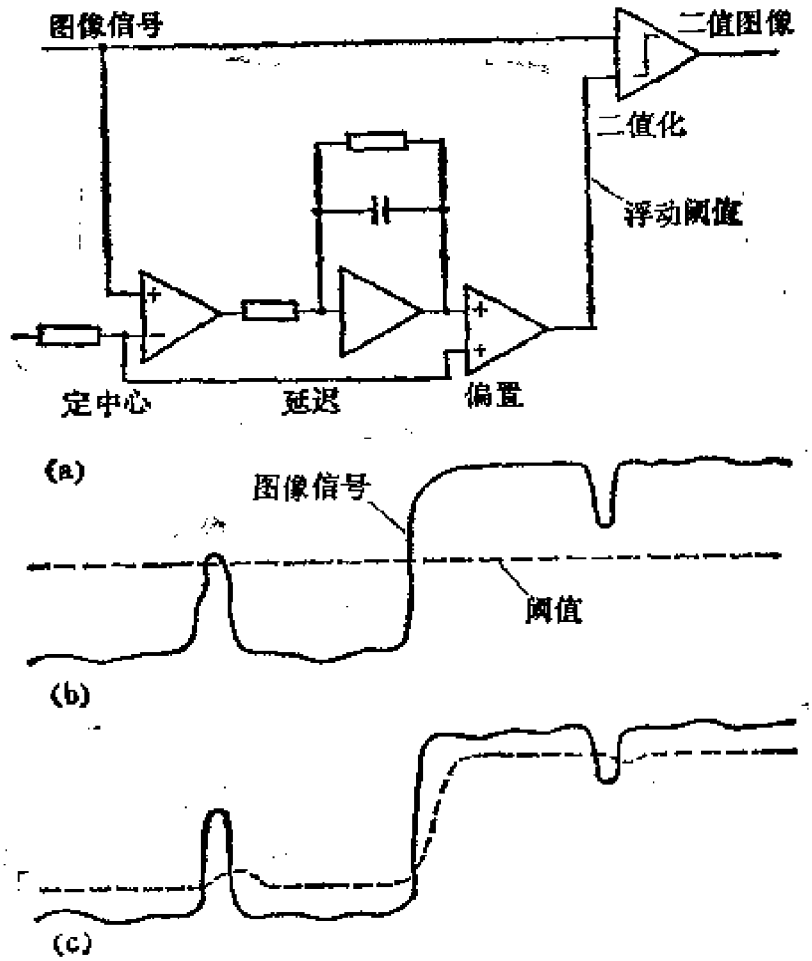


图 10.5　浮动型阈值法

六、边缘信息的提取

对于难以二值化的图像，还是采用灰度图像表示法好. 在物体的边界或棱等具有特征的地方(即边缘)，亮度急剧改变，因此对图像中的各点进行空间微分，检测出亮度变化的边缘，作为灰度图像的基础，计算机把空间微分作为差分计算. 距离和方向的差值的取法有种种办法. 在工业视觉系统中，一般采用 sobol 算法.例

如在图 10.6 中，A、B、$\cdots$、I 是 E 点邻接点的亮度，U、V 分别表示为水平和垂直向上的亮度变化，微分值可由 W 求得，而其方向可由 α 求得。

A	B	C
D	E	F
G	H	I

$$U = (C + 2F + I - A - 2D - G)/8$$
$$V = (A + 2B + C - G - 2H - I)/8$$
$$W = |U| + |V|$$
$$\alpha = \tan^{-1}(V/U)$$

图 10.6　空间微分的 sobel 算法

§ 10.3　工业视觉系统实例

一、图像目视检查自动化系统

目前，最先进的目视检查自动化技术可能主要应用于印刷电路板的制造工艺中，即集成电路及其掩膜图形的伤残检查，由于电路的集成化程度越来越高，对质量检查的要求也就越来越高，因而强烈地要求能自动化；另外，这种电路图像几乎近于理想的二值图像，检测也比较容易。下面引用一些已经实际应用的视觉系统实例，介绍有关的目视检查自动化技巧。

1. 局部特征的利用

复杂的电路图形通常也是由一般的直线和圆等规则图形所构成的，可是，如图 10.7 所示，在有缺陷的地方将出现不规则的形状。因此，利用检查电路形状的有关知识，来确定正常图像所应具有的局部特征，再把输入的图像与之比较，不一致的地方就可认为是检测出来的伤残之处。例如，有一种视觉系统（日本江尼等人研制）通过把印刷电路板的伤残与导线图像相比较得出更细微的图形，再采用放大、缩小的方法探测出伤残。图 10.8 给出了它的原理

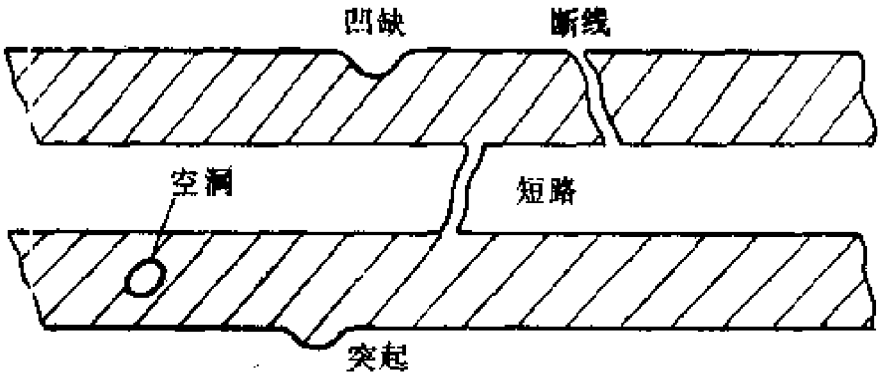


图 10.7　印刷电路板的缺陷

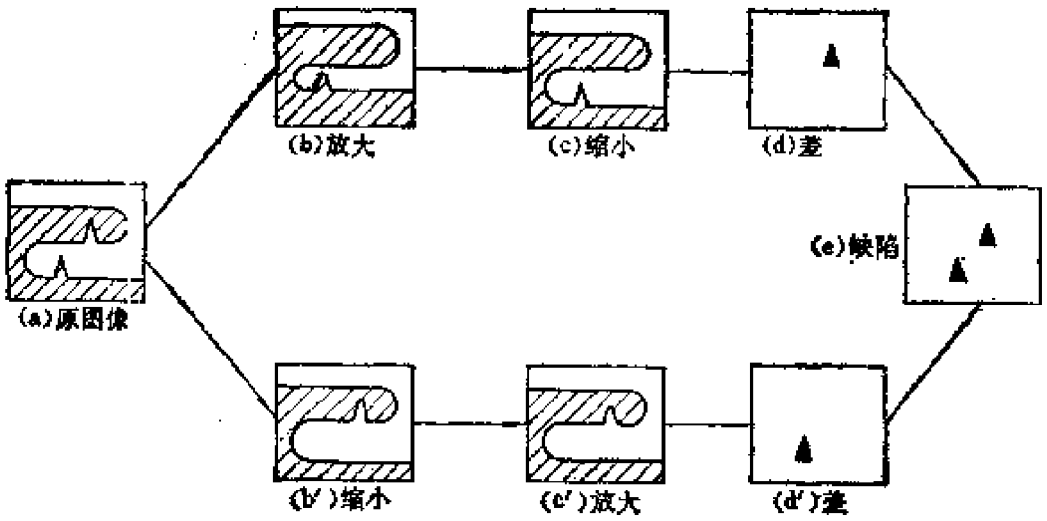


图 10.8　印刷电路板的伤残检测

图，在输入的二值图像中，把相当于导线的值为“1”的部分放大(图中的(b))，然后又等量地缩小，从而得到微小的凹陷部分(被覆盖的区中的(c))，把它与原像比较，可以探测出凹陷的微小部分．以同样的方法可以得到凸起的微小图像(d′)，把(d)与(d′)相综合就检测到全部伤残．因为这种方法容易硬件化，采用前面图 10.3 所示的图像处理器，在电视摄像机一帧的扫描时间内就可把伤残探测出来．

用上述方法构成检测装置，由于电视摄像机的分辨率有限，不可能一次检查整个印刷电路板，只能把电视摄像机的视野限制于某个局部，这样，就要求印刷电路板在 $X-Y$ 方向逐段移动，才能完成全面检测。为了抑制电视摄像机的残像，要使闪光灯和摄像机同步工作。据称，这种系统对印刷电路板伤残的识别能力可与人工方法相比。

还有的系统（Jarvis 研制）在利用局部特征的同时，采用了另一种伤残检测方法。系统预先对正常印刷电路的所有图像一一列表，在检测时，与标准列表相对照，把不在表上的作为伤残。由于伤残总是发生在导线条与基板的边界，所以首先通过简单的处理找出这些边界，即，在 3 × 3 像素的小区域内，正中间的点是导线，在该点的水平与垂直方向上的邻近点若是代表背景基板的点，中间点就作为边界点，针对这个边界上的点，检查靠近它的 5 × 5 象素上的图像，若不在预先制定的正常图像表上就认为是伤残。

可是，把所有的正常图像都列成表，实际上是不合适的，因此，也不能因为不在正常图像的列表上就认为一定是伤残，这样，就必须对伤残的可疑点作进一步的仔细处理，才能最后断定。在对那些可疑点检查时，要计算旁边导线条的面积以及导线条和背景长度，再进一步算出面积和长度比的特征值，最后由此来断定是不是伤残。

2. 反射光的利用

用局部特征来探测伤残的方法，对微小伤残探测的能力比较高，但对类似于电路图形形状的伤残就探测不出来。解决这个问题的一种办法，是把没有伤残的完整电路图像作为绝对标准，然后把检查对象与它重合在一起进行比较，把其中有差别的部分作为伤残，但是这样做就有一个位置重合精度和数据量的问题，有的系统（日本中岛山等人研制）利用激光的反射解决了这个问题。

检查对象为掩膜图像，它由一定角度 0°，45°，90°，135° 的倾

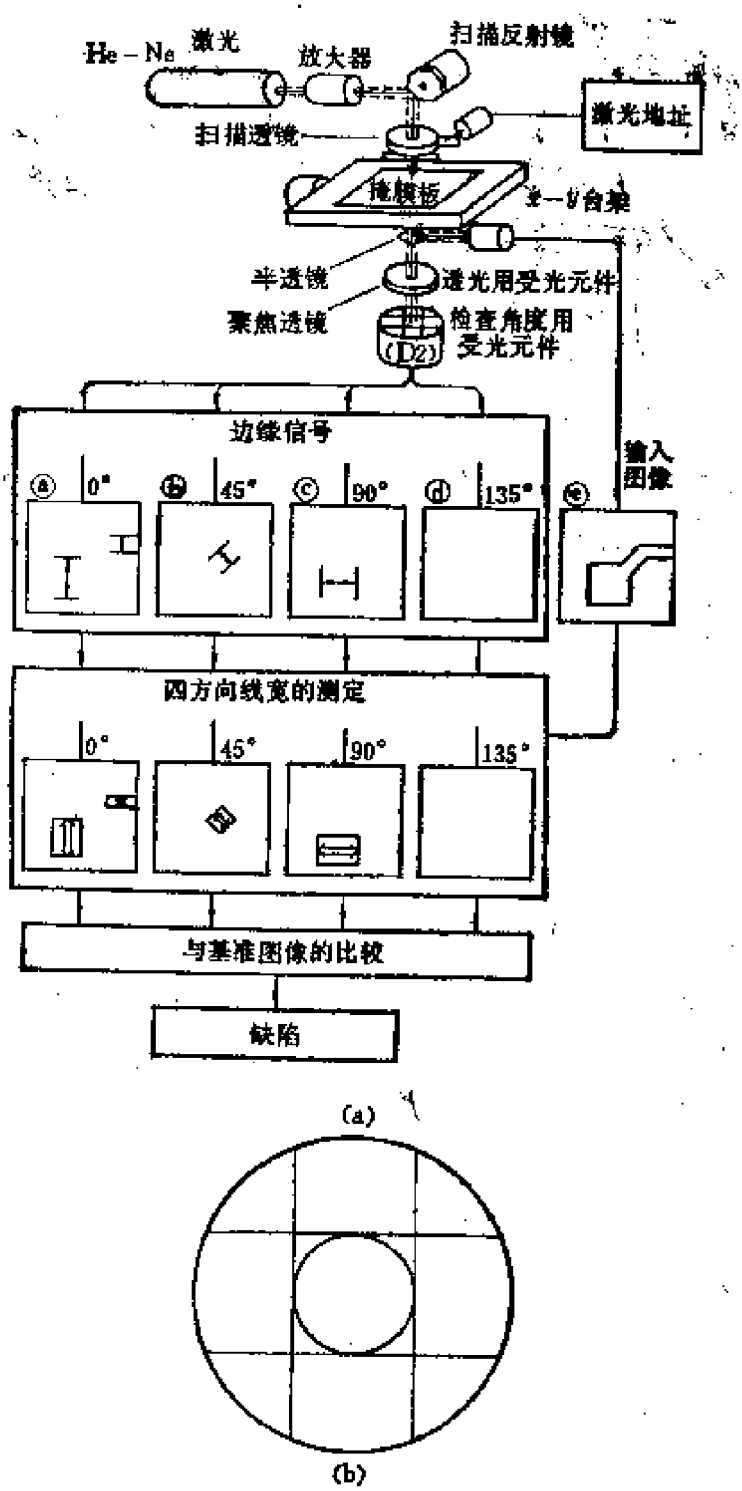


图 10.9　激光衍射法检测伤残

斜线构成，线的宽度用几个规定的值来描绘。在有伤残的地方，线宽就与标准线宽不一样了，这样就能用测量线宽的办法能检测出伤残。图 10.9(a) 表示了系统的构成，在构成掩膜图形的线的方向 (0°, 45°, 90°, 135°) 上使用激光衍射法精确地测出差别。系统的具体工作原理如下所述。

在图 10.9(a) 中来自 He-Ne 激光的光束，通过旋转扫描反射镜在一个方向上扫描。在与它垂直的方向上移动掩膜板，激光透过掩膜板，经过半透镜之后分成两个方向的光束，一束进入透光用的受光元件，另一束进入角度检查用的受光元件。检查用的受光元件配置在如图 10.8(b) 所示的 4 个扇区上。当激光束碰上掩膜图形的边缘时，在其垂直方向上发生反射，由某个角度受光元件扇区检测出这个反射光。这样，根据扇区的方向就可确定边缘的方向，然后在那里把来自角度检测用受光元件的信号二值化以后，按照扫描位置的地址把它写入到相应的存储器 (a)，或 (b)，或 (c)，或 (d)。在图像存储器 (e) 内存入来自穿透光形成的图形。由于知道了分别来自 (a) ~ (d) 的具有特定角度的边缘的位置，再参考 (e) 就定出图像边缘的宽度，与标准线宽相比较，把不同的部分作为伤残而检测出来。系统用这种方法可以发现小到 10μm 的伤残。对于 200 × 200mm 的掩膜板，检查时间约为 5min。

3. 模型的利用

在前面介绍的系统中，并没有利用关于检查对象的知识。因为检查对象是事先已知的，所以有效地利用这方面的知识，可提高处理的效率。

这里所介绍的一个缺陷检查系统(日本高本等人研制)为了检查印刷电路板，利用设计时的信息，自动制成电路的描述模型。模型如图 10.10，其中节点是要检查的电路的端点 (d, f)、拐点 (e)、分支点 (b、c)、终点 (g)，在节点上要记述它们的 (x, y) 坐标、应该检查的导线及其方向。

检查系统利用这个模型，首先检测一下在模型起点的周围有无导线图像，如果发现了导线图像就沿着模型上所表示的方向，跟踪检查导线，一直到终点。跟踪过程中随时测量与跟踪方向垂直的导线条的宽度，当发现与记录在模型上的值存在有某种程度的差异时，就作为伤残取出。在这个方向上希望可以随机存取图像传感器的信息，所以这里使用一个图像分析器，检查的范围是 100 × 100 mm，视野可以从 10 × 10mm 变到 40 × 40mm。相应的最小检查能力可在 10—40μm 内变化，检查的时间与图像密度有关，约在 5—30min 之间。

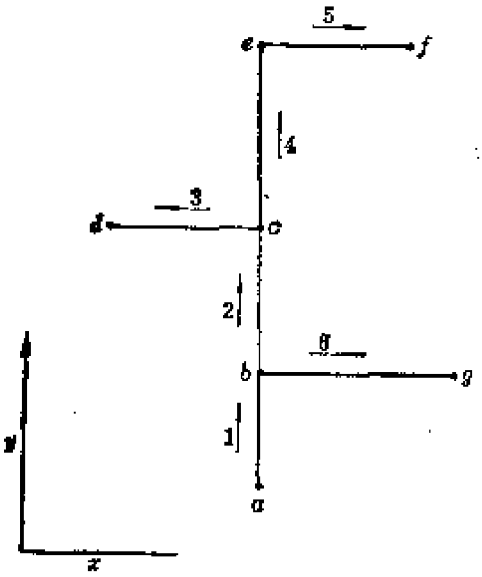


图 10.10　电路描述模型

4. 采用属性测量法的单体检查

上面主要讲述了电路图形的目视检查自动化系统，此外，在药品中对片剂和胶囊等的检查，对黄瓜、鱼类食品等级划分等方面的自动化检测亦已实现。在这些单体检查上都采用了属性测量的方

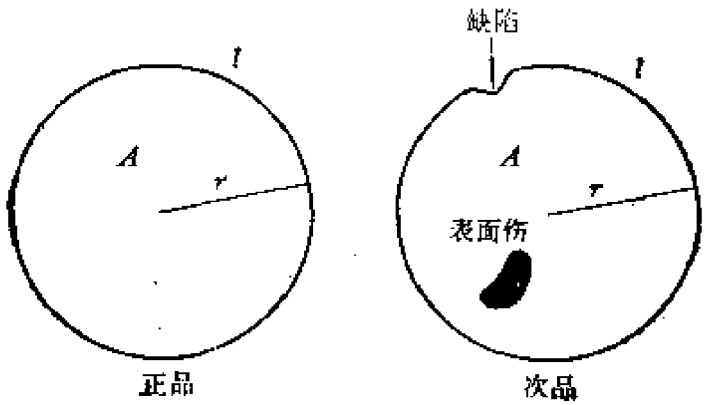


图 10.11　属性检查

法。例如，有一个药品片剂伤残检查系统(日本中村等研制)，就利用了片剂的面积和周长的属性来探测其伤残。

如图 10.11，设 l 为片剂的周长，A 为其面积，r 为片剂的半径，若成品没有伤残时，下式成立：

$$\begin{cases} A = \pi r^2 \\ l = 2\pi r \end{cases} \tag{10-1}$$

当然，

$$l = 2\sqrt{\pi A} \tag{10-2}$$

也成立，但对于有伤残的片剂，式 (10-2) 就不成立，由此可以区别出成品与次品。在实际处理中，则把片剂的二值图像输入到计算机中，计算 l 和 A。

二、视觉定位系统

利用视觉定位，这在零件组装和搬运中是必要的，这时要测量零件的位置和姿态，以及在装配时所需要的螺丝孔和连接端子的位置等。在晶体管和集成电路等半导体装配作业中经常用视觉系统来测定位置。要在附有金属引线的衬底上粘上半导体芯片，同时要把芯片内部的铝电极与衬底上的金属引线相连结，这项工作要耗费大量的人力，而且要在显微镜下操作，工作人员极度疲劳，故自动化的要求极为迫切。

1. 模板比照法

用于连接晶体管金属引线的视觉系统必须确定引线和半导体芯片的正确位置。这里介绍一个晶体管装配系统(日本拍岗等研制)，系统用小型计算机对带有视觉的多台装置(最多达 50 台)进行群控，装有 25 个晶体管的片子及衬底由仓库送到传送带，经过检测机构、连结构，最后由收藏机构送回仓库。在检测机构上，把要连接的芯片用光导摄像机拍出它在显微镜下的图像，检测出其位置后再送往连接机构。系统运用了一种复合模板比照法来确

定晶体管的位置，其原理如图 10.12 所示。

首先，操作人员预先从应该要联接的晶体管的图像中选出具有特征的 12 × 12 像素的局部图像(图中的 *P*1、*P*2、*P*3)，把它作为模板存于计算机中。当晶体管稍许移动一下位置时，如图中的(b)，把 *P*1、*P*2、*P*3 左右各转 10 度时的图像也记录下来。确定位置时，把输入的二值图像(160 × 120 像素)与最初的模板 *P*1 重合比较，找出最好的匹配点，再计算得到的两点间的距离 *d*12 及倾斜角 *θ*12。当这些值接近预先置于计算机内的参数时，就认为 *P*1 和 *P*2 的位置被准确地确定了，然后算出应该连结的电极的位置。

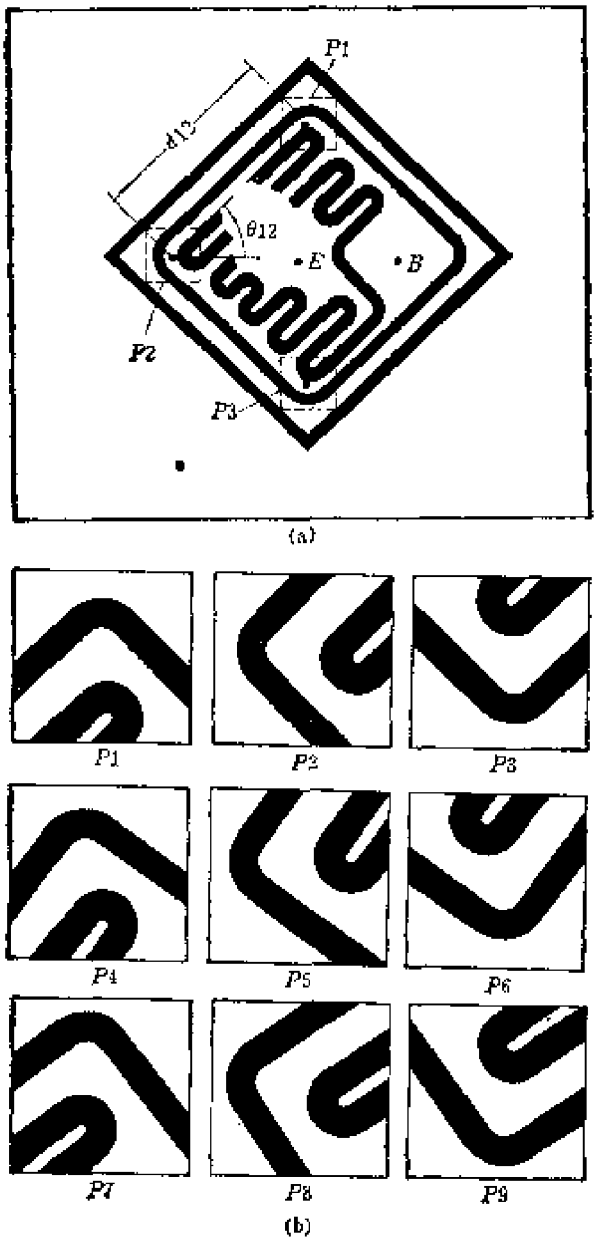


图 10.12　复合模板比照法

如果 *d*12、*θ*12 偏离了预想的值，则认为模板比照失败，再同下面的模板 *P*3 进行比较，把 *P*1 和 *P*3，*P*2 和 *P*3 之间的参数与预置于计算机中的参数进行比较；如果正确的话，再算出电极的位置。反复地进行这种操作，直到正确重合为止。这样就可能相当可靠地确定位置。因为

这种模板比照法只考虑必要的特征，效率高，且易于硬件化，所以在确定位置时常被采用。该系统的识别精度是 $\pm 10\mu m$，识别率是 99.9%，处理速度平均每片在 0.1s 以下。

2. 电极模板法

集成电路金属接线的定位系统，与晶体管的情况有所不同，由于集成电路的电极近于正方形，与芯片其他部分的形状不同，因而可以把它作为模板用；由于要求有很高的定位精度，所以要采取进一步的措施。

图10.13示意了用于集成电路金属接线的一个系统，它用两台电视摄像机通过半透明三棱镜观察电路芯片的两处不同外围部分，各自摄下不同的区域，独立地进行位置确定。这样由两个位置来确定芯片的位置，从而提高了精度。该系统适用于 0.8—0.6mm 的方形芯片(共有 400 种电路)，最大定位误差为 $10\mu m$，连接速度

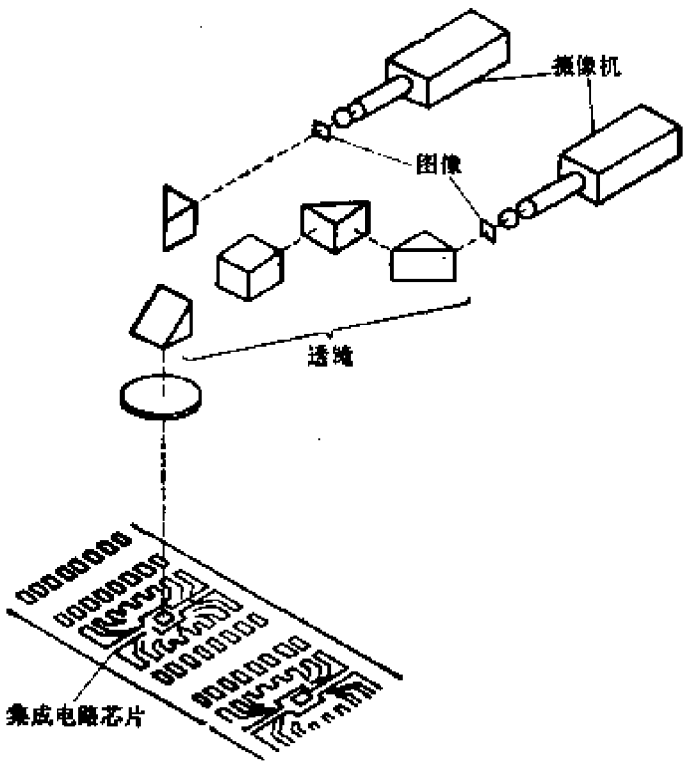


图 10.13　电极模板法

为每根铜丝 0.35s。

此外，还有的系统采用高分辨率的线性图像传感器，得到高分辨率的图像的方法。其原理是通过线性传感器机械地扫描，把所得的信号二值化，并输入存储器。把这个图像与事先输入的电极的标准图像相对照，探测出每个电极，然后从连结各电极中心位置的直线来确定芯片的位置和方向。

3. 两级定位法

模板比照法也可用于许多装配定位工作，这里介绍一个泵体的排水孔位置检测以及自动安装软管的系统。

由于泵的种类不同，所要检测的孔的大小和位置也不同。为此在这个系统中，在机器人的手指上装有小型的 CCD 电视摄像机（100 × 100 像素），对应于泵的不同种类，从不同的位置来观察孔，即由计算机把泵的类别送给机器人，控制机器人的手指动作。由于摄像机在某个确定的位置上变化，所以就把孔眼定在视野的中心，这就使其图像大小一定了。这样设定的摄像机的位置便于进行下面的图像处理。

处理分粗略定位和精确定位两个阶段进行。在粗略定位过程中，对原图像每隔一点就粗略地进行采样，从采样的 50 × 50 像素的图像中用模板比照法探测出洞孔。但因在这个粗略的图像中，孔的大小为 18 × 24 像素，模板太大，所以把孔分成四份，每四分之一圆作为一个模板。当确定了四分之一圆的位置后，就可求出孔的中心位置和直径，从而完成第一阶段的处理。

在精确定位过程中，从 100 × 100 像素的原图像中进行孔的精确测定位置，先设定第一阶段找到的孔的水平轴、垂直轴的长方形区域，由在这个区域内的近似圆内的面积来确定孔的中心定置和直径。图 10.14 是水平轴的区域，在距离原先找到的孔的中心各自为 A 的地方，把设定宽为 B 的区域的面积定为 M_0，M_1，就可求得孔中心的坐标 x 和直径 D：

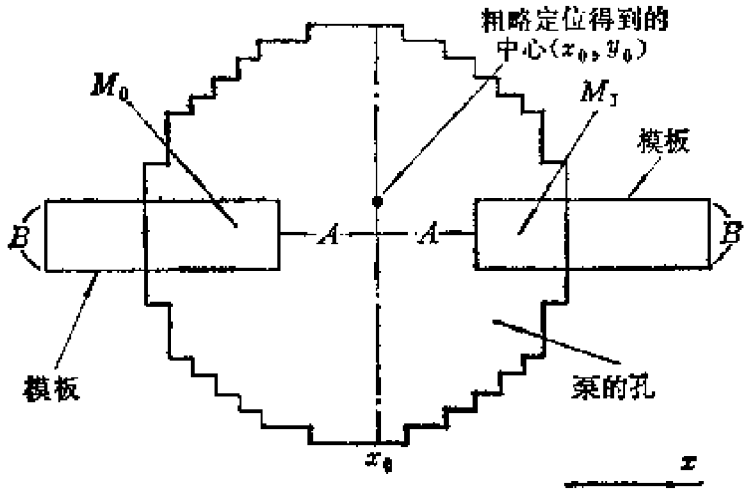


图 10.14　两级定位法

$$\begin{cases} x = x_0 + (M_1 - M_0)/B \\ D = 2A + (M_1 + M_0)/B \end{cases}$$

该系统从找到吸水和排水的两个孔，一直到机器人用手爪把软管连接完毕，全部时间不到 7s。

4. 投影法

如果对象物体是比较规则方正的，就不需要用模板比照法。在一个集成电路芯片连接自动化系统(日本井上等研制)中，芯片位置的确定就用了一种更为简便的投影法。图 10.15 是该系统构成示意图。该系统把约 1000—3000 个芯片排成网格状，放在 X-Y 台架上，电子计算机来控制移动 X-Y 台架，使得电视摄像机在视野内每次可观察到 4 个芯片。由于芯片比基板更明亮，故可由亮度加以区分，输入图像进行二值化以后，可得到 192 × 192 个像素的二值化图像。为了确定各芯片的位置，把相当于芯片的"1"，在水平和垂直两个方向上计数，构成投影图，具体过程如下：

(1) 由全部图像在水平方向的投影图，确定芯片垂直方向的大概位置。

(2) 对于经过(1)所能确定的各个区域，根据在垂直方向上的

投影再来确定芯片在水平方向上的正确位置.

(3) 对于经过(2)所能确定的各个区域,再根据在水平方向上的投影. 确定在垂直方向上的正确位置.

用这种方法,对于 0.4mm 的方形集成电路芯片,可达到 4—5 μm 的定位精度,足以满足要求.

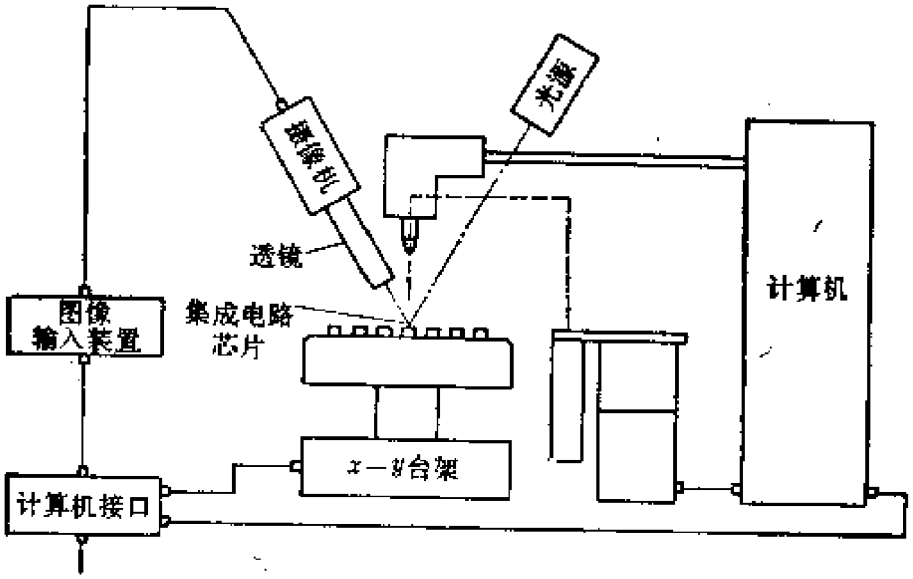


图 10.15 投影法

5. 边缘信息的利用

对于那些由于得不到良好的反差而难于二值化的图像可表示为灰图像,并且采用前面所述的空间微分方法进行处理.

图 10.16 示意了一个用于汽车发动机点火的集成电路芯片安装检查系统的原理. 先探测出安装在散热器上的电路芯片的位置和方向,安装检查通过后,再进行电气试验. 首先把来自固体摄像机的图像变换成 50 × 50 的像素,每个像素为 16 个灰度, 用前面所讲的 sobel 算法,求出各点微分值的大小和方向. 其次作出各点微分方向上的直方图, 对其峰值进行观察研究就知道了芯片的大致方向 (图中的(a)),芯片如有 ±30° 以上的旋转,就不可以进行

电气试验，认为安装检查没有通过。

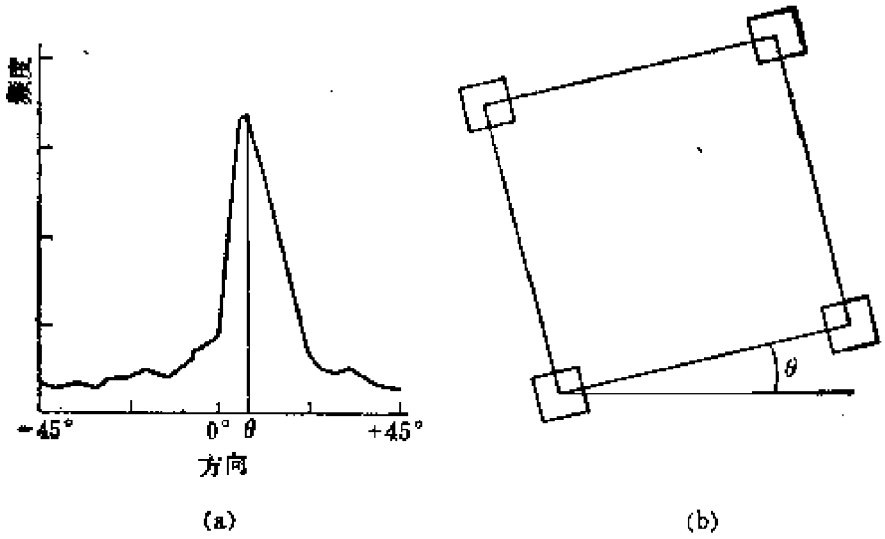


图 10.16　边缘信息的利用

在知道了芯片的大致方向之后，把芯片的四个角的部分作为模板，使用对照法，确定其正确的位置（图中的（b））。前面所讲的均是二值模板，而这里的模板是具有 16 个级别的灰度模型。这个系统没有专用的处理器，而是用微处理机来处理。由于使用的是低分辨率的图像，处理时间很长，差不多需要 1s。

三、零件识别系统

在自动化作业中，在用传送带传送各种零件时，一般要识别零件的形状和种类。迄今为止，所识别的对象主要是机械零件，零件上而且都有几个“稳定的姿态”。对其他对象的研究比较少。以下介绍一些机械零件识别系统。

1. 根据外形特征识别

在一个称为“Vision Module”的系统（Agin 等研制）中，采用了从外形特征来识别机械零件的方法。在传输带的正上方装有固体线性传感器（128 个像素），用以输入图像，进行二值化处理。

这时，传送带起了 Y 方向上的扫描作用，图 10.17 表示由二值图像求得的外形图，对于 4 种不同的零件，考虑到纵向和横向两种不同状态，共有 7 种图像.

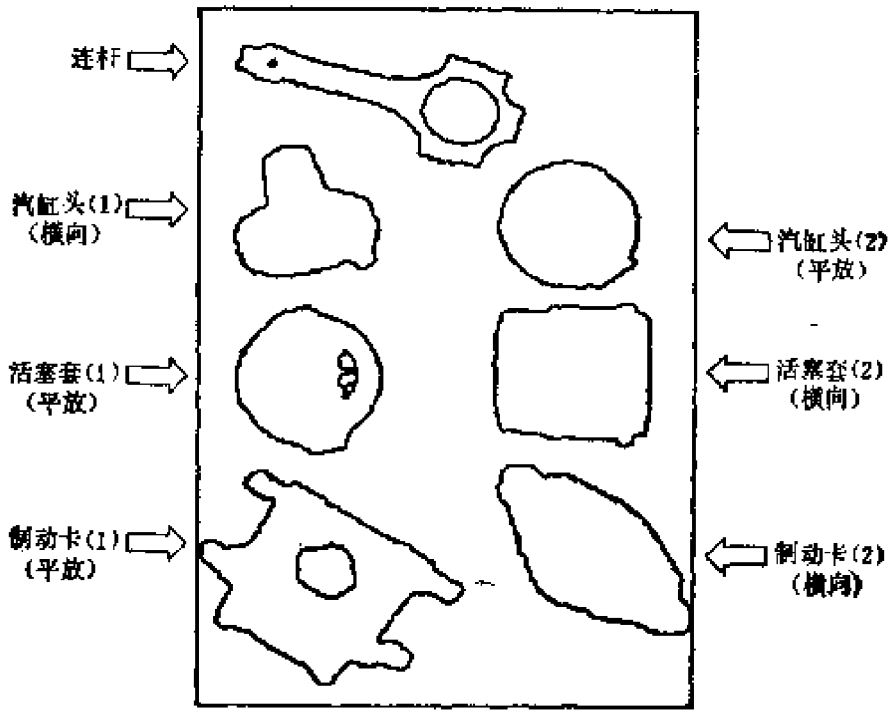


图 10.17　机械零件外形的二值图像

为了区别这些图像，拟定了如下特征量：

x_1 = 周长

x_2 = 面积的平方根

x_3 = 孔的全面积

x_4 = 从重心到外侧轮廓的最小距离

x_5 = 从重心到外侧轮廓的最大距离

x_6 = 从重心到外侧轮廓的平均距离

$x_7 = x_1/x_2$

识别流程如图 10.18 所示，显然，为区别 7 种图像并不需要使用所有的特征量. 系统工作过程中，在首先计算出所有的特征量

x_i 之后，为了识别各个零件，要选取最好的特征作为判别的最初节点（图中为 x_3），依次选取，判别下去。 最后生成一棵如图所示的判别树，与此同时，所有的零件也就顺序得到了识别。

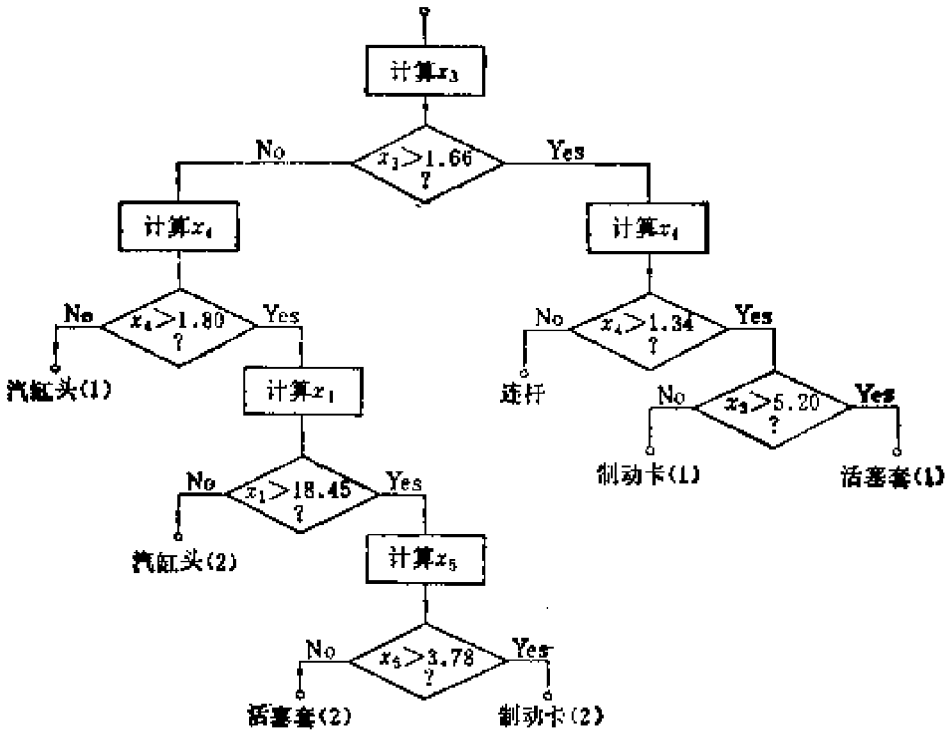


图 10.18 零件外形判别树

上述系统虽然简便，但是因为只能根据外形特征进行识别，所以对那些形状类似的零件就难以胜任了。

2. 根据表面特征识别

当零件种类很多、而且形状变得复杂时，只凭简单的外形特征就不能进行准确的分类。 于是，有的系统（日本谷田等人研制）除了有效地利用轮廓外形信息外，同时还利用零件表面的特征。 识别对象虽然只是 20 种发动机零件，由于每个零件都有“上面”、“底面”、“侧面”等几个稳定的姿态，所以实际上需要识别约 60 种图形。系统的构成如图 10.19 所示。

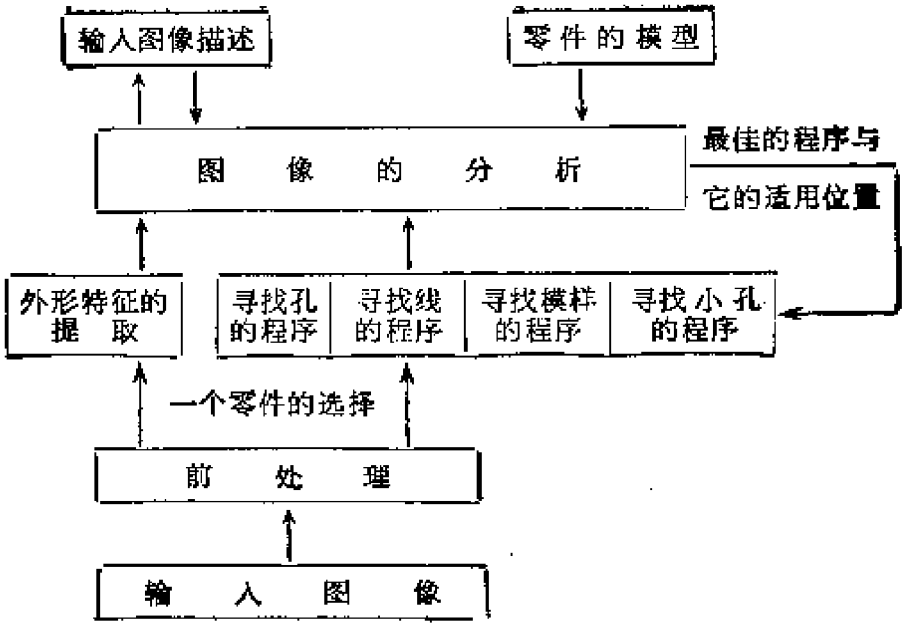


图 10.19　根据表面特征识别零件

该系统用电视摄像机从上方观察零件。通过处理，从视野中选择一个零件并二值化，算出其外形和重心。接着把重心作为中心，每 3° 引一根直线，求出重心到轮廓的距离 r，r 与直线的角度 θ 相对应，在计算机中首先准备好对应于各零件的标准的 $r(\theta)$ 曲线，然后把待测零件的 $r(\theta)$ 曲线与之相比较，找出最相关的，同时能够取出来的一个参量，用来识别零件。

根据外形来进行识别的方法，当然不能用来识别外形相同而表面特征不同的零件，因为这些表面特征难以区别零件上沾有的油污和灰尘等杂质；可是如果有效地利用零件的模式，也可以取出有关的特征（空洞、棱角、形状等）并加以区别。再有，在提取相对于物体重心的位置、性质等模型特征时，要写出最佳特征的提取程序。发动机零件因为外形相似的很多，在用外形判别的过程中，剩下几种根据外形不能判别的零件作为后补。在图像分析研究时，审查这些后补的模式特征，从这许多特征中确定能高效地识别未知零件的特征。然后，按照特征抽取程序从输入图像中提取所确

定的特征．正如图 10.19 所示，虽然有几种特征提取程序，可是最佳的程序由分析部分来确定．再有，应该研究的特征的位置及其性质由模式给出，所以，即使是很弱的特征也可以以很高的可靠性来提取．重复这个步骤，直到未知零件可识别为止．零件识别后的工作将转入测量在发动机装配中必需的螺丝孔的位置．

3. 根据实例的学习

一些生产线上的产品经常发生变化或改进，如果视觉系统每当对象改变一次时就重新编一次程序，当然既麻烦又费时间，于是，计算机自动地学习零件模型的系统得到了研究．当这种系统的摄像机"看"着零件实例时，由操作人员通过终端屏幕上的问答方式把实例的主要特征提示给系统，从而使系统完成一次零件模型的学习．具体工作过程如下所述．

在电视摄像机上看到要学习的零件时，系统首先抽取其轮廓信息，计算 $r(\theta)$ 曲线，而且对这一新零件进行登记，同时，要从外形上审查一下，该零件的登录是否会影响其他零件的知识．系统把不能与当前所学习的零件相区别的零件在屏幕上显示出来．操作人员看了以后，指出为了同其他零件相区别，要学习的零件有哪些主要特征，例如，由于孔是非常重要的，而且是易于抽取的特征，所以先在显示屏幕上指出孔的位置，系统把所指示的这一特征的抽取结果以及用根据这一特征尝试识别的结果显示在屏幕上．这时，如果还有容易混淆的零件存在，操作员就指出零件的第二个特征，例如棱．如此重复进行下去，新学习的零件就可以与其他所有零件相区别了．

采用这种学习方法，操作人员即使没有计算机编程序的知识，也可以很容易地把新的零件模型教示给计算机．

4. 利用窄条光提取外形特征

为了从零件的外形获得准确、稳定的识别信息，巧妙地设置照明光是一条重要的途径，在图 10.20 所示的视觉系统 Consigh

(Hollana 研制)中，从倾斜方向向传送带发送两条窄条缝隙光，用安装在传送带上方的固体线性传感器摄取其图像，而且预先把两条缝隙光调整到刚好在传送带上重合的位置。这样，当传送带上没有零件时，缝隙光合成了一条直线，可是当零件随传送带通过时，缝隙光变成两条线，其分开的距离同零件的厚度成正比。由于光线的分离之处正好就是零件的边界，所以当零件在传感器下通过结束的时刻就可以取出准确的边界信息。求得了边界，则用前述的方法进行零件的识别和位置的确定。如果机器人得到这些识别结果就可以在指定的场地把零件加以区分。

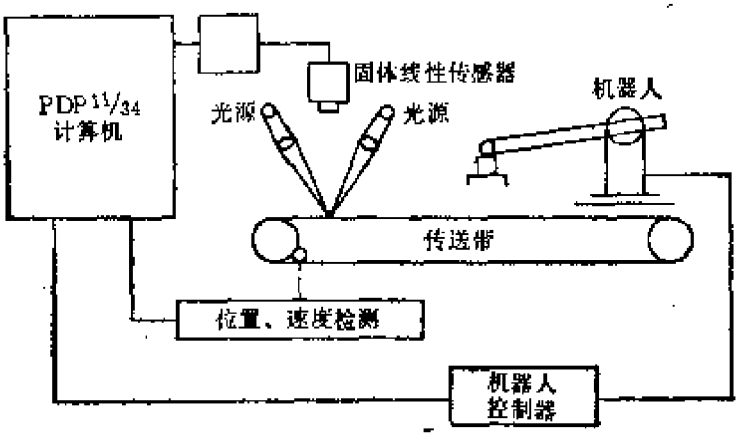


图10.20 Consigh 系统

四、综合视觉系统

以上介绍了三种不同功用的视觉系统，可是在通常的装配自动化方面，常常需要把这些功用和实现方法结合在一起的综合视觉系统，即，既要识别由传送带传送的零件的种类，又要进行位置的确定，再由机器人加以装配时，还要检查装配的结果。例如，一个小型发动机自动装配系统(美国 SRI 试制)就需要综合地运用视觉的多种功能。发动机本体上有 8 个要拧入螺栓的螺孔。在机器人的手指上装有固体摄像机 (100 × 100 像素)，可俯视发动机本

体，在获得二值化图像后，系统用前述的 Vison Module 方法在图像中寻找孔的边缘，根据孔的大小或圆度等性质来同其它特征相区别。但是 100 × 100 像素的分辨率即使能观察到发动机体的全部，而孔的位置也不能确定得十分精确，所以在粗略地测定了孔的位置以后，重新摄取图像，测定孔的精确位置，再让机器人的手爪把螺栓旋进螺孔并拧紧。所有螺栓都紧固完毕，最后再一次从上面摄取图像，检查螺栓是否安装无误。如果前边所检测的孔的图像都消失了，就确认螺栓已全部插入了。

又如，有一个吸尘器装配系统(日本武安等试制)，它配有两只机器人手臂，八个电视摄像机。开始装配时，先由一只机械手从堆放着的过滤器中拣取一个，把它嵌进另一只机械手拿着的积尘箱内，再把本体嵌进箱内，最后用夹板把本体与积尘箱固定紧。整个装配过程中需要使用多个摄像机的视觉反馈对机械手进行控制。当把本体装入积尘箱时，使用了三个摄像机(水平方向的 B、D，垂直方向的 R)，它们分别观测本体同积尘箱的距离，把其信息送给机器人，修正手爪的位置。其间首先用摄像机 B 和 D 测量水平方向的误差，修正手腕位置，然后用垂直方向的摄像机 R 测定本体和积尘箱的转角误差，把该信息送到手腕，使之准确地进行夹板的固定连接。

§ 10.4 工业视觉系统的展望

本章主要介绍了一些已经实用或接近实用的视觉系统。在着手于工程实现的同时，工业视觉领域也在深入地开展一些着眼于未来的基础研究。研究的目标主要是关于图像处理器、图像处理技术、以及系统的适应性等问题。

1. 更高性能的图像处理器

在多数工业视觉系统中，可使用二值图像处理器。随着 LSI

技术的发展和高性能微处理机器的出现，有可能使得高速处理灰度图像的工业视觉系统的价格变得比较便宜，使实用成为可能。这样的处理器正朝着两个方向发展。

一个方向是研制高速进行空间微分等局部处理的专用处理器，例如，能否研究试制由 VLSI 构成单片图像处理器，它使空间微分等基本的灰度图像处理能按电视的扫描速度进行。美国 Hughes 公司把 CCD 传感器和处理器作成一体，制成一种新式的传感器，提供了很有意义的研究结果。

另一个方向是使图像处理器具有一些灵活性、柔软性，可根据用户程序来方便地加以开发，日本森等人的 PPP (Parallel Patten Processor)，Stanberg 等人的 CYTO Computer 就是这种例子。

2. 灰度图像处理技术的引入

在工业视觉系统中，多数情况可以用二值图像。可是灰度图像具有通用性，由于在智能机器人视觉研究方面，已经建立了处理灰度图像的各种方法，这些方法随着前端处理器的研制，可望在工业视觉中被正式采用。这样，即使对于那些不能取得良好的反差，因而也不便于使用二值图像系统的环境，视觉系统也会有用武之地。

3. 更柔软的系统

在中批量和小批量生产中，重要的是，视觉系统要能适应产品的变更，对此，有两种方法正在研究。

一种方法是不必改变程序，而是把产品的模型以对话的形式教示给计算机。虽然在前述几个系统中已采纳了这种方法，但是，其中多数是采用让计算机观察实例，把其特征在显示屏幕上进行教示的方法。在今后的视觉系统中，在配置了具有高速处理灰度图像的微处理机以后，使用上面所说的对话方式，系统就可能具有快速响应、方便使用的特点；而且，已有的一些系统由于处理方式一定，虽然允许对象零件有所变化，可是，如果变更范围较大时显

然就不能适应。当然，要求能处理所有的对象的视觉系统无论从技术上还是从价格上来看都是不合适的，但是研制处理对象能稍加扩充的系统还是有指望的。

其他要研究的还有用 CAD 的设计信息来制成对象模型的方法。这种方法在印刷电路板的伤残检查中正在试用。三维形状的模型化的研究也在迅速进行着。不久的将来，带有阴影的物体的模型化也是有可能的。若作到这一点，则可以把任一视点上看到的物体的图像用计算机来模拟，对这些图像进行前处理，可以得到优化的结果。总之，有效利用设计信息将是今后的研究课题。

在智能机器人的视觉研究中，正在研究除灰度以外的如颜色和距离等信息的视觉系统。关于距离，试图采用投射窄细状光的办法；根据三角测量的原理计算距离的立体方法；以及投射激光，测量从对象反射回来的时间的方法等等。这些技术还有待于进一步的实用化研究。

第十一章　语音识别原理

机器人的听觉指的是对于语音的识别和说话人的识别．其中主要是建立发音系统模型并由此导出语音识别的算法．本章的内容着重于讨论语音识别的基本原理．

§11.1　语音的产生机理

这一节首先论述人的发音系统和语音的产生机理，由此导出得到广泛应用的(人的)发音系统的离散模型．这一模型为语音信号的数字处理、提取瞬语音的特征、实现语音合成提供了原则性的指导．

图 11.1 是人的发音系统断面图．从声门到嘴唇这段空间称为声道，包括咽喉和口腔．对于成年的正常人，声道的长度约 17.5 cm．它的形状随唇、舌、硬腭、软腭的位置而变，横断面积可以从零到大约 20cm^2 的范围内变化．在发音的过程中，声道起共振腔的作用，当肌肉收缩，改变声道的形状时，可以使声道的共振频率(或称为共振峰频率)改变，从而改变发出的声音的特性．当软腭下垂时，鼻腔和声道连通，由于鼻腔和声道的耦合，因而产生语音中的鼻音．

图 11.2 是发音系统的示意图．其中，肺、支气管、气管组成次声门系统．次声门系统提供对声道的激励．当肺收缩时，呼出的气体穿过声带间狭窄的孔道．按贝努里定律，由于声带孔道处的

气流速度较高，因而压力较低．如果声带的张力适当，压力降低的结果使声带靠拢，并将气流阻断；气流阻断后压力回升，声门张开重新使气流通过．于是声带产生张弛振动，在声门处产生如图 11.3 所示的准周期性的空气体速度脉冲．声带振动的频率可以由静止状态声门的大小、肺产生的空气压力，以及声带的劲度加以控制．体速度脉冲激励声道，产生语音中的浊音，改变脉冲的频率形成各种声调的语音．

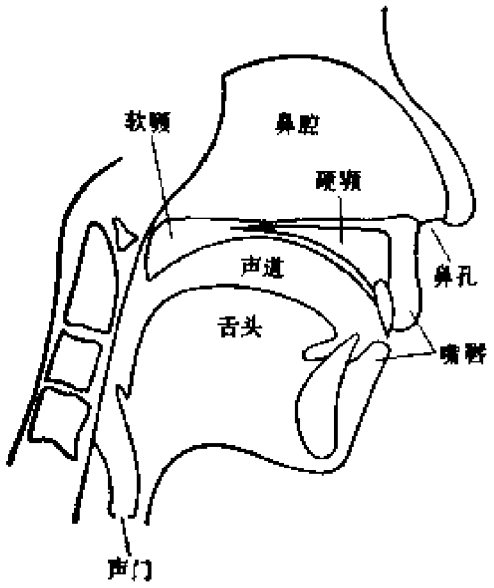


图 11.1　人的发音系统

如果在声道的后端(唇端)某处收缩，呼出的空气被迫通过这收缩的部分而产生湍流，于是形语音中的清擦音．清擦音相当于用白噪声激励声道产生的语音．如果声道后端收缩，声门处的激励是准周期性的脉冲，则形语音中的浊摩音，但是，湍流只在体速度的峰值附近产生．如果在声道后端处完全关闭，于是在关闭处的前面建立起压力，然后突然释放，形成语音中的塞音．

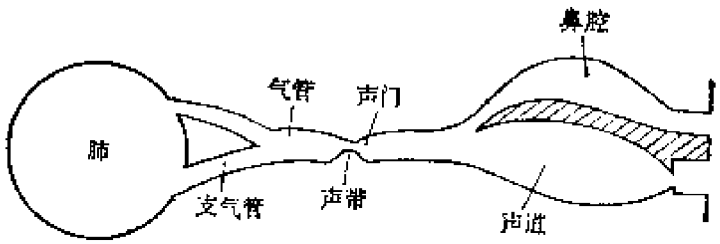


图 11.2　发音系统示意图

按如上所述，可以把人的发音系统看成是声道和激励两部分组成，如图 11.4 所示．图中线性时变系统代表声道和嘴唇的喷射，

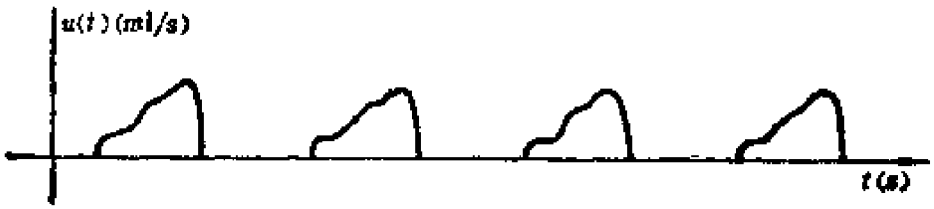


图 11.3　声门处的准周期性体速度脉冲

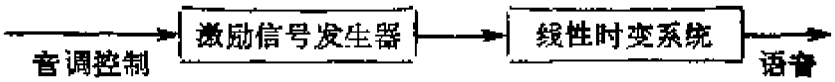


图 11.4　语音生成模型

系统的时变性是由于在发音的过程中声道的形状随时间不断变化的结果．但是，这种变化是由于肌肉的收缩和放松形成的，变化的速度十分缓慢，一般说来，在 20ms 的时间区间里，声道特性的改变可以忽略，因此可以把声道看成是定常的线性系统．

§ 11.2　发音系统的离散模型

一、声门模型

由于声带振动，在声门处产生准周期性的体速度脉冲．若把每一个脉冲看成是声门的冲击响应，于是，声门输出的准周期性脉冲是声门对一个准周期冲击序列的响应．对声门输出脉冲可用如下函数近似：

$$G(t) = te^{-\sigma t} \tag{11-1}$$

语音合成的实验表明，若用式(11-1)代表声门的冲击响应，合成的语音具有良好的自然性．

若语音信号的采样周期为 T，那么，由式(11-1)得声门的转移函数为

$$G^*(z) = \frac{Te^{-\sigma T}z^{-1}}{(1-e^{-\sigma T}z^{-1})^2} \tag{11-2}$$

式 (11-2) 中分子的 z^{-1} 仅仅表示时间的延迟，无实质性的意义；Te^{-cT} 代表增益的大小，可以在整个发音系统的模型中加以考虑。所以，声门的离散模型为

$$G(z) = \frac{1}{(1 - e^{-cT}z^{-1})^2} \tag{11-3}$$

二、均匀无损管道中的行波

现在讨论声波在无损管道中传播的情形。

设有一固定的(时不变的)不均匀的(横截面不为常数)光滑的管道如图 11.5(a) 所示，图 11.5(b) 是这管道的横断面积函数。设管道的横向尺寸比声波的波长短得多（对声道这显然是合理的），因此在管道内声波是轴向传播的平面波。管道是无损的，指的是：在声波的传播过程中，不存在由于管壁的振动和热传导以及粘滞造成的能量损失。

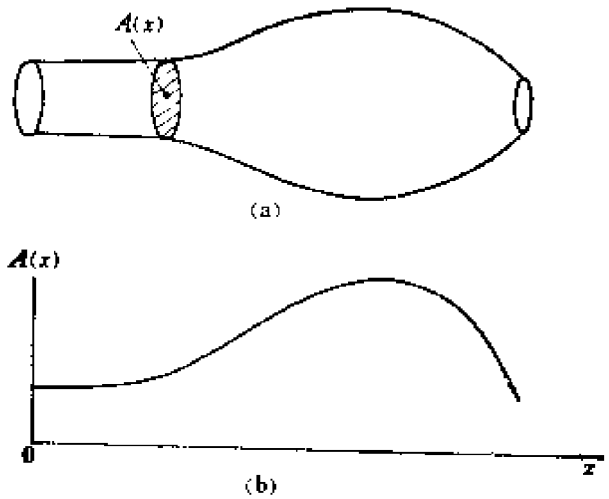


图 11.5　无损管导和它的横断面积函数

在以上条件下，波得诺夫根据质量、能量、动量守恒定律导出了声波在管道中的传播方程为

$$\begin{cases} -\dfrac{\partial p}{\partial x} = \rho \dfrac{\partial (u/A)}{\partial t} & \text{(a)} \\ -\dfrac{\partial u}{\partial x} = \dfrac{1}{\rho C^2}\dfrac{\partial (pA)}{\partial t} + \dfrac{\partial A}{\partial t} & \text{(b)} \end{cases} \tag{11-4}$$

其中：$p = p(x,t)$：管道中的声压；

$u = u(x,t)$：管道中的空气体速度；

$A = A(x,t)$：管道的横断面积函数；

ρ：空气的密度；　C：声速。

由于管道是定常的，所以方程式(11-4)为

$$\begin{cases} -\dfrac{\partial p}{\partial x} = \dfrac{\rho}{A}\dfrac{\partial u}{\partial t} & \text{(a)} \\ -\dfrac{\partial u}{\partial x} = \dfrac{A}{\rho C^2}\dfrac{\partial p}{\partial t} & \text{(b)} \end{cases} \tag{11-5}$$

如果管道均匀（A 为常数），我们立即可以发现，方程(11-5)和均匀无损长线（$r_0 = 0$，$g_0 = 0$）方程在形式上完全一样，p 对应于线间电压 V，体速度 u 对应于线电流 I，ρ/A 对应于单位长度上的线电感 L_0，$A/\rho C^2$ 对应于单位长度上的线间电容 C_0。和均匀长线类比，均匀无损管道的特征声阻抗为

$$z_j = \sqrt{\frac{j\omega\rho/A}{j\omega A/\rho C^2}} = \frac{\rho C}{A} \tag{11-6}$$

与均匀无损长线的通解一样，方程(11-5)的解为

$$\begin{cases} u(x,t) = u^+(t - x/C) - u^-(t + x/C) & \text{(a)} \\ p(x,t) = p^+(t - x/C) + p^-(t + x/C) & \text{(b)} \\ p^+ = z_j u^+ & \text{(c)} \\ p^- = z_j u^- & \text{(d)} \end{cases} \tag{11-7}$$

其中 u^+ 和 u^- 分别是无损管道中的正行和逆行体速度波；p^+、p^- 分别是正行和逆行声压波。声压波和体速度波之间通过特征声阻抗联系起来，如式中的 (c)、(d) 所示。图 11.6 是均匀无损管道中

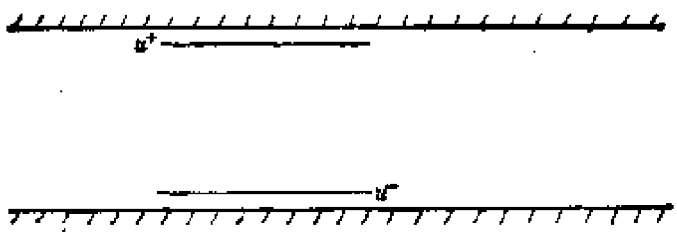


图 11.6　均匀无损管道中的体速度行波示意图

的体速度行波.

三、声道的无损管道模型

对于瞬时语音(如 20ms 以内的语音)或持续音,可以把声道看成是定常的,如图 11.7(a)所示. 很容易想到, 可以根据声道在各个部位的截面积,用 P 段等长的均匀无损管道来逼近声道,图 11.7(b) 是六段逼近的情形. 显然,级联的段数越多,越接近于声道的真实形状. 但是,这种逼近忽略了摩擦、管壁振动和热传导造成的损失. 不过,我们仅希望得到一个声道的终端模型,只要模型的输出近似于实际情况即可, 而不必过分地追求内部过程的细节. 因此,对于如上损失造成的终端效果,可以通过合理地选择嘴唇和声门处的损失考虑进去.

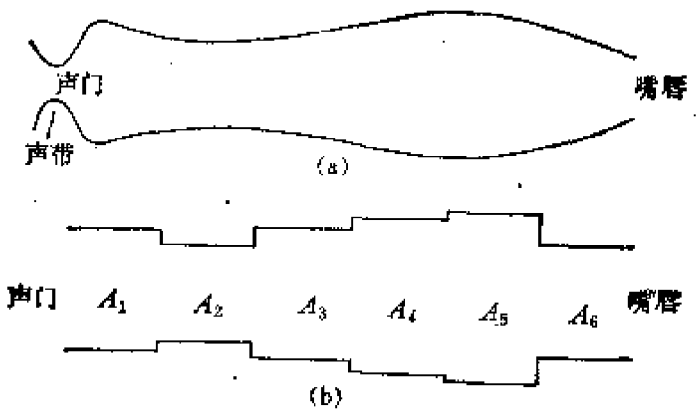


图 11.7　声道的级联均匀无损管道逼近

设段数是 P，每段的长度是 l_0 按均匀无损管道的通解式 (11-7)，在第 k 段内的声压和体速度分别是

$$\begin{cases} p_k(x,t) = \dfrac{\rho C}{A_k}[u_k^+(t-x/C) + u_k^-(t+x/C)] & \text{(a)} \\ u_k(x,t) = u_k^+(t-x/C) - u_k^-(t+x/C) & \text{(b)} \end{cases} \tag{11-8}$$

其中 A_k 是第 k 段的横断面积，x 从第 k 段的左端开始进行度量。考察第 k 段和 $k+1$ 段连接处的情况，见图 11.8。

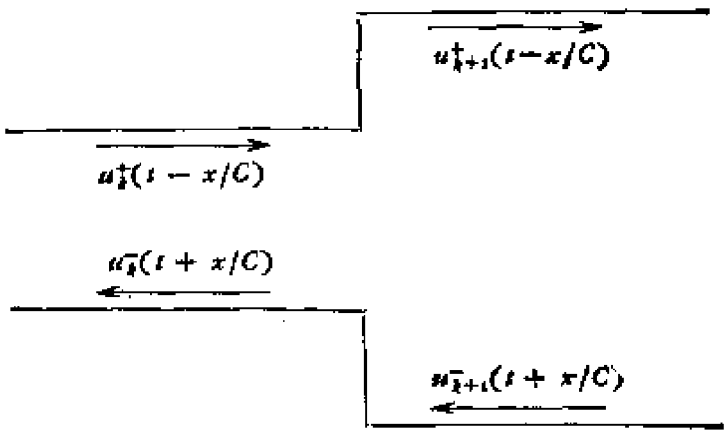


图 11.8　两管连接处波的传播

在两管的连接处，由连续性条件有

$$\begin{cases} p_k(l,t) = p_{k+1}(0,t) & \text{(a)} \\ u_k(l,t) = u_{k+1}(0,t) & \text{(b)} \end{cases} \tag{11-9}$$

将式(11-8)代入式(11-9)，得

$$\begin{cases} \dfrac{\rho C}{A_k}[u_k^+(t-\tau) + u_k^-(t+\tau)] \\ \qquad = \dfrac{\rho C}{A_{k+1}}[u_{k+1}^+(0,t) + u_{k+1}^-(0,t)] & \text{(a)} \\ u_k^+(t-\tau) - u_k^-(t+\tau) \\ \qquad = u_{k+1}^+(0,t) - u_{k+1}^-(0,t) & \text{(b)} \end{cases} \tag{11-10}$$

其中 $\tau = l/C$ 是声波传过一段所需的时间。由式(11-10)解得

$$\begin{cases} u_{k+1}^{+}(0,t) = \dfrac{2A_{k+1}}{A_k + A_{k+1}} u_k^{+}(t-\tau) \\ \qquad + \dfrac{A_{k+1} - A_k}{A_k + A_{k+1}} u_{k+1}^{-}(0,t) \quad \text{(a)} \\ u_k^{-}(t+\tau) = \dfrac{A_k - A_{k+1}}{A_k + A_{k+1}} u_k^{+}(t-\tau) \\ \qquad + \dfrac{2A_k}{A_k + A_{k+1}} u_{k+1}^{-}(0,t) \quad \text{(b)} \end{cases} \tag{11-11}$$

式(11-11)表明，在第 $k+1$ 段的起始处，正行波由两部分组成，第一部分是第 k 段的正行波在交界面处的穿透部分[式 (11-11)(a)右端第一项]，第二部分是第 $k+1$ 段内的逆行波在交界面的反射波.同样，式 (11-11)(b) 表明，第 k 段末端的逆行波也是由两部分组成，它分别是：第 k 段正行波的反射波和第 $k+1$ 段逆行波的穿透波。设 r_k 是交界面处逆行波的反射系数：

$$r_k = \frac{A_{k+1} - A_k}{A_k + A_{k+1}}; \quad -1 \leqslant r_k \leqslant 1 \tag{11-12}$$

于是，式(11-11)可改写为

$$\begin{cases} u_{k+1}^{+}(0,t) = (1+r_k)u_k^{+}(t-\tau) + r_k u_{k+1}^{-}(0,t) & \text{(a)} \\ u_k^{-}(t+\tau) = -r_k u_k^{+}(t-\tau) + (1-r_k)u_{k+1}^{-}(0,t) & \text{(b)} \end{cases} \tag{11-13}$$

式(11-13)可表示为图 11.9。

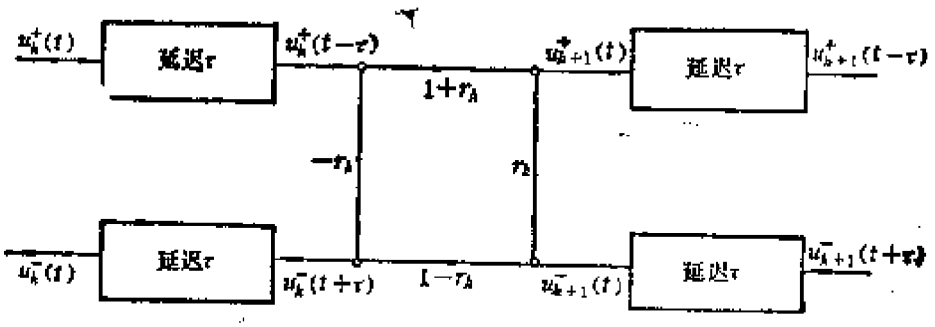


图 11.9　均匀无损管道连接处的信号流图

图中 $u_k^{+}(t) = u_k^{+}(0,t)$，$u_k^{-}(t) = u_k^{-}(0,t)$。

以下讨论由图 11.7 (b) 所示的级联均匀无损管道的边界条件.

首先讨论嘴唇处的边界条件.可以设想,在嘴唇外存在第 $P+1$ 段无限长的均匀无损管道,其截面积 A_{P+1} 应当选择得合适,使得模型输出的语音近似于实际采集到的语音. 由于第 $P+1$ 段无限长,所以,在这段中无反射波. 于是, 唇端的边界条件可以表示成图 11.10 所示的情况. 图中 r_l 是第 P 段和第 $P+1$ 段间的反射系数:

$$r_l = \frac{A_{P+1} - A_P}{A_P + A_{P+1}} \tag{11-14}$$

$u_P(l, t)$ 是嘴唇处的空气体速度.

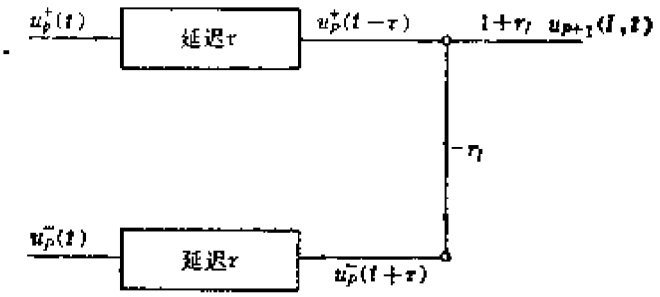


图 11.10 声道唇端的边界条件

以下讨论声道前端(声门端)的边界条件.

声道前端的边界条件决定于声门和声道的耦合. 由于把声道看成是 P 段均匀无损管道的级联,这相当于 N 段无损长线的级联,声门处的声压函数 $p_g(t)$ 类似于电压源,级联无损管道是 $p_g(t)$ 的负载. 设 $p_g(t)$ 的输出阻抗是 R_g, 如图 11.11(a) 所示. 利用化电压源为电流源的原理,得图 11.11(b) 所示的声门体速度和声道的耦合形式.

图 11.11 中 $p_1(t)$、$u_1(t)$ 分别是声道入口处的声压和体速度. 有

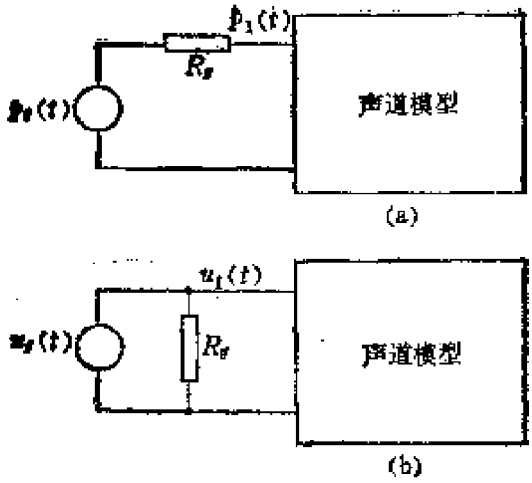


图 11.11　声门和声道的耦合

$$u_1(z) = u_g(z) - p_1(z)/R_g \tag{11-15}$$

已知

$$u_1(z) = u_1^+(z) - u_1^-(z) \tag{11-16}$$

$$p_1(z) = p_1^+(z) + p_1^-(z) = z_{f1}[u_1^+(z) + u_1^-(z)]$$

$$= \frac{\rho C}{A_1}[u_1^+(z) + u_1^-(z)] \tag{11-17}$$

式中 z_{f1} 是第一段无损均匀管道的特征声阻抗。将式(11-16)、(11-17)代入(11-15)可以解得

$$u_1^+(z) = \frac{1 + r_g}{2} u_g(z) + r_g u_1^-(z) \tag{11-18}$$

反射系数：

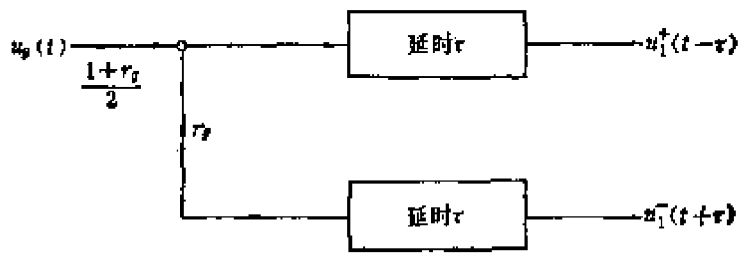


图 11.12　级联无损声道声门端的边界条件

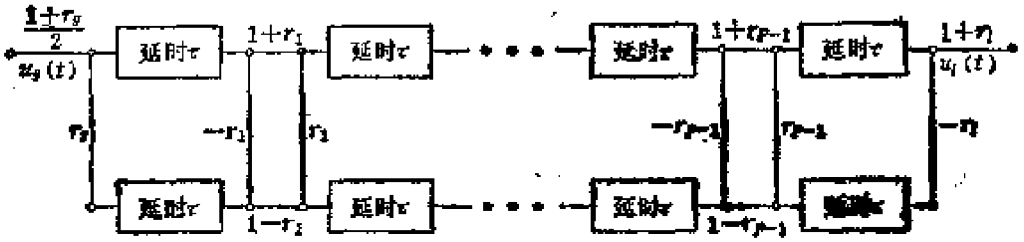


图 11.13　P 段级联均匀无损管道信号流程图

$$r_g = \frac{R_g - \rho C/A_1}{R_g + \rho C/A_1} \tag{11-19}$$

因此，声道在声门端的边界条件如图 11.12 所示。

综上所述，得声道的信号流程图如图 11.13 所示。

四、声道的离散模型

如果激励的频带受限在 $1/4\tau$ 以下，那么，可以用 $T = 2\tau$ 的周期对语音采样。于是由图 11.13 得声道的离散模型如图 11.14(a) 所示。

在图 11.14(a) 中，$z^{-\frac{1}{2}}$ 意味着要在两个采样点之间进行内插，显然，这对于实现增加了困难。仔细分析图 11.14(a) 可以发

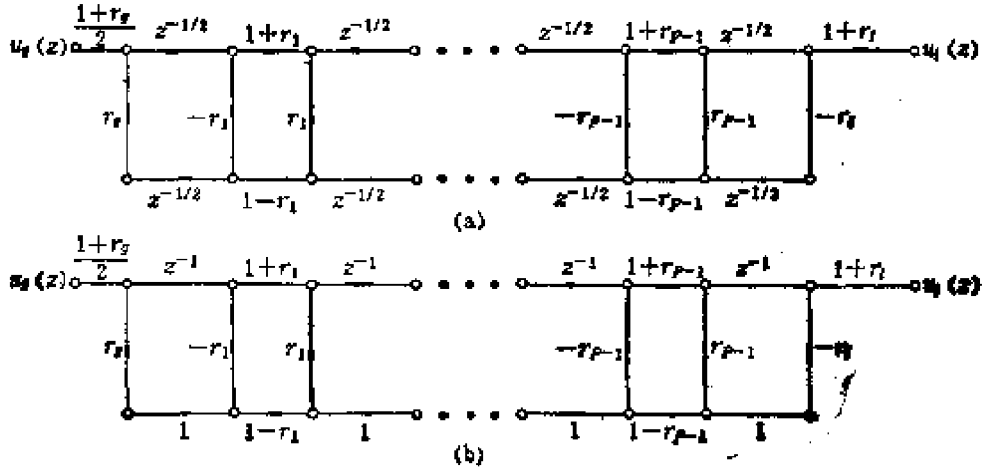


图 11.14　声道的离散模型

现，延迟单元只存在于上下两个路径中，如果把下面路径的延迟移到对应的上面路径上去，得图 11.14(b) 所示的情况。这两个图形沿任意对应迴路的总延时相等，唯一的区别是图 11.14(b) 中从 u_g 到 u_l 的延时增加了 $P\tau/2$，这增加的延时量可以在模型中串联 $z^{\frac{P}{2}}$ 的超前环节补偿。其实，这种补偿没有必要，因为在模型中任何 $z=0$ 的零点或极点对语音的性质不会有任何影响。因此，图 11.14(a) 和 (b) 有相同的转移函数。图 11.14(b) 即是声道的离散模型。

由图 11.14 可以计算出声道的转移函数 $V_{(z)}$。图 11.15 是图 11.14(a) 的一个单元。

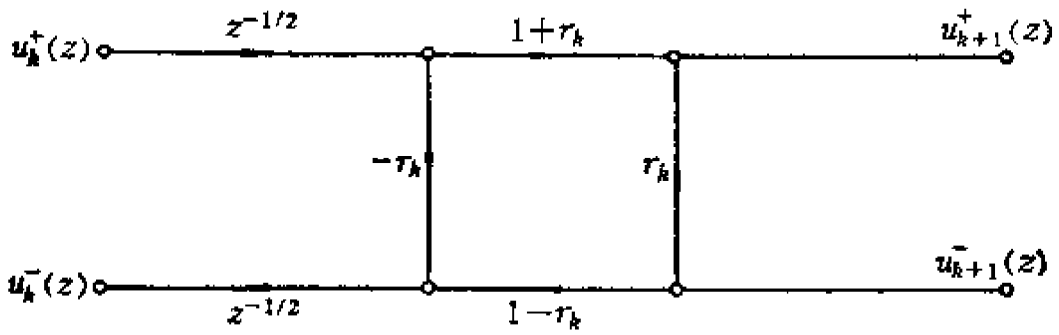


图 11.15　声道离散模型的一个单元

由图可以得

$$\begin{cases} U_{k+1}^+(z) = (1+r_k)z^{-\frac{1}{2}}U_k^+(z) \\ \qquad + r_k U_{k+1}^-(z) \qquad\qquad (a) \\ U_k^-(z) = -r_k z^{-\frac{1}{2}}U_k^+(z) \\ \qquad + (1-r_k)z^{-\frac{1}{2}}U_{k+1}^-(z) \qquad (b) \end{cases} \tag{11-20}$$

由上式可以解得

$$\begin{cases} U_k^+(z) = \dfrac{z^{\frac{1}{2}}}{1+r_k}[U_{k+1}^+(z) - r_k U_{k+1}^-(z)] \qquad (a) \\ U_k^-(z) = \dfrac{z^{\frac{1}{2}}}{1+r_k}[-r_k z^{-1}U_{k+1}^+(z) + z^{-1}U_{k+1}^-(z)] \qquad (b) \end{cases} \tag{11-21}$$

令

$$U_k = [U_k^+(z), U_k^-(z)]^T \tag{11-22}$$

$$Q_k = \begin{bmatrix} 1, & -r_k \\ -r_k z^{-1}, & z^{-1} \end{bmatrix} \tag{11-23}$$

式(11-21)可以写成如下矩阵形式:

$$U_k = \frac{z^{\frac{1}{2}}}{1+r_k} Q_k U_{k+1} \tag{11-24}$$

如前所述,在想象的 $P+1$ 段均匀无损管道中无逆行波,所以有

$$U_{P+1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} U_l(z) \tag{11-25}$$

令 $r_P = r_l$, 于是,反复使用式(11-24)得

$$U_1 = z^{\frac{P}{2}} \left[\prod_{k=1}^{P} \frac{1}{1+r_k} Q_k \right] \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} U_l(z) \tag{11-26}$$

根据声门端的边界条件[见图 11.14(a)], 有

$$U_g(z) = \frac{2}{1+r_g} [1, -r_g] U_1$$

$$= \frac{2}{1+r_g} z^{\frac{P}{2}} \left(\prod_{k=1}^{P} \frac{1}{1+r_k} Q_k \right) \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} U_l(z) \tag{11-27}$$

所以,声道的转移函数为

$$V(z) = \frac{U_l(z)}{U_g(z)} = \frac{0.5(1+r_g) z^{-\frac{P}{2}} \prod_{k=1}^{P} (1+r_k)}{D(z)} \tag{11-28}$$

$$D(z) = [1, -r_g] \cdot \begin{bmatrix} 1, & -r_1 \\ -r_1 z^{-1}, & z^{-1} \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1, & -r_2 \\ -r_2 z^{-1}, & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} 1, & -r_P \\ -r_P z^{-1}, & z^{-1} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{11-29}$$

其中 $r_p = r_l$。在式(11-28)中，因 $z^{-\frac{p}{2}}$ 无实际意义，可以不予考虑。由式(11-29)可以看出 $D(z)$ 是一形如 $1 + \sum_{k=1}^{p} a_k z^{-k}$ 的多项式，所以，声道为如下全极点模型：

$$V(z) = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{11-30}$$

由常识知，只要各级管道的截面积 $A_k \geqslant 0$，级联管道必稳定，所以，$V(z)$ 的所有极点在 z 平面单位圆内。相反，如果任意给定一形如式(11-30)所示的转移函数，只要 $V(z)$ 的全部极点在单位圆内，可以证明，必可找到一个级联的均匀无损管道具有所给定的转移函数。

我们注意到，对于式(11-30)所示的转移函数，输出指的是嘴唇处的空气体速度，但是，实际中使用压敏元件将声音信号转变成电信号，因此，测量的不是体速度，而是声压。为此，需要知道从体速度到声压的转换关系，即嘴唇的喷射作用。可以证明，嘴唇的喷射的声阻抗是一"感性"阻抗，因此，从体速度到声压是高通滤波关系。利用数字滤波器的理论，可以得喷射阻抗的近似离散形式为

$$L(z) = P_l(z)/U_l(z) = R_l(1 - z^{-1}) \tag{11-31}$$

因此，若以声压为输出，声道的转移函数为

$$V(z)L(z) = \frac{G(1 - z^{-1})}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{11-32}$$

在上式中，比例系数 R_l 归入到总增益 G 中。

五、发音系统的离散模型

考虑到式(11-3)所示的声门模型，得人的发音系统的离散模型如图 11.16 所示。

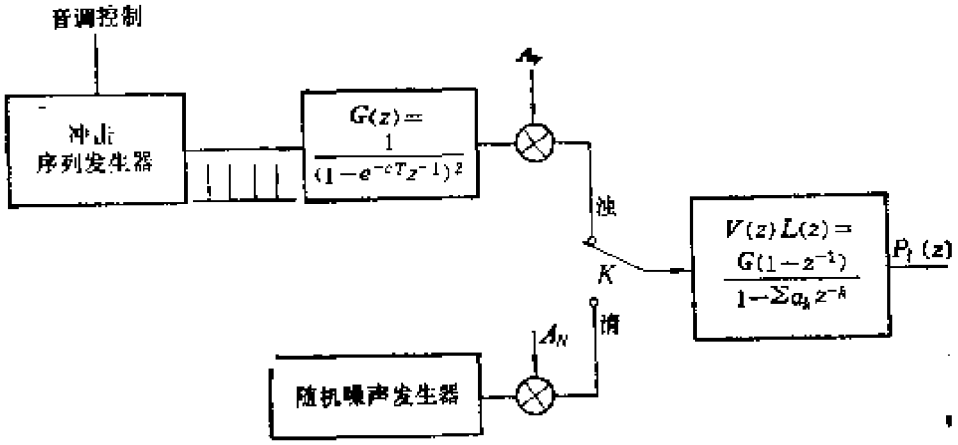


图 11.16　发音系统的离散模型

图中 A_V 是浊音音量控制；A_N 是清音音量控制；K 是清音浊音判别开关．随机噪声发生器就是一随机数发生器．

以上是应用最广泛的发音系统的离散模型．在进行语音识别时，正是根据这一模型，从采集到的语音推算出在各个时刻发音系统的参数 a_k 和激励的性质，以此作为语音的特征进行识别．相反，若已知发音系统的参数 a_k 和激励的性质，应用以上模型，就能合成自然性良好的语音．

应当指出，以上只是发音系统的一种模型，例如，下一节我们将看到，在进行线性预测 (LPC) 分析时，将把声门模型、声道模型、喷射阻抗串联起来，从而得到一个全极点的模型：

$$H(z) = G(z)V(z)L(z)$$

这样作将更为方便．

最后，还应注意到这一模型存在如下局限性：

(1) 这是一个终端模型，不能准确地描述发音系统内部的细节．

(2) 它只适用于瞬时语音 (20ms 区间内的语音) 和持续语音；对塞音的效果不十分理想，但仍可使用．

(3) 没有考虑鼻腔的影响．当发鼻音时由于口腔对某些频率

的吸收作用，声道模型具有零点；细致的分析表明，对摩擦音，$V(z)$ 中也应存在零点。但是，只要 P 足够大，声道的全部极点模型可以用来完满地表述所有语音。

（4）简单地把语音分成清音或浊音两个类别是不准确的，因为对浊擦音，摩擦只发生在体速度的峰值附近；又如，一般认为是浊音的韵母 ü，在气流的峰值附近也存在摩擦。由于以上原因把浊音激励简单地加起来也是不合适的。

§11.3 瞬时语音信号的时域特征(一)——LPC 系数

从本节开始，以三节的篇幅论述怎样抽取语音的特征。凡是对模式识别原理有基本了解的人都知道，抽取模式的特征是实现机器自动识别的关键。在一般的论述语音信号处理的著作里，把抽取语音特征称为语音分解，着眼点是对语音信号实现数据压缩，以便对语音信号进行传输或存储，或进行别的特殊处理。两种称呼指的是一回事，只不过目的不相同而已。

本节论述瞬时语音的线性预测系数。线性预测分析是最有效的语音分析技术，它之所以引起语音研究方面的人们的极大兴趣，是由于这种分析技术能从一组采集到的语音数据，用一种有效的时域递归算法迅速地得到相应的发音系统的精确的模型。它已广泛地用于语音信号处理和语音识别，并取得了很大的成功。随着计算机速度的提高和内存的加大，这种方法将越来越表现出它的广阔前景。

一、瞬时语音的线性预测系数

在这里把人的发音系统模型看成是图 11.17 所示的情况。

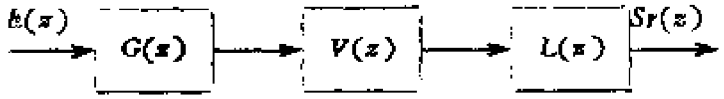


图 11.17　发音系统模型

图中，$E(z)$ 是激励信号，$S_r(z)$ 是语音．发音系统的转移函数为

$$H(z) = G(z)V(z)L(z)$$

$$= \frac{G(1-z^{-1})}{(1-e^{-cT}z^{-1})^2\left(1+\sum_{i=1}^{P} a_i z^{-i}\right)} \tag{11-33}$$

由于 G 是一常数，可以归入到激励信号中去；以及 $cT \ll 1$，所以可将 $H(z)$ 表示成

$$H(z) = \frac{1}{(1-z^{-1})\left(1+\sum_{i=1}^{P} a_i z^{-i}\right)}$$

$$= \frac{1}{(1-z^{-1})A(z)} \tag{11-34}$$

$$A(z) = \sum_{i=0}^{P} a_i z^{-i} \qquad a_0 = 1 \tag{11-35}$$

所以，发音系统是一个全极点模型．输出语音的 z 变换为

$$S_r(z) = \frac{1}{(1-z^{-1})A(z)} E(z) \tag{11-36}$$

式(11-36)称为语音的综合模型．语音的分析模型为

$$E(z) = A(z)(1-z^{-1})S_r(z)$$

$$= A(z)S(z) \tag{11-37}$$

$$S(z) = (1-z^{-1})S_r(z) \tag{11-38}$$

$S(z)$ 表示经过预强调后的语音．由分析模型知

$$e(n) = S(n) - \sum_{i=1}^{P} - a_i S(n-i)$$

$$= S(n) - \hat{S}(n) \tag{11-39}$$

$\hat{S}(n)$ 是过去的 P 个语音采样信号的线性组合。若把 $\hat{S}(n)$ 看成是对 $S(n)$ 的预测，那么，$e(n)$ 就是预测误差。P 称为线性预测的阶数。

所谓线性预测，指的是：对给定的由 N 个采样 $S(0), S(1), \cdots S(N-1)$ 组成的一个语音段落（称为一帧）和事先给定的阶数 P，确定系数 $A = [a_1, a_2, \cdots, a_P]^T$，使得如下指标函数取极小值。

$$\alpha_P(A) = \sum_n e^2(n) \tag{11-40}$$

A 称为线性预测系数，α_P 称为预测误差的平方和。由式(11-39)和(11-40)，有

$$\begin{aligned}
\alpha_P &= \sum_n e^2(n) \\
&= \sum_n \left[S(n) - \sum_{i=1}^{P} - a_i S(n-i)\right]^2 \\
&= \sum_n \left[S^2(n) + \left(\sum_{i=1}^{P} a_i S(n-i)\right)^2 \right. \\
&\quad \left. + 2S(n) \sum_{i=1}^{P} a_i S(n-i)\right] \\
&= \sum_{i=1}^{P} \sum_{j=1}^{P} \sum_n a_i S(n-i) S(n-j) a_j \\
&\quad + 2 \sum_{i=1}^{P} \sum_n a_i S(n) S(n-i) + \sum_n S^2(n) \\
&= A^T C A + 2 A^T C_0 + \sum_n S^2(n)
\end{aligned} \tag{11-41}$$

其中 C 是协方差矩阵：

$$\begin{cases}
C = [c_{ij}] \quad i, j = 1, 2, \cdots, P & \text{(a)} \\
c_{ij} = \sum_n S(n-i) S(n-j) & \text{(b)} \\
C_0 = [c_{01}, c_{02}, \cdots, c_{0P}]^T & \text{(c)}
\end{cases} \tag{11-42}$$

所以

$$\frac{\partial \alpha_P}{\partial A} = 2CA + 2C_0 = 0$$

于是，得求解线性预测系数的方程为

$$CA = -C_0 \tag{11-43}$$

即

$$\begin{bmatrix} c_{11}, c_{12}, \cdots, c_{1P} \\ c_{21}, c_{22}, \cdots, c_{2P} \\ \cdots\cdots\cdots\cdots \\ c_{P1}, c_{P2}, \cdots, c_{PP} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{bmatrix} = -\begin{bmatrix} c_{01} \\ c_{02} \\ \vdots \\ c_{0P} \end{bmatrix}.$$

C 是对称的 P 阶方阵。

按照计算 α_P 和 c_{ij} 的累加限的不同，线性预测分为如下两种不同的方法。

1. 协方差法

对协方差法，$S(n)$ 的取值范围是 $0 \leqslant n \leqslant N-1$。因此可小结如下：

线性联立方程组

$$\sum_{i=1}^{P} c_{ij} a_i = -C_{0j} \quad j = 1, 2, \cdots, P$$

$$c_{ij} = \sum_{n=P}^{N-1} S(n-i)S(n-j)$$

$$e(n) = \sum_{i=0}^{P} a_i S(n-i)$$

$$a_0 = 1; \quad n = P, P+1, \cdots, N-1$$

$$\alpha_P = \sum_{n=P}^{N-1} e^2(n)$$

2. 自相关法

对自相关法，$S(n)$ 是语音信号过窗后再经预强调的结果，窗

宽为N。 因此 $S(n)$ 的取值范围是 $|n| < \infty$，只不过在窗外 $S(n) = 0$ 而以。于是，对自相关法

$$\begin{aligned} c_{ij} &= \sum_{n=-\infty}^{\infty} S(n-i)S(n-j) \\ &= \sum_{n=-\infty}^{\infty} S(n)S(n+|i-j|) \\ &= \sum_{n=0}^{N-l-1} S(n)S(n+l) = r(l) \quad l = |i-j| \end{aligned} \tag{11-44}$$

因此，矩阵 C 为自相关矩阵：

$$C = R = \begin{bmatrix} r(0), r(1), \cdots, r(P-1) \\ r(1), r(0), \cdots, r(P-2) \\ r(2), r(1), \cdots, r(P-3) \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ r(P-1), r(P-2), \cdots, r(0) \end{bmatrix} \tag{11-45}$$

$$C_0 = [r(1), r(2), \cdots, r(P)]^T$$

注意到 R 矩阵不仅是对称方阵，而且，它的任意一条对角线上的元素相等；这样的方阵称为托也勃利兹 (Toeplits) 矩阵。正是由于 R 具有这样的性质，对于自相关法求解 A 的过程才可以简化。

对自相关法小结如下：

线性联立方程组

$$\sum_{i=1}^{P} r(|i-j|)a_i = -r(j) \qquad j = 1, 2, \cdots, P$$

$$r(l) = \sum_{n=0}^{N-l-1} S(n)S(n+l) \qquad l \geqslant 0$$

$$e(n) = \sum_{i=0}^{P} a_i S(n-i) \qquad a_0 = 1;$$

$$n = 0, 1, \cdots, N+P-1$$

$$\alpha_p = \sum_{n=0}^{N+p-1} e^2(n)$$

以后将会看到，用自相关法得到的滤波器 $1/A(z)$ 在原理上是稳定的，而协方差法则不具备这性质，不过，当语音采样点的数目 N 足够大时，这两种方法的结果趋于一致，用协方差法得到的滤波器的稳定性仍可得到保证。

二、正交原理

为求线性预测系数，原则上解相应的线性联立方程组即可，但是，由于矩阵 C 所具备的特有性质，我们能建立比线性联立方程组的一般解效率高得多的解法。作为寻求这一解法的理论准备，先讨论所谓的正交原理。

所有形如

$$A_p(z) = \sum_{i=0}^{p} a_i z^{-i} \qquad a_0 = 1, \quad a_i \text{ 为实数}$$

的 m 阶多项式的集合构成一多项式线性空间 Q_p。设 $\boldsymbol{F}(z)$，$\boldsymbol{G}(z)$ 是这空间的任意二向量*，定义它们的内积为

$$\langle \boldsymbol{F}(z), \boldsymbol{G}(z) \rangle = \sum_{n=n_0}^{n_1} U(n)V(n) \qquad (11\text{-}46)$$

其中 $U(n)$、$V(n)$ 分别是以 $S(n)$ 为输入，$\boldsymbol{F}(z)$、$\boldsymbol{G}(z)$ 为转移函数的输出系列，如图 11.18 所示。

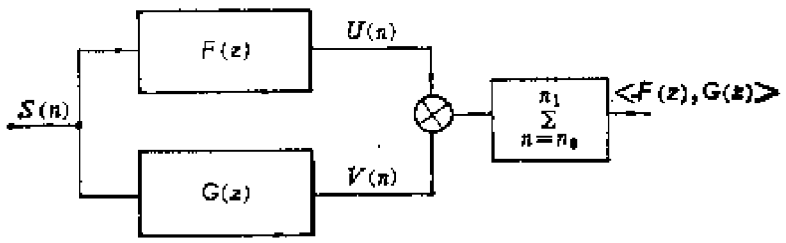


图 11.18 $\boldsymbol{F}(z)$、$\boldsymbol{G}(z)$ 的内积

* 由于这原因，在本节把 z 的函数表示成向量的形式。

显然，$\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle$ 是一实数，其大小和输入 $S(n)$ 有关。以上定义的内积具有如下性质：

(1) $\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle = \langle \boldsymbol{G}(z), \boldsymbol{F}(z)\rangle$

(2) $\|\boldsymbol{F}(z)\|^2 = \langle \boldsymbol{F}(z), \boldsymbol{F}(z)\rangle \geqslant 0$

(3) g, h 是任意二常数

$$\langle \boldsymbol{F}(z), g\boldsymbol{G}(z) + h\boldsymbol{H}(z)\rangle = g\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle + h\langle \boldsymbol{F}(z), \boldsymbol{H}(z)\rangle$$

(4) 若 $\|\boldsymbol{F}(z)\|^2 = 0$，则有

i $\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle = 0$

ii $\|\boldsymbol{F}(z) + \boldsymbol{G}(z)\|^2 = \|\boldsymbol{G}(z)\|^2$

iii 当 $n_0 = -\infty$, $n = \infty$ 时

$$\boldsymbol{F}(z) = 0$$

(5) 满足柯西-许瓦尔兹不等式

$$|\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle| \leqslant \|\boldsymbol{F}(z)\| \cdot \|\boldsymbol{G}(z)\|$$

证明：

若 $\|\boldsymbol{F}(z)\| = 0$，以上结论是显然的。

设 $\|\boldsymbol{F}(z)\| \neq 0$，则有

$$\|a\boldsymbol{F}(z) - \boldsymbol{G}(z)\|^2 = a^2\|\boldsymbol{F}(z)\|^2 + \|\boldsymbol{G}(z)\|^2 - 2a\langle \boldsymbol{F}(z), G(z)\rangle \geqslant 0$$

令

$$a = \langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle / \|\boldsymbol{F}(z)\|^2$$

所以

$$\|a\boldsymbol{F}(z) - \boldsymbol{G}(z)\|^2 = \frac{(\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle)^2}{\|\boldsymbol{F}(z)\|^2} + \|\boldsymbol{G}(z)\|^2 - 2\frac{(\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle)^2}{\|\boldsymbol{F}(z)\|^2}$$

$$= \|\boldsymbol{G}(z)\|^2 - \frac{(\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle)^2}{\|\boldsymbol{F}(z)\|^2} \geqslant 0$$

于是

$$|\langle \boldsymbol{F}(z), \boldsymbol{G}(z)\rangle| \leqslant \|\boldsymbol{F}(z)\| \cdot \|\boldsymbol{G}(z)\|$$

由于以上这些性质，式(11-46)关于内积的定义是可行的．多项式空间 Ω_P 是欧几里德空间．

由内积定义：

$$\langle z^{-i}, z^{-j}\rangle = \sum_n S(n-i)S(n-j) = c_{ij} \qquad (11\text{-}47)$$

对自相关法：

$$\langle z^{-i}, z^{-j}\rangle = r(|i-j|) \qquad (11\text{-}48)$$

设 $\boldsymbol{A}_P(z)$ 是 P 阶多项式，则

$$\alpha_P = \langle \boldsymbol{A}_P(z), \boldsymbol{A}_P(z)\rangle = \|\boldsymbol{A}_P(z)\|^2 \qquad (11\text{-}49)$$

因此，线性预测问题就是选择 $\boldsymbol{A}_P(z)$ 的系数 a_{Pi}，$i = 1, 2, \cdots, P$，使得 $\boldsymbol{A}_P(z)$ 的模平方取极小值．

如图 11.19 所示，所谓线性预测，是用 $S(r-i)$，$i = 1, 2, \cdots, P$ 的线性组合 $\sum_{i=1}^{P} -a_{Pi}S(n-i)$ 来预测 $S(n)$．这样的预测称为正向预测，预测误差

$$e_P^+ = \sum_{i=0}^{P} a_{Pi}S(n-i) \qquad a_{P0} = 1 \qquad (11\text{-}50)$$

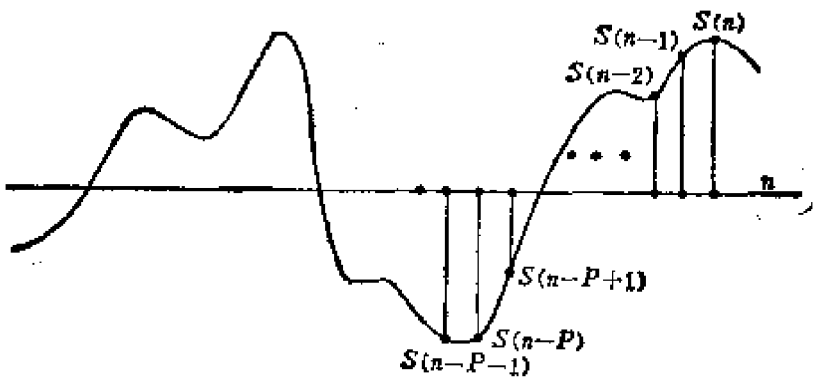


图 11.19　预测示意图

称为正向预测误差，而且

$$\alpha_P = \sum_n [e_P^+(n)]^2 = \|\boldsymbol{A}_P(z)\|^2 \qquad (11\text{-}51)$$

同理，我们可以用 $S(n-i)$（其中，$i=1, 2, \cdots, P$）的另一线性组合 $\sum_{i=1}^{P} -b_{Pi}S(n-i)$ 预测 $S(n-P-1)$，这称为反向预测，预测误差

$$e_P^-(n) = \sum_{i=1}^{P+1} b_{Pi}S(n-i) \quad b_{P,P+1} = 1 \qquad (11\text{-}52)$$

称为反向预测误差，同样可以定义反向预测误差的平方和为

$$\beta_P = \sum_n [e_P^-(n)]^2 = \|\boldsymbol{B}_P(z)\|^2 \qquad (11\text{-}53)$$

$$\boldsymbol{B}_P(z) = \sum_{i=1}^{P+1} b_{Pi}z^{-i} \quad b_{P,P+1} = 1 \qquad (11\text{-}54)$$

同样，可以选择系数 b_{Pi}，使得 β_P 取极小值，从而得到最优反向线性预测。

设对给定的一帧语音采样，$\boldsymbol{A}_P(z)$ 已使 $\|\boldsymbol{A}_P(z)\|^2$ 取极小值，那么，对于任意给定的常数 K，必有

$$\|\boldsymbol{A}_P(z) + Kz^{-j}\|^2 \geqslant \|\boldsymbol{A}_P(z)\|^2 \quad j = 1, 2, \cdots, P \qquad (11\text{-}55)$$

将上式展开，得

$$2K\langle \boldsymbol{A}_P(z),\ z^{-j}\rangle + K^2\|z^{-j}\|^2 \geqslant 0 \qquad (11\text{-}56)$$

若 $\|z^{-j}\|^2 = 0$，则令

$$K = -\langle \boldsymbol{A}_P(z);\ z^{-j}\rangle$$

将 K 值代入(11-56)，得

$$-2[\langle \boldsymbol{A}_P(z), z^{-j}\rangle]^2 \geqslant 0$$

所以

$$\langle \boldsymbol{A}_P(z), z^{-j}\rangle = 0 \qquad j = 1, 2, \cdots, P \qquad (11\text{-}57)$$

若 $\|z^{-j}\|^2 \neq 0$，可令

$$K = -\langle \boldsymbol{A}_P(z), z^{-j}\rangle / \|z^{-j}\|^2$$

将这 K 值代入(11-56),得

$$-[\langle \boldsymbol{A}_P(z), z^{-i}\rangle]^2 \geqslant 0$$

同样可得

$$\langle \boldsymbol{A}_P(z), z^{-i}\rangle = 0, \quad i = 1, 2, \cdots, P$$

因此，$\boldsymbol{A}_P(z)$ 是 P 阶线性预测逆滤波器的必要条件是 $\boldsymbol{A}_P(z)$ 和 z^{-i}（其中 $i = 1, 2, \cdots, P$）正交.

下面证明以上正交条件的充分性.

设 $\boldsymbol{A}_P(z)$ 和 z^{-i}（其中 $i = 1, 2, \cdots, P$）正交，而且存在 $\boldsymbol{A}'_P(z)$ 满足 $\|\boldsymbol{A}'_P(z)\| \leqslant \|\boldsymbol{A}_P(z)\|^2$. 显然，可将 $\boldsymbol{A}'_P(z)$ 表示成

$$\boldsymbol{A}'_P(z) = \boldsymbol{A}_P(z) + \boldsymbol{G}(z)$$

$$\boldsymbol{G}(z) = \sum_{i=1}^{P} g_i z^{-i}$$

于是

$$\|\boldsymbol{A}'_P(z)\|^2 = \|\boldsymbol{A}_P(z)\|^2 + \|\boldsymbol{G}(z)\|^2 + 2\langle \boldsymbol{A}_P(z), \boldsymbol{G}(z)\rangle$$

$$= \|\boldsymbol{A}_P(z)\|^2 + \|\boldsymbol{G}(z)\|^2$$

由于 $\|\boldsymbol{G}(z)\|^2 \geqslant 0$，并考虑到一开始的假设，故必有

$$\boldsymbol{A}'_P(z) = \boldsymbol{A}_P(z)$$

充分性得证.

这不就是关于正向线性预测系数的方程组吗. 同理,关于 $B_p(z)$ 的正交性原理,实质上就是反向线性预测系数的联立方程组.

三、LPC 方程的 Gram-Schmidt 递归解法

Gram-Schmidt 递归解法无论对自相关法还是协方差法都是适用的.

已知

$$A_p(z) = \sum_{i=0}^{p} a_{pi} z^{-i} \quad a_{p0} = 1$$

$$B_p(z) = \sum_{i=1}^{p+1} b_{pi} z^{-i} \quad b_{p,p+1} = 1$$

$$\langle A_p(z), z^{-j} \rangle = \langle B_p(z), z^{-j} \rangle = 0 \qquad j = 1, 2, \cdots, p$$

设 $A_p(z)$ 和 $B_p(z)$ 已经得到,现在讨论怎样从 $A_p(z)$ 和 $B_p(z)$ 得到 $A_{p+1}(z)$ 和 $B_{p+1}(z)$. 因为 $A_p(z)$ 和 $B_p(z)$都和 z^{-j}正交,而且 $A_p(z)$ 和 $B_p(z)$ 线性无关,所以由 $A_p(z)$ 和 $B_p(z)$ 的线性组合构成的二维子空间和 z^{-j} (其中 $j=1, 2, \cdots, p$) 正交. 显然,向量 $z^{-(p+1)}$ 不在这二维子空间里,于是,在这二维子空间中必可找到如下向量

$$A_{p+1}(z) = A_p(z) + K_{p+1} B_p(z) \qquad (11\text{-}58)$$

和 $z^{-(p+1)}$ 正交,从而实现 $A_{p+1}(z)$ 和 z^{-j} (其中 $j=1, 2, \cdots, p+1$) 正交. 式(11-58)中 K_{p+1} 是待定系数,称为 PARCOR 系数(部分相关系数). 现在确定 K_{p+1} 的值.

$$\beta_p = \|B_p(z)\|^2$$

$$= \left\langle B_p(z), \sum_{i=1}^{p+1} b_{pi} z^{-i} \right\rangle$$

$$= \langle B_p(z), z^{-(p+1)} \rangle$$

$$= -\sum_{i=1}^{P+k} b_{Pi}c_{P+1,i} \tag{11-59}$$

按正交原理

$$\langle \boldsymbol{A}_{P+1}(z), z^{-(P+1)}\rangle = \langle \boldsymbol{A}_P(z) + K_{P+1}\boldsymbol{B}_P(z), z^{-(P+1)}\rangle$$
$$= \langle \boldsymbol{A}_P(z), z^{-(P+1)}\rangle + K_{P+1}\beta_P = 0$$

所以

$$K_{P+1} = -\langle \boldsymbol{A}_P(z), z^{-(P+1)}\rangle/\beta_P$$
$$= \frac{-1}{\beta_P}\sum_{i=0}^{P} a_{Pi}c_{P+1,i} \tag{11-60}$$

为实现递归还必须确定 $\boldsymbol{B}_{P+1}(z)$。按正交原理，只要 $l \neq k$ 就有

$$\langle \boldsymbol{B}_l(z), \boldsymbol{B}_k(z)\rangle = 0$$

从而可把 $\boldsymbol{B}_0(z), \boldsymbol{B}_1(z), \cdots, \boldsymbol{B}_P(z)$ 取作由形如 $\sum_{i=1}^{P+1} b_i z^{-i}$ 的多项式构成的空间的基。于是，可将 $\boldsymbol{B}_{P+1}(z)$ 表示成

$$\boldsymbol{B}_{P+1}(z) = z^{-(P+2)} + \sum_{i=1}^{P+1} b_{P+1,i}z^{-i}$$
$$= z^{-(P+2)} - \sum_{i=0}^{P} r_{P+1,i}\boldsymbol{B}_i(z) \tag{11-61}$$

式中 $r_{P+1,i}$ 是待定常数，它必须使 $\boldsymbol{B}_{P+1}(z)$ 满足正交条件：

$$\langle \boldsymbol{B}_j(z), \boldsymbol{B}_{P+1}(z)\rangle = \left\langle \boldsymbol{B}_j(z), z^{-(P+2)} - \sum_{i=1}^{P} r_{P+1,i}\boldsymbol{B}_i(z)\right\rangle$$
$$= \langle \boldsymbol{B}_j(z), z^{-(P+2)}\rangle - \left\langle \boldsymbol{B}_j(z), \sum_{i=1}^{P} r_{P+1,i}\boldsymbol{B}_i(z)\right\rangle$$
$$= \langle \boldsymbol{B}_j(z), z^{-(P+2)}\rangle - r_{P+1,j}\beta_j$$
$$= 0 \qquad j = 0, 1, 2, \cdots, P \tag{11-62}$$

由(11-62)知，若 $\beta_j = 0$，那么 $r_{P+1,j}$ 可为任意值；若 $\beta_j \neq 0$，则

$$r_{P+1,j} = \frac{1}{\beta_j}\langle \boldsymbol{B}_j(z), z^{-(P+2)}\rangle$$

$$= \frac{1}{\beta_j}\sum_{i=1}^{j+1} c_{P+2,i}b_{ji} \quad j = 0, 1, 2, \cdots, P \qquad (11\text{-}63a)$$

将上式代入式(11-61)得 $\boldsymbol{B}_{P+1}(z)$ 的系数为

$$b_{P+1,P+2} = 1$$

$$b_{P+1,i} = -\sum_{j=0}^{P} r_{P+1,j}b_{ji} = -\sum_{j=i-1}^{P} r_{P+1,j}b_{ji}$$

$$i = 1, 2, \cdots, P+1 \qquad (11\text{-}63b)$$

LPC 方程的 Gram Schmidt 递归解法小结如下：

$$\boldsymbol{A}_0(z) = 1, \quad \boldsymbol{B}_0(z) = z^{-1}, \quad a_0 = c_{00}, \quad \beta_0 = c_{11}$$

$$\beta_P = \sum_{i=1}^{P+1} b_{Pi}c_{P+1,i} \qquad (11\text{-}59)$$

$$K_{P+1} = \frac{-1}{\beta_P}\sum_{i=0}^{P} a_{P,i}c_{P+1,i} \qquad (11\text{-}60)$$

$$\boldsymbol{A}_{P+1}(z) = \boldsymbol{A}_P(z) + K_{P+1}\boldsymbol{B}_P(z) \qquad (11\text{-}58)$$

$$r_{P+1,j} = \frac{1}{\beta_j}\sum_{i=1}^{j+1} c_{P+2,i}b_{ji} \quad j = 1, 2, \cdots, P \qquad (11\text{-}63a)$$

$$b_{P+1,i} = -\sum_{j=i-1}^{P} r_{P+1,j}b_{ji} \quad i = 1, 2, \cdots, P+1 \qquad (11\text{-}63b)$$

$$\boldsymbol{B}_{P+1}(z) = z^{-(P+2)} + \sum_{i=1}^{P+1} b_{P+1}z^{-i} \qquad (11\text{-}61)$$

正向预测误差的平方和 α_P 也可递归计算。由式(11-58)可以得

$$\boldsymbol{A}_P(z) = 1 + \sum_{i=1}^{P} K_i\boldsymbol{B}_{i-1}(z)$$

所以按正交原理，有

$$\|\boldsymbol{A}_P(z) - \boldsymbol{1}\|^2 = \|\boldsymbol{A}_P(z)\|^2 + \|\boldsymbol{1}\|^2 - 2\langle \boldsymbol{A}_P(z), 1\rangle$$

$$= \alpha_P - 2\alpha_P + \|\boldsymbol{1}\|^2 = \|\boldsymbol{1}\|^2 - \alpha_P$$

$$\|\boldsymbol{A}_P(z)-\mathbf{1}\|^2=\left\|\sum_{i=1}^{P}K_i\boldsymbol{B}_{i-1}(z)\right\|^2=\sum_{i=1}^{P}K_i^2\beta_{i-1}$$

因此

$$\alpha_P=\|\mathbf{1}\|^2-\sum_{i=1}^{P}K_i^2\beta_{i-1}$$

同理

$$\alpha_{P+1}=\|\mathbf{1}\|^2-\sum_{i=1}^{P+1}K_i^2\beta_{i-1}=\|\mathbf{1}\|^2-\sum_{i=1}^{P}K_i^2\beta_{i-1}-K_{P+1}^2\beta_P$$

$$=\alpha_P-K_{P+1}^2\beta_P \qquad (11\text{-}64)$$

四、自相关 LPC 方程的 Levinson 递归解法

对自相关法，递归过程可以大大简化。和 Gram-Schmidt 递归同理

$$\boldsymbol{A}_{P+1}(z)=\boldsymbol{A}_P(z)+K_{P+1}\boldsymbol{B}_P(z)$$

$$K_{P+1}=\frac{-1}{\beta_P}\langle\boldsymbol{A}_P(z),z^{-(P+1)}\rangle$$

但是，对自相关法存在

$$\boldsymbol{B}_P(z)=z^{-(P+1)}\boldsymbol{A}_P(1/z) \qquad (11\text{-}65)$$

$$\alpha_P=\beta_P \qquad (11\text{-}66)$$

$$\alpha_{P+1}=\alpha_P(1-K_{P+1}^2) \qquad (11\text{-}67)$$

因此递归过程变得十分简单。

现证明关系式(11-65)。关于 a_{Pi} 的线性联立方程组为

$$\langle\boldsymbol{A}_P(z),z^{-j}\rangle=\sum_{i=0}^{P}a_{Pi}\langle z^{-i},z^{-j}\rangle$$

$$=\sum_{i=0}^{P}a_{Pi}r(|i-j|)=0$$

$$j=1,2,\cdots,P$$

令：$l=P+1-j$，$k=P+1-i$，以上方程组变为

$$\sum_{k=1}^{P+1} a_{P,P+1-k} r(|l-k|) = 0 \qquad l = 1, 2, \cdots, P$$

无任何实质性的变化，将方程改写为

$$\sum_{i=1}^{P+1} a_{P,P+1-i} r(|i-j|) = 0 \qquad j = 1, 2, \cdots, P$$

而关于 b_{Pi} 的线性联立方程组为

$$\langle \boldsymbol{B}_P(z), z^{-j} \rangle = \sum_{i=1}^{P+1} b_{Pi} r(|i-j|) = 0$$

$$j = 1, 2, \cdots, P$$

比较这两个联立方程组，得

$$b_{Pi} = a_{P,P+1-i}$$

因此 $\boldsymbol{B}_P(z)$ 为

$$\begin{aligned}\boldsymbol{B}_P(z) &= a_{PP}z^{-1} + a_{P,P-1}z^{-2} + \cdots + a_{P,2}z^{-(P-1)} \\ &\quad + a_{P,1}z^{-P} + z^{-(P+1)} = z^{-(P+1)} \sum_{i=0}^{P} a_{Pi} z^{i} \\ &= z^{-(P+1)} \boldsymbol{A}(1/z)\end{aligned}$$

以下证明 $\alpha_P = \beta_P$. 按正交原理，有

$$\begin{aligned}\alpha_P &= \langle \boldsymbol{A}_P(z), \boldsymbol{A}_P(z) \rangle = \langle \boldsymbol{A}_P(z), z^0 \rangle \\ &= \sum_{i=0}^{P} a_{Pi} r(i)\end{aligned}$$

而

$$\begin{aligned}\beta_P &= \langle \boldsymbol{B}_P(z), \boldsymbol{B}_P(z) \rangle = \langle \boldsymbol{B}_P(z), z^{-(P+1)} \rangle \\ &= \left\langle \sum_{i=0}^{P} a_{Pi} z^{-(P+1)+i}, z^{-(P+1)} \right\rangle \\ &= \sum_{i=0}^{P} a_{Pi} \langle z^{-(P+1)+i}, z^{-(P+1)} \rangle \\ &= \sum_{i=0}^{P} a_{Pi} r(i) = \alpha_P\end{aligned}$$

考虑到关于 α_P 的递归公式(11-64)，有 $\alpha_{P+1} = \alpha_P(1 - K_{P+1}^2)$.

对 Levinson 递归小结如下：

$$A_0(z) = 1, \quad \alpha_0 = r(0)$$

$$K_{p+1} = -\frac{1}{\alpha_p}\langle A_p(z), z^{-(p+1)}\rangle$$

$$= -\frac{1}{\alpha_p}\sum_{i=0}^{p} a_{p,i} r(|p+1-i|)$$

$$A_{p+1}(z) = A_p(z) + K_{p+1} z^{-(p+1)} A_p(1/z)$$

$$\alpha_{p+1} = \alpha_p(1 - K_{p+1}^2)$$

五、线性预测滤波器和发音系统的无损管道模型间的关系

对于自相关法，由 Levinson 递归公式

$$K_{p+1} = \frac{-1}{\alpha_p}\langle A_p(z), z^{-(p+1)}\rangle$$

$$= \frac{-1}{\sqrt{\alpha_p \beta_p}}\langle A_p(z), B_p(z)\rangle$$

$$= \frac{-\sum_{n=0}^{N-1} e_p^+(n) e_p^-(n)}{\left[\left(\sum_{n=0}^{N+p-1} e_p^{+2}(n)\right)\left(\sum_{n=-p}^{N-1} e_p^{-2}(n)\right)\right]^{\frac{1}{2}}} \quad (11\text{-}68)$$

上式表明，K_{p+1} 是正向预测误差和反向预测误差的归一化互相关系数，这就是 PARCOR (Partial Correlation) 这名称的由来。从式(11-68)得

$$-1 \leqslant K_p \leqslant 1$$

从 Levinson 递归可以清楚地看到，给定了 PARCOR 系数序列 $K_1, K_2, \cdots, K_p$，实质上就是给定了 $A_p(z)$。所以 PARCOR 系数 $K_1, K_2, \cdots, K_p$ 和线性预测系数是等价的。

在 § 11.2 (四)论述声道的离散模型时曾指出过，声道的转移函数为

$$V(z)=\frac{U_1(z)}{U_g(z)}=\frac{G}{D_p(z)}$$

$$D_p(z)=[1,\ -r_g]\begin{bmatrix}1, & -r_1\\ -r_1z^{-1}, & z^{-1}\end{bmatrix}\cdots$$

$$\begin{bmatrix}1, & -r_p\\ -r_pz^{-1}, & z^{-1}\end{bmatrix}\cdot\begin{bmatrix}1\\ 0\end{bmatrix}$$

式中 r_i 是反射系数；G 是声道的增益，可以将它归入到激励信号中加以考虑。设 $r_g=1$（相当于声门损耗为零的情况），在这种情况下，$D_p(z)$ 可以递归求得

令

$$M_1=[1,\ -1]\begin{bmatrix}1, & -r_1\\ -r_1z^{-1}, & z^{-1}\end{bmatrix} \tag{11-69}$$

$$M_p=M_{p-1}\begin{bmatrix}1, & -r_p\\ -r_pz^{-1}, & z^{-1}\end{bmatrix} \tag{11-70}$$

于是

$$M_1=[(1+r_1z^{-1}),\ -z^{-1}(1+r_1z)]$$
$$=[D_1(z),\ -z^{-1}D_1(z^{-1})]$$

$$M_2=M_1\begin{bmatrix}1, & -r_2\\ -r_2z^{-1}, & z^{-1}\end{bmatrix}=[D_2(z),\ -z^{-2}D_2(z^{-1})]$$

$$D_2(z)=D_1(z)+r_2z^{-2}D_1(z^{-1})$$

利用数学归纳法可以证明

$$M_p=M_{p-1}\begin{bmatrix}1, & -r_p\\ -r_pz^{-1}, & z^{-1}\end{bmatrix}$$
$$=[D_p(z),\ -z^{-p}D_p(z^{-1})] \tag{11-71}$$

$$D_p(z)=D_{p-1}(z)+r_pz^{-p}D_{p-1}(z^{-1}) \tag{11-72}$$

式(11-72)就是求 $D_p(z)$ 的逆归公式。比较式(11-72)和自相关法求 $A_p(z)$ 的递归公式，立即可以发现，$A_p(z)$ 就是 $D_p(z)$，PARCOR 系数 K_p 就是反射系数 r_p。所以，用自相关法计算线

性预测滤波器，实质上就是根据采集到的语音反算发出这语音的 P 段级联无损管道的模型。

由于 K_P 就是反射系数，所以，由式(11-14)得计算声道横断面积的递归公式为

$$A_P = \frac{1 + K_{P-1}}{1 - K_{P-1}} A_{P-1} \tag{11-73}$$

因为对自相关法 $|K_P| \leqslant 1$，由式(11-73)知，只要 $A_1 > 0$，就可以保证 $A_P > 0$. 因此和 $1/A_P(z)$ 对应的 P 段级联均匀无损管道在物理上可实现. 一个无源有损耗线性系统必稳定. 这就是由自相关法得到的 LPC 滤波器必定稳定的原因。

六、窗宽和阶数 P 的选择

由 LPC 滤波器和声道的离散模型间的关系，LPC 的阶数 P 就是级联均匀无损管道的数目，显然 P 越大级联管道越接近于真实的声道形状. 但是在进行递归时，乘法的次数和 P^2 成正比，因此，实际应用时怎样合理地选择阶次 P 是人们关心的问题。

由于 LPC 系数是实数，因此，可将 LPC 滤波器表示成共轭极点的形式：

$$\frac{1}{A(z)} = \frac{1}{\prod_{i=1}^{K} [1 - 2e^{-C_i T} z^{-1} \cos b_i T + e^{-2C_i T} z^{-2}]} \tag{11-74}$$

其中每一对共轭极点代表声道的一个共振峰，共振峰的频率为

$$f_i = b_i / 2\pi \tag{11-75}$$

共振峰的带宽近似为

$$B_i = C_i / \pi \tag{11-76}$$

实验表明，第一共振峰在 $f_1 = 500$Hz 左右；共振峰间的间隔大约是 1000Hz，前三个共振峰是主要的。例如，希望线性预测滤波器能反映声道的前四个共振峰，P 至少应当是 8。为了使所得

的第四共振峰满足一定的精度要求，P 应当比 8 更大一些，例如取 $P=10$。

若定义自相关法的归一化均方根预测误差为

$$r_{ms}=\left(\sum_{n=0}^{N+P-1}e^2(n)\Big/\sum_{n=0}^{N-1}S^2(n)\right)^{\frac{1}{2}}=\left(\frac{\alpha_P}{\alpha_0}\right)^{\frac{1}{2}}$$

$$=\left(\frac{\alpha_0\prod_{i=1}^{P}(1-K_i^2)}{\alpha_0}\right)^{\frac{1}{2}}=\prod_{i=1}^{P}(1-K_i^2)^{\frac{1}{2}}\quad(11\text{-}77)$$

由于 $(1-K_i^2)^{\frac{1}{2}}\leqslant 1$，所以加大 P 将使得 r_{ms} 降低。图 11.20 是 r_{ms} 和阶次 P 间的关系曲线。由这曲线清楚见到，随着 P 的增加，一开始 r_{ms} 迅速减小，但是当进一步加大 P 时曲线慢慢变得平坦起来，当 P 加大到 14 以后，再加大 P 对减小 r_{ms} 效果极小，因此 P 值一般不超过 14。由这曲线还可以看到，对清音的预测误差远大于对浊音的预测误差。这是因为用全极点模型来描述发清音的声道效果较差，而且对清音，图 11.17所示的发音系统模型中 $G(z)$ 是人为地强加的，这也是对清音预测精度较差的一个原因。

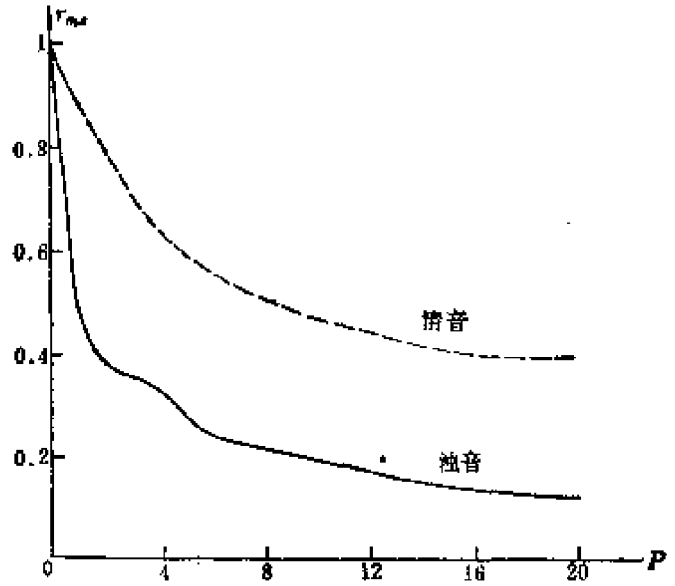


图 11.20　归一化均方根预测误差和预测阶次的关系

最后讨论怎样选择时窗的宽度 N. 由于发音系统的时变性，只有在一个短的时间间隔里（例如 20ms 以内）才能把它看成是定常的，这是进行 LPC 分析的前提. 由于这个原因，一般不希望时窗的宽度大于 20ms.

如果时窗宽度选得过小，则会产生一种"位置效应"，即对于持续音，由于窗的位置不同，在理论上将得到不同的 LPC 系数. 用相关匹配的原理可以证明，对于浊音只要窗的宽度大于二倍音调周期，就不会产生"位置效应". 但是考虑到窗边缘部分的衰减作用，窗宽应比二倍音调周期略大. 实验表明，对绝大多数人基频范围为 100Hz ～ 400Hz，为了消除"位置效应"，窗宽应当大于 20 ms.

克服系统的时变性的要和克服"位置效应"的要求往往是互相矛盾的，实际工作中需根据任务的要求折中.

§11.4 瞬时语音信号的时域特征(二)
——强度、过零率、基频

一、瞬时语音的能量和平均幅度

设 $S(t)$ 是语音信号，语音在 t 时刻的能量定义为

$$E(t)=\int_{t-T}^{t}S^2(\tau)d\tau=\int_{-\infty}^{\infty}S^2(\tau)W^2(t-\tau)d\tau \quad (11\text{-}78)$$

其中 $W(t)$ 是窗函数：

$$W(t)=\begin{cases}1 & 0\leqslant t\leqslant T\\ 0 & \text{其它}\end{cases} \quad (11\text{-}79)$$

对于离散情况，可将(11-78)和(11-79)改写为

$$E(n)=\sum_{m=-\infty}^{\infty}S^2(m)W^2(n-m)=S^2(n)*W^2(n) \quad (11\text{-}80)$$

$$W(n)=\begin{cases}1 & 0\leqslant n\leqslant N\\ 0 & \text{其他}\end{cases} \tag{11-81}$$

式(11-80)中"*"表示求褶积。因此语音的瞬时能量指的是以语音的平方 $S^2(n)$ 为输入，以窗函数的平方 $W^2(n)$ 为冲击响应的环节的输出。以上是矩形窗的情况，完全可以推广到任意时窗的情况。众所周知，以窗函数的平方为冲击响应的环节是一个低通滤波器，所以语音的瞬时能量是语音信号平方后的低通滤波输出，如图 11.21 所示。为了使输出 $E(t)$ 能正确反应语音的瞬时能量的变化情况，在设计低通滤波器时一般将其冲击响应时间选择为 20 ms 左右。而且将其转移函数的极点设计在实轴上，以保证冲击响应始终保持为正。

图 11.21　语音的瞬时能量

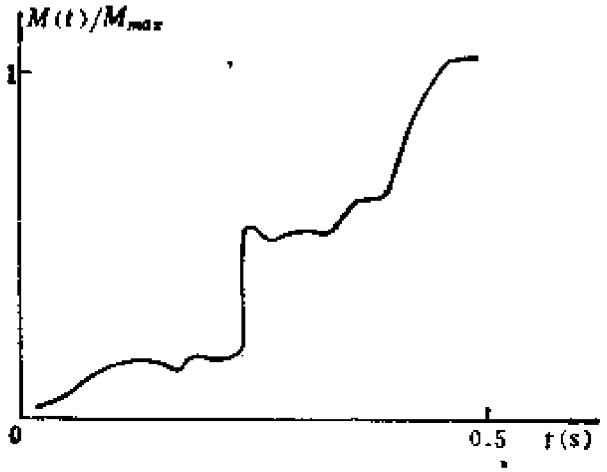


图 11.22　"史"字语音的幅度曲线

语音的瞬时平均幅度函数定义为

$$M(t)=\int_{-\infty}^{\infty}|S(\tau)|W(t-\tau)d\tau=|S(t)|*W(t) \tag{11-82a}$$

$M(t)$ 的含义是在 t 时刻语音的幅度。对于离散情况：

$$M(n) = |S(n)| * W(n) \qquad (11\text{-}82b)$$

因此，只要将语音信号整流然后进行低通滤波，滤波器的输出就是语音的瞬时幅度。

图 11.22 是“史”字语音的瞬时平均幅度。由图 11.22 可以清楚地看到清音的音量远小于浊音的音量，因此语音的瞬时能量或瞬时平均幅度函数主要是用作判别语音的“清-浊-静(无声)”的参数之一。

二、语音信号的过零率

过零率是语音信号的谱分布的粗略描述，指的是在窗口范围内语音信号经过零点的次数，如图 11.23 所示。

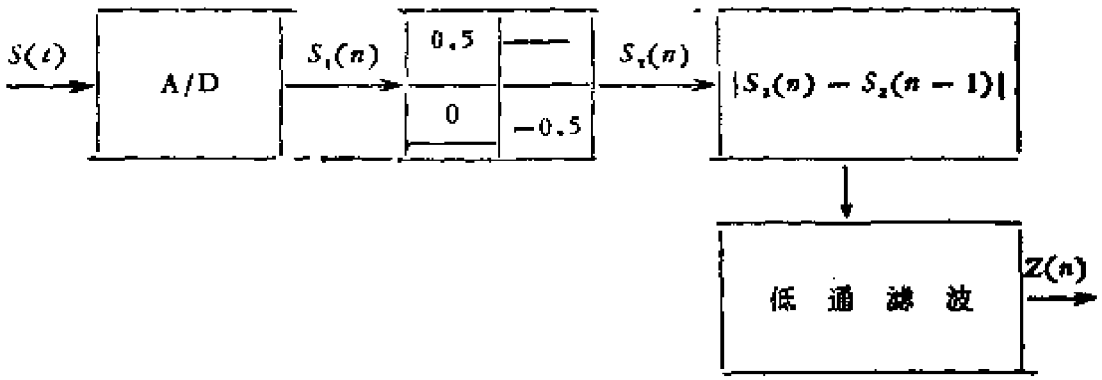


图 11.23 语音信号的过零率

由于浊音的激励是准周期性的脉冲，因此，浊音的音量大，谱分布集中在低频段；清音的激励是白噪声，因此，清音的音量小，谱分布很宽。由于两种语音的谱分布不同，反应在过零率上，清音的过零率比浊音的过零率高。实验表明，若窗宽是 20ms，浊音的过零率主要分布在 60 以下，清音的过零率主要分布在 60 以上。

过零率是对语音进行“清-浊-静”判别的另一主要参数。例如，对孤立词语音识别系统，常常用如下方法判别语音的开始（或结

束)。

如果

$$(Z(n) > \theta_1) \wedge (M(n) > \theta_2) \vee (M(n) > \theta_3) = T$$

那么，语音的起始点为 $n - N - 1$. 其中，$Z(n)$ 是语音的过零率，$M(n)$ 是语音的瞬时幅度，N 是窗宽，θ_1，θ_2，θ_3 是阈值。

三、瞬时语音的基频

瞬时语音的基频指的是在窗规定的区间里声带的振动频率。它是实现语音识别和识别说话人的重要特征；对于语音合成，它是必不可少的参数，尤其是对于有调语音，因为声调具有辨意作用，因此，提取语音基频更具有特别的意义。由于其重要性，因此，发展了多种提取语音基频的方法，例如倒频谱法，短时傅里叶分析法，并行时域检测法数据压缩法，线性预测反滤波法，等等。这里只论述线性预测反滤波法。

发音系统的分析模型：

$$E(z) = (1 - z^{-1})A(z)S(z)$$

其中 $A(z)$ 是声道逆滤器的转移函数，按线性预测原理，只要 $A(z)$ 的阶次足够高，而且，语音是浊音，那么，分析模型的输出将接近于周期性的冲击序列，冲击序列的频率就是瞬时语音的基频。于是用 LPC 方法提取语音基频的方案如图 11.24 所示。

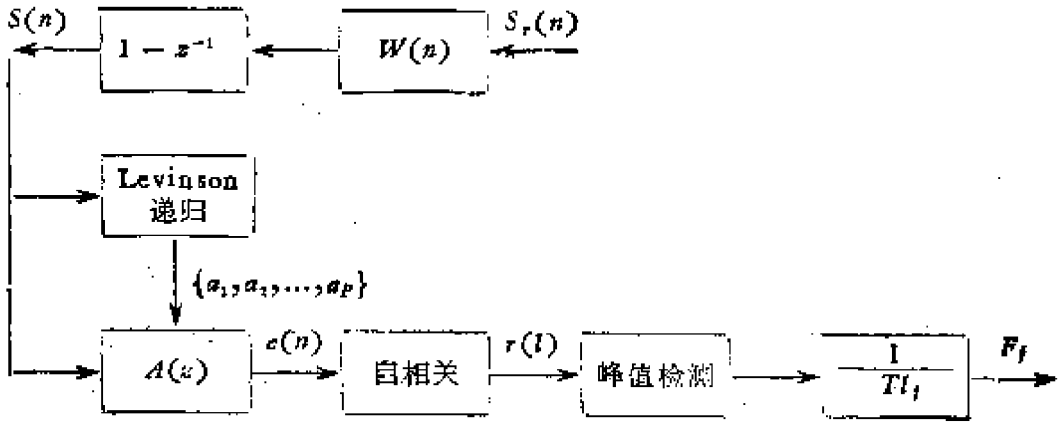


图 11.24 LPC 方法提取语音基频原理

其中求自相关的公式为

$$r(l)=\sum_{n=0}^{N+P-l-1}e(n)e(n+l) \qquad (11\text{-}83)$$

峰值检测器输出自相关函数的第一个极大值点 l_1. 若 T 是采样周期，$1/Tl_1$ 即是语音基频。图 11.25 是对一男性发音人所发的韵母 ā 提取的 $e(n)$ 和 $r(l)$ 曲线。

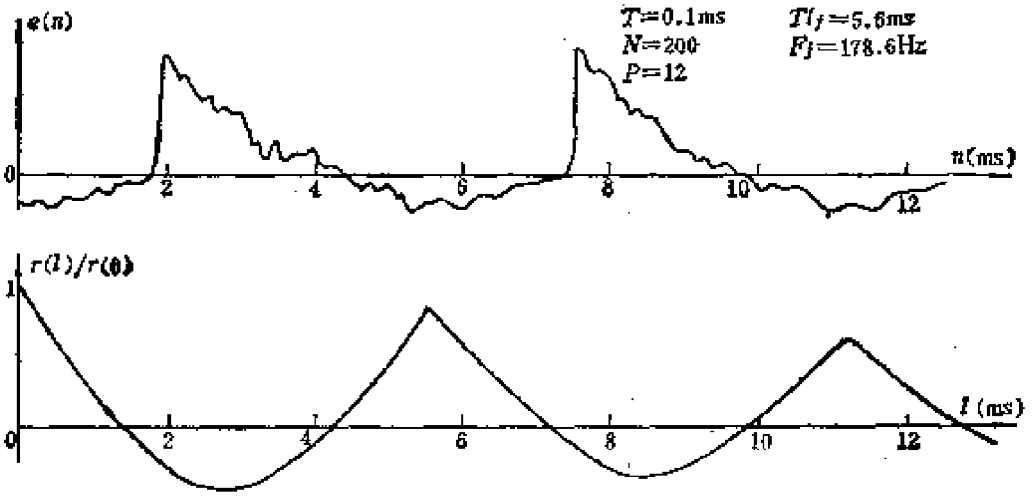


图 11.25 语音 ā 的 $e(n)$ 和 $r(l)$ 曲线

§ 11.5 瞬时语音信号的频域特征

一、通道式瞬时谱分析码

用 $S(t)$ 和 $W(t)$ 分别表示语音信号和时窗，瞬时语音的傅里叶变换为

$$\begin{aligned}S(j\omega,t)&=\int_{-\infty}^{\infty}S(\tau)W(t-\tau)e^{-j\omega\tau}d\tau\\&=e^{-j\omega t}\int_{-\infty}^{\infty}S(t-\tau)W(\tau)e^{j\omega\tau}d\tau\end{aligned} \qquad (11\text{-}84)$$

$S(j\omega,t)$ 是一维语音信号 $S(t)$ 的二维表示，它的含义是：在时

刻 t 附近的一个短时间区间里语音信号的傅里叶变换。语音性特性的改变是由于发音系统的肌肉运动的结果，因此，瞬时谱 $S(j\omega, t)$ 在时间上是慢变化的信号。设 ω_k 是某一确定的角频率，由(11-84)式得

$$\begin{cases} S(j\omega_k, t) = e^{-j\omega_k t}\int_{-\infty}^{\infty} S(t-\tau)W(\tau)e^{j\omega_k \tau}d\tau \\ \qquad\qquad = e^{-j\omega_k t}V_k(t) \qquad\qquad (a) \\ V_k(t) = \int_{-\infty}^{\infty} S(t-\tau)W(\tau)e^{j\omega_k t}d\tau \qquad\qquad (11\text{-}85) \\ \qquad\qquad = S(t) * [W(t)e^{j\omega_k t}] \\ \qquad\qquad = S(t) * h_k(t) \qquad\qquad (b) \\ h_k(t) = W(t)e^{j\omega_k t} \qquad\qquad (c) \end{cases}$$

(11-85b) 和 (11-85c) 表示，$V_k(t)$ 是以语音信号为输入，以 $W(t)e^{j\omega_k t}$ 为冲击响应的环节的输出。以时窗 $W(t)$ 为冲击响应的环节是一个低通滤波器，设 $W(t)$ 的傅里叶变换是 $W(j\omega)$，那么，冲击响应 $h_k(t)$ 的傅里叶变换为

$$H_k(j\omega) = W(j(\omega - \omega_k)) \qquad (11\text{-}86)$$

这表明，冲击响应 $h_k(t)$ 代表一个中心频率为 ω_k 的带通滤波器。于是，语音的瞬时傅里叶变换 $S(j\omega, t)$ 在频率 ω_k 的采样 $S(j\omega_k, t)$ 可以表示为图 11.26 所示的通道的输出。由图可见，$S(j\omega_k, t)$ 是 $V_k(t)$ 的检波输出。已知 $V_k(t)$，意味着 $S(j\omega_k, t)$ 已知。

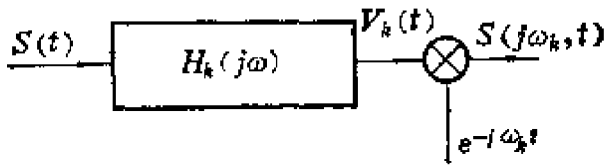


图 11.26 瞬时谱的频率采样

设时窗 $W(t)$ 是如图 11.27 所示的理想低通滤波器，而且取

$$\omega_k = 2k\omega_k \quad k = 0, 1, 2, \cdots, N-1 \qquad (11\text{-}87)$$

组成图 11.28(a) 所示的低通滤波器组。滤波器组的组合频率响应

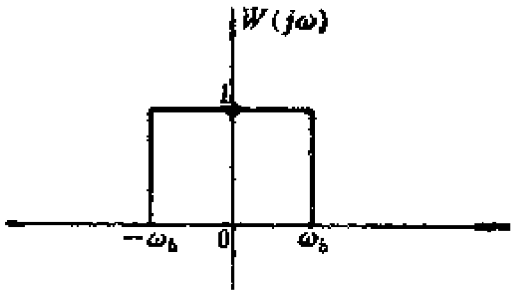


图 11.27　理想低通滤波器

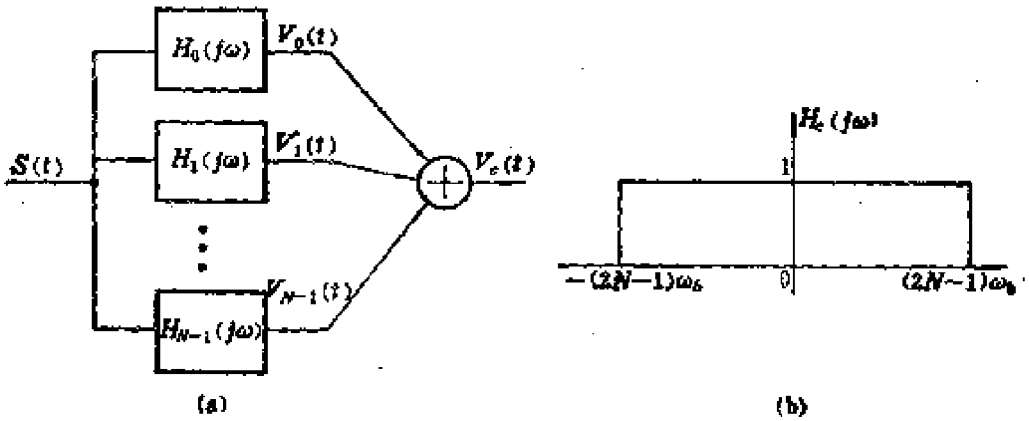


图 11.28　带通滤波器组和它的组合频率响应

为

$$H_e(j\omega) = \sum_{k=0}^{N-1} V_k(j\omega)/S(j\omega)$$

$$= \begin{cases} 0 & |\omega| > (2N-1)\omega_b \\ 1 & |\omega| \leqslant (2N-1)\omega_b \end{cases} \tag{11-88}$$

因此，只要语音信号的上限频率 ω_m 满足

$$\omega_m \leqslant (2N-1)\omega_b$$

那么带通滤波器组的组合输出和语音相等，即

$$V_e(t) = \sum_{k=0}^{N-1} V_k(t) = S(t)$$

$$\omega_m < (2N-1)\omega_b \tag{11-89}$$

这表明，由 $S(j\omega, t)$ 的采样信号 $S(j\omega_k, t)$ 完全可以恢复出 $S(t)$

(至少在理论上是这样)。

注意到幅度 $|S(j\omega_k, t)|$ 就是时间信号 $V_k(t)$ 的包络。求 $V_k(t)$ 的包络在工程上可以对 $V_k(t)$ 整流再经过低通滤波近似实现。图 11.29 是实现通道式谱分析的框图。向量

$$C(t) = [C_0(t), C_1(t), \cdots, C_{N-1}(t)]^T \qquad (11\text{-}90)$$

代表瞬时语音的特征，由于 $C(t)$ 是低通滤波器的输出，因此，可以用很低的频率对 $C(t)$ 采样，用这样的方法可将比特率压缩到 500bit/s 左右(不包括激励信号)。$C(t)$ 的采样输出就是通道式瞬时语音谱分析码。

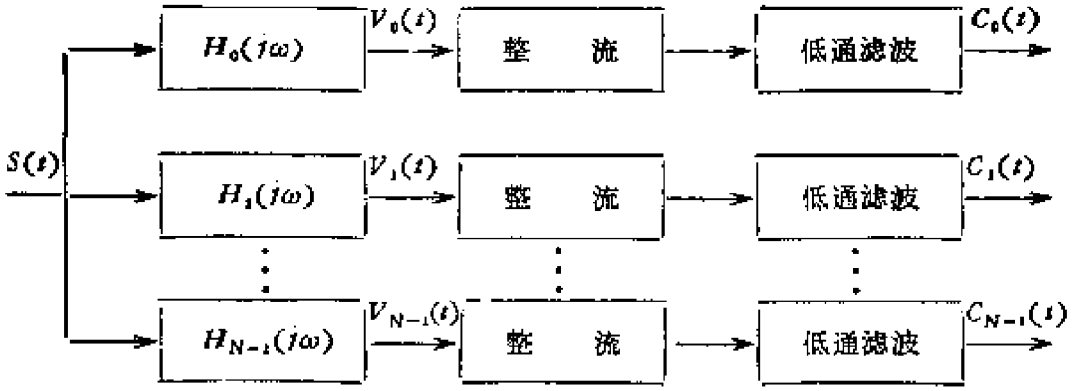


图 11.29 通道式瞬时谱分析

以上抽取瞬时语音特征的方法是从古老的通道式声码器沿袭下来的，由于在抽取的过程中舍弃了语音瞬时傅里叶变换的相位，而在求包络时采用的是简单的求近似包络的方法，因此 $C(t)$ 是不完全的精确度不高的语音特征，而且，输出码随音量而变。但是，抽取通道式瞬时谱分析码不像抽取 LPC 系数那样，计算量很小，抽取过程很容易作到实时，这是它的十分显著的优越性。因此，这种瞬时语音特征适合于对实时性要求高而又希望造价低的简单的语音识别系统。

在实际实现时，通道的数目 N 决定于所考虑的语音的带宽，N 的大小一般是十几；带通滤波器的带宽也并不要求相等，一般随 k

的加大，带宽也随之增加。

二、共振峰

发音系统的转移函数可以表示为共轭极点的形式：

$$V(z)=\frac{1}{\prod_{i=1}^{K}(1-2e^{-C_iT}z^{-1}\cos b_iT+e^{-C_iT}z^{-2})} \tag{11-91}$$

每一对共轭极点表示系统的一个共振峰，共振峰的频率为

$$f_i=b_i/2\pi \quad (\mathrm{Hz}) \tag{11-92}$$

共振峰的带宽近似为

$$B_i=C_i/\pi \quad (\mathrm{Hz}) \tag{11-93}$$

若用 LPC 方法，则可对 $A(z)$ 进行因式分解，并在得到的共轭极点对中排除那些带宽比共振峰带宽宽得多的共轭极点对，可以准确得到共振峰频率和带宽。

显然可以用共振峰频率、带宽和激励作为瞬时语音的特征，但是，这种语音特征很少用于语音识别；它主要用于语音分解，实现语音的低比特率传输和便于存储，以及合成自然度良好的语音。

§ 11.6 语音识别系统举例

一、声控自动查报电话号码系统

这个系统(清华大学计算机科学与技术系方棣棠、吴文虎、房贺祥研制)用语音输入，自动查部 8000 部电话的号码。工作过程是：打电话的人用语言告诉总机室的话务员所要通话的单位，然后话务员用语音对系统重述单位的标准名称，系统立即显示出该单位的电话号码(或实现自动转接)。

这个系统用 BP-7 动态频谱分析仪抽取语音的通道式瞬时谱分析码作为语音的特征，用 Apple-II 微型计算机作识别的处理。

系统把输入的单位名称作为一个完整的（不加划分）语音进行识别，因此，是一个孤立词语音识别系统。

1. 特征抽取

为了作到实时，系统用 BP-7 动态频谱分析仪抽取通道式瞬时谱分析码作为瞬时语音的特征。BP-7 频谱分析仪的工作原理在上节已经进行了论述，唯一的一点区别是对图 11.29 每个通道的低通滤波器输出取对数后再作为该通道的输出。

语音特征的抽取过程如图 11.30 所示。

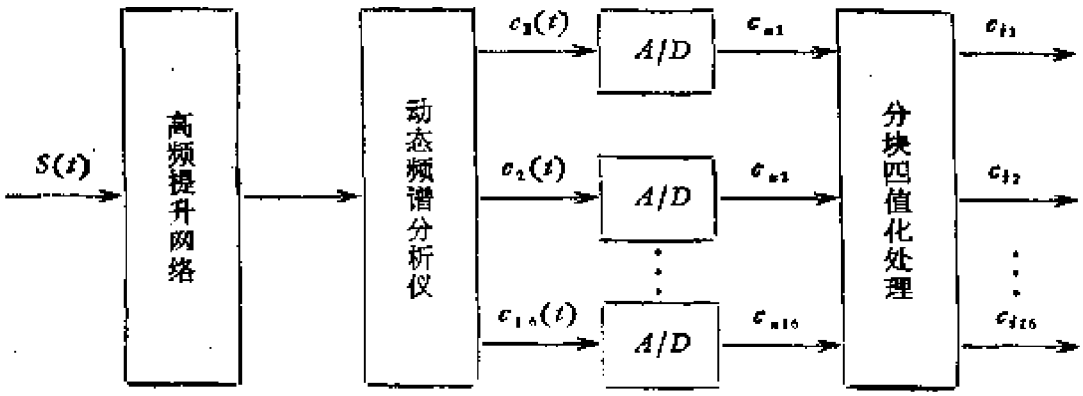


图 11.30　语音特征的抽取过程

其中 $S(t)$ 是从话筒得到的语音信号。由于清音的音量小频率高，BP-7 的输出随音量而变，为了使音量小的清音能得到同等的对待，因此在 BP-7 动态频谱分析仪前设置了一个高频提升网络，它的频率响应为

$$W(j\omega) = \frac{1.77 \times 10^{-4} j\omega + 1}{7.95 \times 10^{-6} j\omega + 1}$$

通道的数目 $N = 16$。由于基频随音量变化很大，为了减少基频对识别的影响，舍弃 BP-7 的 315Hz 以下各通道的输出，于是十六个通道的中心频率为：315，400，500，630，800，1000，1250，1600，2000，2500，3150，4000，5000，6300，8000，10000Hz。A/D 转换的变样率为 100Hz，因此，每 1/100 s 可以获得一帧通道式瞬

时谱分析码。

$$C_n = [C_{n1}, C_{n2}, \cdots, C_{n16}] \tag{11-94}$$

对一个完整的语音(单位名称)，各帧通道式码组成一矩阵 C

$$C = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_m \end{bmatrix} = \begin{bmatrix} C_{1,1}, & C_{1,2}, \cdots, & C_{1,16} \\ C_{2,1}, & C_{2,2}, \cdots, & C_{2,16} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots \\ C_{m,1}, & C_{m,2}, \cdots, & C_{m,16} \end{bmatrix} \tag{11-95}$$

C 矩阵的行表示时间。

2. 非线性分块时间归正

定义列向量

$$\boldsymbol{D}^* = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} \tag{11-96}$$

$$d_i = \sum_{j=1}^{16} |c_{ij} - c_{i-1,j}|, \quad c_{0j} = 0, \quad j = 1, 2, \cdots, 16$$

其中 c_{ij} 是矩阵 C 的元素，d_i 是 C 矩阵的第 i 行(帧)和第 $i-1$ 行对应列(通道)的差的绝对值和。定义

$$D = \sum_{i=1}^{m} d_i \tag{11-97}$$

它表示在语音持续的期间里语谱总的变化量。按顺序确定整数 Q_1、Q_2、Q_3、Q_4、Q_5，使得

$$\sum_{i=Q_{k-1}+1}^{Q_k} d_i \doteq \frac{1}{6} D \tag{11-98}$$

按所求得的 Q_k 值，将矩阵 C 分成六块，如下式所示。

$$\left.\begin{bmatrix} c_{11}, \cdots, & c_{1,16} \\ \vdots & \vdots \\ c_{Q_1,1}, \cdots, & c_{Q_1,16} \\ \vdots & \vdots \end{bmatrix}\right\}\text{第一块}$$

$$C=\begin{bmatrix} c_{Q_{k-1}+1,1} & \cdots & c_{Q_{k-1}+1,16} \\ \vdots & & \vdots \\ c_{Q_k,1} & \cdots & c_{Q_k,16} \\ \vdots & & \vdots \\ c_{Q_i+1,1} & \cdots & c_{Q_i+1,16} \\ \vdots & & \vdots \\ c_{m,1} & \cdots & c_{m,16} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{第 } k \text{ 块} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{第六块} \end{matrix} \quad (11\text{-}99)$$

构造 B 矩阵

$$B=\begin{bmatrix} b_{1,1} & \cdots & b_{1,16} \\ b_{2,1} & \cdots & b_{2,16} \\ \vdots & & \vdots \\ b_{6,1} & \cdots & b_{6,16} \end{bmatrix} \quad (11\text{-}100)$$

$$b_{ij}=\frac{1}{Q_i-Q_{i-1}}\sum_{k=Q_{i-1}+1}^{Q_i} c_{kj};\quad Q_0=0$$

再对 B 中的元素逐行进行处理，同行元素按相对大小分别取 0，1，2，3 四值之一，这就是所谓四值化。对 B 矩阵经四值化处理后得矩阵 A. 这样，每一个单词的语音和一个矩阵 A 对应。

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,16} \\ & \cdots\cdots & & \\ a_{6,1} & a_{6,2} & \cdots & a_{6,16} \end{bmatrix} \quad (11\text{-}101)$$

它所占的比特数为

$$6\times 16\times 2=192$$

数据量得到很大的压缩，因此才有可能用较小的内存空间存储大量的语音样板。

3. 识别和实验结果

在进行识别前，先由发音人自己对系统进行训练，把要进行识别的词按规定的顺序念一遍，系统按 1 和 2 论述的过程建立所有单词的参考样板 $\{A_k\}$, $k=1,2,\cdots,K_d$.

在进行识别时系统根据发音人(对系统进行训练的人)发出的语音 $s_x(t)$，自动抽取出矩阵 A_x，然后用最近邻识别方法进行识

别。所谓最近邻识别方法，就是计算出 A_x 和每一个语音样板间的距离 $D(A_x, A_k)$，$k=1, 2, \cdots, K$，A_x 和哪个语音样板间的距离最小，就把 A_x 判定为是和这个样板同样的语音。声控自动查报电话号码系统使用 Chebyshev 距离，于是 A_x 和 A_k 间的距离为

$$D(A_x, A_k) = \sum_{i=1}^{6} \sum_{j=1}^{6} |a_{ij}^x - a_{ij}^k| \qquad (11\text{-}102)$$

其中 a_{ij}^x 和 a_{ij}^k 分别是 A_x 和 A_k 的元素。识别规则为

若 $D(A_x, A_l) = \min_k D(A_x, A_k) \quad \forall k$

$\rightarrow A_x$ 是第 l 个语音。

这样，系统转入执行由 l 定义的单元，自动查出电话号码（或实现自动转接）。

用以上系统对总数为 8000 部电话进行自动查报电话号码的实验，识别率为 99.4%。

4. 评论

优点：

（1）非线性分块时间归正是一种很好的归正方法，它着重语音拼读的变化部分，很大程度上抑制了由于韵母音持续的时间较长(采集的帧数较多)声母音持续的时间短(帧数较少)，因此，在求语音间的距离时韵母淹没声母的现象。而且计算次数很少，有利于作到实时。

（2）识别过程的计算次数很少，因此，对硬件的要求低。

（3）在一机单人(一台机器只识别一个人的语音)，而且单词的音节较多的情况下识别率高。

不足：

（1）通道式瞬时谱的幅度和音量的大小有关。由于清音的音量比浊音的音量小得多，因此清音的瞬时谱的幅度远比浊音的瞬时谱幅度小。在求 Chebyshev 距离时，在(在块归正时)用每块的

平均谱代替该块时都强调了强音的作用。因此对清音和浊音的"待遇"不同;但这缺点容易克服，只要先对各帧谱进行归一化处理即可。

(2) 通道式瞬时谱分析是一种近似方法，四值化处理显得粗了些。

(3) 只能一机单人。要求识别时的语音音量和训练时的语音音量一致。

二、大字表孤立词语音识别系统

这个系统(F. Itakura 研制)用于识别电话输出的语音，能识别的单词或字的数目(字表的大小)为 200。同样，它要求语音词与词之间有明显的时间间隔，因此仍属于孤立词语音识别系统。

1. 特征抽取和距离度量

系统用六阶自相关 LPC 系数作为瞬时语音的特征。语音信号先经过带宽为 3kHz 的低通滤波器。然后以 6.66Hz 的采样率进行采样，采样信号经过宽度为 200 点的海明窗(每帧 30ms)，最后用 Levinson 递归求出瞬时语音的六个 LPC 系数。

设 A_k 和 A_x

$$A_k = [1, a_{k1}, a_{k2}, \cdots, a_{k6}]^T \qquad (11\text{-}103)$$

$$A_x = [1, a_{x1}, a_{x2}, \cdots, a_{x6}]^T \qquad (11\text{-}104)$$

分别是两个语音段(瞬时语音)的自相关 LPC 系数，板仓从模式识别常用的对数似然函数比的角度，导出了 A_k 和 A_x 间的距离。其实从线性预测很容易得到板仓的结果。设 A_x 是由 $S_x(n)$ 求得的线性预测系数，由分析模型得预测误差的平方和为

$$\alpha_x = \sum_{n=0}^{N+P-1} e_x^2(n) = \sum_{n=0}^{N+P-1} \left(\sum_{i=0}^{P} a_{xi} S_x(n-i) \right)^2$$

$$= \sum_{i=0}^{P} \sum_{j=0}^{P} a_{xi} \left(\sum_{n=0}^{N+P-1} S_x(n-i) S_x(n-j) \right) a_{xj}$$

$$= A_x R_x A_x \qquad (11\text{-}105)$$

令

$$R_x = [r(|i-j|)]$$

$$r(|i-j|) = \sum_{n=0}^{N-|i-j|-1} S_x(n) S_x(n+|i-j|)$$

若用另外一组线性预测系数 A_k 对 $S_x(n)$ 进行预测，同理可以得预测误差的平方和为

$$a_k = A_k^T R_x A_k \geqslant a_x \qquad (11\text{-}106)$$

因此，定义 A_x 和 A_k 间的距离为

$$d(A_x, A_k) = \log \frac{a_k}{a_x} = \log \frac{A_k^T R_x A_k}{A_x^T R_x A_k} \qquad (11\text{-}107)$$

这就是板仓导出的 LPC 系数间的距离度量。显然，当 $A_k = A_x$ 时，$d(A_x, A_k) = 0$，而且，$d(A_x, A_k)$ 和音量大小无关。但是，$d(A_x, A_k) \neq d(A_k, A_x)$。

为了减少求 $d(A_x, A_k)$ 的计算次数，利用 A_x 满足如下 LPC 方程组的性质

$$R_x A_x = [r(0) r^T A_x, 0, \cdots, 0]^T \qquad (11\text{-}108)$$

其中

$$r = [1, r(1)/r(0), \cdots, r(P)/r(0)]^T$$

可把求距离的公式简化为

$$d(A_x, A_k) = c + \log(b^T r / A_x^T r) \qquad (11\text{-}109)$$

其中

$$c = \log(A_k^T A_k)$$

$$b_i = 2 \sum_{j=0}^{P-i} a_{k,j} a_{k,j+i} / A_k^T A_k$$

板仓把 c，b 称为改进的 LPC 系数。

2. 用动态规则的时间归正方法

一个词的语音由若干帧组成。把每个词的参考样板表示成如

下矩阵的形式：

$$R(k) = [c(m, k), b^T(m, k)]$$
$$m = 1, 2, \cdots, M(k);\ k = 1, \cdots, K \qquad (11\text{-}110)$$

其中 $c(m, k)$ 和 $b(m, k)$ 是第 k 个参考样板 $R(k)$ 的第 m 帧的改进 LPC 系数；K 是字表的大小。参考样板可以在训练系统时得到。

在进行识别时，系统根据输入的语音 $S_x(t)$，计算出如式(11-108)中所示的这语音各帧的自相关函数，以及各帧的 LPC 系数，分别为

$$r_x(n),\ A_x(n),\quad n = 1, 2, \cdots, N$$

N 是待识别的语音的总的帧数。待识别的语音的第 n 帧和第 k 个参考样板的第 m 帧间的距离为

$$d(n, m, k) = c(m, k) + \log[b^T(m, k) r_x(n) / r_x^T(n) A_x(n)] \qquad (11\text{-}111)$$

设时间归正函数为

$$m = W(n)$$

它实现从待识别的语音的帧数到参考样板的帧数的映射，并按下式计算输入语音和参考样板之间的距离：

$$D_x(k) = \min_{(W(n))} \sum_{n=1}^{N} d_x(n, W(n), k) \qquad (11\text{-}112)$$

式中 $d_x(n, W(n), k)$ 是待识别的语音的第 n 帧和第 k 个参考样板的第 $W(n)$ 帧间的距离。N 是待识别的语音的帧数。$W(n)$ 是时间归正函数，它应满足：

(1) 边界条件：$W(1) = 1$；$W(N) = M(k)$

$M(k)$ 是第 k 个参考样板的帧数；

(2) 连续性条件：系统规定如下连续性条件：

$$W(n+1) - W(n) = \begin{cases} 0, 1, 2 & W(n) \neq W(n-1) \\ 1, 2 & W(n) = W(n-1) \end{cases}$$

由以上的边界条件和连续性条件可知，$W(n)$ 的取值范围如图 11.31 所示。

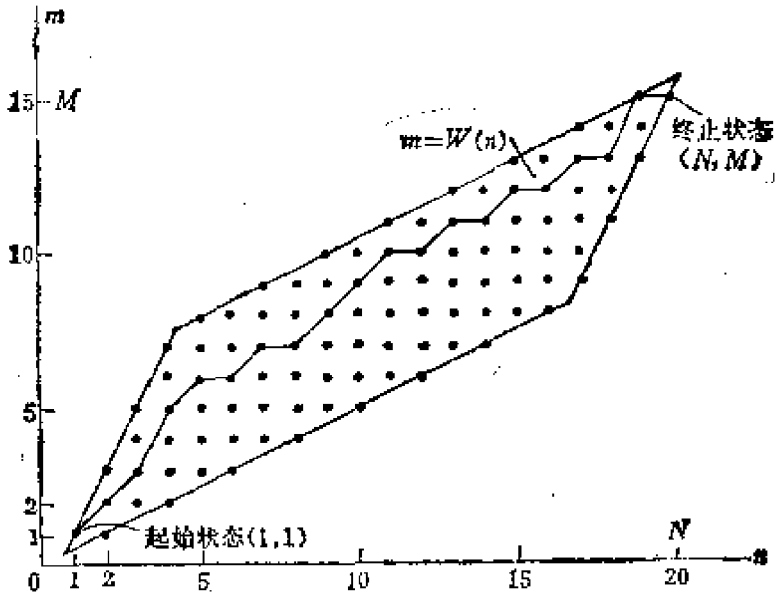


图 11.31　时间归正函数 $W(n)$ 和其取值范围

到此可以清楚地看到，求 $D_x(k)$ 实值是在 n，m 组成的网格结构中从 (1, 1) 点出发到 $(N, M(k))$ 点止的分段决策问题。按动态规划，得计算 $D_x(k)$ 的递归算法如下

$$D_x(n+1, m, k) = d(n+1, m, k) + \min\{D_x(n, m, k)g(n, m), D_x(n, m-1, k), D_x(n, m-2, k)\} \quad (11\text{-}113)$$

$$g(n, m) = \begin{cases} 1 & W(n) \neq W(n-1) \\ \infty & W(n) = W(n-1) \end{cases}$$

按以上动态规划方法，为得到 $D(k)$ 大约需进行

$$L \doteq (2N - M(k) + 1)(2M(k) - N + 1)/3 \quad (11\text{-}114)$$

次求 $d(n, m, k)$ 的运算。例如 $N = 40$，$M(k) = 40$，$K = 200$（字表的大小），为用最邻近识别法识别出一个语音大约要进行 112000 次求 d 的计算。

为了缩短计算 $D_x(k)$ 的时间，提出了如下序贯判别方法．对每一级(每一 n) 除计算出 $D_x(n, m, k)$ 外，还计算出

$$D_x(n, k) = \min_m D_x(n, m, k) \qquad (11\text{-}115)$$

可以想像，若样板 $R(k)$ 与输入语音匹配（距离最小），那么 $D_x(n, k)$ (其中 $n=1, 2, \cdots, N$) 始终是一较小的值．设 $T(n)$ 是阈值函数，在动态规划的递归运算过程中，若 $D(n, k) \geqslant T(n)$，则在第 n 级终止递归，排除 $R(k)$ 和输入语音匹配的可能．板仓取

$$T(n) = 4 + F(n) \qquad (11\text{-}116)$$

在进行识别时，一开始 $F(n)$ 选得足够大，使得对所有可能的匹配情况 $D_k(n, k)$ 都不会超过 $T(n)$．若在递归的过程中 $D(n, k) < T(n)$，则对 $F(n+1)$ 进行修改（例如 $F(n+1) = D(n, k)$）．这样一来，经过对几个参考样板的递归运算后使得 $T(n)$ 足够地低，以致只有少数参考样板能完成全部递归．最后在这少数的参考样板中用最邻近识别方法对输入的语音进行识别．如果对所有的参考样板都不能完成全部递归，表示系统拒识输入语音．实验表明，使用序贯判别之后，使得求 $D_x(k)$ 的计算减少到原来的 12%．

3. 识别流程图和实验结果

板仓使用语音的瞬时能量超过噪声水平判断语音的开始和结束．为了补偿由于传感器、传输线的响应和其它物理因素的影响，板仓对语音信号进行了所谓的长时间谱的归一化处理．方法是求出各帧语音的前两个自相关函数的平均值，由这平均值计算出二阶 LPC 逆滤波器，然后让逆滤波器冲击响应的自相关函数和各帧语音的自相关函数求卷积，将所得的结果用作求 LPC 系数．于是语音识别系统的信号流程图如图 11.32 所示．

系统使用 DDP-516 计算机，若参考样板为 200 个日本地名，平均音节数为 3.5，发音人指定，在两个星期里进行的 2000 次识

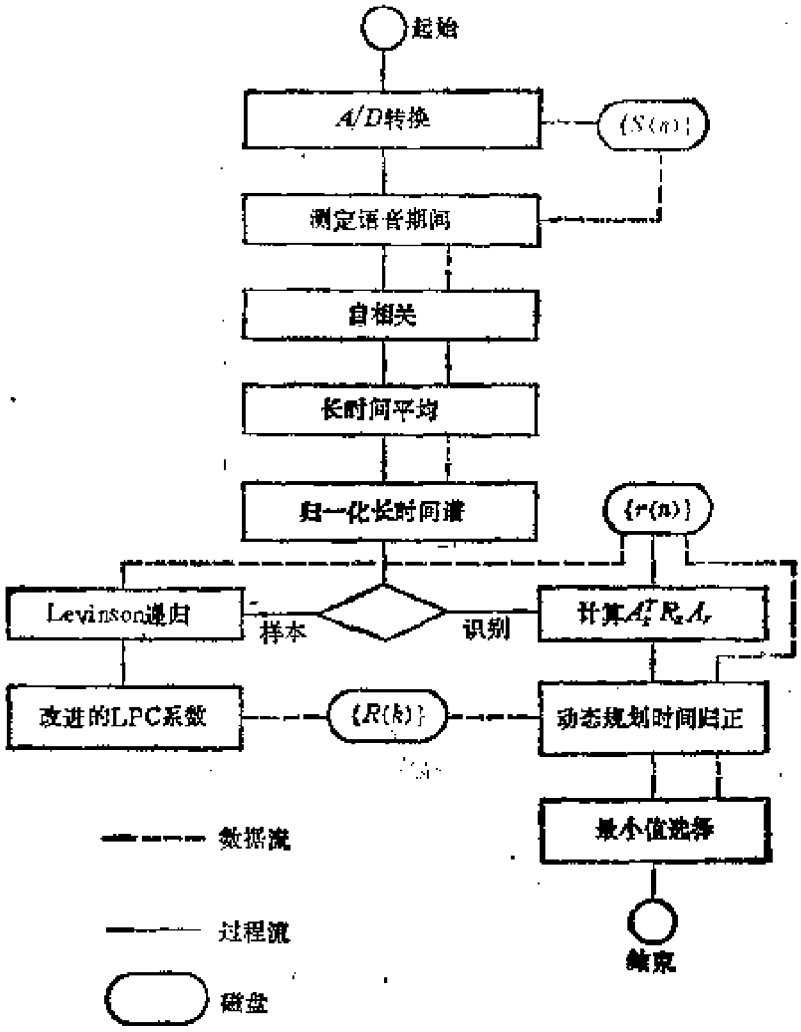


图 11.32　语音识别系统信号流程图

别试验表明：

(1) 识别率 97.5%，拒视率 1.65%；

(2) 响应时间 12s。

4. 评论

优点：

(1) LPC 分析方法具有精度高，能把声道模型和激励分开进行处理的优越性，因此，对建立高质量的语音识别系统有着美好的前景。

(2) 导出的瞬时语音的距离度量函数合理。距离度量不受语音音量大小的影响。

不足:

(1) LPC 系数只是描述声道的参数，仅用 LPC 系数进行识别而不考虑激励是不合理的。如果在识别时考虑激励的性质和参数，系统的性能指标将会进一步提高。作者在其发表的文章中申明，他建立这个语音识别系统的目的，是为了检验他导出的瞬时语音的距离度量函数。

(2) 动态规划时间归正方法常常被认为是最好的归正方法。其实不一定。姑且不论计算量的大小，动态规划法还存在另一种不足。例如语音"妈"，音母 a 的音拖得再长也不会改变语音的性质，语音的性质决定于拼读时变化的部分，动态规划归正法没有考虑语音的这一特点，而是不加区分地帧帧测量距离 d，势必造成由于元音的帧数多，辅音的帧数少，因而在求语音的距离 D 时有元音淹没辅音的现象，例如，系统常常把"H"误识别为"EIGHT"。

(3) 提取 LPC 系数的计算工作量很大，要作到实时需强大的硬件支持。

第十二章　机器人的触觉

§ 12.1　概述

对环境、对象状况的感知，是机器人智能化的最重要的特征．在机器人的各种感知功能中，触觉与视觉一样，是必不可少的基本方式．

据研究，人类的大脑功能有三分之二用于管理触觉信息．人体的触觉信息直接来源于密布在整个皮肤组织中的触觉细胞、神经末梢、触角小体等基本感受元素，它们在皮肤的所有部位都能感受到触、压、痛等机械刺激，以及冷、热等温度刺激．与视觉、听觉不同，人体的触觉不存在于身体的确定部位，也没有集中的构造，而是与肌肉、腱、关节这些更深部位的感受器一起，形成复杂的综合信息．例如，压力、硬度、形状、大小、滑动、节律、质地等，传向神经系统进行处理，最后，或是得知对象物的几何模式及其物理性质，或是产生统一的控制命令，指挥身体的运动组织作出合适的反应．

机器人的触觉是对人体触觉功能的一定程度的模仿．当然，机器人并不需要也不可能在其整个表面装设触觉感受器，而一般都设置在手爪、足、关节等主要操作部位．机器人的触觉功能也主要在于检测对象物体与这些操作部位的接触状态以及相互之间的作用力，并将获得的信息经过处理传送到控制系统，产生能够适应对象状态变化的操作．

一般认为，机器人的触觉应具有四种基本功能：接触觉、压觉、滑觉和力觉.

接触觉检测手爪与对象物之间有无接触. 根据手爪触感点的输出，机器人可以感受、搜索对象物，感受手爪与对象物之间的相对位置或姿态，并修正手爪的操作状态(见图 12.1(a)、(b)). 在触感点足够密的情况下，机器人还可通过接触觉判断对象物的形状、大小.

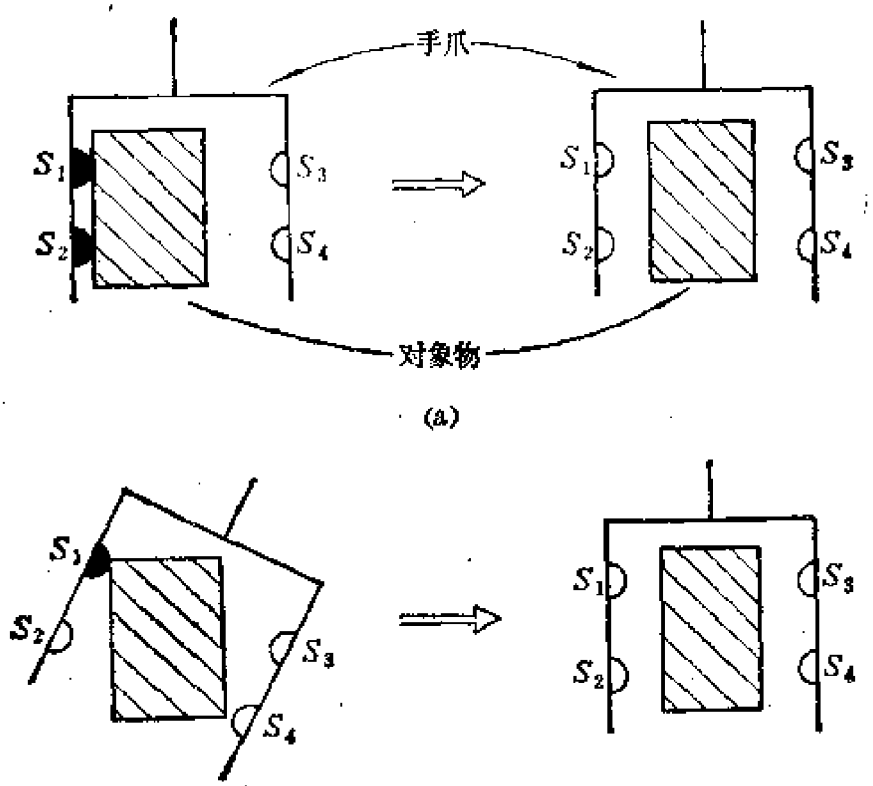


图 12.1 (a) 通过接触觉修正手爪位置；
(b) 通过接触觉修正手爪姿态.

压觉检测手爪与对象物之间接触面上的法向力. 根据手爪上压感点阵的输出，机器人可以对对象物采取不同的握取方式：硬握取——用最大的力夹紧对象物，以确保握牢；软握取——用适当

的力夹住对象物，以免对象损坏；零握取——一种临界握取方式，不用力夹对象物，目的只是确定对象物的存在或形状和大小。此外，压感点阵的输出还可同时给出对象物软硬质地的信息。

滑觉检测手爪与对象物之间接触面上的切向力，以感受对象物在手爪中的滑移。滑移检测一般是间接进行的，或者检测手爪与对象物之间由于滑移发生的振动（这与对象物的表面性质有关），或者检测手爪与对象物之间相对滑移的位置变化。机器人的滑觉和压觉常常配合工作(图 12.2)，例如在抓握对象物、进行提取或移动时，滑觉信息可用来产生适合对象物重量及表面摩擦系数的握紧力，实现不滑落的最小握紧力控制。

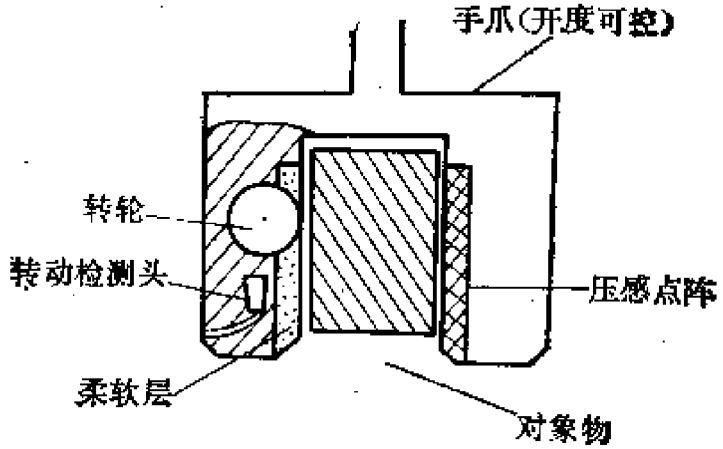
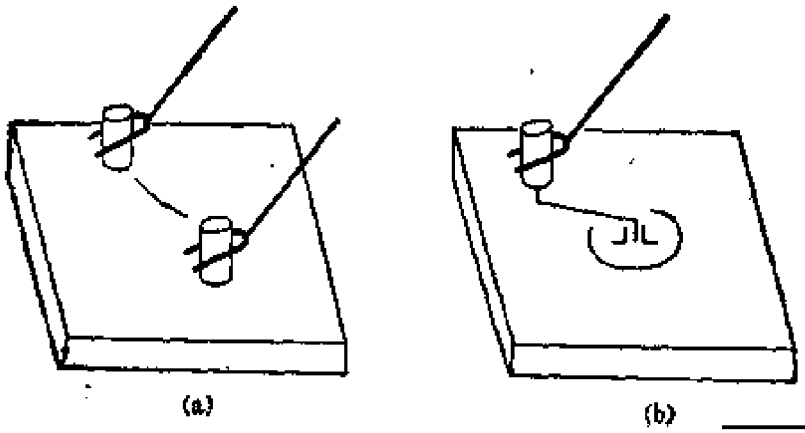


图 12.2　滑觉与压觉配合工作

力觉是指机器人动作时各自由度的力(力矩)感觉。机器人的基本动作是手爪的移动，而在移动过程中，手爪往往受到各种约束。例如手爪抓住对象物在某一面上移动（图 12.3(a)）时要受到这个面的约束，手爪的移动轨迹也可能受到对象物的约束(图 12.3(b))。机器人通过安装在关节上的力觉感受器，就可以检测外界对象物施加于手爪和臂的这类约束力(力矩)，产生满足约束的柔性动作。显然，力觉感受器也可以检测机器人操作时施加于对象物的力(力矩)，因此，力觉感受器具有双向传递力信息的性能。



(a) (b)

简单、明显，因此，传感器的研究成为目前触觉技术中的一个关键问题。

一、触觉传感器的结构原理

触觉信息是通过传感器与目标物体的实际接触得到的，因而，触觉传感器的输出信号基本上是由两者接触而产生的力以及位置偏移的函数。随后，输出信号被解释成接触界面的性质、目标物体的特征以及接触状态，这些信息主要包括：(1) 目标物体的存在与否，(2) 目标物体的形状、位置、姿态，(3) 接触面上的压力、压力的分布，(4) 力的大小、方向和作用点，(5) 力矩的大小、方向和作用面。此外，还可包括静态与动态的信息，目标物体的柔顺性、质地、粘弹性等等组合信息。

简单的触觉传感器一般只传送上述信息中的一种，例如各种限位开关、接触开关，它们是最简单的触觉传感器，只感受目标物体的存在与否。而比较复杂的触觉传感器则包含着许许多多的甚至具有复合传感作用的敏感单元，它们可以在不同的负载状况下提供各种类型的接触信息，而且，传感器还与专用的计算装置或微处理器相连，形成触觉传感系统，完整地实现信号获取及解释处理的功能。

触觉传感器主要有三个部分组成(图 12.4)。

接触界面，由多个敏感单元按一定的方式排列配置而成，与目标物体直接接触。

转换媒介，即为把接触界面传递来的力或位置偏移转换成输出信号的敏感材料或敏感机构。

检测和控制，按预定方式、次序收集触感输出信号，并把它们传送给处理装置进行解释。

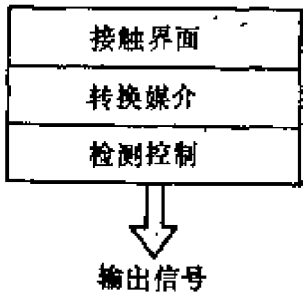


图 12.4 触觉传感器的组成

几乎所有的触觉传感器都输出最便于处理的电信号，因此，它们的转换原理和性能评价都具有非电量-电量的一般传感器的特点。

触觉传感器采用的转换原理如下所述。

光电式：把接触界面的压力变为机械位移，再利用此机械位移改变光源与光敏检测器之间的距离，或遮挡光源形成阴影，从而使检测器的电信号发生变化。

压阻式：利用各种电阻率随所受压力大小而变化的材料，例如硅导橡胶、可导纤维等压敏电阻材料，把接触界面的压力变为电信号。

电阻应变式：与压阻式原理类似，利用金属导体(或半导体材料)变形时电阻值也发生变化的电阻应变效应，常称应变计。

压电式：利用压电陶瓷、压电晶体等材料的压电效应，把接触面上的压力转换成电信号。

磁致弹性式：利用某些磁性材料在外力作用下磁场发生变化的效应，感受接触界面上的压力，磁场的变化而后又通过各种类型的磁路系统转换为电信号。

此外，还有利用力或位移转换为电容量变化的电容式等等。

触觉传感器的性能评价主要为如下几个方面：

(1) 感受范围：传感器检测接触界面上空间位置或力的上、下限值。

(2) 灵敏度：传感器输出量的变化值与相应的被感受量的变化值之比。

(3) 分辨率：传感器可能感受的空间位置或力的最小变化值。

(4) 重复性：同一工作条件(接触状态、环境)下，在整个感知范围内传感器连续多次与目标物体接触，传感器重复输出读值的能力。

(5) 迟滞：被感受量先逐渐增加，后又逐渐减少，在规定的感

变范围内的任一被感受量的最大差值。

（6）响应速度：传感器感受、转换、检测整个接触界面信息的时间。

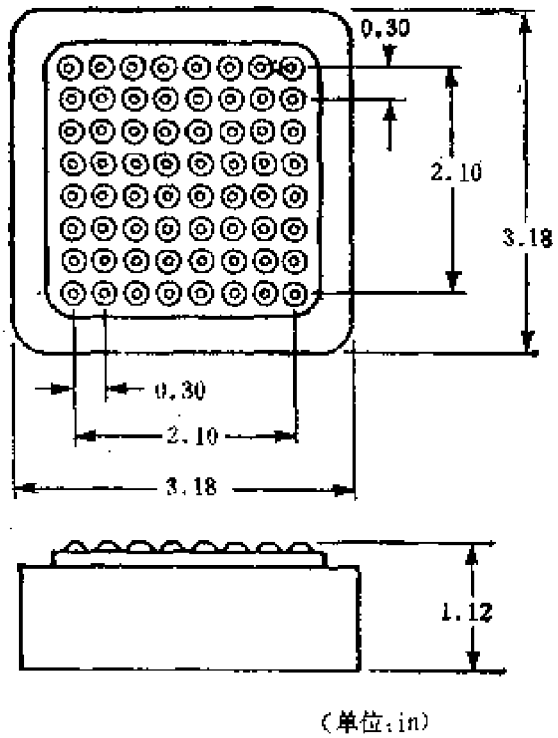
（7）温飘、抗干扰等受环境影响的性能变化情况。

当然，由于为机器人所用，在要求性能的同时，触觉传感器的整体结构一定要尽量轻巧纤薄、皮实可靠。

下面具体介绍两类触觉传感器的一些实验研究成果。

二、成像型触觉传感器

成像型触觉传感器着重于感受接触界面上目标物体的形状，再通过进一步处理输出信号，最终可识别物体，判断它的位置和姿态。传感器一般都采用由若干感受单元组成的阵列型结构。由于形状信息是将目标物体对接触界面的压力转换处理后得到的，所以这种传感器实际上同时体现了“接触觉”和“压觉”这两种功能。



（单位：in）

图 12.5　LTS-100 的外形

例 12.1 光电式成像触觉传感器 LTS-100（美国 LORD 公司研制）.

传感器有 64 个敏感单元，每个单元都有一个突起的触头，它们排成 8 × 8 阵列，形成接触界面，传感器的整个尺寸如图 12.5 所示（包括检测、控制电路），相邻触头间的距离 0.3in 即为它的分辨率.

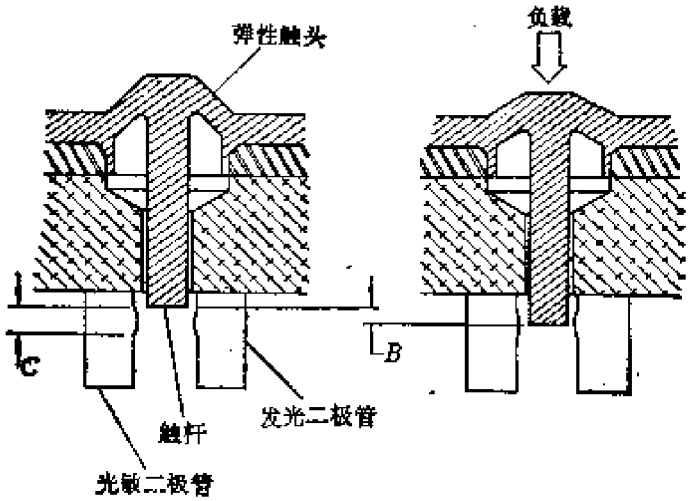


图 12.6 敏感单元的转换原理

图 12.6 说明了传感器一个敏感单元的转换原理. 当由弹性材料制成的触头受到法向压力时，触杆下伸，遮挡了发光二极管射向光敏二极管的一部分光，于是光敏二极管的电信号输出间接地随触头所受压力的大小而连续改变，提供灰度读数. 触杆的下伸拉移范围可达 0.8in，传感器的灵敏度约为 0.03N. 转换的一致性取决于触头、触杆的性能以及二极管对的特性. 一般都经过硬件调整和软件修正以达到规定的指标.

传感器与一台微处理机相连，形成传感系统(图 12.7).

触感阵列的输出电流由多路模拟开关选通检测，放大后经 8 位模/数转换器变为灰度不同的触觉数字信号，送往微处理机解释处理. 传感器的控制电路接收来自微处理机的选通信号并接着提供顺序检测地址及触发脉冲，扫描整个阵列. 选通信号之间有一

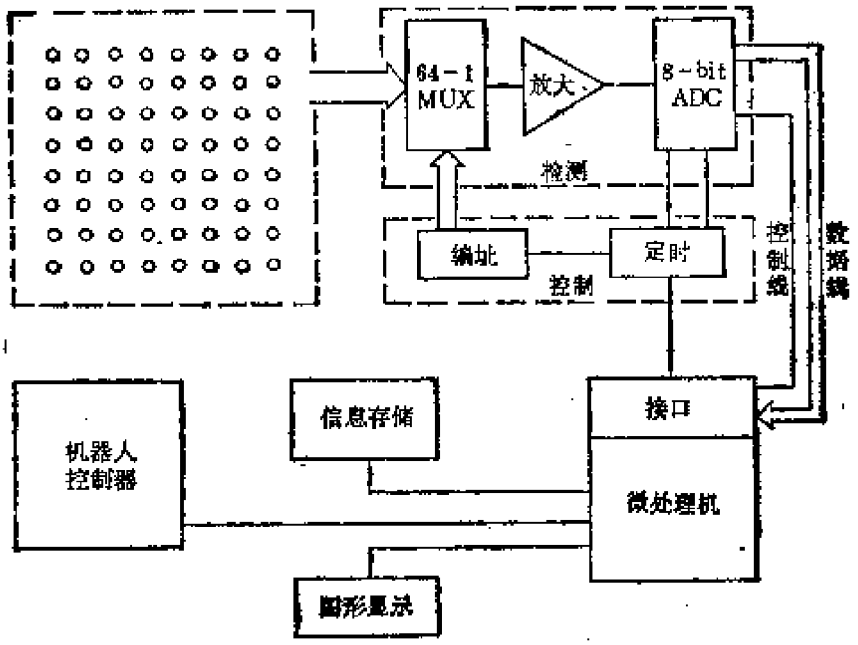


图 12.7　LTS-100 传感系统

定的延迟间隔，以保证放大器和转换器的可靠工作.

LTS-100 可配合机器人工作，手爪把工件放在传感器上，经过微处理机处理、解释的形状、位置信息送往机器人控制器，控制手爪以合适的姿态抓握工件进行装配操作.

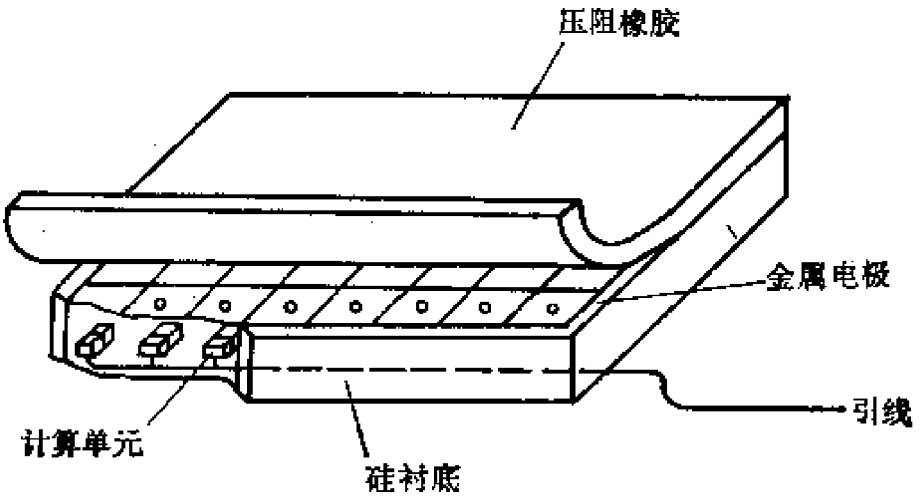


图 12.8　VLSI 触觉传感器结构

例 12.2 VLSI 触觉传感器（美国 Carnegie-Mellon 大学和 California 工学院研制）.

无论采用哪种转换原理，成像触觉传感器都存在如何提高分辨率的问题. 分辨率愈高，每个触感单元的尺寸愈受限制，而且，触感单元的引线也愈多，多路模拟开关电路愈复杂，这就会使传感器的体积、重量和计算、通讯难以符合机器人的实用要求. 而集成电路技术的应用则为解决这些困难提供了可能的途径.

图 12.8 表示了一种 VLSI 触觉传感器的物理结构. 作为接触

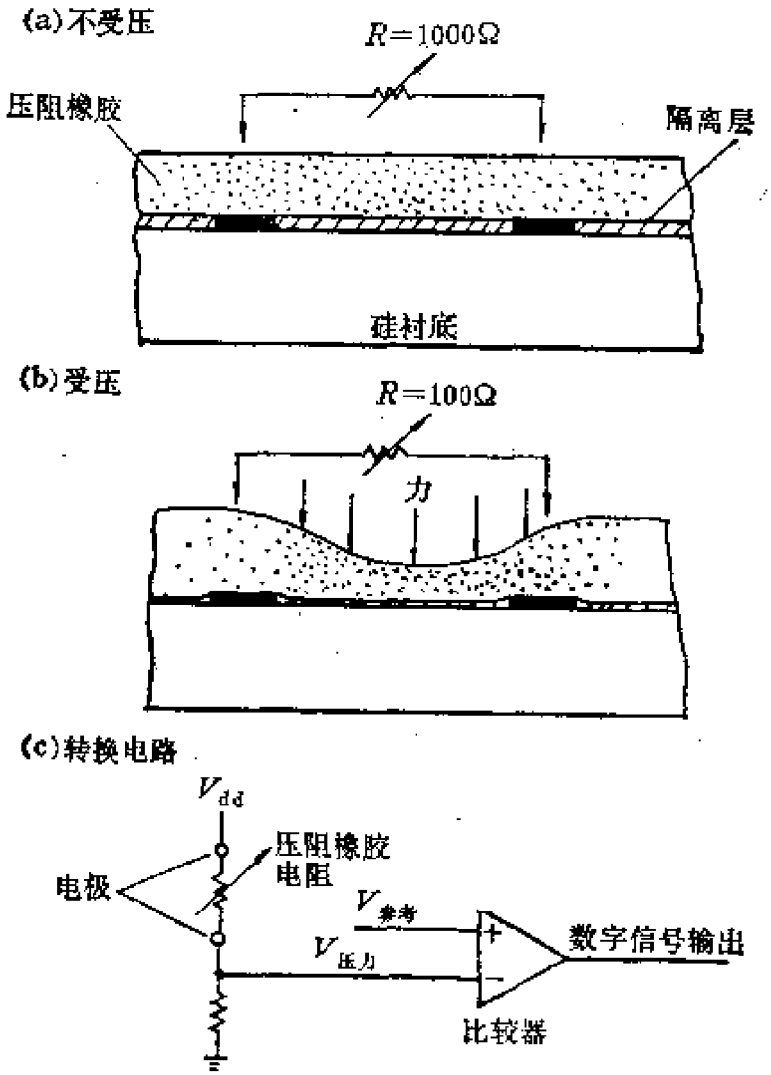


图 12.9 VLSI 触觉传感器的转换原理

界面的一种压阻橡胶放在一块特制的超大规模集成电路芯片上，两者之间夹有一块 SiO_2 隔离层，隔离层上刻有窗口，露出与电路相连的金属电极，可与橡胶直接接触．所有的电极形成一个阵列．而每个电极下面则是一个个的模拟、数字计算单元．

图 12.9 进一步说明了传感器的转换原理．当压阻橡胶在某处受压变形时，其中的导电粒子密度变大，电阻率变小，附近的相邻电极间的电压变小，从而得到与压力大小成比例的触感信号．

传感器的电极面积仅约 $1mm^2$，相应地，每个电极下面的计算单元集成电路面积也不超过 $1mm^2$，其中含有一个阈值可调的模拟比较器，一个锁存器，一个简单的累加器以及一个指令寄存器．所有的计算单元与控制总线相连，接受一台微处理器的控制指令，并行地处理各自获取的信号．

三、操作型触觉传感器

成像型触觉传感器主要感受的是接触界面上的法向力，在传感器安装在机器人手爪上的情况下，为了完成各种操作任务，传感器必须感受目标物体相对于手爪的滑动、扭转，这就要求传感器也能感受接触界面上的切向力和力矩．另外，手爪操作物体运动时，还要有感受各个自由度受到的力和力矩的传感器．着重于感受目标物体的动态情况的触觉传感器，可称为操作型触觉传感器．

例 12.3 磁阻式操作型触觉传感器（美国 California 大学和 Bell 实验室研制）．

如图 12.10、12.11 所示，这种传感器安于机器人手爪的内侧．若干个面积为 $2 \times 2mm^2$ 的触感单元排成阵列形式，每个单元由三部分构成，一片厚约 $25\mu m$ 的永磁合金制成磁偶极子，它埋置于一种硅树脂弹性介质中，弹性介质厚约 1mm，粘连在装有半导体磁阻检测器的衬底上．

传感器的转换原理比较简单．磁偶极子原理上可看作是一个

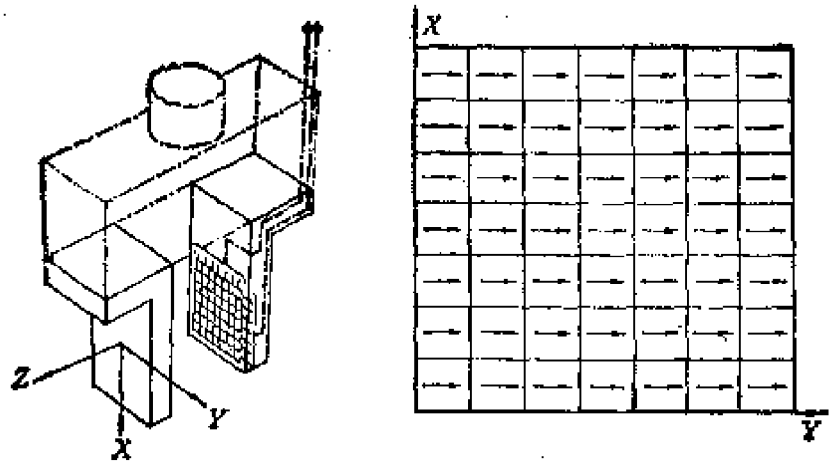


图 12.10　磁阻式触觉传感器阵列

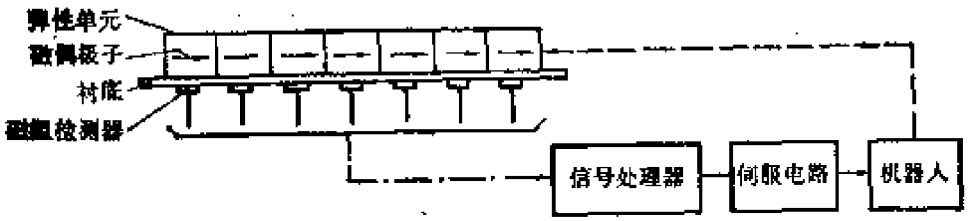


图 12.11　磁阻式触觉传感器系统

永磁小棒，当接触界面上的作用力使弹性介质变形时，它也随着相对于磁阻检测器发生位置偏移，于是，磁场发生变化，检测器根据磁阻效应产生电信号输出.

磁偶极子可有五个自由度：沿 X、Y、Z 三轴上的平移及绕 X、Z 轴的旋转. 每个弹性单元下面放置五个磁阻检测器，就可测出相应的变化量. 实际上，由于接触界面上法向应变分量很小，而且弹性单元的厚-宽比率较大，传感器只感受切向力（X、Y 轴上的平移）以及扭转力矩（绕 Z 轴的旋转）.

例 12.4　六自由度腕力/力矩传感器（瑞士 Lausanne 联合工学院研制）.

机器人的腕力/力矩传感器主要用于装配操作中，提供从手爪到臂控制系统的广义力反馈信息：三个自由度的力及三个自由度

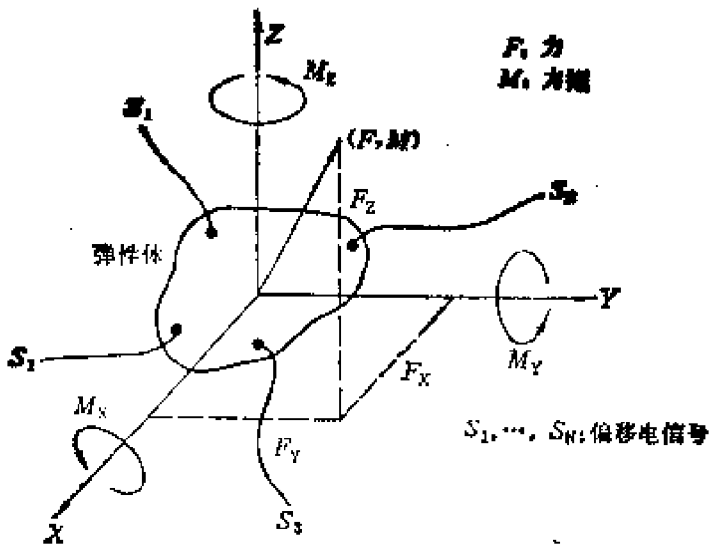


图 12.12　弹性体受到的广义力

的力矩(图 12.12). 一般,这类传感器都采用电阻应变式,在弹性结构上粘贴若干应变片,把广义位移(直线位移和转角)转换为电信号,但应变片难免存在温度、时间的稳定性问题. 本例中的传感器于是采用了一种新的原理,它由敏感广义力的一个弹性单元和敏感广义位移的六个检测器组成.

图 12.13 略示了传感器的原理模型. 刚性块 B_2 与机器人的臂固连,B_1 作为可动端面与手爪相连,B_1、B_2 之间由三个均匀分布的弹性支柱 L_1、L_2、L_3 联结在一起. 六个位移检测器均匀分布在 B_1 下方. 每个检测器以一个活动块与 B_1 相连,下有一个刚性小杆,杆端为一电磁感应线圈,处于一对通电线圈的缝隙中.

当 B_1 受到广义力作用时,解耦装置使不同检测器的刚性小杆只敏感六个自由度之一的广义位移(平移或旋转),而且转换为小杆沿其轴的伸缩,于是,感应线圈切割通电线圈对的磁力线,输出电信号.

检测器的输出通过多路模拟开关经模数变换后送往微处理

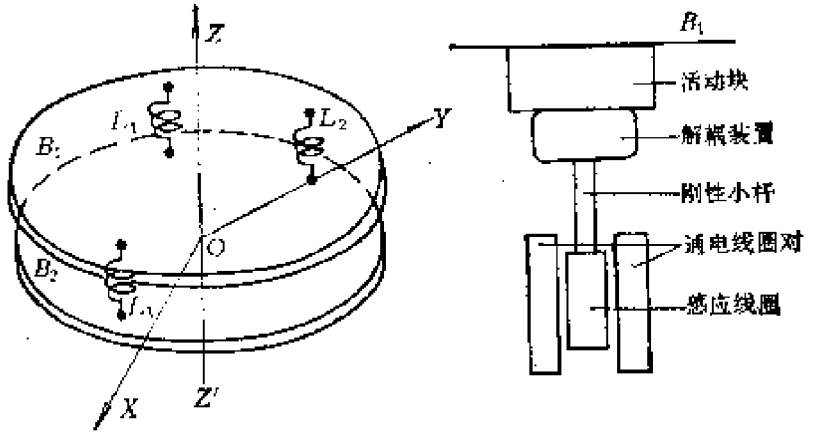


图 12.13　腕力/力矩传感器的原理模型

机，根据弹性支柱的刚度特性，以及检测器坐标系统与参考坐标系统间的坐标变换，计算出 B_1 所受的广义力.

整个传感器的分辨率为 0.1N，感受范围达 ±20N，误差与一般机器人的精度相一致，而且具有较好的线性度.

§12.3　具有触觉的机器人系统

触觉传感器无论装设在机器人本体(腕、手爪)，或是安置在机器人的操作台上，都必须通过硬件和软件的方式与机器人有效地结合，形成协调的工作系统. 本节扼要介绍实际构造触觉机器人工作系统所涉及到的通讯技术问题，然后给出应用实例.

一、触觉传感器与机器人的通讯

要使触觉传感器与机器人有效地结合，首先要有一个合适的系统工作环境，其中比较重要的是系统的内部通讯问题.

系统内部通讯常用的有下列三种接口. 离散 *I/O* 接口：所传送的是一些单个的输入、输出信号，而且是二值的，用于传感器与机器人之间的简单通讯. 例如，相互进行开、关等驱动控制，这适用于机器人仅依靠接触觉工作的系统. 并行口：它由多个二值

的输入、输出信号线组成，用于传感器与机器人之间的数码信号的同时读写，能满足限时通讯方式的要求。RS-232C 接口：它依据标准化的硬件约定串行地传送数据，是一种最简单的通讯方式，要求机器人与传感器采用统一的通讯方式和传送速率。其它可用的通讯接口还有 RS-422、IEEE-488 和 DMA（直接存取访问）等。

数据通讯速度与下列三个方面有关。硬件接口：如前所述，并行口显然比串行口的通讯速度快，但并行口要求合适定义的双向通讯标准和一定的软件支持，实现比较困难。数据格式：可选用 ASCII 编码或二进制编码，对于数值数据，二进制编码便于快速通讯，但难以表示字符类数据，一般的传感器和机器人都采用 ASCII 编码。数据类型：触觉传感器输出的原始数据一般先处理为某种类型的参数再送往机器人，原始数据量往往比参数量大达一个量级以上，因此，设法压缩原始数据，选用适当的参数类型就成为提高通讯速度的关键。

二、应用实例

机器人系统（图 12.14）的工作任务是把一个很小的尼龙衬套装配到一个节流阀杆的孔中。待用的节流阀杆是一些金属回旋体冲压件，杂乱地放在盒内。尼龙衬套为圆柱形，一端呈法兰盘状，由一个振动式送料带提供。

1. 系统构成

机器人为 UNIMATE PUMA 560（配有 VAL II 语言），一个六自由度力/力矩传感器（LORD 公司研制）装于机器人腕部，一块电磁铁取代手爪与腕力传感器弹性联结，这种安装方式可提供一定的柔顺性，补偿机械手响应输入的固有滞后。机械手面前的操作台上放置着一个 LTS-300 触觉阵列传感器（LORD 公司研制）。

LTS-300 是一种高分辨率的成像型传感器，它有 6400 个触

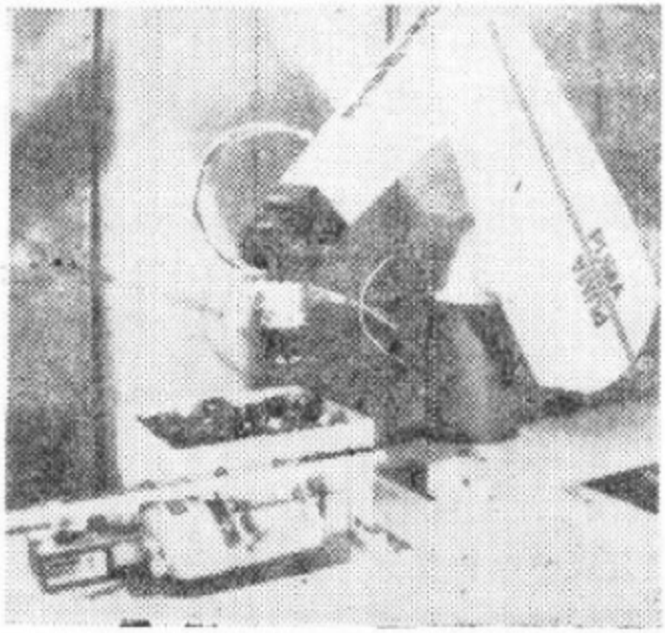


图 12.14　装配任务工作系统

感单元，排成中心距离为 0.08in 的 80 × 80 阵列．触觉单元的灵敏度量级为 10^{-2}N．传感器通过转换接口电路与 MC 68000 微处理机联成传感系统．

腕力传感器带有专用的预处理器（装在机械手上）和控制器．控制器可以把原始数据转换为力/力矩参数，且有可供用户使用的 FTL 语言．控制器还随时监视传感器的工作状态，如发现故障，即可发出信号，使机器人立即停止工作．

PUMA 560 的控制器即为整个系统的控制器，由于没有现成的实时串行接口，它通过 PUMA 560 的 *I/O* Module 与 MC 68000 以两个 8 位并行口相连．MC 68000 也用来对腕力传感器的数据作某些处理，传送通过 RS-232C 实现．这样，两个传感器的数据处理结果都依靠并行口与机器人通讯．另外，机器人的控制器还通过两条并行 *I/O* 输出线控制电磁铁的操作状态．

整个系统的构成如图 12.15 所示．

2．工作过程

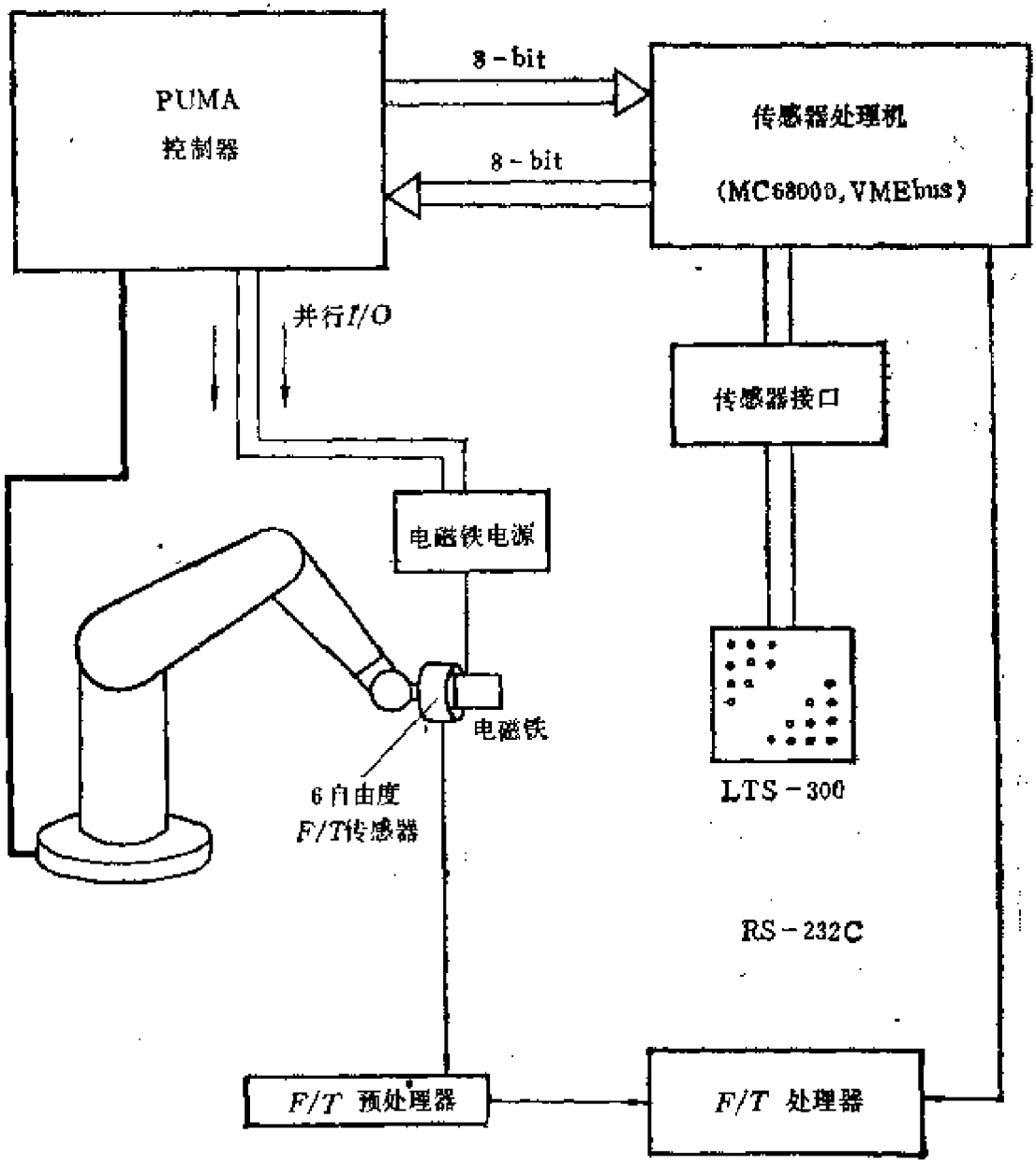


图 12.15　具有触觉的机器人系统构成

系统的工作过程如下：

(1) 尼龙衬套的获取和定位.机器人腕部装有一个真空吸管，它从振动式送料带上拣取衬套，并把衬套插入操作台上一个装配槽的夹具中，使得最后装配时可把节流阀杆放在衬套上方，并可滑

入装配槽.

(2) 节流阀杆的获取. 电磁铁从料盒吸起阀杆，腕力传感器判断是否接触阀杆并计算拣起物的重量，接触信号和重量数据都通过并行口送往机器人控制器，如果没有吸到阀杆，控制器接着运行搜索程序；如果拣起的阀杆多于一个，机器人则把它们放在一边，另行搜索，直到只拣起一个阀杆，然后把它放在 LTS-300 的触感面上.

(3) 节流阀杆的放置. 在重量计算过程中，阀杆重心作用线相对于腕的坐标可同时算出，如果作用线垂直于工作面 X-Y 平面，据图 12.16 就有

$$X = -\frac{M_Y}{F_Z}, \quad Y = \frac{M_X}{F_Z}$$

式中，M_X、M_Y、F_Z 分别为腕力传感器所得的力矩和力.

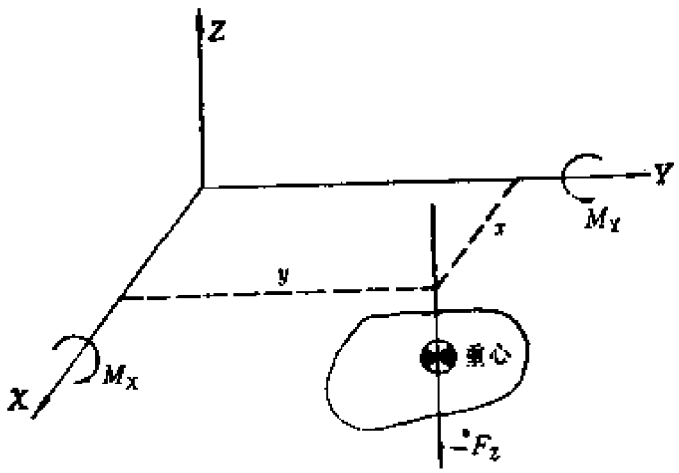


图 12.16 阀杆的重心计算

这一结果被送往 PUMA 控制器，转换成机器人的坐标，可用来较为精确地保证把阀杆放在 LTS-300 的中心位置附近. 当阀杆与 LTS-300 一有接触，腕力传感器即可测出，这时，机器人停止动作，转去计算 (利用 VAL II) 电磁铁到触感面的距离，如果这一距离大于某个定值，说明阀杆处于"悬挂"状态，机器人简单地

"撒手"就不容易使整个阀杆都落于触感面之内，因此就需要重新调整放落位置。

(4) 节流阀杆的辨识和定位。阀杆放置在触感面上有三种可能的"稳态"，每种"稳态"都有唯一对应的"着陆"面形状。LTS-300 经过扫描、阈值处理得到"着陆"面的二值图像，然后再把它处理成一种三角形。把三角形的面积和纵横比(图 12.17)这一组合特征与预存的标准模型一一比较，就可判定阀杆所取的是那一种"稳态"，而阀杆的几何尺寸是已知的，在确定的"稳态"下，"着陆"面三角形的位置、姿态也就确定了整个阀杆的位置和姿态。

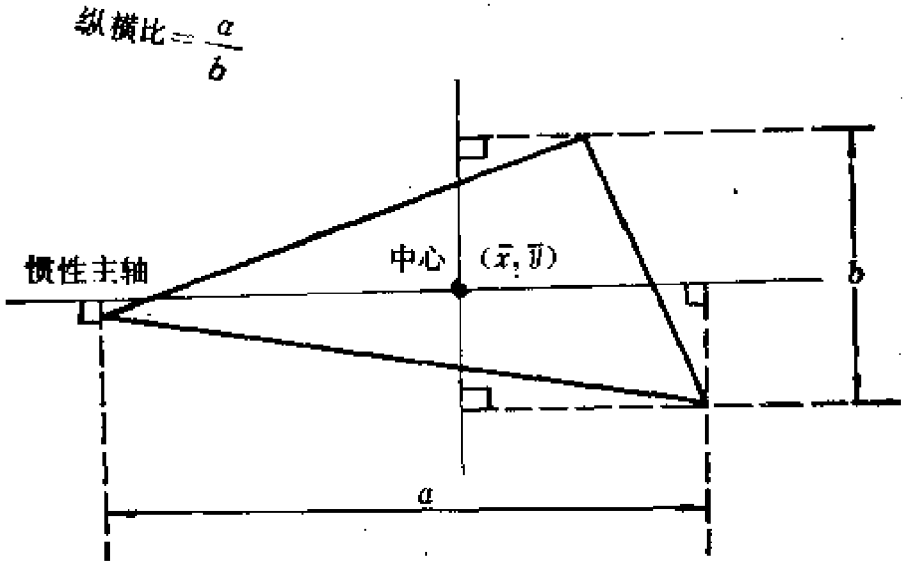


图 12.17 "着陆"面三角形的特征

(5) 装配。三角形的位置由其中心表示，姿态由其惯性主轴的角度表示，这些参数连同"稳态"序号送往 PUMA 控制器，换算成与当前"稳态"对应的机器人吸取阀杆的位置和姿态，阀杆被送往装配槽上方。按照一个示教程序，机器人把姿态合适的阀杆沿槽下推一定距离，使它与夹具中的尼龙衬套紧紧配合。

整个装配过程约需 25—50s，人工完成这项操作的平均时间为 6s。

由于 LTS-300 具有足够的识别精度，而且衬套的位置、姿态也已固定，在上述装配实例的最后步骤中，仅仅利用示教程序，就能保证衬套与阀杆上孔的装配成功. 在较为复杂的装配任务中，当然还可进一步利用腕力传感器的力觉信息，自动搜索、校正配合位置.

§ 12.4 接近觉传感器

如前所述，机器人触觉所提供的信息仅当它与对象物体发生实际接触时才能获得，视觉是在二者并未接触并且相隔一定距离时获取物体的图像信息，而机器人在实际工作过程中，往往需要感知正在接近、即将接触的物体，以便作出降速、迴避、跟踪等反应，这就要求具备一种"接近"的感觉. 接近觉与触觉略有相关，内容简单，我们把它附于本章，对它的传感器作扼要介绍.

接近觉传感器主要敏感机器人手爪与对象物体的逼近程度，感受范围可小至几十 mm，甚至一个 mm. 大多数接近觉传感器只关心感知范围内是否存在物体，也有的可量测它们与物体之间的精确距离.

各种磁场式传感元件可制成性能优异的接近觉传感器. 例如涡流式传感器，它在一根探针端部产生交变磁场，磁场在灵敏区域内的导磁物体中感应出涡流，而涡流本身又产生磁场反抗传感器的磁场，于是，装在探针上的线圈敏感出磁力线密度的变化，从而可判断物体的存在. 这种传感器的接近觉精度可做到不大于 1 mm. 又如，在传感器端部安装一小块永久磁铁，当有导磁物体存在时，可构成一个闭合磁路，进而触发一种舌簧接点开关的接通，发出接近觉信号. 这种传感器具有无功耗的特点.

接近觉传感器也可依据声学原理构造. 例如圆柱形空腔谐振器，一端敞口，封闭端发射驻波，当有物体接近敞口时，谐振器有所

阻塞，空腔内的驻波分布状况发生变化，空腔壁上的拾音器则能测出波形变动时所产生的声压变化，这种传感器可精确地量测出物体与传感器的距离。

利用光学原理制成的接近觉传感器一般都是量测物体的反射光，传感器的可靠性主要取决于光源的寿命及其经受振动的能力。此外，还有利用静电感应，接触电容、射流等技术制成的各种接近觉传感器。

第十三章　机器人规划

§13.1　概述

规划一词的日常理解是行动之前拟定行动步骤。在人工智能的研究范围中，规划实际上就是一种问题求解技术，即从某个特定问题的初始状态出发，发现一系列行为或构造一系列操作(也称算子)步骤，达到解决该问题的目标状态。规划之所以另树一帜，是因为它比起一般的问题求解，更注重于解决问题的过程，而不是求解的结果。而且，它所面对的通常是真实的问题世界，而不单是抽象的数学世界。因此，往往要牵涉到环境的动态变化、行为或操作的执行时间、多个作用因素等复杂的问题。规划系统与专家系统在实现技术上存在许多共同之处，例如知识表达、搜索策略、推理机制等，因而在文献中，有时把具体的规划系统归类于专家系统加以介绍。

规划系统用于机器人，即为机器人规划，也称机器人问题求解。感知能力使机器人认识对象和环境，而运用感知信息，产生适应对象和环境的动作，最后解决问题，还要依靠规划功能。例如，给定工件装配任务，机器人按照什么步骤去操作每个工件？在杂乱的环境下，机器人如何寻求避免与障碍碰撞的路径，去接近某个目标？规划功能的强弱反映了智能机器人的智能水平。在智能化程度最高的机器人语言，即所谓的作业目标水平级(任务级)语言中，它的重要功能就基于自动规划生成。

机器人规划系统的基本任务是：在一个特定的工作区域中自动地生成从初始状态到目标状态的动作序列、运动路径和轨迹的控制程序。图 13.1 表明了典型系统的组成，各基本单元的功能如下。

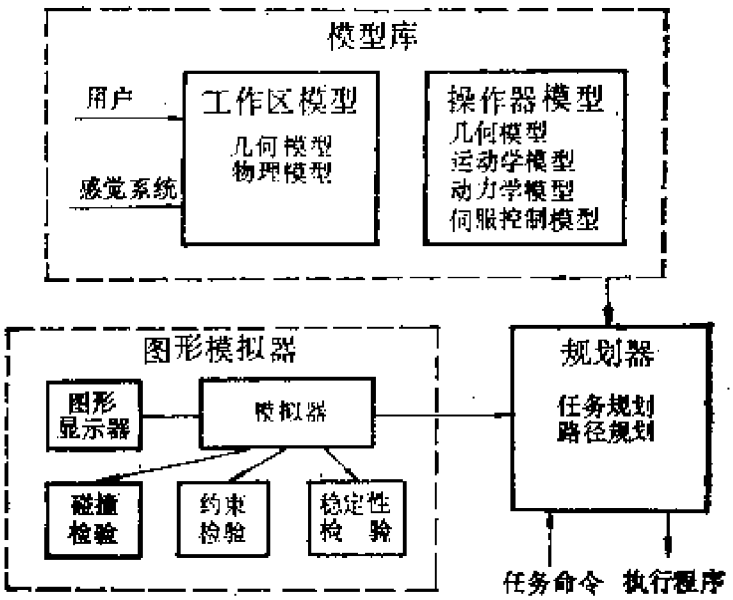


图 13.1　典型的机器人规划系统

工作区模型描述机器人工作区内物体的几何、物理性质，提供对象和环境的信息。这些信息可由用户或感觉系统输入或修正。

操作器模型的内容主要为规划时考虑具体实现所用。

图形模拟器用以检验规划所产生的操作器的动作、运动轨迹的可行性。

规划器中包括两级不同规划问题的子系统。任务规划子系统根据任务命令，自动生成完成该任务的机器人执行程序，如将任务理解为工作区的状态变化，则它生成的即为把初始状态一步步变为目标状态的操作序列。运动规划子系统调用工作区模型和操作器模型的信息，首先将任务规划的结果变成一个无碰撞的操作器运动路径，这称为路径规划；然后再将路径变为操作器各关节的空间坐标，形成运动轨迹，这称为轨迹规划。

任务规划与运动规划虽然有逐步细化的关系，但它们求解的方法并不相同，实际研究中也分属两类不同的规划问题．前者旨在产生动作序列，后者旨在设计空间路径．以下将分别给予扼要介绍．轨迹的生成与机器人控制问题有关，第三章中已经作了讨论．

§ 13.2 任务规划

图 13.2 例示了一个典型的机器人任务规划问题．在这个简单的"积木世界"中，要求机器人把三个积木块 A、B、C 重新堆叠，从堆叠关系的初始状态变为目标状态，机器人的操作行为只有两种：拿起某一块积木，或放下某一块积木．规划的求解则是从初始状态出发，确定一个操作序列，使得"积木世界"在每个操作行为的相继作用下，经过一系列中间状态，最后达到目标状态．

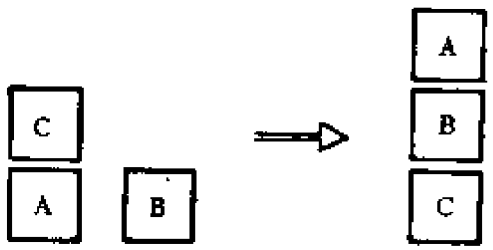


图 13.2 机器人任务规划问题

规划问题在求解过程中，或者使用所谓的"目的-手段"分析法（means-ends analysis），逐步比较并设法缩小当前世界状态与目标状态之间的差别，最后达到目标状态；或者使用简化问题目标的方法，把目标不断分解成较为容易解决的子目标，逐步搜索、确定出操作序列．

任务规划的求解需要解决下述三个基本问题．

知识表示问题．其中包括：怎样描述世界状态，例如积木块之间的堆叠关系；怎样描述操作行为，例如机器人"抓"、"放"动作

的作用条件、作用效果；怎样描述世界状态在被操作后发生的变化；从总的求解思路上看，怎样表示问题空间，例如把总的目标分解为不同层次意义的子目标；怎样表示问题空间中的多作用因素以及动态变化；另外还包括怎样描述各种推理知识等等.

搜索问题，即发现、确定能够达到目标状态的操作序列的过程. 当各种可能序列的数目极大（而且其中大多数都不能达到目标状态）时，搜索过程中的计算工作量可能随操作种数呈指数增长，称为"组合爆炸"问题. 因此，在规划过程中，常常采用各种启发式信息来限制搜索的工作量.

子目标冲突问题. 当存在若干合取子目标时，即规划问题要满足一个以上的条件时，由于各个子目标的完成次序并未指定，而且往往某个子目标的实现可能会破坏另一个子目标的实现，如何查找、排除这类故障，便成为规划求解的关键. 子目标冲突亦称"约束"或"制约"问题. 例如，"油漆天花板"和"油漆梯子"就是"油漆任务规划"中的一对合取子目标，任何人也不会登着刚油漆好的梯子去油漆天花板，因此天花板必须先油漆，这一信息要在推理过程中得到发现、解决. 前述"积木世界"的规划问题也会出现类似的"冲突".

知识表示是一般问题求解技术的共同问题. 搜索问题虽然也常存在，但在规划过程中，它与"冲突"问题密切相关. 合取子目标的完成次序如果过早确定，就可能造成规划的失败，因而不得不回溯到某个选择点，导致搜索工作量的增加.

至今为止，人工智能方面已初步形成了四种有关任务规划问题的方法. 非层次规划、层次规划、骨架式规划和机遇式规划，以下着重介绍前两种方法.

一、非层次规划

几乎所有的规划问题都固有一种层次结构，即递归的目标与

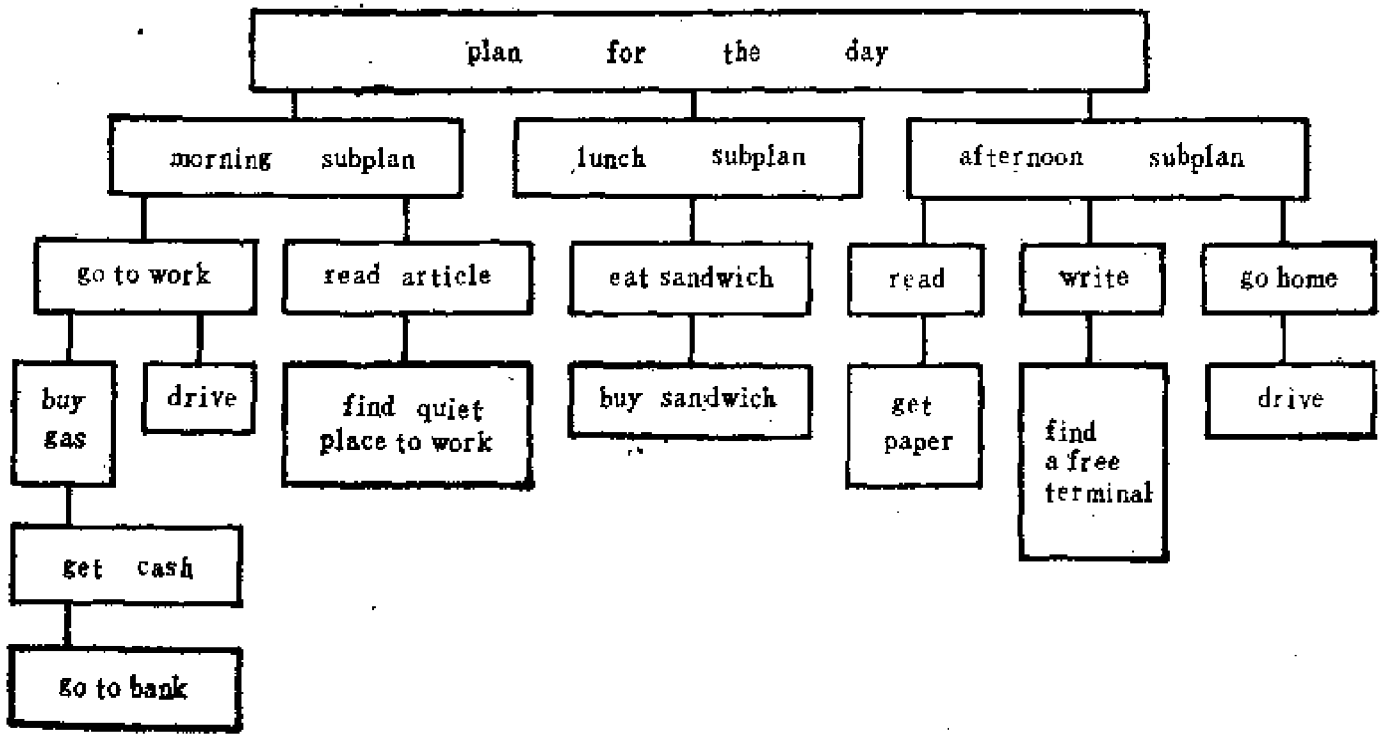


图 13.3 规划的层次结构

子目标之间的组成体系，上层与下层目标之间的关系实际上都相应于某个操作与其先决条件之间的关系，例如图 13.3 所示。但就规划问题的表示意义来说，却可以把目标分成抽象与具体，或重要与次要的程度不同的层次。也可以对所有子目标一视同仁，即在表示意义上只认为是一个层次。前者称为层次规划方法，后者称为非层次规划方法。

HACKER 系统（M.I.T. Gerald Sussman 研制）是一个非层次规划的代表性例子。它在功能上属于一种“技能获取”模型。一项“技能”是一些“过程”的集合，一个“过程”解决某一类问题，随着解决问题的深入，系统不断选定已有的“过程”或产生新的“过程”来满足需要，而新的“过程”可能出现“故障”，系统又必须对“过程”加以合适的“修正”。系统产生的“过程”序列实际上就是规划的解，而系统对“故障”的排除则体现了解决子目标冲突的方法。

HACKER 系统采用了过程式的知识表示方法。它有一个“答案库”（answer library），存放解决各类问题的“过程”；一个“知识库”（Knowledge library），存放有关问题范围的事实；一个“编程技术库”（programming-techniques library），当“答案库”中没有合适的“过程”可选时，可用它产生新的合适的“过程”；另外，系统还有一些存放“故障”模式及其“修正”程序的库。

下面，以图 13.2 所示的机器人规划问题为例，说明 HACKER 的方法。

规划问题可表示为一对合取子目标：

ACHIEVE (AND (ON A B) (ON B C))

这对合取子目标实际上是相互制约的，但系统并无子目标完成次序的任何信息，它采用了一种“线性假设”（linear assumption）策略，即假定子目标为相互独立，因而可以随意的次序完成它们。

系统按此假设，构造一个两步规划；子目标的完成次序为

(1) ACHIEVE(ON A B)

(2) ACHIEVE (ON B C)

在答案库中寻找与子目标(1)相匹配的"过程":

(TO (MAKE (ON X Y))
    (PUTON (X Y)))

将 X、Y 代之以 A、B，得知，要使得A在B上，只须执行操作

p1: (puton A B)

即把A放在B上.

模拟这步操作的执行，发现不能成功，因为A上有C. 系统查找故障库和知识库，确认当前障碍是缺少一个当前操作的先决条件:

(FACT(PREREQUISTE(EXPRESSION(CLEARTOP OBJECT))
                 (HAVE()(MOVES EXPRESSION OBJE(T)))))

即，要移动某物体，它的顶上必须没有其它东西.

HACKER 对于"缺少先决条件"故障采用了"插入"调整方法：把某操作缺乏的条件视为第二层子目标，在此例中即为ACHIEVE(CL A)，再寻找与之匹配的"过程"，把所得到的一个操作插在原操作之前. 于是已有的规划解为

p1: (clear A)

p2: (puton A B)

p1 意即把A顶上的东西挪走. p1、p2 的模拟执行都获得成功，如图 13.4 所示，"积木世界"由初始状态 s0 依次变为中间状态 s1、s2. 至此，子目标(1)完成.

子目标(2)以类似的方式进行，得到操作

p3: (clear B)

p4: (puton B C)

这时，在模拟执行 p3 时，发现了另一种故障："子目标冲突"，即 p3 生成的状态 s3，破坏了已完成的子目标状态 s2.

HACKER 对于"子目标冲突"故障采用的方法是，允许破坏，

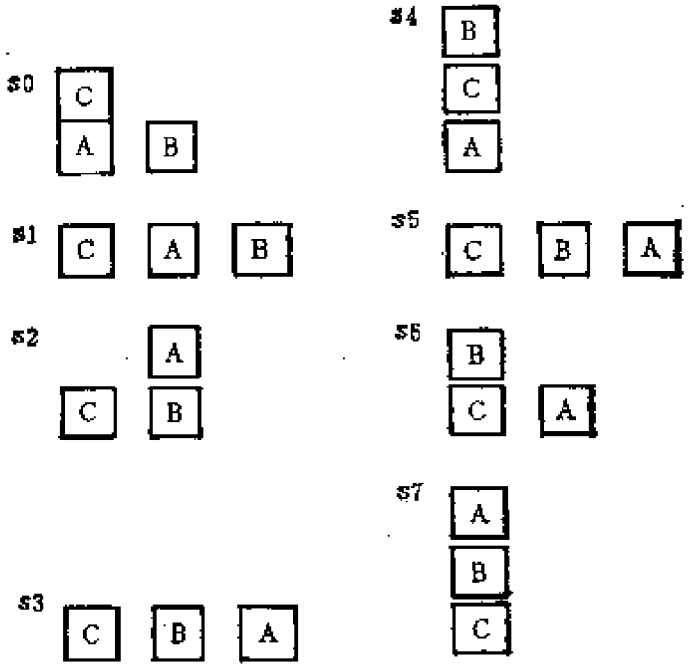


图 13.4 "积木世界"的状态

但在当前操作执行之后再试一次上步操作，如果无故障发生，则顺序进展，如果故障仍然存在，则回溯到初始状态，改变第一层子目标完成次序．在现在的情况下，系统显然不得不回溯，按照先子目标(2)后子目标(1)的次序，重新生成规划的操作序列．即有

p1：(puton B C)

p2：(clear A)

p3：(puton A B)

当系统在模拟执行 p2 时，又发现了"子目标冲突"故障，但这次按它的调整方法再试 p1，却获得了成功．所得规划的解为

p1：(puton B C)

p2：(clear A)

p3：(puton B C)

p4：(puton A B)

相应的状态变化为 s0，s1，s4，s5，s6，s7．s7 即为目标状态．

由上可知，HACKER 最后所得的规划并非最优。另一个非层次规划系统 INTERPLAN (Edinburgh 大学 Austin Tate 1974 年研制)，对于"子目标冲突"的处理方法有所不同，它能够在不同层次的子目标之间调整次序。在上例中，INTERPLAN 起初如同 HACKER 一样，先尝试第一层子目标(1)、(2)的次序调整，当发现重复产生冲突时，不是采取再试上步操作的作法，而是把造成冲突的第二层子目标 ACHIEVE (CL A) 提前执行，它最后所得的规划则为

p1: (clear A)

p2: (puton B C)

p3: (puton A B)

相应的状态即为 s0, s1, s6, s7。显然，INTERPLAN 通过"子目标提升"的办法得到了最优的规划解，但它的排序空间远大于 HACKER，因而也必须化费很大的工作量。

在合取子目标对数较多的情况下，HACKER 和 INTERPLAN 的方法效率显然很低。为此，Richard Waldinger 1977 年提出用所谓"目标回归" (goal regression) 的方法处理冲突现象，并且据此设计了一个构造性的规划系统，系统逐步扩充达到各子目标的操作步骤，及时检测、处理冲突现象。

"目标回归"方法的基本思想是：对于合取子目标 $P \wedge Q$，若先完成 P 后，Q 无法达到，说明为完成 P 所取的操作步骤 $E_1, E_2, \cdots, E_m$ 中，必有某个 E_i 破坏了为完成 Q 所要取的操作步骤 $F_1, F_2, \cdots, F_n$ 中的某个 F_j 的前提条件，如果能确定 E_i 的位置，把 Q 从后面移动到 F_j 的前面完成，这就能避免了冲突；如果不能找到 F_j，则说明先完成 P 后完成 Q 的尝试不合理，这时可交换 P、Q 的完成次序，再重复上述过程。

对于前述例子，在先完成子目标 (1) 即 ACHIEVE (ON A B) 之后，得到两步操作：

p1: (clear A)

p2: (puton A B)

要接着完成子目标(2)即 ACHIEVE(ON B C)，须有操作 (puton B C)，而它的前提条件是"B 上无物"，显然，这一条件已被 p2 破坏，这样，只要把 ACHIEVE(ON B C) 从后面移到 p2 之前，即可解决冲突问题．规划的最后结果为

p1: (clear A)

p2: (puton B C)

p3: (puton A B)

相应的状态变化为 s0，s1，s6，s7 (图 13.4)．

实际上，"目标回归"方法必须等到发现Q不能完成时，把Q一步步地试着往前移动，直到发现 F_i 的位置，即通过检测搜索的方式确定插入位置．如果冲突现象很多，反复搜索的工作量很大，但这种方法避免了 HACKER 与 INTERPLAN 盲目回溯、重复完成某些子目标的缺点．

以上介绍了非层次规划解决"子目标冲突"问题的三种方法．前两个例子基于"线性假设"，依靠回溯、重新排序；后一个例子基于"目标回归"，实际上也是一种排序，而排序的工作量之大将造成复杂问题的低效，这正是非层次规划对目标的意义层次不加区分、缺乏总体考虑的后果．

二、层次规划

层次规划方法首先勾画出一个完整但又比较粗略的规则，然后逐步细化、逐步明确，直到足以具体完成整个规划的各步操作．层次规划方法实际上把不同性质的问题放在不同层次上加以考虑．

NOAH (Earl Sacerdoti 1975 年前后研制于 SRI)是个典型的层次规划系统，它是作为一个计算机咨询系统的一部分而设计的．

NOAH 意即“行为层次网络”（Nets of Action Hierarchies），它通过不断扩充一个所谓“过程网络”（procedural net）来有层次地求解规划问题。

系统具有多种知识表示形式：过程知识，亦称领域知识，包括了一些函数，它们可把目标语句扩充为一些子目标，而且能够模拟用以转换世界状态的操作的行为；陈述知识，亦称规划知识，用来表示函数的执行效果，另外还有一组所谓“评论家”（critics）的过程，通过对陈述知识的全局检测、推理，发挥调解冲突现象，消除冗余操作等功用。

每一层次的“过程网络”都由节点和弧构成。节点表示相应层次的子目标（也称抽象操作），它们依靠指针与一些函数相链接，节点上还附有增删表（陈述知识），在扩充节点的函数执行后，表中就记录着有关在完成该节点后将会增加或删除的世界状态的某些描述。网络中的弧表示子目标的执行关系。起初，作为第一层的“过程网络”只有一个节点——规划问题的目标，接着，执行与之相链接的函数，把这个节点扩展为若干表示子目标的第二层节点，完成第二层子目标后就意味着可以达到第一层子目标。随后，类似地扩充下去，最后一层节点则全部都是通过一些简单的操作就可立即达到的子目标。每一层节点的确定都必须经过“评论家”的检测，形成一级规划。

下面仍以原先的积木世界规划问题（再示于图 13.5）为例说明 NOAH 的方法。

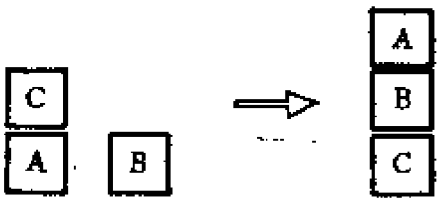


图 13.5　积木世界规划问题

系统以三个命题表示世界的初始状态：

(ON C A)　　；C 在 A 上面
(CLEAPTOP C)　；C 上面无物
(CLEARTOP B)　；B 上面无物

目标状态也用一个命题表示：

(AND(ON A B) (ON B C)) ；A放在B上面，而且B放在C上面.

规划目标 ACHIEVE (AND(ON A B) (ON B C)) 是网络的第一个节点，形成第1级规划(图13.6).

图13.6 第1级规划

对于合取子目标对型节点，系统采取简单的扩展，只是把两个子目标拆成并行的节点，而不确定任何前后执行次序. 这里，NOAH 实际上采用了所谓的“最小约定”(least-commitment)策略，即，在没有理由对子目标排序的情况下就推迟排序，直到理由成熟为止. 而且，NOAH 还不断检测规划是否有子目标冲突的可能，同时把冲突现象调整在形成一级规划之前.

于是，第2级规划就如图13.7所示，其中，S和J是两个特殊节点，S表示“分离”，J表示“合并”. 在规划形成之前，照例要由“评论家”们加以检测，但这一级并未发现问题.

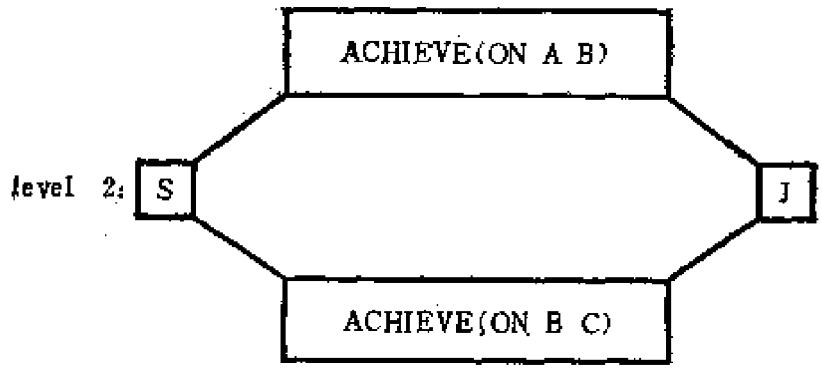


图13.7 第2级规划

接着，执行负责扩展 ACHIEVE 型节点的 PUTON 函数，得到图13.8的 level 3”，这时有两个“评论家”对网络进行检测调

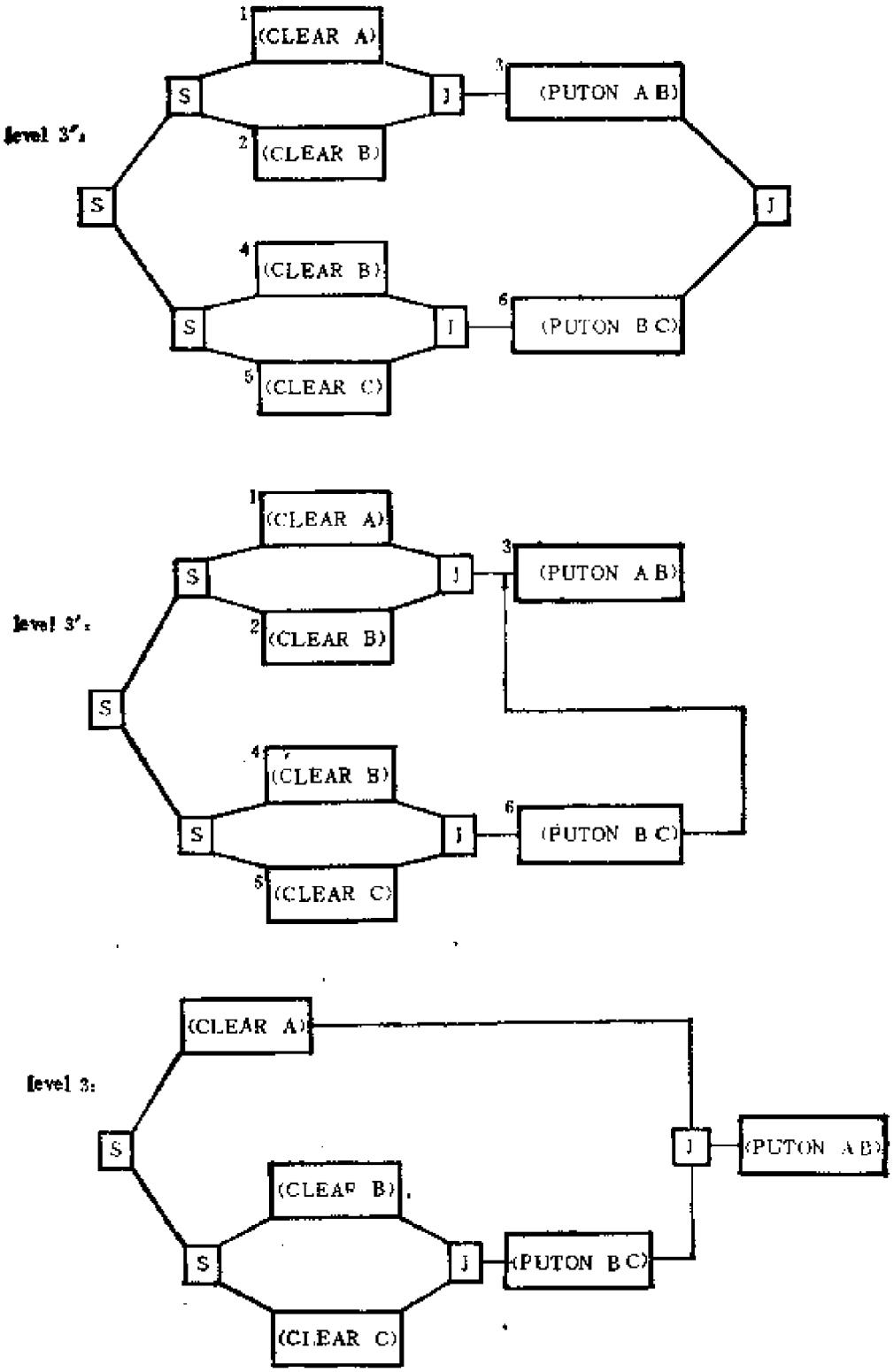


图 13.8　第 3 级规划的形成

整. “冲突调解评论家”,它的作用是,若为达到一个目标所取的某个行为消除了另一目标的某个行为的先决条件时，就对这两个行为局部排除,使两者互不侵犯.在 level 3′ 的情况下,由于节点 3的删除表上已有(CLEARTOP B) 一项,表明完成节点 3 后,B 上面

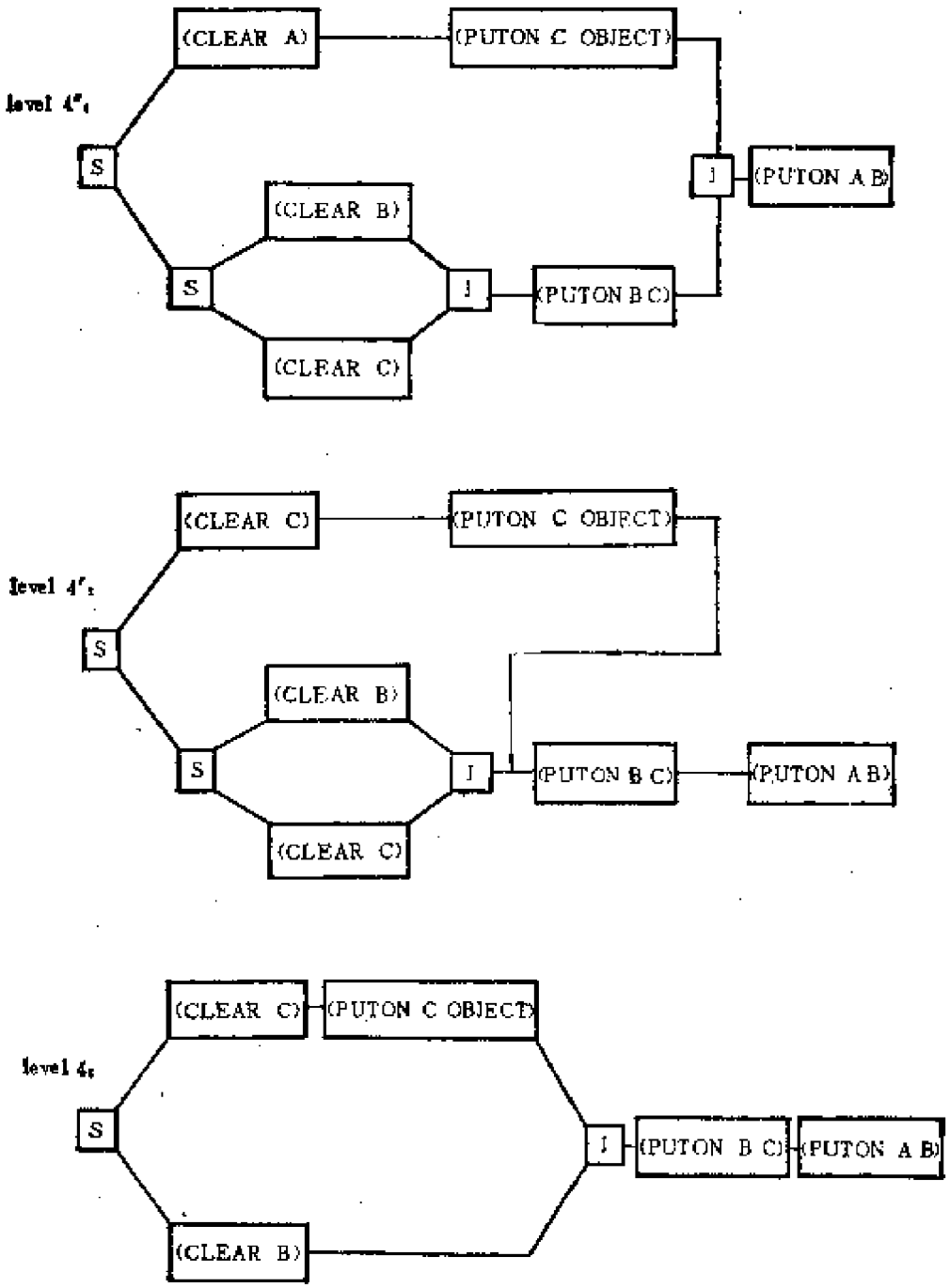


图 13.9　第 4 级规划的形成

NOAH 用作机械装配方面的咨询系统时，对于训练有素的机械师，它只需提供高层次的规划指令："把安装支架用螺钉固定在机座上"，而对于初学者，它可以提供低层次的规划指令，甚至包括

"螺钉如何拧"这样的操作细节.

在层次规划方法中,"层次"的区分原则不止一种, NOAH 的层次概念是子目标的详略程度,而在另一个关于分子遗传学实验的规划系统 MOLGEN(M. Stefik 1980 年研制于 Stanford 大学)中,是从控制结构上把规划过程分为三层: 顶层为"策略层",作出"最小约定"、"启发式"等规划策略型式的选择; 中间层为"设计层",确定提出目标、细化算子等规划步骤;最低层为"规划层",针对具体的问题领域,作出具体的可实行的规划. 由于"策略层"相对于"设计层"实际上是一种"规划的规划",于是由此提出了所谓"元规划" (meta-planning) 的概念. 另外,该系统还运用了一种"约束传递" (constraint propagation) 的方法,把众多子目标之间的"冲突"可能形式化为各种约束条件,通过相互传递的方式综合起来,限制求解空间,这种方法较好地实现了"最小约束"策略.

三、骨架式规划和机遇式规划

骨架式规划 (skeletal planning) 和机遇式规划 (opportunistic planning) 都是模拟日常生活中人的规划思路,这里仅介绍它们的基本思想.

工程技术人员在设计某个新的实验过程时,很少一切都从头开始,"定做"规划,而常常是参照以往的经验,先形成一个只包括大体步骤的构思,然后再根据实际条件把每一步骤具体化. 骨架式规划方法正是如此. 系统事先存放了一批解决许多不同类型问题的轮廓过程,称之为"骨架". 骨架有粗有细,规划过程中首先根据细化工作量最小的原则选择一个目标与给定问题相类似的骨架,然后用一些不同层次的算子,即解决具体问题的方法,逐步填入比较抽象的骨架步骤中,并根据某些原则加以评定,确定完整的规划结果. 与层次规划方法相比,骨架式规划另有特点,它的初始规划骨架不是当时生成的,而是原来存有的,只需着重于选择与细

化工作;它的骨架步骤是相互独立的,细化过程中出现的冲突问题由较小的子规划解决;它的知识表示问题更为突出,特别需要大量的问题领域的专门知识. 总之,骨架式规划方法比较简单,擅长于实验过程设计这类共性明显的规划问题. 骨架式知识表示的概念原先在自然语言理解方面已有研究应用,作为规划方法也已在 Peter Friedland 1979 年研制的 MOLGEN 系统(与前述 MOLGEN 系统类似,也用于分子遗传学的实验设计)中得到了实现.

Barbara Hayes-Roth 和 Frederick Hayes-Roth 1978 年进行了一项关于人的规划过程的试验. 受试者们面对一张城市地图各自作出包括十件事情的"一日活动"计划. 通过分析记录,试验者发现人的规划过程远比已有的人工智能规划系统复杂,甚至极不相同. 他们认为,人的规划过程是机遇式的,存在"由底向上"的成分,即当世界状态中出现某些刺激规划者注意的机遇(例如日常活动需要考虑的时间、地点、兴趣、常识等)时,一些具体的行为步骤随即就会引入规划,而不像层次规划方法那样是一个"由顶向下"的细化过程. 另外,试验者认为人的规划过程又具有"多向性",即规划过程可以在几个不同的抽象层次上同时展开,往往先形成一个一个的可喻之为"岛群"的独立的局部规划,然后再通过一系列的规划行为链接在一起,相互扩展,构成整体,而层次规划则要求在每一个抽象层次上都形成一个完整的规划.

Hayes-Roth 和 Hayes-Roth 针对"一日活动"建立了一个机遇式规划的模型,主要采用了一种带有五个"面"的"黑板"来控制规划的进行. 其中有: "规划面",记录着各项活动的具体过程;"元规划面",记录着问题类型、规划结果评价等推理判定;"规划抽象面",记录着把各种活动抽象成轻重缓急程度不同的判定;"知识基面",存放着活动事件表、地图等知识;"执行面",编排所有的规划决策. 前四个"面"中的信息都由若干"专家"程序提供,它们的运行次序并不一定,而是随机地相互触发、相互调用,运行结果分

送"黑板"的有关面，最后经过"执行面"综合成规划结果。

模型的作者们认为，机遇式规划方法尽管在解决不了目标冲突现象时也许会放弃某些目标或局部修改规划，但由于它的灵活性，解不唯一等特点，对于复杂的规划问题，效率上要优于层次规划方法。总之，这对于真正用机器智能实现任务规划是一种有意义的尝试。

§ 13.3 空间路径规划

当机器人的手爪，臂或本体要穿行存在障碍物的外部世界，去达到某个目标位置时，就需要在空间确定一条无碰撞的穿行路径，形成空间路径规划问题，也称无碰路径规划问题。与任务规划有所不同，在此，"规划"的含意实际上是直观地求解带有约束的几何问题，而不是操作序列或行为步骤。另一方面，如果把运动物体看作要研究的问题的某种状态，把障碍物看作问题的约束条件，而无碰路径则为满足约束条件的解，那末空间路径规划就是一种多约束的问题求解过程，这不但符合"规划"的广义理解，而且为复杂问题的描述和求解提供了新的思路。

空间路径规划一般分解为两个子问题：

寻空间问题，在某个指定区域R中，确定物体A的安全位置，使它不与已有的其它物体 $B_i(i=1,2,\cdots,m)$ 相碰。

寻路径问题，在某个指定区域R中，确定物体A从初始位置移动到目标位置的安全路径，使得移动过程中不发生A与B_i的碰撞。

显然，一系列安全位置就可连成安全路径。

对于空间路径规划的求解，早先有过一种"假设与测试法"，即，先假设一条初始位置到目标位置的路径，然后查看路径上所选的状态是否安全，如可能碰撞，则修正路径，这样反复试验直到达

到目标位置。这种方法直观，但需要反复计算立体相交问题，而且没有任何启发信息，既复杂又费时间。

还有一种"罚函数法"，即根据障碍物的分布，对空间状态定义一种罚函数，例如，对可能碰撞的状态，罚函数取值无穷大，运动物体与障碍物相距愈远，罚函数取值愈小，把所有障碍物在运动物体所在状态的罚函数求和，得到总罚函数的值，然后按最小值决策路径。这种方法把障碍物的限制程度用简单方法加以组合，有一定的方便，但只解决局部优化，很难通盘考虑整个空间，保证算法的有效性。

另有一类方法，从无碰撞空间的显式表示入手，设法回避求物体相交的复杂问题，得到不少有效的研究成果，下面分别介绍其中的两种。

一、位姿空间法

位姿空间法(由 Tomás Lozano-Pérez 自 1979 年起开始研究)把运动物体的位置和姿态的描述简化成所谓"位姿空间"(Configuration Space)中的一个点，由于障碍物的存在，运动物体在该空间中就有一个相应的禁区，称为"位姿空间障碍域"(Configuration Space Obstacls)。这实际上是构造了一个虚拟的数据结构，把运动物体、障碍物及其几何约束关系作了等效的变换，简化了问题的解决。

下面按二维空间的情况说明方法的基本思想，所讨论的物体都为刚性凸多边形。

设在运动物体 A 上固联一个坐标系，其原点为 A 的某个顶点 rv_A。开始，固联坐标系与参考坐标系重合。由于构成 A 的所有点的位置和姿态都可相对于固联坐标系得到确定，因此 A 在空间的任意位置和姿态就可以借助于一个集合 $(x、y、\theta)$ 来表示。其中，$(x、y)$ 为 rv_A 相对于参考坐标系的位置，θ 为固联坐标系围绕

rv_A 旋转后与参考坐标系的夹角(三维情况下,可选为欧拉角).

集合 $(x、y、\theta)$ 的一个取值表示 A 在空间某种状况 p 下的位置和姿态,称为 A 的一个位姿 p. A 的初始位姿点设为 s,这时显然有 $x=0, y=0, \theta=0$. A 的所有位姿点构成一个位姿空间,记为 Cspace_A, x, y, θ 则为该空间中的三个坐标.由于 A 的位置、姿态变化可由位姿点来描述,在这种意义下,一个运动物体在位姿空间中就被映射为一个点 rv_A.

对于障碍物 B_i,由于它们的位置、姿态固定不变,只需把它们对于 A 的约束关系映射到位姿空间中去.

运动物体 A 与障碍物 B 发生碰撞的所有状况在位姿空间中构成一个位姿空间障碍域,记为 $CO_A(B)$,定义为位姿点的集合

$$CO_A(B) \equiv \{p \in \mathrm{Cspace}_A \mid (A)_p \cap B \neq \phi\}$$

式中 $(A)_p$ 表示了所有构成与位姿点 p 相应的 A 的点集. 上式说明,如果 $p \in CO_A(B)$,那末 $(A)_p$ 与 B 相交,因而 p 是不安全的. 反之,任何 $p \notin CO_A(B_i)$, $i=1, 2, \cdots, m$, 都是安全的.这样,关于 A 的寻空间问题和寻路径问题,就分别等价于在 Cspace_A 中 $CO_A(B_i)$ 之外的区域里寻找一个 A 的一个位姿点或依次连接的位姿点序列.

例如,当 A 的姿态固定时,Cspace_A 将是一个平面,而 $CO_A(B)$ 则是一个比 B 更大的凸多边形,即如图 13.10 所示的阴影区.

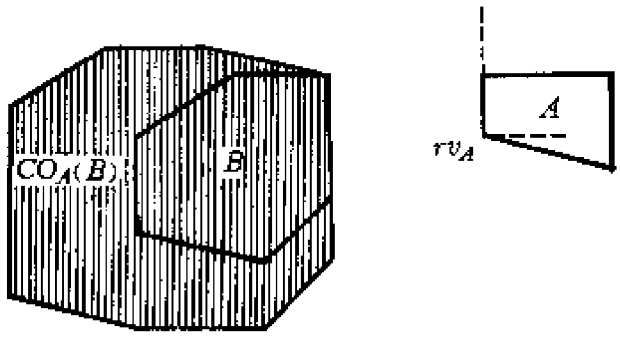


图 13.10 位姿空间障碍域

这时只要 rv_A 不进入该区，A 就不会与 B 碰撞．而且，只要把 A 的初始姿态点 s 与 $CO_A(B_i)$ 的某些顶点直至目标位姿点 g 连成一条折线，就得到 A 在 B_i 中的一条最短安全路径(图 13.11)．

到此，寻路径问题可进一步形式化为一个"可见"图的搜索问题．图的节点即为所有 $CO_A(B_i)$ 的顶点，包括 A 的初始位姿点 s 和目标位姿点 g，图中每一对可以相互"看得见"(即可用直线相连而不与任何障碍域相交）的节点之间用弧连结．搜索从 s 到 g 的最短路径就可得到 A 在 B_i 中的最短安全路径．

上述过程的关键问题是如何求得 $CO_A(B_i)$．对于 A 的姿态固定的情况，位姿空间法给出了一条定理：对于二维点集 A、B，位姿空间障碍域内各点的坐标值(以上标 X、Y 表示)为

$$CO_A^{XY}(B) = B\ominus(A)s = B\oplus(\ominus(A)s))$$

式中，$(A)s$ 表示所有构成与初始位姿点 s 相应的 A 的点集，$\ominus$ 表示求点集的坐标差，或者表示把点集中各点的坐标值反号．

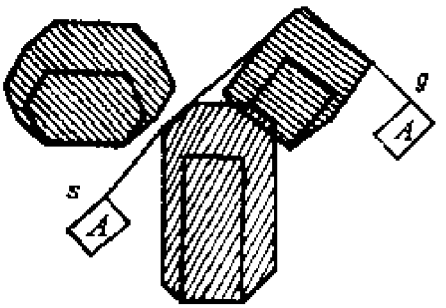


图 13.11　最短安全路径

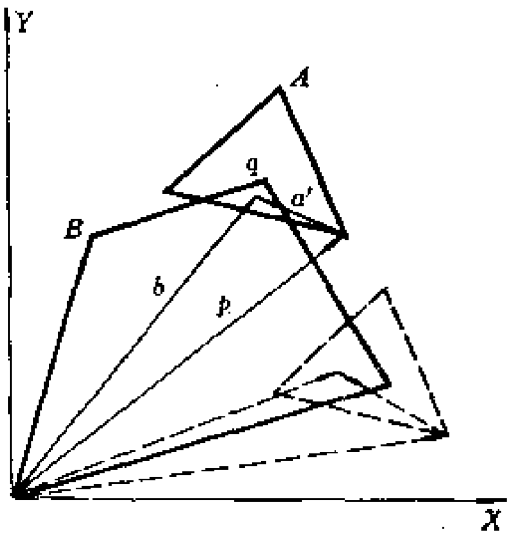


图 13.12　位姿空间障碍域的计算

图 13.12 例示了上述定理的正确性．$CO_A^{XY}(B)$ 实际上是指处于位姿 p 的 A 与 B 相交时参考顶点 rv_A 的所有位姿坐标．当 A 与 B 相交时，设共有一点 q，q 在 $(A)_p$ 中的坐标用向量 a' 表示，

q、rv_A 在 $(A)_r$ 中的坐标用向量 b、p 表示，由图可知，$p = b - a'$，因此，考虑所有的相交情况，就有

$$CO_A^{XY}(B) = B \ominus (A)_r = B \oplus [\ominus (A)_r)]$$

根据上述定理，可以建立一种算法，通过对 $\ominus(A)_r$ 与 B 的边、顶点进行相加计算，就可以确定 $CO_A(B)$ 的边和顶点，运算结果还可以从几何图形上给以直观的说明. 如图 13.13 所示，使 A 以固定的姿态保持与 B 的边接触，并且沿着 B 的边界滑动一周，参考顶点 rv_A 的轨迹形成的凸多边形（粗线）即为位姿空间障碍域 $CO_A(B)$，显然，只要 rv_A 不进入这个凸多边形，A 就不会与 B 相交. 由图可知，凸多边形 $CO_A(B)$ 的边实际上是 A 和 B 的边、顶点按一定规律"混合"的结果.

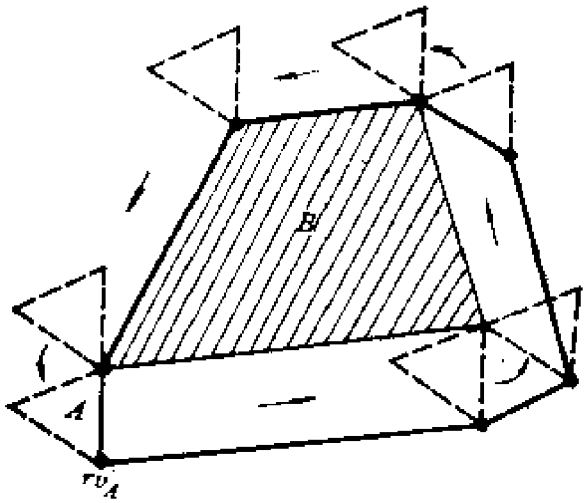


图 13.13　位姿空间障碍域的形成

如果物体 A 在由初始点 s 运动到目标点 g 的过程中姿态有变化，需要将姿态转角 θ 分成许多段，对每一个转角段 θ_i 计算相应的 $CO_A^{XY\theta_i}(B_i)$. 这时，由于所得结果不是一个简单的二维多边形，因而不能再用前面所说的"可见图"来搜索安全路径. 对此，位姿空间法提出一种"子分割"方法，把位姿空间分割成许多矩形单元，从中搜索包括初始点和目标点的自由空间单元，连成安全路径. 整个过程可以推广到 A 为多面体的情况.

二、旋转映射图法

旋转映射图法(由张铃、张钹自 1981 年起开始研究)兼顾运动物体的平移和旋转情况，利用状态空间和旋转映射的概念，把运动物体在障碍物中间的位置及其相应的无碰撞姿态表示为所谓状态空间中的某个集合——旋转映射图 RMG (Rotation Mapping Graph)，从而把安全路径的搜索问题转换为考虑 RMG 的连通性问题，借助于拓扑方法从理论上方便了问题的解决.

在笛卡尔坐标系统 $X_1OX_2OX_3$ (或 X_1OX_2) 中，假设障碍物是由若干空间曲面(或平面曲线)所组成. 点 P 为运动刚体 A 上的一点，PT 为通过 P 点的一个参考轴，那末 A 的运动可分解为 P 点的平移和 PT 轴围绕 P 的旋转. 设 θ 为 PT 在 X_1X_2 平面上的投影与 X_1 轴的夹角 (在平面中，θ 为 PT 与 X_1 轴的夹角)，ϕ 为 PT 与 X_3 轴的夹角. 利用 P 的坐标 x 以及 PT 轴的姿态角 θ、ϕ 就可定义 A 在空间的一个状态 (x, θ, ϕ) (在平面中，则为 (x, θ)). 对于 P 在坐标系统中所有可能位置 x 形成的全体状态 (x, θ, ϕ) (或 (x, θ)) 称为 A 的状态空间.

设 D 为 P 点的坐标 x 的域，$f(x)$ 为相应于 x 的 PT 轴在障碍物中间的无碰撞姿态角的集合，R 为集合 $f(x)$ 的元素的可能取值范围，则称映射 $f: D \to 2^R$ 为 A 的旋转映射，其中 2^R 为 R 的所有可能子集. 显然，对于二维情况，$f(x)$ 即为实轴上的一个子集，而对于三维情况，$f(x)$ 则是平面中的一个子集. A 的旋转映射图 RMG 定义为: $G(f) = \{(x, f(x)) | x \in D\}$，它是 A 状态空间中的一个集合.

例如，物体 A 为一杆件，长 r，PT 轴即为 A 本身，当它在图 13.14(a) 所示的障碍 (斜线区) 环境中运动时，假定 D 限于正 X_1 轴，那末 PT 的 RMG 就可表示为图 13.14(b) 中的斜线区.

由 RMG 的定义可以直接推出旋转映射图法的基本定理：给

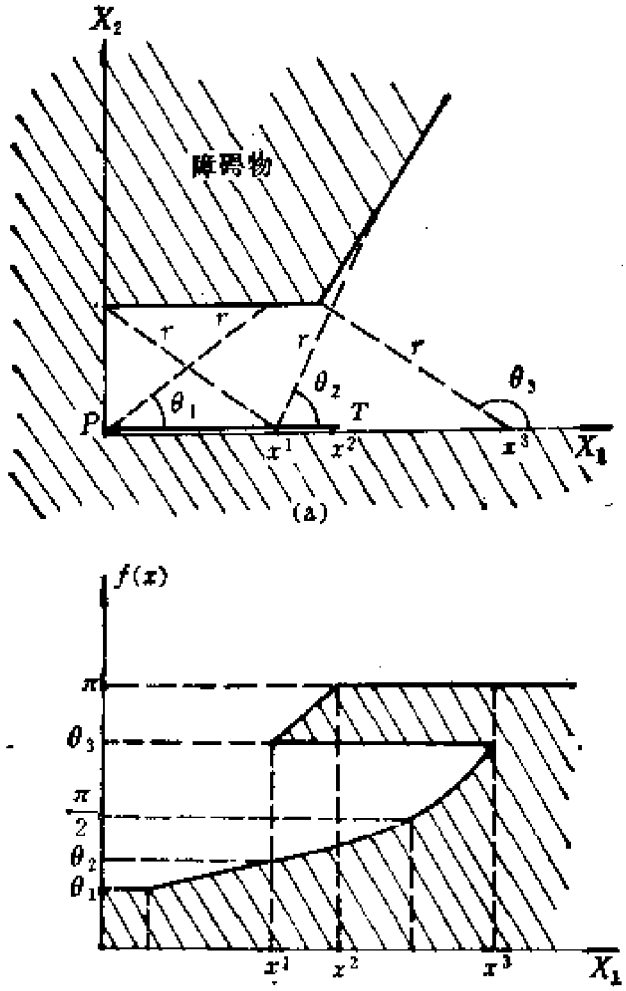


图 13.14　杆件的 RMG

定 A 的初始状态 s 和目标状态 g，当且仅当 s、g 属于 $G(f)$ 的某个连通子集时，则必存在一条从 s 到 g 的无碰撞路径．这样，是否存在安全路径的问题就转化为考虑 $G(f)$ 的连通性问题了．

下面按二维情况说明方法的要点，仍以杆件 PT 为例，并设障碍物 B 为多边形．

首先确定旋转映射．对于任何 $x^0 \in D$，以 x^0 为中心，杆长 r 为半径，逆时针画圆 R，如图 13.15(a) 所示．任取一点 x，经过 x 沿 x^0x 方向作一射线 $t(x)$，令点集 $N(x) = t(x) \cap R$，那末点

集 $N(B)=\cup_{x\in B}N(x)$. 显然，杆件端点 P 位于 x^0 点的无碰姿态角集合 $f(x^0)$ 就可由三段有向弧 (1, 2)、(3,4) 和 (5, 6) 表示 (图 13.15(b))，而且 $(1,2)\cup(3,4)\cup(5,6)=R-N(B)$. 相应于每一段弧的角度范围称作 $f(x^0)$ 的一个姿态分支.

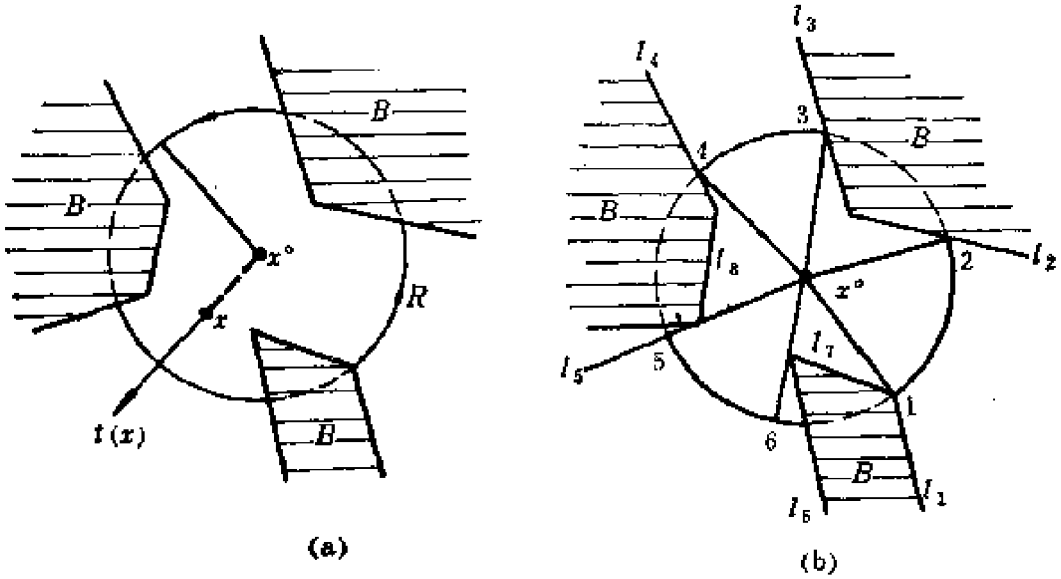


图 13.15　杆件的旋转映射

为了考虑连通性问题，需要把 D 划分为若干个由各障碍物 B 的边 l_i，r-扩展边 l_i'、p_i' 或者阴影区边界 s_i 围成的连接区 D_i，后三种边界的形成原则是：r-扩展边界 l_i'，对于 B 的边 l_i，作一距离为 r 的平行线即为 l_i'，它表明了满足 T 与 l_i 接触而 PT 垂直于 l_i 这一约束的 P 的所有位置(图 13.16(a))；r-扩展边界 p_i'，对于 B 的顶点 p_i，以 p_i 为中心、r 为半径作一圆弧 $\overset{\frown}{AB}$ 即为 p_i'(图 13.16(b)、(c) 和 (d))；阴影区边界 s_i，对于如图 13.17 所示的这种障碍物布局，B_1 的一条边 l_3 位于界边 l_1' 之内，B_1、B_2 之间的区域形成一个"叉口"，这时，可以求出位于 l_2 下面的一条边界 s，当杆件端点 P 进入 s 与 l_2 围成的"阴影区"，位于其中某点 x 处时，由于 B_2 的遮挡，朝着叉口方向的姿态分支 $f(x)$ 就不存在.

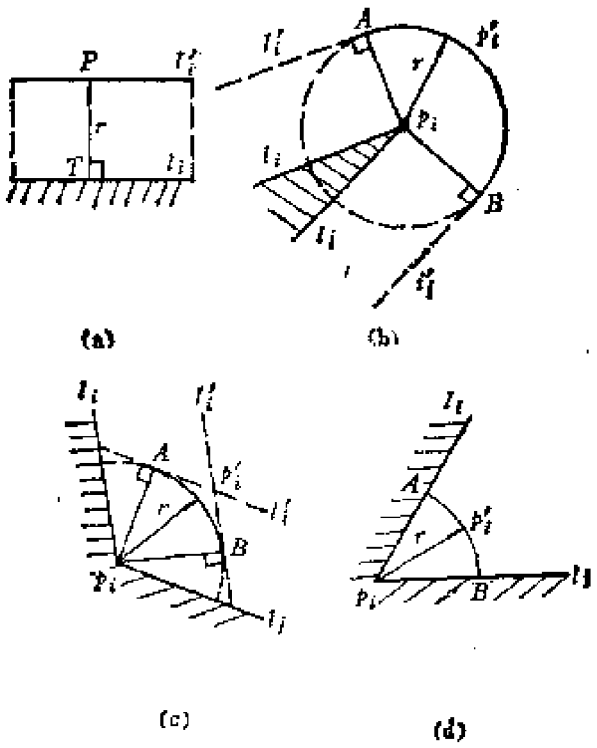


图 13.16 r-扩展边界

根据上述 D_i 的形成原则，可以作出一个具体例子的各种边界，如图 13.18 所示。图中，实线为障碍物的边或 r-扩展边界，虚线为阴影区边界，阿拉伯数字为连接区号。

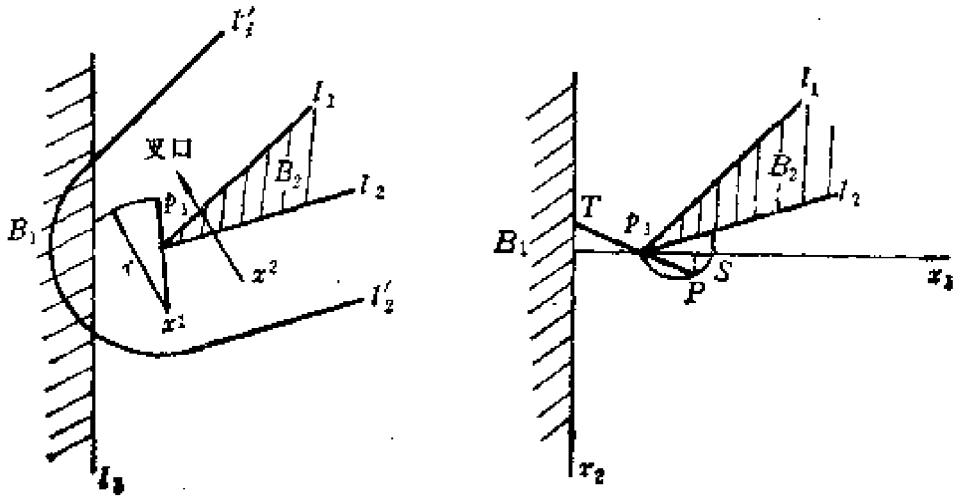


图 13.17 阴影区边界

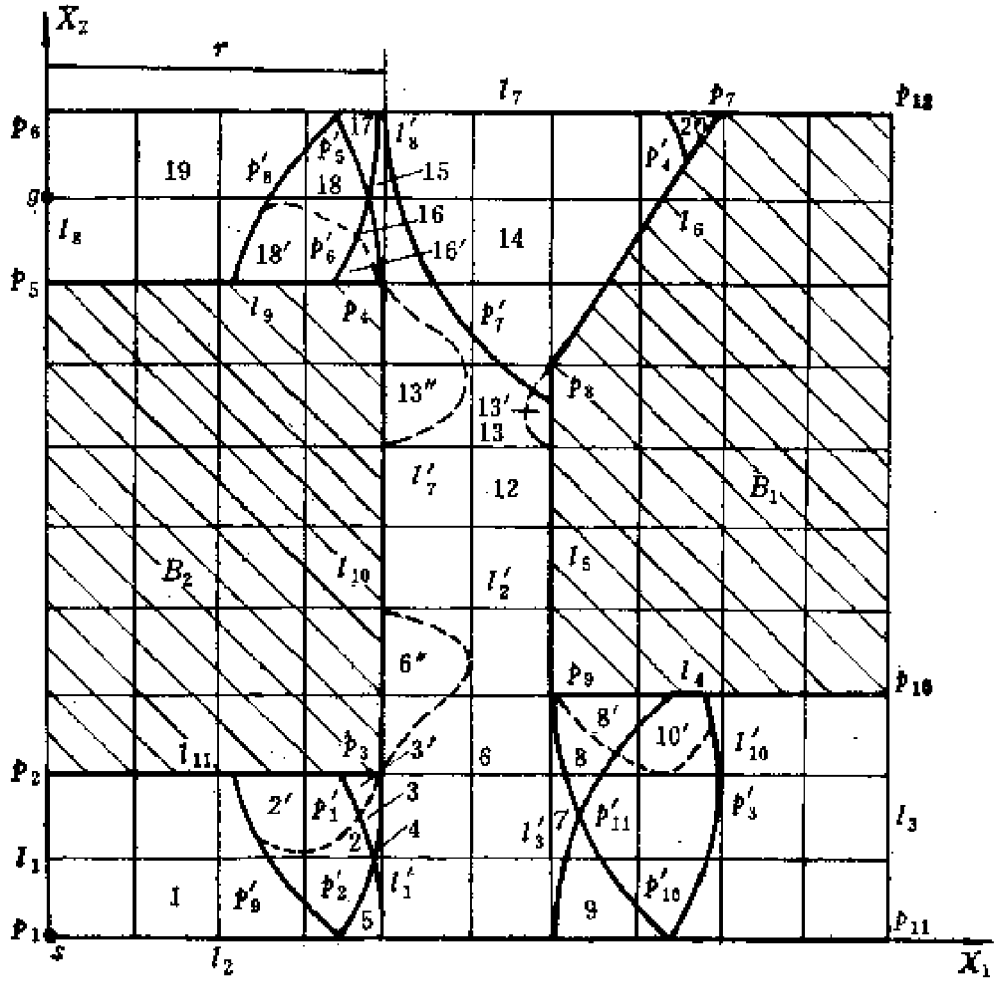


图 13.18　连接区的划分

可以证明，同一个连接区中的任意两点 x_1、x_2 都具有等价的姿态分支构造，即，$f(x_1)$ 与 $f(x_2)$ 中的分支数目相同，而且存在一一对应的关系。换句话说，只要在给定连接区中任选一点，就可确定杆件 PT 在该区中的姿态角范围。

另外，确定每一个连接区中的姿态分支还需服从下列三条规则：

融合(分解)规则.假定连接区 D^1、D^2 被 r-扩展边界 l'_k 分隔，D^1 中的姿态分支 (i, k)（表示位于边 l_i、l_k 之间的形如图 13.15 中所示的弧段，下同）、(k, j) 就被融合，跨越 l_k，变为 D^2 中的姿

态分支 (i, j). 反之，D^1 中的 (i, j) 必须被分解，跨越 l'_k，变为 D^2 中的 (i, k) 和 (k, j). 例如图 13.18 中，6 区的(11,2)与(2,4)要穿过 l'_2，融合为 12 区中的 (11.4). 对于 D^1、D^2 的分隔线是 r-扩展边界 p'_k 的情况，设 p_k 为 l_m、l_n 所成的顶点，那末 D^1 中的 (i, j) 则要被分解，跨越 p'_k，变为 D^2 中的 (i, m) 和 (n, j). 反之则被融合. 例如图 13.18 中，1 区的(2.10)要跨越 p'_4，被分解为 2 区中的(2,4)和(5,10). 显然，由于杆件的旋转角度能连续地"衔接"起来、过渡下去，具有上述这种融合(分解)关系的姿态分支都是连通的.

消失规则. 前面已经说到，朝着"叉口"方向的姿态分支在杆件穿过阴影区边界进入阴影区时就不存在. 例如图 13.18 中，在杆件从 2 区进入 2′ 区时，姿态分支(5, 10)就要被取消.

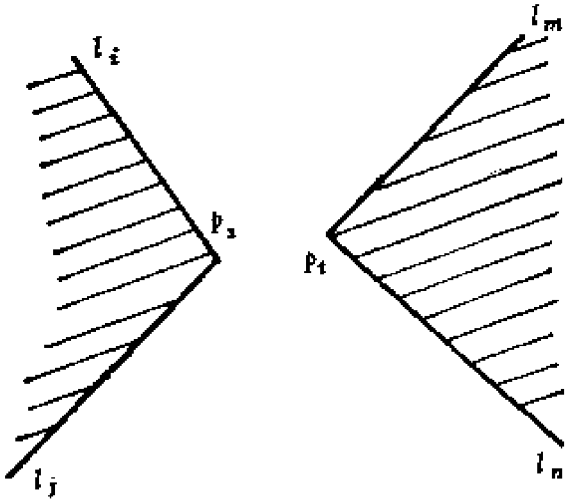


图 13.19 等价规则

等价规则. 如图 13.19 所示，p_s、p_t 分别为 l_i 与 l_j，l_m 与 l_n 的公共点，在同一连接区或相邻连接区中，姿态分支 (i, m) 与 (i, n)、(j, m) 与 (j, n)、(m, i) 与 (m, j)、(n, i) 与 (n, j) 实际上都是同一姿态分支的不同表示，因而是等价的.

按照图 13.15 所示的方法，以及上述三条原则，对于图 13.18

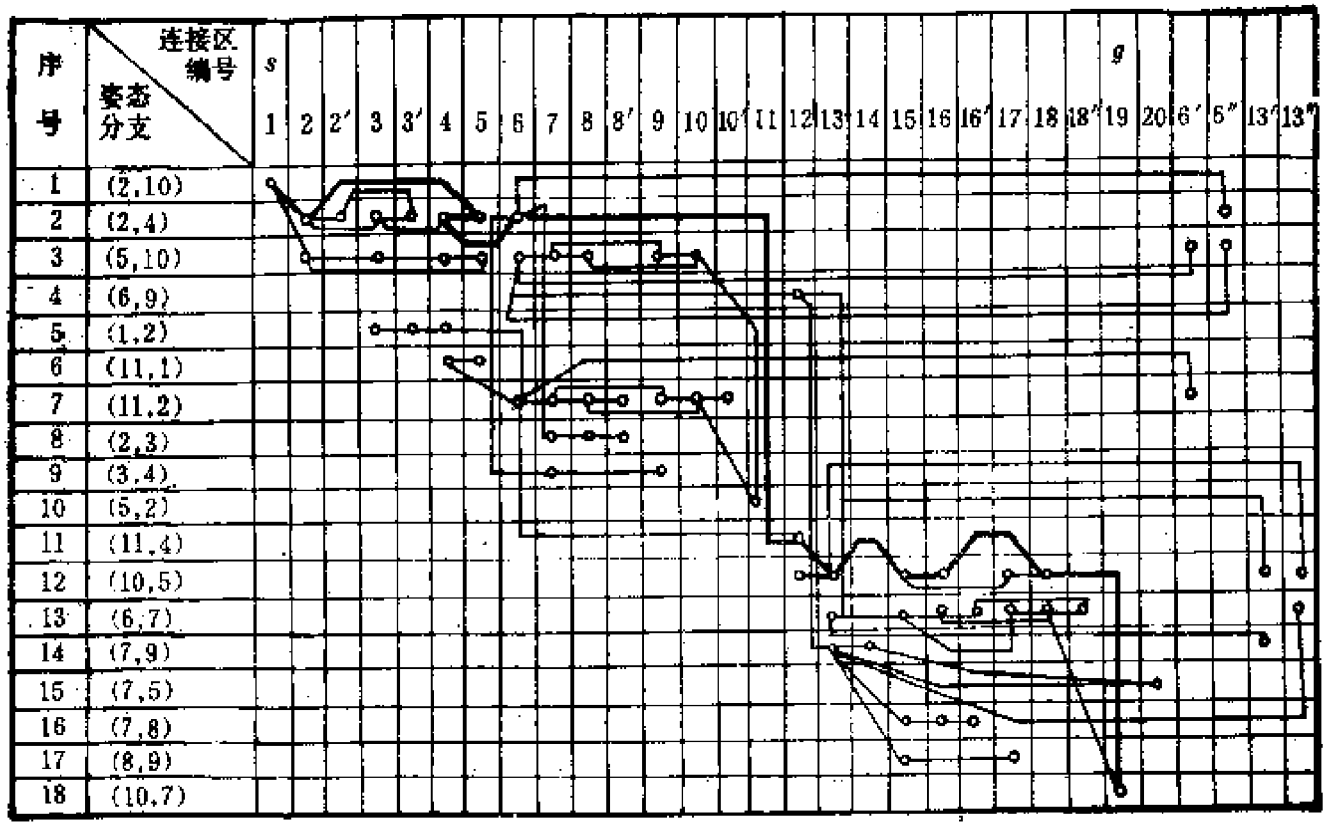


图 13.20　杆件的特征网络

的具体例子，每一个连接区的旋转映射 $f(x)$ 都能得到确定。图 13.20 列出了这些结果。

把相邻连接区中所有等价的姿态分支，或者具有融合(分解)关系的姿态分支都连接起来，就形成一个特征网络，如果从中能找到任意一条由包括初始状态 s 的连接区到包括目标状态 g 的连接区之间的连通路径，那就完成了实际空间中无碰路径的规划问题。

图 13.20 中的粗线表示从 $s(0,0)$ 到 $g(0,9)$ 的一条无碰路径：

[1, 1][2, 2][5, 2][4, 2][6, 2][12, 11][13, 12][15, 12]
[16, 12][18, 12][19, 18]

在 $[I, J]$ 中，I 为连接区序号，即杆件端点 P 的位置；J 为姿态分支序号，即杆件 PT 的姿态角范围。

上述旋转映射图法虽然以二维的杆件问题为例，但基本思路和求解过程可以推广到三维的情况。

把求无碰路问题化为分析 RMG 图的连通结构问题，可以通过一种所谓的"降维法"进行计算，从而避免了分析高维空间的困难。"降维法"的主要思想是，一个多维的旋转映射图 $G(f)$ 是 $X \times Y$ 中的一个子集，可看成是 $f: X \to Y$ 上的一个映射(集值映射)的一个图像，并具有以下性质：

若 $f: X \to Y$ 是半连续的，则当 f 的定义域 $D(f)$ 连通，并且 $\forall x \in D(f)$，$f(x)$ 是 Y 上的连通集，则 $G(f)$ 是 $X \times Y$ 上的连通集。

在障碍是由有限个凸多面体组成时，其 RMG 图所对应的 f 基本上是半连续的，于是研究高维空间中集合 G 的连通性问题，可化成研究低维空间中的 $D(f)$ 与 $f(x)$ $(\forall x \in D(f))$ 集合的连通性问题，从而可大大降低计算的复杂性。比如原 $G(f)$ 是六维图形，就可简化为两个三维问题。

杆件的无碰路问题可以应用于机器人的问题，实际上，一个多

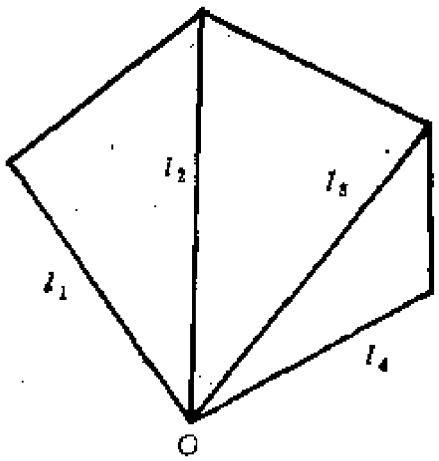


图 13.21　多边形由杆件组成

关节的机械手可以看成是由多个作旋转运动的杆件连接而成，而任一个多边形(或多面体)都可以看成由交于一点的多个杆件组成的(图 13.21)。

附录

AL 语言用户手册

Shahid Mujtaba
Ron Goldman

斯坦福人工智能实验室
1979 年 1 月

附录　AL 语言用户手册*

1. 如何使用本手册

这份文件试图把用户在斯坦福人工智能实验室（SAIL）使用 AL 语言对机械手编程所需要的所有信息汇集于一处。本文件可以满足那些想了解关于 SAIL 中该系统的现行状态和结构的详细情况的研究单位的需要，此外，它也是对于作为语言设计说明的原有 AL 文件的修改结果。在 SAIL，通过监控命令 DO ALXGP. DOC [DOC, HE] 可以得到文件的最新版本。特定章节可以用命令 DO ALXGP[DOC. HE] (n) 得到，其中 n 是你所希望的章节号。随着新的功能的实现和使用，AL 系统正在不断地发展、进化。手册的这一版本包括了一些在第一版(1977 年 11 月)中未包括的内容。

第 II 部分描述了 AL 程序设计系统以及 SAIL 配备的有关硬件和软件。它是针对一般读者的综述。

第 III 部分及似后的部分是提供给 AL 用户的。第 III 部分通过例子逐个说明了简单的 AL 指令的使用。这一部分假设读者已经熟悉了使用 FORTRAN 这类高级语言进行交互式编程的方

* 本附录内容主要译自 Shahid Mujtaba 和 Ron Goldman 所写的 AL Users' Manual 一书。

法;机械手编程或 ALGOL 程序设计的预备知识倒不一定必需.完成这一部分后,用户应该掌握部分 AL 指令,从而能编写简单的程序.

第 IV 部分描述了 AL 语言,并且以简明的方式给出了迄今已经实现的完整的 AL 指令集.对一个有经验的程序员来说,这一部分作为参考手册来使用是足够的.

第 V 部分介绍了如何执行 AL 程序.

第 VI 部分介绍 POINTY 系统,它允许用户交互地生成 AL 程序的坐标树数据结构.

II. SAIL 的 AL 系统

2.1 AL 的设计思想

2.1.1 引言和历史

1973 年，Lou Paul 在 Scheinman 斯坦福模型臂上设计并实现了用于机械手控制的 WAVE 系统，该系统由 Lou Paul 和 Bob Bolles 使用。

在研制 WAVE 所总结的经验基础上，他们与 Jevvy Feldman Ray Finkel 和 Russ Taylor 一起于 1974 年完成了 AL 的最初的说明书。

AL 的编程器和运行系统由 Frinkel 和 Taylor 初步完成，后来由 Ron Goldman 接手工作。

现在 SAIL 使用的机械手是由 Vic Scheinman 设计的，而 Tom Gafford 和 Ted Panofsky 负责计算机与机械手的接口。现在，由 Ken Salisbury 和 Gene Salamin 研制硬件。

Paul 和 Bruce Shimano 的工作是负责研究 Shimano 运动学控制和臂的伺服计算程序。后来，Shimano 完成了力的柔顺控制，而 Tatsuzo Ishida 已经完成了双臂协调控制的理论分析。现在，Salisbury 编制机械手的伺服计算程序。

AL 程序的调试工具 ALAID 是由 Finkel 首先完成的。一个较新的版本即将由 Hamid Nabari 完成。

第一个 AL 句法分析程序是由 Bill Laaser 和 Pitts Jarvis 写成的，后来由 Shahid Mujtaba 接手这项工作。

1975 年，Dave Gvossman 和 Taylor 提出了一个有关的系统 POINTY 的设想，而这个系统是由 Taylor 首先实现的。后来，Maria Gini Pina Gini 和 Mujtaba 又完成了一个新的版本，Enrico Pagello 也做了一些工作。现在，POINTY 系统由 Mujtaba 负责。

根据新的经验和情况，在 Tom Binford 的全面指导下，Grossman, Shimano, Goldman 和 Mujtaba 对 AL 的设计进行了不断的修改与更新。

AL 系统适合于准备时间成为关键因素的批量加工情况。为了减少编程时间，我们采用的方法有：依赖符号数据库和预定义的装配原语以及把希望机械手所做的任务变成程序的方法。我们对在机械手和机器人学习方面具有很少经验的工科大学生使用该系统情况进行了测试，发现学习使用 AL 系统是比较简单的，而且在使用前不必了解系统的全局。

我们假设批量加工的环境是相当结构化的，而且是已知的，并且从一项装配任务到另一项装配任务之间不希望有明显的批量变化。通过模拟，可以预测出每一物体在任意时刻的位置，若要移动物体，可以预先知道物体是否在机械手手中，还可以预先得知所给的信息是否够用，这样就可以使执行过程中与用户的通讯最少。由于生产环境高度的结构化，我们设法在装配前完成尽可能多的计算。

2.1.2 规划系统和运行系统

WAVE (AL 系统的先驱)的经验已经表明，机械手轨迹的计算是必需的一步。因而我们决定，轨迹的计算以及其它所有仅需一次完成的计算可以在编译时一次完成（假设这种方案可以减少执行时的计算工作量，并且每当一组动作执行时可以省去重复性计算）。

这种规划和执行的顺序导致了两个系统的产生：规划系统和时间系统。

规划系统由 AL 编译器构成，其功能是接收用户写好的 AL 程序，对程序进行模拟，给用户指出错误，向运行系统输出指令。编译器完成对程序进行的模拟(称为环境模拟)，用来校验用户所要求 AL 做的是否属于它的能力范围，并就不希望的动作(例如用户因偶然失误而要机械手穿过桌子)向用户发出警告。运行系统接收规划系统的输出，进而完成动作。

这种方法因后来的研究工作而不断变化。计算的耗费已经明显地下降，这使得在分布式计算中进一步使用多处理器成为可能。更好的机械手伺服软件，更快的机械手求解和更复杂的路径计算算法都将降低计算的工作量。因而在运行时间内允许制定更多的决策。我们还认识到某些轨迹最好在运行时间内计算，例如柔性运动、移动传送带，这些都是工作空间属于高度非结构化的情况。

2.1.3 数据和控制结构

AL 的主要输入方式是逐字的输入，使用符号进行程序设计。以机械手处理托板上的部件为例，如果部件之间的距离是已知的(通常是常量)，那么就不必定义所有部件的位置。一旦告诉了托板的棱并已知部件之间的间隔，那么就不必再在辅助的小型机上配备相应的软件进行繁琐的记录再现式的程序设计。符号程序设计简化了 AL 与其它的生成环境模型的手段(如交互图形和计算机辅助设计)的接口工作。它允许建立库程序，这些库程序可以通过填入相应的参数而调用。如果有些复杂的运动可以被参数化或者通过代数方法来描述，那么使用符号程序设计可以便于对这些运动进行说明。例如，当多个关节必须同时校准时，命令机械手沿着任意方向移动一定距离要比用手动方式容易得多。相反，除非只需要关心运动的起止点和终止点，而这两点之间的路径的形

状无关紧要，否则，示教方式就需要记录大量的点（磁带记录方式）。

通过符号文本这种接口可以在人-机之间传递多种复杂的信息。由于需要满足多重逻辑关系，双臂的同时运动；终止条件的描述以及出错条件很可能要通过文本中介作出明确的说明。对于定义目标位置，提出避免碰撞的轨迹，工位初始化以及其它此类目的，非文本形式的输入是一种非常有用的手段。

与其它常用的高级语言相比，AL 使用了更多的数据类型。除标量 SCALAR 外，它还允许使用向量 VECTOR、旋转 ROTATION、坐标系 FRAME、变换 TRANS 及事件 EVENT 数据类型。一个 VECTOR 由一个三元实数组构成，一个 ROT 由一个方向向量和一个表示旋转量的角度组成。一个 FRAME 描述物体的位置和姿态，而一个 TRANS 则描述 FRAME 之间的关系。此外，还可以定义由这些数据类型构成的数组。算术操作不仅可以用于标准的标量操作，而且可以用于像旋转和平移这类操作。

我们希望用比较自然的方式书写程序。像现行的控制语言这种机器语言，如果用来书写较长的结构化程序是非常麻烦的。我们需要一种适合于更系统化的而且又易读的程序设计风格的语言。为此，我们使用类似于 ALGOL 的控制结构，这比起带有跳转的线性的机器语言码是一种改进。ALGOL 的块结构在 AL 中也被采用。

像 SAIL 和 WAVE 这类的语言的研制经验告诉我们，文本宏命令是非常有用的，它们可以减少重复敲键的工作，并且能运用否则无法实现的方式对常量和变量进行符号定义，AL 具有一个通用的文本宏命令系统。

像其它语言一样，AL 也提供了过程，当相似的计算或操作需要在同一程序中多处完成时，过程可以减少编码的数量。

通过分割程序的控制流程，AL 还允许并行处理的控制，这就能使某些操作（例如不同机械手的同时运动）可以被同时执行。然后，各种过程再合并到一起。系统提供同步化原语。

2.1.4 物体的运动

当装配件移动时，AL 具有一种自动记录装配件上一个元件的位置的方法，称之为"附着"，它与 FRAME 概念一起，广泛地应用于物体的描述。坐标系可以彼此"附着"，因此，把一个物体"附着"于机械手之后，用户便可以完全忘掉机械手而只考虑物体在什么地方必须与其它物体相接就可以了。用户不必关心怎样移动机械手，只需说清楚物体的最终姿态和位置即可。AL 将完成机械手该做什么以实现所陈述的目标。用户只需考虑物体的运动而不必考虑机械手的运动。这一点与其它程序设计系统有明显的不同。在其它的程序设计系统中，程序由机械手的一系列运动构成，这些运动在真实世界中与物体的关系只有用户知道；再有就是用户必须为每条运动语句向机械手明确地提供物体间的距离和角度关系。

2.1.5 传感器信息

在运行时间内，AL 允许根据传感器的输入信息对动作进行选择、替换。其中的做法是：检查某些条件是否已经超过了给定的阈值，若已超过则去执行预先定好的动作，这称做条件监视。当遇到错误条件时，有可能把一组动作插入到运动中去，以便恢复正常运动。线性控制做不到这点，它必须终止程序的执行。

2.1.6 程序设计辅助工具

AL 可以在程序编译和执行的不同阶段为用户提供帮助，以便保证尽早发现错误，并简化程序设计工作。

2.1.6.1 AL 句法分析器

AL 的句法分析器接收用户书写的 AL 程序并检查句法是否正确，必要时产生错误信息，如有要求还可使用由 POINTY 产生的 AL 的说明．它使程序能够通过由文本编辑程序产生磁盘文件或电传打字机的方式输入给系统．

AL 句法分析器试图尽早发现错误、纠正错误，因此，若需中止编译过程只浪费较少的时间．此外，由于较早发现错误，有可能在用户源程序的上下文产生错误信息．AL 使用下述两种主要的检查方式来消除一类重要的错误．赋值语句和表达式的量纲检查，保证用户说明的单位必须正确而且与所希望的相符．赋值语句和表达式的项及因子的类型检查保证所要完成的操作必须使用数据类型正确的参数，而且表达式的赋值与变量应是同一数据类型．

AL 允许交互式的纠错过程，它允许用户请求一个标准的纠错过程，或者改变(修补)错误较小的、损坏了的源代码并且从那里继续执行，而不必求助于系统文本编辑程序和重新编译．错误的恢复是局部的，从而可以回溯到最里面的当前语句的开始地方，要进一步回溯，就必须使用文本编辑程序．根据用户的选择将产生一个修改过的源文件的复本．

2.1.6.2 AL 编译器

AL 编译器为用户程序提供了很多语义检查．当有下列情况发生时，AL 将提出警告：试图使机械手移动到无法达到的位置(如穿过桌面)；没有为一个运动分配足够的时间；提出不合适的力的要求；试图移动未与机械臂相连接的物体；程序执行完后机械手未位于在停放位置．

为了帮助找到错误，用户可以要求在程序中某些希望的地方打印出规划值．这些值是 AL 编译过程中环境建模阶段为每个变量保存的．

2.1.6.3 交互式模型的建立

POINTY（详细描述见第 VI 部分）允许用户借助机械手交互地产生 AL 程序的坐标系，还可以检验简单的运动语句.在组成一个较大的 AL 程序之前， POINTY 的交互式特点对测试小的程序段是有用的.

2.1.6.4 调试程序

在程序执行期间，为使用户可以通过修补程序来改正错误，检查和改变变量的值，用户可以使用几个调试程序.

执行过程中调试一个 AL 程序包括检查和修改变量，改变控制流程、触发条件监视程序，修补代码. 设计 ALAID 就是用它来完成这些动作，并且辅助程序员准备正确的机械手代码. ALAID 在 PDP-10 和 PDP-11 之间建立了一条通讯链(参见 2.2 节)，并允许从任意一台机开始调试. 它在 PDP-11 上运行的 AL 程序和 PDP-10 上的高层的策略程序之间建立了一个规正的接口. ALAID 能使两个进程使用 AL 中的同步原语互相发信号. 它还允许 PDP-10 上运行的程序在 PDP-11 上的 AL 机器手程序的存储空间中检查和设置变量.

目前，ALAID 联结两台机器，能检查和设置变量，发信号和等待事件，使运行系统进入 11 DDT，允许用户停止和重新开始 PDP-11 上的 AL 程序的执行，检查和修改程序代码.

11DDT 是 PDP-11 机器语言的符号调试程序，AL 专家用它调试运行系统，用户用它继续或重新启动其程序的执行.

2.2 AL 系统硬件

AL 系统硬件主要由一台用于编译和装入 AL 程序的 PDP KL10 计算机(以后称为 PDP-10)、一台执行 AL 程序的 PDP-11/45 计算机(以后称为 PDP-11)、两台 Sfanford 型 Scheinman 机械手，以及如终端和磁盘等各种外设组成.

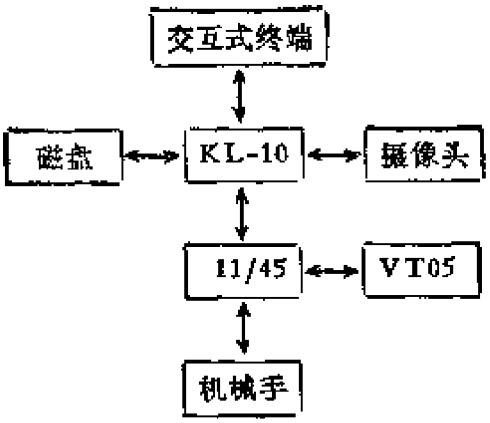


图 2.1 SAIL 的 AL 的硬件配置

各部件之间的相互关系如图 2.1 所示：PDP-11 系统与机械手和 PDP-10 系统相连接。因为 PDP-10 与机械手之间没有直接接口，所以，PDP-10 与机械手之间的所有通讯必须通过 PDP-11 运行系统。

2.3 软件

在 SAIL，当前的 AL 系统的软件结构如图 2.2 所示。每个方块表示能一次调入内存的一组程序，每组程序需要和产生的文件写在程序组连线的旁边。

数据和程序存储在文件中，文件名的形式是 ABCDEF XYZ，其中 ABCDEF 是一至六个字母数字字符，构成名字，XYZ 是零(此时"·"省去)至三个字符，构成扩展名。扩展名用于区别同名的文件族中的文件。

附加信息可由 POINTY 用 AL 语句形式产生并存在一个说明文件中。运动的程序和数据可用文本编辑在磁盘文件 FOO.AL (其中 FOO 为文件名，AL 是扩展名，用来表示 AL 源程序) 上准备和存储，也可由电传打字机直接输入到 AL 编译器。

AL 句法分析器接收用户书写的 AL 程序，并检查其句法上

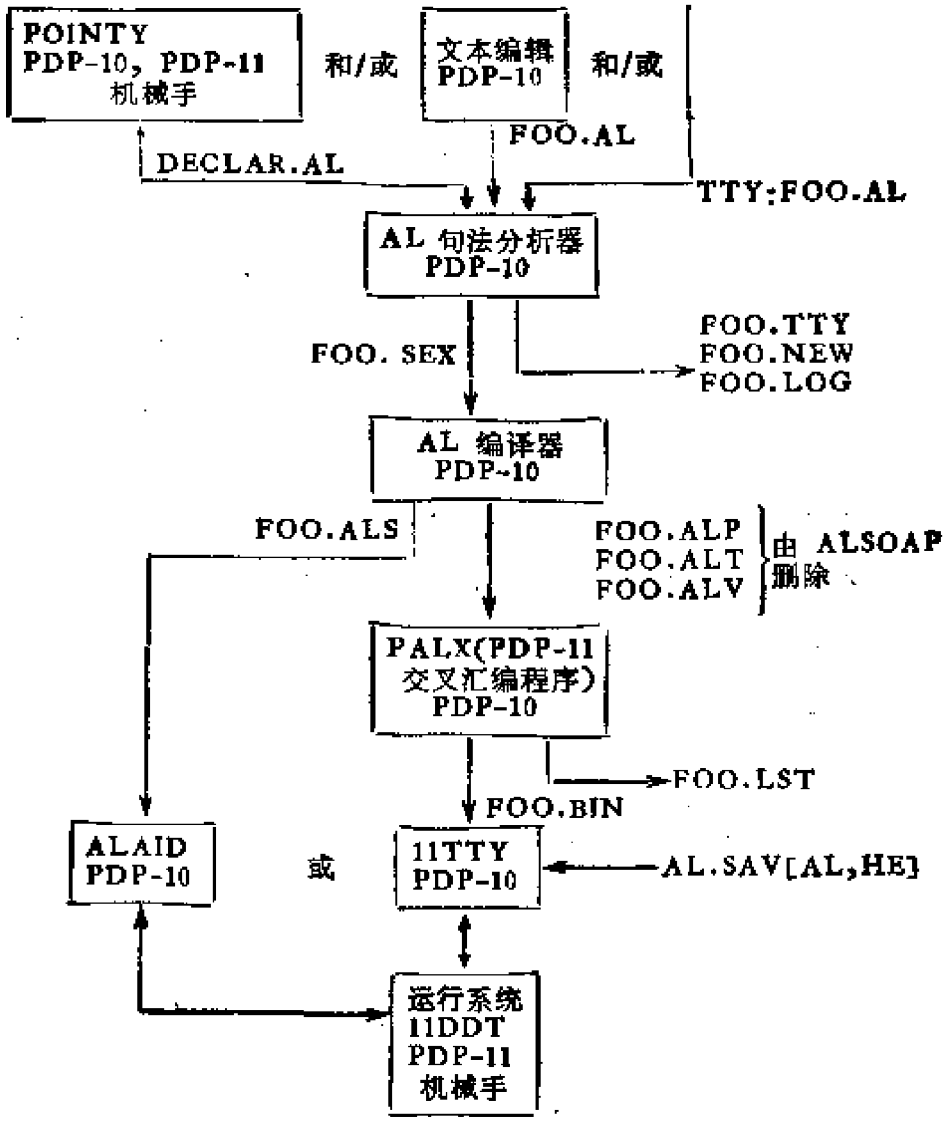


图 2.2　软件结构

是否正确，必要时产生错误信息．它产生一个带有扩展名 .SEX 的中间文件(用 S-表达式)并传送给AL编译器．根据用户的选择，AL句法分析器将产生一个包括所有错误信息的以 .LOG 为扩展名的存入文件和一个以 .NEW 为扩展名的修改过的源文件的复本．对由电传打字机直接输入的程序，将产生一个以 .TTY 为扩展名的程序的磁盘文件的复本．

AL 编译器读入由 AL 句法分析器产生的 S-表达式文件，并

把它变为内部形式．然后对程序进行模拟(称为环境模拟)，接着是轨迹计算和代码生成．最后，生成扩展名分别为 .ALP、.ALV、.ALT、.ALS 的四个文件，对应地包含如下信息：程序码、数值常量，运动轨迹和符号．

前三个文件由 PALX 使用，它是 PDP-11 的交叉汇编程序，把一个带有扩展名 .BIN 的二进制装入模块进行汇编后交给运行系统．

带扩展名 .BIN 的二进制文件由一个称为 11TTY 的程序，与 AL 程序码解释程序和运行系统一起装入．

带有扩展名 .ALP、.ALV、.ALT 的中间文件一般由程序 ALSOAP 删除，这个程序是在 AL 编译后自动运行的．

若希望的话，ALAID 可在 PDP-10 上同时运行，提供一个 PDP-10 和 PDP-11 之间的通讯链．ALAID 使用 .ALS 文件．

2.4 AL 程序设计

为了用 AL 编制一个装配任务程序，首先应该制定一个规划，其中包括部件的粗略安排，完成装配任务的运动序列．

部件和卡具应放置在工作空间中希望的物理位置．物体布置的信息必须输给 AL，可用尺和其它测量设备直接测量，或者借助于机械手和 POINTY，后者是一个使用机械手的交互式数据采集辅助工具（参见第 VI 部分）．这些数据必须并入一个文件中，这个文件含有说明为完成希望的装配而如何移动部件的 AL 语句．

得到程序后，用户借助某些手段（在 SAIL，是通过一个交互式终端）把它输入到计算机系统中去．

像其它的程序一样，程序要进行编译，装入执行和调试．

III. 使用 AL 系统

3.1 基本构造

这一部分的目的是要向读者介绍 AL 语言，并且通过一系列例子，说明 AL 语言在机械手运动的程序设计中的应用．本节将描述 AL 语言的基本构造．其他指令将在以后各节介绍．

先说明一下记号的约定：在程序和例子中，保留字用大写，变量和预定义的常量用小写．

3.1.1 数据类型

能被处理的数据的类型是各种计算机语言的核心．例如，FORTRAN 有 INTEGER 型和 REAL 型数据，其他一些语言能处理字符串．AL 中的数据类型被选作处理特殊问题，即控制机械手，以及处理真实世界中的三维物体(具有距离、位置和方向)过程中产生的问题．

一个变量就是一个能具有不同值的标识符．标识符由字母数字字符串（字母和数字）和横线符“-”组成．例如：pump_base, handle, screw_hole-2 和 P132．注意所有标识符必须以字母开头(例如 3inch_screw 就不行)．大写与小写是等价的，例如 ABC, abc 和 aBc 代表一个变量．

变量可由一个赋值语句给予一个值，赋值语句由一个变量名、左箭头“←”和一个正确类型的表达式组成．当一个赋值语句执行时，先计算右边表达式的值，其结果成为左边变量的值．

像 ALGOL 一样，AL 需要说明每个变量，即在使用前必须

先说明变量的数据类型。AL 还采用 ALGOL 型的块结构，这就意味着所有在一对特定的 BEGIN 和 END 之间说明的变量仅能与同一个 BEGIN-END 对中的码相联系。同样的变量名还可以用在不同的块中而不发生冲突。在以后将对块结构作更详细的说明(见 3.1.2 节)。我们现在来看一看 AL 语言中使用的数据类型。

3.1.1.1 标量 SCALAR AL 中最基本的数据类型是标量 SCALAR，它在内部的表示是一个浮点数。标量可用作无量纲的量，如某操作重复的次数，或者有量纲的量，如物体的长度成两部件之间的角度。标量允许的算术操作是加法、减法、乘法、除法和指数，用常用的算术操作符分别表示为："+"、"—"、"*"、"/"和"↑"。指数运算的优先级高于乘法和除法，与其他代数语言一样，乘除法的优先级高于加法和减法。几个常用的函数也可以使用：平方根函数SQRT；三角函数 SIN，CON，TAN，ACOS，ASIN 用一个参数，ATAN2 用两个参数；自然对数 LOG；指数函数 EXP。

标量常数写成十进制数，可以带小数点和小数部分。如：2，1.，3.14159，—123.45 都是标量常数。

下面通过一个例子来介绍标量变量的说明和使用。在本节的例子中，我们使用助记符来命名变量以弄清每一个个体的类型。注意，AL 语句用分号";"来分隔，花括号"{}"中为注释。

```
SCARLAR s1, s2,  {说明由一种数据类型跟着一个变量名
                  表组成，变量用逗号分隔，最后用分号
                  结束}
  s1 ← 2;
  s2 ← 3.50;  {s1 值为 2.0，s2 值为 3.50}
  s1 ← s2 * (s1 — 3.2);  {现在 s1 = —4.20}
```

经常希望把一个物理单位量纲与一个变量联系起来。AL 提供给标量的量纲有 TIME，DISTANCE，ANGLE 和 FORCE。应该注意的是一般认为 ANGLE 是无量纲的，但为了我们的目的，

这里的定义稍有点灵活性，即对于定义旋转有用的个体允许有量纲。有量纲的变量与规则的标量变量是一样的，只不过它们带一个适当的单位：sec, inches, deg 或 ounces, 这些单位的意义是明显的。AL 还能处理 cm, oz, lbs, gm 和 radians。

除了 AL 要检查用法的一致性外，有量纲的变量与简单的变量在使用时完全一样。每步算术操作，每一次赋值，都要进行量纲检查。加法，减法和赋值需要确切的量纲匹配，如果匹配失败且两个参数之一是简单的(无量纲的)，那么在给用户适当的信息提示之后将强行按正确的类型操作。乘法和除法不需要量纲匹配，它们产生的量纲的结果与经过表达式传递的参数的量纲不同。在这一点上，中间结果可以有未说明的量纲。除非这样的结果用在赋值语句中，否则是不会有问题的。平方根函数可用于任意物理量纲的标量，结果的量纲将是那个参数量纲的平方根。SIN, CON, TAN 函数要用于具有 ANGLE 量纲的标量，并且假设其单位是"度"，结果是无量纲的。反函数 ASIN、ACOS、ATAN2 使用无量纲的参数，结果值具有 ANGLE 量纲，而且单位是"度"。指数和对数函数用于无量纲的参数，运算结果也是无量纲的。由于在句法分析期间，基数自乘的幂的值是不知道的，这样，指数运算就给句法分析程序提出一个问题。如果基数或者指数不是无量纲的，那么将通过给出错误信息来指出这个问题。

下面是使用有量纲标量和函数的例子。

```
SCALAR s1, s2;
DISTANCE SCALAR ds1;
TIME SCALAR tm1, tm2;
FORCE SCALAR fs1;
ANGLE SCALAR theta, ph1;
ds1 ← 1.0 * inch;
tm1 ← 3 * sec;
```

```
fs1 ← 2.2 * ounces;
tm2 ← tm1 + 4.5;   {在给出相应的错误信息之后，常量
                    4.5被转换为秒}
theta ← 90 * deg;
phi ← theta * 4 * deg;{这是错误的：右边量纲为 ANGLE
                    * ANGLE}
s1 ← SIN (30 * deg);
theta ← ACOS (.7);
ds1 ← SQRT (ds1 * 3 * inches);
phi ← ATAN2 (s1, s2);  {即 arctangent (s1/s2)}
s1 ← LOG(33.0);
s2 ← s1↑3;
AL 中有几个预先说明过的标量：
SCALAR PI; {3.14159···}
```

π 也作为常数 3.14159…而被识别。

```
DISTANCE SCALAR bhand, yhand;
        {这些变量指的是"蓝手"和"黄手"的开度}
```

按通常可以接受的方式，VELOCITY（速度）、ANGULAR-VELOCITY（角速度），TORQUE（力矩）定义为基本量纲。

利用量纲语句还可以定义新的量纲，如加速度 acceleration，新量纲的单位可用宏指令(见 4.5.8 节)定义，如 feet，例如：

```
DEFINE feet = ⊂〈12 * inches〉⊃;
DIMENSION acceleration VELOCITY/TIME;
acceleration SCALAR as1;
as1 ← 6.7 * feet/(sec * sec); {=6.7 * 12 * inches/sec/sec}
```

3.1.1.2 向量 VECTOR AL 程序的工作环境是三维的，因此仅有标量是不够的。我们现在介绍另一种数据类型：向量 VECTOR。在如何组成算术表达式和赋值式方面，它和以后介绍

的其他代数数据类型与标量很相似.

我们所描述的环境是一个具有三个相互垂直且交于原点的基轴的欧几里得空间. 这些基准轴的实际排列是与实现相关的，尽管在 SAIL 和本手册的其他部分我们总假设 Z 轴是向上指的.

向量可以表示成具有方向和大小的个体，如位移、速度、加速度. 与标量一样,它们可以是有量纲的. 利用函数 VECTOR，可以由三个标量表达式来构造向量. 所有的标量表达式必须具有相同的量纲,此即为结果向量的量纲.

向量之间允许的操作包括：加法、减法、点积和叉积. 可以用一个标量去乘或除一个向量. 单位方向向量(无量纲的)可用函数 UNIT 得到. 只有对相同量纲的向量才定义加法和减法. 两个向量的点积、叉积和用一标量乘一个向量,其结果的量纲就是两个参数的量纲之积. 在向量符号两边加上两条直线,表示向量的模. 操作符按通常形式定义,例如,我们有标量 s 和两个向量:

v1 ← VECTOR (x1, y1, z1) 和 v2 ← VECTOR(x2, y2, z2)

则我们可有:

```
s*v1 = v1*s = VECTOR (s*x1, s*y1, s*z1)
v1 + v2 = VECTOR (x1 + x2, y1 + y2, z1 + z2)
v1 - v2 = VECTOR (x1 - x2,y1 - y2,z1 - z2)
v1 · v2 = x1*x2 + y1* y2 + z1*z2
```

在 AL 中有几个预先说明过的向量:

```
VECTOR xhat, yhat, zhat, nilvect {其值如下}
xhat ← VECTOR (1, 0, 0);
yhat ← VECTOR (0, 1, 0);
zhat ← VECTOR (0, 0, 1);
nilvect ← VECTOR (0, 0, 0);
```

下面是几个向量应用的例子:

```
VECTOR v;
```

```
DISTANCE VECTOR dv1, dv2, dv3;
SCALAR S;
DISTANCE SCALAR ds1, ds2;
ds1 ← 2 * inches;
dv1 ← VECTOR (4, 2, 6) * inches;
ds2 ← dv1、yhat  {所以 ds2 = 2 * inches}
v ← VECTOR(2, 1, 3);
v ← v_zhat;  {所以 v = VECTOR (2, 1, 2)}
dv2 ← VECTOR (3,0,4) * inches;
ds1 ← |dv2|;  {向量 dv2 的大小赋给 ds1，是一个具有
                适当量纲的标量，所以 ds1 = 5 * inches}
dv3 ← VECTOR (4 * inches, 2 * inches, 6 * inches);
                                {dv3 与 dv1 一样}
v ← VNIT(v);  {所以 v = VECTOR(2/3, 1/3, 2/3)}
```

3.1.1.3 旋转 ROTATION 我们要讨论的下一种数据类型是旋转即ROT，它代表一个轴的方向或绕轴的旋转．旋转能作用于向量并使它们绕原点旋转(其长度不变)．它们也可作用于其他旋转(通过矩阵相乘)．为使一个向量绕基准原点旋转，用旋转乘以该向量(向量在右，旋转在左)，为了构成复合旋转，可把它们乘起来，右边的旋转先起作用． 用函数 AXIS 可以得到旋转的轴，在旋转表达式的两边加上竖线就可以得到旋转的角度．旋转是无量纲的，用户不必给这种数据类型规定量纲，但绕轴旋转的量却有单位 ANGLE．

一个旋转可用函数 ROT 来构造．ROT 函数有两个参数，一个是代表旋转轴的简单向量，另一个是表示旋转量的角度．旋转的方向遵循右手规则，因此，一个绕 X 轴 90 度的旋转使 Y 轴移到 Z 轴．这种表示方法比纯矩阵运算更加易写易懂．下面是说明旋转应用的一个例子：

```
ROT r1, r2, r3, r4;
ANGLE SCALAR alpha, beta, gamma;
VECTOR v;
r1 ← ROT (xhat, 90*deg);
v ← r1*zhat;  {v 使 Z 绕 X 旋转 90°,所以 v = VECTOR
                (0, -1, 0)}
r2 ← ROT (yhat, 45*deg);
r3 ← r2*r1;  {这样,就意味着先绕 X 轴旋转 90°,然后再
              绕原来的 Y 轴旋转 45°.换一种解释是先绕
              Y 轴旋转45°,然后再绕新的 X 轴旋转 90°}
v ← AXIS(r2);  {把 r2 的旋转轴赋值给 v, v = yhat}
alpha ← |r2|;{把 r2 旋转的角度赋值给 alpha, alpha =
              45°}
r1 ← ROT (xhat, alpha);
r2 ← ROT (yhat, beta);
r3 ← ROT (zhat, gamma);
r4 ← r3*r2*r1;
    {r4 是具有两种意义的一个旋转: 绕基准 X 轴旋转
     alpha 度,然后绕基准 Y 轴旋转 beta 度, 最后绕基
     准 Z 轴旋转 gamma 度;或者换一种说法: 绕基准
     Z 轴旋转 gamma 度,然后绕新的 Y 轴旋转 beta 度,
     最后绕经两次旋转后得到的 X 轴 alpha 度.两种解释
     得到同样结果, 可选最方便的一种使用.}
```

有一个称为 nilrot 的预先说明过的旋转，它定义为 ROT (zhat, 0*deg):

3.1.1.4 坐标系 FRAMES 在真实环境中处理物体，我们需要规定它们的位置和姿态. 为此,我们引入一种新的数据类型,固联坐标系(以后简称坐标系). 它包括两部分: 原点的位置 (一个

距离向量)和轴的方向（一个旋转）。一个物体的特征可用与物体固联的坐标系来说明。

AL 中有几个预先说明过的坐标系：station 代表工作空间的基准坐标系。每台机械手都联有一个坐标系，其值(每次运动结束时修改)是机械手的位置。现有两个这样的坐标系：barm 和 yarm，分别对应于蓝机械手和黄机械手。与每台机械手相联的还有一个坐标系，或称"停放位置"，它们是 bpark 和 ypark。

一个坐标系可通过调用函数 FRAME 来构成，该函数有两个参数：一个表示姿态的旋转，另一个表示位置的距离向量。一个坐标系的姿态和位置可由函数 ORIENT 和 POS 得到。为了把一个点(该点用某坐标系中一个距离向量来说明)变换到基准坐标系，可用向量(在右边)乘该坐标系(在左边)；为使坐标系平移一定距离，简单地加上或减去一个距离向量即可。最后，为了在基准坐标系中构造一个与某坐标系 f1 中的向量具有相同方向的向量 xhat，可使用"相对于"操作符 WRT，写为 xhat WRT f1。对任意向量 v 和坐标系 f，下式是等价的（结果与 v 具有相同的量纲)：

$$v \text{ WRT } f \equiv (f * v) - POS(f) \equiv ORIENT\ (f) * v$$

下面是应用坐标系的几个例子。

```
FRAME f1, f2;
f1 ← FRAME (ROT(zhat, 90 * deg), 2 * xhat * inches);
        {坐标系 f1 在X方向上与基准坐标相距 2 英寸，该
        坐标系的X 轴与基准坐标的 Y 轴指向相同}
v1 ← xhat WRT f1;  {值为 VECTOR (0, 1, 0)}
f2 ← f1 + v1 * inches;  {与 f1 几乎一样，只是原点在(2,
                        1,0)}
v2 ← f1 * (zhat * inch);  {值为 VECTOR (2,0,1)}
```

3.1.1.5 变换 TRANS 最后一种代数数据类型是变换

TRANS. 变换用于把某个坐标系和向量从它们所在的一个坐标系统变换到另一个坐标系统. 与 FRAME 一样，TRANS 包括两部分：一个旋转和一个向量. 一个变换的应用首先是绕基准原点旋转其运算对象，然后再把结果平移. TRANS 可以用与 ROT 同样的方式构成，右边的首先作用.

一个 TRANS 由一个以角度为单位的旋转部分和一个具有其他物理单位(通常为长度单位)的平移部分(向量)构成. 当用一个 TRANS 乘时，实际上先用旋转部分相乘然后再加上向量部分. TRANS 相乘的矩阵运算仍得到一个 TRANS. 两个相乘的 TRANS 的向量部分必须具有相同的量纲，而且积的向量部分的量纲与它们一样. 为了简便，我们把 TRANS 的向量部分的量纲就称为 TRANS 的量纲. 当一个 TRANS 作用于一个向量时，两者必须具有相同的量纲，TRANS 的量纲如上定义. 结果向量也具有同样的量纲. 当一个 TRANS 作用于一个坐标系时，它必然是一个距离 TRANS. 在构成复合 TRANS 时，必须统一量纲，结果具有相同的量纲. 除非另有说明，否则我们将认为 TRANS 具有长度的量纲.

我们可以用函数 TRANS 来构造一个变换，这个函数有两个参数：一个表示旋转部分的 ROT，一个表示平移部分的向量. 另一种规定一个变换的惯用方法是用两个坐标系来构成. 当算术运算符"→"作用于两个坐标系时，产生一个变换，它把第一个坐标系的原点移到第二个坐标系的原点，再经过旋转使其轴一致. 当在要求一个变换的上下文中使用一个坐标系时，就把它理解为一个简写的距离变换，即偏离基准坐标系统的变换.

下面是一些使用 TRANS 的例子.

```
TRANS t1, t2, t3, t4;
t1 ← TRANS(ROT (xhat, 60*deg),2*zhat*inches);
v1 ← t1*yhat*inches;
```

```
        {t1 使 xhat 绕X 轴旋转 60°，然后沿 Z 轴平移 2 英
          寸，即有 (0, 0.866, 2.5)}
    t2 ← f1 → f2;  {有 f1 * t2 = f2}
    v2 ← t2 * (xhat * inches);
                    {从 f1 看 v2 是 f2 的X 轴}
    t3 ← t2 * t1;  {t3 表示首先完成 t1 代表的变换，然后再
                    完成 t2 代表的变换}
    f3 ← f1 * f2;  {此式表示 f2 在 f1 坐标系中的位置，等价
                    于 (station → f1) * f2}
    t5 ← INV (t1);  {这表示 t1 的逆变换}
```

零变换等价于 TRANS (nilrot, nilvect)，记为 niltrans。

与 AL 的工作取得进展的同时，坐标系和变换之间原先的区别也在缩小。目前，二者的差别是坐标系可以彼此附着，而且有与之相关的远离点（参见 3.4 节）。一般来说，能使用坐标系的地方也能使用变换，反之亦然。例如要得一个变换的两个组成部分，可以使用算子 ORIEN 和 DOS。这两种数据类型是否将被合并还有待于进一步研究。有一种观点认为，坐标系是空间中与实际物体或位置相固联的标记，而变换则是这些物体之间的相互关系。在这种情况下，坐标系不必有与之相关的量纲，但在它们和其他坐标系之间将存在有某种关系。

3.1.2 块结构——"程序是什么"

一个 AL 程序由一系列语句组成，这些语句使得机械手成功地完成某项希望的任务。最简单的 AL 程序由一个简单语句组成，任何合理的程序将由很多语句 S1，S2，S3···组成，它们用分号分隔，由保留字 BEGIN 和 END 包围，形如

```
BEGIN
  S1;
```

```
    S2;
    :
    S3;
  END
```

的这种复合结构就是一个块语句。块内的语句（S1，S2，···）本身也可以是其他块语句，空格在程序中不起作用而只是为了使程序更加可读。

为了记录位于其他块内的块，可以用紧跟在 BEGIN 和对应的 END 后面的双引号内的字符串给这些块命名。跟在对应的 BEGIN 和 END 之后的字符串应该是相同的，在一般的 END 之后是不用字符串的，否则将产生错误信息。下面是命名块的一个例子：

```
BEGIN"MAIN"
  S1;
  S2;
  BEGIN"INNER"
    S3a;
    S3b;
  END"INNER";
  S4
END
```

像 ALGOL 一样，AL 在使用一个标识符之前要先说明。标识符仅仅在说明该标识符的块内生效，在此块之外，引用这些标识符将得到错误信息。除非在给定块内部的块有子说明，否则在一个给定块内如果同一标识符有多于一次的说明，那么将产生错误信息。

考虑下面例子：

```
BEGIN "BLK_1"
  SCALAR i, k, m;
```

```
i ← 1;
BEGIN "BLK_2"
  SCALAR i;  {注意新变量 i 不同于 BLK_1 中说明
              的 i}
  i ← 2;
  m ← i;  {所以 m = 2, i 代表第二次说明的 i}
END "BLK_2"
k ← i;  {因为已退出 BLK_2, 所以 k = 1}
END "BLK_1"
```

在内部块 "BLK_2", 变量 i 是不同于 "BLK_1" 中的 i 的新变量. 如果去掉在块 "BLK_2" 中的语句 SCALAR i, 那么在执行结束时 k 和 i 的值都是 2.

3.1.3 一个简单的程序

如前所述,一个赋值语句由三部分组成: 一个变量名,一个左向箭头"←",和一个类型正确的表达式. 当执行一个赋值语句时,先计算右边表达式的值, 其结果就是左边变量的新值. 为保证表达式和变量具有同样的数据类型, 必须小心谨慎. 在编译期间, AL 要检查箭头两边的类型和量纲的一致性,若发现不一致则要产生错误信息.

在执行期间, 打印语句将打印出变量和字符串的值. 其构成是, 保留字 PRINT, 跟着是一个开括号"(",一个参数表(逗号为其分隔符), 和一个闭括号")". 参数可以是变量名或预定义的常量名,也可以是由双引号包围的字符组成的字符串常量.

下面是一个简单的 AL 程序,它计算并打印出当前机械手的位置及它们之间的距离:

```
BEGIN
  DISTANCE SCALAR s1;
```

```
DISTANCE VECTOR v1;
PRINT ("THE BLUE ARM IS AT", barm);
PRINT ("THE YELLOWARM IS AT", yarm);
v1 ← POS (barm)-POS (yarm);
        {v1 是两机械手中心的距离向量}
s1 ← |v1|;  {s1 是两手中心的绝对距离}
PRINT("THE DISTANCE BETWEEN BLUE AND
        YELLOW FINGERS IS", s1, "INCHES");
END
```

在以后各节将讨论可能在块内出现其他语句.

3.2 简单的 MOVE 语句

最简单的运动程序是把一台机械手移到一个已知的位置. 当两台机械手 barm 或 yarm 不使用时, 它们手指向下静止地处于平衡位置, 这样当发生供电故障时机械手不会倾倒. 机械手的停放位置(手指的指向已描述过)记为 bpark 和 ypark.

在这份文件中,当我们提到一台机械手时,除非与上下文有明显矛盾,否则我们都指的是蓝机械手.

假设机械手在任意位置, 我们要在计算机的控制下把它移到停放位置,所用的语句是

```
MOVE barm TO bpark;
```

在编译期间, AL 将计算出从当前位置到停放位置的一条轨迹, 以便在电机所能提供的最大加速度和力矩的条件下尽快地完成运动. 一条轨迹是从初始位置到最终位置各个关节位置关于时间的函数. 可是, 在编译期间, AL 不能读取机械手的位置, 因此,必须为机械手提供一个规划位置 (用户可以规定). 除非另外告知,否则 AL 将假定程序开始时机械手位于停放位置. 在执行

时，如果实际位置与假定的开始位置不同，那么运行系统将试图修改运动轨迹，以便在原先规划的时间长度内完成运动．这样，如果某个关节要比原来规划的走更长距离，那么为了在同样的时间内完成运动，运动就要比规划的更快．

在上面的语句中，如果 AL 假定 barm 已经位于停放位置，那末它会明智地判定这时不需任何运动，而且因此可计算出一条运动时间为零的轨迹．如果在执行时出现机械手最初不在停放位置这种情况，那末修改了的轨迹就要使机械手运动到停放位置，而且运行时间为零．这将导致需要很大的加速度和超过限度的电机力矩．为了降低运动速度以确保成功，我们应该使用一个 DURATION 子句来修改规划时间为零的轨迹，如下所示：

```
MOVE barm TO bpark WITH DURATION = 4 * seconds;
```

这条语句告诉计算机我们要在 4s 间隔内把机械手移动到停放位置．注意，bpark 后面没有分号，分号是在 DURATION 子句之后也就是整个运动语句的末尾．

规定不同方式的运动也是可以的．符号“⊗”用来表示机械手当前的位置，下面这条语句将使机械手向下移动 2in：

```
MOVE barm TO ⊗ − 2 * zhat * inches;
```

3.2.1 barm 和 bpark 的更多应用

我们来考虑一下 barm 和 bpark．bpark 完整地说明了机械手停放的位置．它用笛卡尔坐标给出了机械手的中心位置，此外，它还指出机械手是向下指向的．由于有六个关节，仅仅给定手的笛卡尔坐标是不够的，这是因为手指尖中心在同样位置但手所指的不同方向可能有无限多个．

barm 是一个坐标系的名字，其原点位于机械手的手指中间，其 Z 轴与手指指向相同，Y 轴通过手指的中心，X 轴根据这两个轴按右手定则定义．barm 的值取决于手的位置和姿态．barm 由一

个向量和一个旋转构成，向量决定在基准坐标系中手中心的位置，旋转决定在基准坐标系中机械手坐标系是如何旋转的。基准坐标系 station 是一个作为参考的坐标系统，其向量部分设在(0,0,0)。我们的基准坐标系 Z 轴向上，Y 轴水平方向且与桌子的短边平行并指向窗户（就是指向从黄机械手的基柱到蓝机械手的基柱），X 轴方向水平且与桌子的长边平行并指向远处的墙。

在停放位置，机械手向下指，手中心的坐标为 (43.53, 56.86, 9.96)*inches。坐标系绕 Y 轴旋转 180°，这样 bpark 的值如下：

```
FRAME (ROT(YHAT,180*degrees), VECTOR(43.53,
        56.86, 9.96)*inches)
```

指令 MOVE barm TO bpark 的作用是把名叫 barm 的坐标系移到由 bpark 描述的新的位置和姿态。

3.3 手指的使用：OPEN, CLOSE 和 CENTER

我们的机械手末端效应器（手）由两个手指组成，当使用指令 OPEN 和 CLOSE 时手指可以根据指令张开或合拢，此时要给出手指之间的开度。下面是这种指令的一个例子：

```
OPEN BHAND TO 2.5*inches
```

这种指令的一般形式是

```
OPEN 〈手〉 TO 〈标量表达式〉
CLOSE 〈手〉 TO 〈标量表达式〉
```

其中〈手〉是保留字 bhand 和 yhand 之一，〈标量表达式〉具有长度量纲，也就是说其表达式的运算结果应为 in、cm 或其他度量长度的单位。

指令 OPEN 和 CLOSE 以相同的速度同时移动两个手指。OPEN 命令把手张开到所要求的开度，CLOSE 指令不断合拢手

指，直到达到接触传感器触发为止.若手指的最大开度比需要的开度小则发一错误信号(CLOSE 指令将在近期内实现).若在手指中间有一个很重的物体，那末手指或电机有可能损坏.CENTER命令可以防止这些不希望的结果的发生，它使两个手指慢慢地合拢，直到其中一个接触传感器触发以使系统知道已经碰到了物体，这时，整个机械手将移动以使手指保持在接触物体的位置. 手指和臂不停地移动，直到两个接触传感器都触发为止. 这时，可以读取机械手新的位置以决定物体的位置. 应该注意的是，CENTER 命令并不是使物体位于手指的中间，而是保证机械手在不移动物体的前提下抓取物体. 当要抓取物体的位置精确地知道或物体要移到某个精确的地方时才使用 OPEN 和 CLOSE 命令. CENTER 把机械手作为其参数，如下所示:

CENTER〈机械手〉

上述几条语句的应用将在下面的例子中得到说明，这个例子是抓取一个 2in 的立方体，沿 X 方向移动 10in，然后把它放下.

```
BEGIN
  FRAME cube, new_place;
  cube ← FRAME (ROT (XHAT, 180 * deg, VECTOR
                                    (20, 30, 1) * inch);
  {定义立方体中心的位置}
  new_place ← cube + 10 * xhat * inches;
  MOVE barm TO bpark WITH DURATION = 4 * seconds;
  OPEN bhand TO 3 * inches;
  {保证使 barm 和 bhand 到已知的位置}
  MOVE barm TO cube;
  {使蓝机械手到给定的立方体的位置}
  CENTER barm;
  {抓取立方体，立方体不动}
```

```
MOVE barm TO new_place;
{把立方体放到我们要放之处}
OPEN bhand TO 3.0 inches;
{打开手,放下块}
```

预说明的宏指令 DIRECTLY 与下面两个子句达到相同的目的：

WITH APPROACH = NILDEPROACH

WITH DEPARTURE = NILDEPROACH

子句 APPROACH 和 DEPARTURE 允许用户至多给出一个三段运动：从当前位置到离开点，从离开点到接近点，从接近点到目标。这些中间点通常是就当前位置坐标系和目标坐标系而言的。

有时需要使物体移动经过空间中附加的点，即有多于如上所述的三段运动。例如，移动着的机械手必须躲避其路径上的物体，或者机械手必须通过一个开口。在这种情况下，可以用 VIA 子句来给出机械手必须通过的坐标系。VIA 子句通常应与规划时可决定其值的点有关。仅能在运行系统决定的点会给轨迹计算带来问题。

在下面这个例子中，机械手拾起地上的一块砖，把它放到烤箱的底部，烤箱的底部与地面在同一水平面上，但机械手必须通过高于地面的烤箱门。

```
BEGIN "Put brick into oven"
  FRAME brick, oven oven_door;
  brick ← FRAME (ROT (yhat, 90 * degrees),
          VECTOR (10, 30, 3) * inches);
          {定义砖的初始位置}
  oven ← FRAME (ROT (yhat, 90 * degrees),
          VECTOR(10, 40, 3) * inches);
          {定义砖的最终位置}
  oven_door ← FRAME (ROT (yhat, 90 * degrees),
          VECTOR (15, 40, 4) * inches);
          {定义烤箱门的位置}
```

```
MOVE  barm TO bpark WITH DURATION =
        4 * seconds;
OPEN  bhand TO 3 * inches;
      {保证臂和手在已知位置}
MOVE  barm TO brick WITH APPROACH =
      3 * zhat * inches;
      {去取砖,手在水平位置,注意,砖的Z轴与基准
      X轴平行}
CLOSE  bhand TO 1.7 * inches;
      {抓取砖}
MOVE  barm TO oven VIA oven_door
    WITH DEPARTURE = -3 * xhat * inches;
      {垂直提起后把砖通过烤箱门移进烤箱}
OPEN  bhand TO 3.0 * inches;
      {放下砖}
MOVE  barm TO bpark VIA oven_door;
      {回到停放位置}
END
```

3.5 物体模型化——"附着"和间接运动

由于装配任务中经常要把一个附加零件安装到另一个主装配件体上,因此,当主装配件移动时, AL 有一种自动的方法来跟踪附加零件的位置。这个方法称之为"附着"。例如,可能有一个 pump 的坐标系和另一个 pump_base 坐标系,在装配的某一阶段, pump 被安装在 pump_base 上,这时应执行语句:

```
AFFIX pump TO pump_base
```

这条语句告诉 AL, pump_base 的运动要影响 pump 的位置。

请注意，实际上 AFFIX 语句没有调用任何子程序来生成完成安装操作的该语句，只是告诉 AL，在程序执行的这个阶段，pump 要被认为附着在 pump_base 之上。

物体的坐标系与机械手的坐标系联在一起这种特殊情况非常重要。例如，一旦 pump "附着"到 barm，那么用户就可以忘掉机械手，而把注意力集中到 pump 必须移动到何处和怎样移动，AL 将去关心怎样移动机械手才能得到希望的结果。这是一种用户不必为机械手规定运动的间接移动。

当坐标系彼此"附着"时，用户必须给出它们之间的相对变换关系，以及"附着"是否是"刚性的"或"非刚性的"。相对变换关系可在"附着"语句中给出，或者如果两个坐标系的位置已经定义，那么只说明它们要被"附着"就会自动地计算出必要的变换。

"附着"语句的形式如下：

```
part ←〈坐标系表达式〉;
fixture ←〈坐标系表达式〉;
AFFIX part TO fixture NONRIGIDLY;
```

或者换种方式：

```
AFFIX pump TO pump_base AT 〈变换表达式〉
                          RIGIDLY;
```

RIGIDLY 意味着"附着"是对称的，因此，一个坐标系的值的变化也意味着另一个坐标系的值的变化。一个 RIGID "附着"语句一般用在物理上是"刚性地"连接在一起的，例如，pump 要固定在 pump_base 上或一台机械手抓着一个物体。在上面的例子中，pump 的移动将影响 pump_base，而 pump_base 的运动也将影响 pump。

当一个物体是"放"在另一物体之上时，要使用NONRIGID"附着"。例如：工件放在支架上，工件随支架而动，但如果只移动工件，支架仍在原地不动。

一个坐标系可以"附着"到更多的坐标系上，而且"附着"可以链结在一起。"附着"关系可以用下面的 UNFIX 语句解除：

UNFIX pump FROM barm;

所有固定在 pump 上的坐标系(如 pump_base) 仍旧固定在 pump 上，而且将不再受 barm 及其运动的影响。

下面是把一个积木块摞在另一积木块上的例子，通过"附着"语句的使用与不使用来说明该语句在程序设计中的用法和方便之处。

```
BEGIN "block stacking without affixment"
  FRAME blk1, blk1_grasp, blk1_top,
    blk2, blk2_grasp, finplace;
  DISTANCE SCALAR graspheight, blk1length,
    blk2length, blk1width, blk2width,
    blk1height;
  ROT  stand;
  stand ← ROT (XHAT, 180 * degress);
  blk1width ← 1.5 * inches;
  blk2width ← 1.5 * inches;
  blk1length ← 2.4 * inches;
  blk2length ← 2.4 * inches;
  blk1height ← 2 * inches;
  graspheight ← 0.75 * inches;
  {定义积木块的尺寸}
  blk1 ← FRAME (nilrot, VECTOR (10, 30, 0)
          * inches);
  blk2 ← FRAME (nilrot, VECTOR (6, 30, 0)
          * inches);
  {定义积木块的底角}
```

```
finplace ← FRAME (nilrot, VECTOR (8, 40, 9)
          * inches);
{定义第一个积木块底面的最终位置}
blk1_grasp ← FRAME(stand, VECTOR(blk1 length/2,
          blk1 width/2, graspheight));
{定义第一个积木块抓取位置}
blk1_top ← FRAME (wilrot, VECTOR (0, 0, blk1
          height));
{定义第一个积木块顶面位置}
blk2_grasp ← FRAME (stand, VECTOR (blk2length/2,
          blk2width/2, graspheight));
{定义第二个积木块抓取位置}
MOVE barm TO bpark WITH DURATION = 3 * se
          conds;
OPEN bhand TO 3.6 * inches;
MOVE barm TO blk1 * blk1_grasp WITH APPROACH =
          3 * zhat * inches;
{移动机械手到第一块积木抓取位置}
CENTER barm;
{抓第一个积木块}
MOVE barm TO finplace * blp1_grasp
  WITH APPROACH = 3 * zhat * inches;
{移动机械手,结果 blk1 位于最终位置}
OPEN bhand TO 3.6 * inches;
{放下第一个积木块}
MOVE barm TO blk2 * blk2_grasp
  WITH APPROACH = 3 * zhat * inches;
{机械手移到第二个积木块的抓取位置}
```

```
    CENTER barm;
    {抓第二个积木块}
    MOVE barm TO finplace * blk1_top * blk2_grasp
          WITH APPROACH = 3 * zhat * inches;
    {移动机械手把第二个积木块放到第一个积木块之上}
    OPEN bhand TO 3.6 * inches;
    {放下第二个积木块}
    MOVE barm TO bpark; DRINT ("all done");
  END "block stacking without affixment";
```

应该注意的是，对每一步运动，其目标位置都是由一个局部坐标系和其中的一个点（例如 blk1 * blk1_grasp）组成的一个表达式。写同样程序的另一种方法如下所示，这里，AL 自动地安排所要用到的坐标系。说明语句的数量还是一样多，但现在的运动语句却更加清楚了。需要注意的是，因为每步运动的目标位置已不再是表达式，AL 将自动地使用标准接近点。

```
  BEGIN "black stacking using affixment"
    FRAME blk1, blk1_grasp, blk1_top blk2,
                blk2_grasp, finplace;
    DISTANCE SCALAR graspheight, blk1 length,
                blk2length, blk1width,
                blk2width, blk1height;
    ROT stand;
    stand ← ROT (XHAT, 180 * degrees);
    blk1width ← 1.5 * inches; blk2width ← 1.5 * inches;
    blk1length ← 2.4 * inches; blk2length ← 2.4 * inches;
    blk1height ← 2 * inches; graspheight ← 0.75 * inches;
    blk1 ← FRAME(nilrot, VECTOR(10, 30, 0) * inches);
    blk2 ← FRAME(nilrot, VECTOR(6, 30, 0) * inches);
```

```
finplace ← FRAME (nilrot, VECTOR (8,40,0)* inches);
AFFIX blk1_grasp TO blk1 at
  TRANS (stand, VECTOR (blk1length/2,
    blk1width/2, graspheight)) RIGIDLY;
AFFIX blk1_top TO blk1 at
  TRANS (nilrot, VECTOR (0, 0, blk1height))
                                RIGIDLY;
{相对于底来定义第一个积木块的顶和抓取位置}
AFFIX blk2_grasp TO blk2 at
TRANS (stand, VECTOR (blk2length/2,
    blk2width/2, graspheight)) RIGIDLY;
{相对于底定义第二个积木块的抓取位置}
MOVE barm TO bpark WITH DURATION = 3*
                                seconds;
OPEN bhand TO 3.6* inches;
{校准机械手的位置}
MOVE barm TO blk1_grasp;
{机械手移到第一个积木块的抓取位置}
CENTER barm;
{抓取第一个积木块}
AFFIX blk1 to barm RIGIDLY;
{第一个积木块及其各部分都"附着"到机械手之上}
MOVE blk1 TO finplace;
{注意,移动的是第一个积木块而不是 barm}
OPEN bhand TO 3.6* inches;
{实际放下积木块}
UNFIX blk1 from barm;
{在环境模型中 blk1 从机械手上"脱离"出来}
```

```
    MOVE barm TO blk2_grasp;
    CENTER barm;
    AFFIX blk2 to barm RIGIDLY;
    MOVE blk2 TO blk1_top;
    {把第二个积木块的底面移到第一个积木块的顶面上}
    OPEN bhand TO 3.6*inches;
    UNFIX blk2 from barm;
    MOVE barm TO bpark; PRINT ("all done");
END "block stacking using affixment";
```

3.6 力觉——简单条件监控

当我们要使用传感器信息的阈值来完成一定的动作时，我们使用条件监控子句。其句法如下：

ON〈条件〉 DO〈动作〉

一个简单的例子是旋转机械手的腕关节(假定是垂直的)，当遇到50oz·in 的力矩时便停止(这或许表示我们以希望的力矩拧紧了某个物体)这种语句的一个例子可以是

```
MOVE barm TO barm * FRAME (ROT (zhat, 90 *
        degrees), nilvect * inches) ON TDRQUE
        (zhat) ≥ 50 * ounces * inches DO STOP barm;
```

这条语句的作用是显然的，遇到力之后，STOP 命令机械手立即停止运动。注意要被检查的力矩方向规定为 zhat，阈值为50oz·in。

假设我们要求一个物体的高度，已知物体在给定的位置，而其高度位于 2 in 和 12in 之间。

```
BEGIN
  FRAME object;
```

```
DISTANCE SCALAR height;
MOVE barm TO bpark WITH DURATION =
                                    3 * seconds,
CLOSE bhand TO 0 * inches;
        {合拢手指}
MOVE barm TO object + 14 * zhat * inches;
        {机械手垂直位于物体之上}
MOVE barm TO ⊗_13 * zhat * inches
        {此处⊗表示 barm 的当前位置}
        WITH DURATION = 10 * seconds
    ON FORCE (ZHAT) ≥ 10 * ounce DO stop;
        {试着使机械手慢慢地向下移动 13in，当遇到力
          时便停止，即已经接触上了}
height ← POS (barm) · zhat_0.3 * inches;
        {取机械手当前位置的Z方向分量并减去手指中心
          和边沿之间的距离就得到了物体的实际高度}
PRINT ("HEIGHT OF OBJECT IS", height,
        "INCHES");
END
```

3.7 施加力和柔顺性

除了检测力之外，AL 允许应用指定的力。由于力既有方向又有大小，因此所施加的力必须给出这两项，或者按合成的大小和方向给出，或者按沿给定坐标系主轴的正交分量给出。使用一个大小为零的力意味着机械手将会是柔顺的，即可在任何外力的作用下沿其方向移动。在下面的例子中，机械手对力在X和Y方向上是柔顺的（即在这两个方向的任何外力的作用都会使机械手移动），

同时，在Z方向施加一个10oz的向下的力。FORCE_FRAME 子句指示所给定的力的分量的坐标系，而且总是需要使用两个或更多的力的分量（必须是正交的或是沿着主轴的）。对力矩也是这样。在下面的例子中，这个坐标系在固定的环境坐标系中并且具有基准方向。第 IV 部分将对 FORCE_FRAME 作更详细的介绍。

```
BEGIN "insert peg into hole"
  FRAME peg_bottom, peg_grasp, hole_bottom,
                        hole_top;
  MOVE barm TO bpark WITH DURATION =
                                    3 * seconds;
  OPEN bhand TO 3 * inches;
        {正常初始化}
  peg_bottom ← FRAME (nilrot, VECTOR (20, 30, 0)
                                    * inches);
  hole_bottom ← FRAME (nilrot, VECTOR (25, 35, 0)
                                    * inches);
  AFFIX peg_grasp TO peg_bottom RIGIDLY
    AT TRANS (ROT (xhat, 180 * degrees),
                          3 * zhat * inches);
  AFFIX hole_top TO hole_bottom RIGIDLY
    AT TRANS (nilrot, 3 * zhat * inches);
  MOVE barm TO peg-grasp;
  CENTER barm;
        {取螺钉}
  AFFIX peg_grasp TO barm RIGIDLY;
  MOVE peg_bottom TO hole_top;
  MOVE peg_bottom TO hole_bottom DIRECTLY
```

```
        {防止机械手升起和落下}
    WITH FORCE_FRAME = station IN WORLD
    WITH FORCE (zhat) = -10 * ounces
        {力的分量在基准坐标系中}
    WITH FORCE (xhat) = 0 * ounces
    WITH FORCE (yhat) = 0 * ounces
    SLOWLY;
        {SLOWLY 是一个减慢运动 3 倍的宏命令，参考
         3.14.4 节}
END "insert peg into hole";
```

当施加力时，应该有一个阻力，否则按牛顿第二定律机械手将会在给定方向加速。可是，即使在没有任何东西碰机械手时，柔性运动也会产生自发性运动(目前有这种现象)，这是因为按照牛顿第一定律，干扰信号的放大和机械手负载和几何模型的不精确性可能使机械手试图保持初速度。我们期望通过使用阻尼来使这个问题有所解决。

3.8 控制结构：IF，FOR 和 WHILE 语句

AL 具有很多传统的 ALGOL 的控制结构，包括条件和循环。在 AL 中没有转移指令，因为它们给保持规划值所需的流程分析带来混乱。在这一节，我们将讨论 IF，FOR 和 WHILE 语句。

IF 语句具有如下形式：

```
IF〈条件〉
  THEN〈语句〉
  ELSE〈语句〉
```

ELSE 部分是可选择的。条件部分是含有一个运算符(<，>，≤，≥，=，≠)的某个布尔表达式。布尔表达式可以由关系运算符，逻

辑连接符∧(与)、∨(或)、¬(非)、⊗(异或)、≡(EQV，逻辑等价)或者是逻辑常量 TRUE、FALSE 构成。条件部分也可以是某个算术标量表达式。如果条件为真(非零)，那么执行 THEN 后的语句，否则执行 ELSE 后的语句(若存在)。

FOR 循环的形式是

FOR〈s 变量〉←— 〈s 表达式〉STEP 〈s 表达式〉UNTIL 〈s 表达式〉DO 〈语句〉

其中〈s 变量〉表示标量变量，〈s 表达式〉代表具有相同量纲的标量表达式。变量的初始值就是第一个表达式的值，每执行一次语句，其值增加第二个表达式的值，重复该过程，直到其值超过第三个表达式的值。

WHILE 循环如下：

WHILE 〈条件〉DO 〈语句〉

其中〈条件〉与前面的相同，检查条件，若为真则执行语句。重复该过程，直到条件为假。

下面的例子说明 IF，FOR，WHILE 语句在一个程序中的应用。机械手从一个地方拾起铸件，把好的放在一个 6 × 4 的托板上，把有缺陷的丢掉，铸件每批来 50 个，但事先不知将有多少批。

```
BEGIN "sort castings"
  FRAME pickup, garbage_bin, pallet;
  SCALAR pallet_row, pallet_column, good, bad;
  DISTANCE SCALAR packing_distance;
  SCALAR ok, more_batches, casting_number;
  packing_distance ← 4 * inches;
  MOVE barm TO pickup WITH DURATION =
                          3 * seconds;
  OPEN bhand TO 3 * inches;
```

```
pallet_row ← 1; pallet_column ← 0; good ← 0;
                                    bad ← 0;
casting ← pickup;
MOVE barm TO pickup DIRECTLY;
CENTER barm;
IF (bhand < 1.5 * inches) THEN more_batches
                    ← FALSE
        ELSE more_batches ← TRUE;
WHILE more_batches DO
  BEGIN "sort 50 castings"
  FOR casting_number ← 1 STEP 1 UNTIL
                          50 DO
    BEGIN "sort casting in hand"
    ok ← FALSE;
    AFFIX casting TO barm RIGIDLY;
    MOVE casting TO pickup + 3 * zhat* ihches
    ON FORCE (zhat) ≥ 20 * ounces DO ok
                                    ← TRUE;
        {检查重量是否够}
    IF ok THEN
    BEGIN "good casting"
    good ← good + 1;
    IF pallet_column = 4
    THEN BEGIN pallet_column ← 0;
        pallet_row ← pallet_row + 1;  END
    ELSE pallet_column ← pallet_column + 1;
    MOVE casting TO pallet + VECTOR (pallet_
        column * packing_distance, pallet_rew *
```

```
                packing_distance; 0 * inches)
          WITH APPROACH = 3 *zhat * inches;
        UNFIX casting FROM barm;
        OPEN bhand TO 3 *inches;
        IF (pallet_column = 4) AND (pallet_row = 6)
          THEN BEGIN "pallet full" pallet_column ← 0;
                                   pallet_row ← 1;
              {移走这块托板,取新的托板的程序}
          END "pallet full";
        MOVE barm TO pickup;
        END "good casting"
      ELSE
        BEGIN "defective casting"
        bad ← bad + 1;
        MOVE casting TO garbage_bin DIRECTLY;
        OPEN bhand TO 3 * inches;
        UNFIX casting FROM barm;
        MOVE barm TO pickup;
        END "defective casting;
      casting ← pickup;
      CENTER barm;
      END "sort costing in hand";
MOVE barm TO bpark;
PRINT ("THERE WERE", good, "GOOD CASTINGS
                                    AND",
                bad, "DEFECTIVE CASTINGS");
END "sort castings";
```

3.9 控制结构(续): CASE 和 UNTIL 语句

AL 中另外两个传统的 ALGOL 控制结构是 CASE 和 UNTIL 语句.

CASE 语句有多种形式,正规的 CASE 语句的形式是

```
CASE 〈索引〉 OF
    BEGIN
      〈语句 0 〉;
      〈语句 1 〉;
      〈语句 2 〉;
         ⋮
      〈语句 n〉
      END
```

先计算标量索引表达式,根据其值的整数部分,执行下面的一条语句. 如果索引是零,那么选择语句 0 ,如果索引是 1 ,那么,选择语句 1 ,如此下去, 直到 n. 如果索引是负的或者是大于语句的数,那么将报告一个错误. 任何语句都可以是空语句, 例如: "〈语句 1 〉;〈语句 3 〉",在这种情况下,当索引是 2 时,什么也不执行.

CASE 语句还有一种有标号的形式:

```
CASE 〈索引〉 OF
  BEGIN
[C 0] 〈语句〉;
[C 1] 〈语句〉;
[C 2] 〈语句〉;
         ⋮
[Cn] 〈语句〉;
ELSE 〈语句〉
```

```
END
```

这里每条语句都有一个或几个注明它的非负的标量常数。与前面的一样，先计算索引表达式的值，如果其整数部分与 Ci 中的一个相同，那么就执行带有该标号的语句。否则，如果有 ELSE，那么执行它后面的语句，如果没有 ELSE，那么若索引的整数部分是负的或大于最大的 Ci 则出现错误，否则什么都不执行。注意，ELSE 语句可以出现在语句表中的任何地方，不必在最后。

下面是一个使用有标号的 CASE 语句的例子，当给出几个可能部件的一个时，选择适当的完成动作。

```
BEGIN
  SCALAR part_number;
  FRAME pick_up, base, base_grasp, cover,
        cover_grasp, side,
          side_grasp, ···;
  {初始化代码包括如下宏命令定义:
    DEFIN base_num = ···;
    DEFIN cover_num = ···;
    DEFIN side_num = ···;
    它们将用在有标号的 CASE 语句中以便更加清楚}
  {现在，去取 pick_up 处的部件，并对它做适当的处理}
  PRINT ("enter the part's number:");
  part_number ← INSCALAR;
  {INSCALAR 从键盘读入一标量}
  {由用户键入部件的数目，将来，这步可用视觉自动完成}
  CASE part_number OF
    BEGIN
      [base_number] BEGIN
                      {处理底座程序}
```

```
                    base ← pick_up;
                    MOVE barm TO base_grasp;
                    CENTER barm; {抓}
                    AFFIX base TO barm;
                    {处理底座的其他程序}
                END;
            [cover_num] BEGIN
                {处理盖子的程序}
                END;
            {重复其他已知部件：边，…}
            ELSE BEGIN
                PRINT ("Unknown part number",
                        crlf);
                {修改错误程序}
              END;
          END;
        {程序的其他部分。注意，上面 CASE 语句中的每条语
          句应该使机械手停留在相同的位置，若不然则需要一
          个规划时间分配（参考 4.5.1 节）以使环境模拟程序知
          道当 CASE 语句完成时机械手的位置。}
      END;
```

UNTIL 语句如下：

DO〈语句〉UNTIL〈条件〉

其中〈语句〉被重复执行，直到条件为真。它与上一节介绍的 WHILE 语句相似，只是 WHILE 循环是当条件为真时，而 UNTIL 循环是直到条件为真时。要注意的是一个 UNTIL 循环总是至少执行一次。

作为 UNTIL 应用的例子，下面摘录一段程序，即要得到一

个好的铸件，把处理过程中发现的坏的铸件扔掉。它与前节中的例子很相似。

```
BEGIN
  SCALAR success;
  {初始化程序}
  success ← false;
  casting ← pickup;
  MOVE barm TO casting_grasp;
  DO BEGIN          {试着得到一个好的铸件}
    CENTER barm;
    AFFIX casting TO barm RIGIDLY;
    MOVE casting TO pickup + 3 * zhat * inches,
         {看是否够重}
      ON FORCE ≥ 20 * ounces
      ALONG zhat OF station DO success ← true;
    IF ¬ success THEN {扔掉坏铸件}
      BEGIN
      MOVE casting TO garbage_bin DIRECTLY;
      OPEN bhand TO 3* inches;
      UNFIX casting FROM barm;
      casting ← pickup;
      MOVE barm TO casting_grasp
      END
      END UNTIL success;
  barm ← ←pick_up + 3 * zhat * inches;
      {规划时间分配以使环境模拟程序"满意"}
  {程序的其他代码}
END;
```

3.10 同时运动：COBEGIN-COEND，SIGNAL-WAIT

到目前为止，我们已经考虑了单臂的运动。为了完成机械手的同时运动，需要引进两个新的概念。除了块内的语句要同时执行外，COBEGIN_COEND 块与 BEGIN_END 块的作用是相同的。

这样，下面的语句将在同一时间使两台机械手运动到停放位置：

```
COBEGIN
    MOVE barm TO bpark;
    MOVE yarm TO ypark;
COEND;
```

在同时执行的前后关系中简单的同步是有可能的。这一点可用明确的事件和 SIGNAL、WAIT 语句来完成。用户要使用的每一个不同的事件都应该在如下所示的说明语句中说明：

```
EVENT e1, e2, e3
```

EVENT 不同于代数数据类型(如标量)而且用户在其程序中不能用正规的赋值语句赋给它一个特定的值。与每个事件有关的是接收信号的次数，最初时，这个数为零，即没出现信号，而且没有正在等待的进程。语句

```
SIGNAL  e1
```

使与事件 e1 有关次数值增 1，如果结果是零或是负值，等待 e1 的一个进程从等待状态释放出来，并准备执行。语句

```
WAIT  e1
```

使与事件 e1 有关的次数值减 1，如果结果是负值，发出 WAIT 命令的进程在连续运行状态下被阻塞，直到发来一个信号。如果

其值为零或正，无需任何等待。

下面的例子用来说明 SIGNAL 和 WAIT 命令的应用(当然可以不必用这种结构)。 蓝机械手拾起一个物体并移动到一个交接位置，在松开之前它必须确保黄机械手已经抓住了物体。

```
BEGIN
  EVENT passed, caught, ready_pass;
  FRAME sted_beam, pass, catch;
  COBEGIN
    BEGIN "blue"
    MOVE barm TO steel_beam;
    CENTER barm;
    AFFIX steel_beam TO barm;
          {barm 拿钢梁}
    MOVE steel_beam TO pass;
          {到传递位置}
    SIGNAL ready_pass; {barm 已准备好}
    WAIT caught;       {等 yarm 来接}
    OPEN bhand TO 3.0 * inches;
      {当黄机械手准备好后蓝手放开}
    UNFIX steel_beam FROM barm;
    SIGNAL passed;
      {蓝机械手宣布它已经放开}
    END "blue"
    BEGIN "yellow"
    OPEN yhand TO 3.0 * inches;
      {黄手与此同时张开}
    MOVE yarm TO catch;
      {黄手到"接收"位置}
```

```
        WAIT ready_pass;
          {黄机械手等待,直到有东西可抓}
        CENTER yarm; {抓物体}
        SIGNAL caught;
          {黄机械手宣布已经抓住}
        WAIT passed;
          {等待蓝机械手放开}
        MOVE yarm TO pallet;
        END "yellow";
      COEND;
    END;
```

下面第二个例子说明 SIGNAL 和 WAIT 在资源共享方面的应用。例子沿用上节中铸件分类的情况，但假设两台机械手做类似的工作，还有一个吊车，它取走装满东西的托板带来空托板。蓝色托板和黄色托板分别与各机械手对应。程序代码与前一节的相似，只是在标有“pallet_full”的块内说明“移走这块托板取新托板的程序”的那段有所不同外，这一段要使用 SIGNAL 和 WAIT，以保证在相同的时刻不让吊车去两个位置而且只有在需要它时才告诉它去某一位置。

```
    BEGIN
      EVENT blue_pallet_full, blue_pallet_empty;
      EVENT yellow_pallet_full, yellow_pallet_empty;
      EVENT crane_free;
      SCALAR more_blue_pallets, more_yellow_pallets;
      more_blue_pallets ← TRUE;
      more_yellow_pallets → TRUE;
      SIGNAL crane_free;
      COBEGIN
```

```
BEGIN "load blue pallets"
  BEGIN "sort castings"
                {3.8 节代码}
            ⋮
    IF (pallet_column = 4) AND (pallet_row = 6)
    THEN BEGIN "pallet full"
      pallet_column ← 0; pallet_row ← 1;
      SIGNAL blue_pallet_full;
      WAIT blue_pallet_empty;
      END "pallet full";
        ⋮
  END "sort castings";
  SIGNAL blue_pallet_full;
  {为了取走最后一个托板}
WAIT blue_pallet_empty;
more_blue_pallets ← FALSE;
  {为了停止吊车等待蓝托板,否则吊车程序将在"change blue pallet" 块中停留}
END;
BEGIN "load yellow pallets"
  BEGIN "sort castings"
      {除了使用黄机械手和黄托板外,其余与蓝托板程序相似}
  ⋮
  IF (pallet_column = 4) AND (pallet_ row = 6)
  THEN BEGIN "pallet full"
      pallet_column ← 0; pallet_row ← 1;
      SIGNAL yellow_pallet_full;
```

```
            WAIT yellow_pallet_empty;
          END "pallet full";
        :
      END"sort castings"
      SIGNAL yellow_pallet_full;
      WAIT yellow_pallet_empty;
      more_yellow_pallets ← FALSE;
    END;
    WHILE more_blue_pallets
    DO BEGIN "change blue pallet"
          WAIT blue_pallet_full;
          WAIT crane_free; {等待吊车空闲}
          {使用吊车改换蓝托板的程序代码}
          SIGNAL blue_pallet_empty;
          SIGNAL crane_free;
        END;
      WHILE more_yellow-pallets
      DO BEGIN "change yellow pallets"
        WAIT yellow_pallet_full;
        WAIT crane_free;
                    {等待吊车空闲}
      {用吊车改换黄托板的程序代码}
        SIGNAL yellow_pallet_empty;
        SIGNAL crane_free;
      END;
    COEND;
  END;
```

3.11 数组

我们有时希望一个变量代表多个值。例如假设在基座板上有三个螺孔，在装配期间，要编写把一个螺钉插进每个螺孔的程序，我们不想重复地给每个螺孔书写相同的代码，最好是只写一次，对所有的孔只要按某种方法使用 FOR 循环语句重复即可。数组允许我们这样做。

一个数组就是一个可以有多个值的变量。在上面的例子中，我们有三个坐标系：first_hole，second_hole 和 third_hole。我们可以定义一个坐标系数组：hole [1:3]，它允许我们按 hole [1]，hole [2]，hole [3] 来引用这三个螺孔。一个比较正式的数组定义为

〈类型〉 ARRAY 〈数组名 1〉[范围],〈数组名 2〉[范围]

其中〈类型〉给出了该数组的数据类型，[范围]表明数组的大小和怎样引用其元素。我们上面的例子使用了一个一维数组。一个二维数组的例子是

```
SCALAR ARRAY foo[1:3, 1:4]
```

它表示

```
foo[1, 1]  foo[1, 2]  foo[1, 3]  foo[1, 4]
foo[2, 1]  foo[2, 2]  foo[2, 3]  foo[2, 4]
foo[3, 1]  foo[3, 2]  foo[3, 3]  foo[3, 4]
```

一个数组可以有没有上限的维数。数组的范围可以是常量、变量或表达式。范围可以是正值，也可以是负值，只要下限小于上限即可。例如：

```
VECTOR ARRAY u[-3:3], v[n:n + 5], w[0:3, 1:m]
```

其中 n 和 m 是标量变量。在进入定义数组的块时才给数组分配空间，因此，v 和 w 的大小将依赖于定义出现时 n 和 m 的值。

在程序中使用数组与一般的变量一样。例如：

```
FOR i ← 1  STEP  1 UNTIL  4 DO  foo [1, i] ←
                                foo[ 2, i] * foo [3, i]
```

在运行时要做一次检查，看每个下标是否落在给定维数的上、下限之内。下标超出范围便打印一个错误信息。只使用下标的整数部分。

下面是一个例子，完成在本节开始时提到的螺钉插入的任务。

```
BEGIN
  FRAME ARRAY hole [1:3];
  FRAME base_plate;
  SCALAR i;
  {程序的初始化和开始，包括 base_plate 和螺孔位置的定
   义：
   base_plate ← FRAME (····);
   AFFIX hole [1] TO base_plate RIGIDLY
                          AT TRANS (····);
   AFFIX hole [2] TO base_plate RIGIDLY
                          AT TRANS (····);
   AFFIX hole [3] TO base_plate RIGIDLY
                          AT TRANS (····);
   螺钉要在 Z 轴向下方向定义。还包括把改锥拿到手里
   的程序代码}
   {现在开始插入三个螺钉}
   FOR  i ← 1  STEP  1  UNTIL  3  DO
  BEGIN
   screw ← screw_dispenser;
     {定义新螺钉的位置}
   MOVE driver_tip TO screw;
```

```
    {得到一个螺钉——实际并非这般容易}
    AFFIX screw TO driver;
    MOVE screw_tip TO hole [i];
      {螺钉正在螺孔上方}
    COBEGIN
      MOVE screw TO ⊗—0.75 * zhat * inches
        WITH FORCE = 20 * ounces ALONG zhat
                    OF screw
        WITH DURATION = 2.5 * seconds;
      {用手下推}
      OPERATE driver {拧螺钉}
      WITH VELOCITY = 200 * rpm
      WITH DURATION = 3 * seconds;
    COEND;
    UNFIX screw FROM driver
                    {放开螺钉}
  END;
END
```

注意，上面用的“driver”现在尚未实现。

3.12 过程

有时，需要在程序的几个地方完成相同的操作。我们不希望在每个地方都放一个完整的序列，而是希望把一个过程或子程序的主体只编一次码，在程序中需要这种操作的每个地方调用一次该过程。例如，在一次装配过程中，可能需要插入很多螺钉，这项操作可以写成一个过程。我们先来解释过程的句法。

过程定义如下：

〈类型〉PROCEDURE〈过程名〉(参量表);

〈语句〉

其中〈语句〉在每次过程调用时都要执行。停放机械手并打开手指的一个简单过程可以写成

```
PROCEDURE park;
  BEGIN
  MOVE barm TO bpark WITH DURATION
                        = 3 * sec;
  OPEN bhand TO 3* inches;
  END;
```

在程序中，每当用户要把机械手移到停放位置并张开手指时，所需键入的只是语句：

```
park
```

这就可以调用上述过程。有时要用过程回送一个计算需要的结果，即把过程作为一个函数。这可以用 RETURN 语句来完成：

```
RETURN (value)
```

其回送值就是过程的结果。例如，决定蓝机械手的高度的一个过程可以写成

```
DISTANCE SCALAR PROCEDURE height_barm;
    RETURN (POS (barm) · zhat);
```

当需要蓝机械手的高度时便可以调用这个过程。注意过程回送的数据类型的说明。我们可以把这个过程一般化，使得对于任何坐标系，它都可以回送该坐标系的高度。为了做到这一点，我们可以用一个参量给过程传递一个值。下面是一般化过程的形式及其应用例子：

```
DISTANCE SCALAR PROCEDURE height (FRAME f);
    RETURN (POS (f)、zhat);
PRINT ("The height of the pallet is:",
```

```
        height (pallet_top));
```

当调用过程时，参数 f 被 pallet_top 的值约束，因而在过程的主体中每次引用 f 时都是指 pallet_top. 参数可以通过引用来传递(对，变量和数组是默认的)，或者通过值来传递(这是传递表达式的唯一方式)。如果由引用传递一个变量，那么其值可由过程修改.例如，为了得到一个坐标系的更精确的位置，可以用机械手抓住它，然后再读取机械手的位置，这个过程可写成

```
PROCEDURE refine(REFERENCE FRAME obj);
  BEGIN
    OPEN bhand TO 3*  inches;
    MOVE barm TO obj;
    CENTER barm;
      {这可以"感知"物体的位置}
    obj ← barm
  END;
```

当过程返回时，作为其参数传递的坐标系具有一个新的值。

在很多程序设计教程中，所采用的关于过程的传统例子是阶乘函数：fact(1) = 1, fact(2) = 2 * 1, fact (3) = 3 * 2 * 1, ···. 在 AL 中有两种求阶乘的方法，第一种是重复，第二种是递归(即调用自身)。

```
SCALAR PROCEDURE rfact (SCALAR n);
  BEGIN
    SCALAR i, prod;
    prod ← 1;
    FOR i ← 2 STEP 1 UNTIL n DO prod ←
                                prod * i;
    RETURN (prod);
  END;
```

```
SCALAR PROCEDURE rfact (SCALAR n);
  IF n > 1 THEN RETURN (n* rfact (n — 1))
       ELSE RETURN (1);
```

一个完成螺钉插入操作的过程如下：

```
PROCEDURE insert_screw (FRAME hole_location);
  BEGIN
  screw ← screw_dispenser;
  MOVE driver_tip TO screw;
  {得到一个螺钉——实际并非如此容易}
  AFFIX screw TO driver;
  MOVE screw_tip TO hole_location;
  {螺钉正位于螺孔之上}
  COBEGIN
    MOVE screw TO ⊗ — 0.75* zhat * inches
      WITH FORCE = 20* ounces ALONG
                          zhat OF screw
  WITH DURATION = 2.5* seconds;
  {用机械手推下}
  OPERATE driver    {拧螺钉}
  WITH VELOCITY = 200 * rpm
  WITH DURATION = 3* seconds;
  COEND;
  UNFIX screw FROM driver {放开螺钉}
  END;
```

现在，前一节例子中插入三个螺钉的循环将是：

```
FOR i ← 1 STEP 1 UNTIL 3 DO insert_
                    screw (hole [i]);
```

应该提到的是，过程给 AL 编译程序中的环境模拟程序带来

了某些困难。关于这类问题及其解的讨论请参考 4.6 节。

3.13 程序员须知

3.13.1 向上抓取的位置

AL 用户很快就会发现，在正常使用时，坐标系 barm 通常使其 Z 轴在基准坐标系中指向下方。由于我们已经习惯于按向上的正 Z 轴方向来考虑问题，因此有时定义另外一个坐标系倒方便，它"刚性地""附着"在 barm 上，但其 Z 轴指向上方，Y 轴与基准 Y 轴平行或反向平行。如果机械手指向下方，用户可以用这样一个坐标系定义具有基准姿态的"抓取坐标系"。下面的语句将建立一个称为 bgrasp 的坐标系以完成我们需要的工作：

```
FRAME bgrasp;
AFFIX bgrasp TO barm
  AT TRANS (ROT (xhat, 180 * deg), nilvect
                    * inches) RIGIDLY;
```

3.13.2 初始化和程序的结束

在开始工作前，把机械手及其手初始化到已知位置，这是一个很好的想法，这样可以确保从一个未知位置开始的第一步运动不致使机械手移动太快。

建议使用下面语句：

```
MOVE barm TO bpark WITH DURATION =
                              3 * seconds;
OPEN bhand TO 3 * inches;
```

在程序结束时用下面语句使机械手到停放位置是好的策略：

```
MOVE barm TO bpark
```

在程序完成时，如果机械手不在停放位置，那么 AL 编译程

序将给出一个警告信息。

3.13.3 降低运动速度

当第一次试验一个程序并且不知道机械手将如何运动时，一个大于 1 的 speed_factor 的使用将使程序中的所有运动降低速度（详见 4.4.6 节）。在程序的开头，用户应按如下所示给 speed_factor 赋一个值：

```
speed-factor ← 2.0
```

为了方便起见，可用两个已经预先说明过的宏命令 SLOW 和 CAUTIONS（分别给 speed-factor 赋值 2.0 和 3.0）来代替上面介绍的赋值语句。

IV. AL 语 言

AL 是一种类似于 ALGOL 的源语言，它经过扩展后用于处理机械手控制问题．前一部分是个别指导式的 AL 语言介绍，下面将描述 AL 语言的一些特点．

4.1 基本结构

4.1.1 程序

AL 程序按照 ALGOL 传统的块结构组织．一个 AL 写的程序或者由一个单句组成，或者由一个块语句组成．块语句是用分号分隔、由保留字 BEGIN 和 END (或 COBEGIN 和 COEND) 包围的一个语句序列．块语句可以用紧跟在 BEGIN 或 COBEGIN 之后的一个字符串常量来命名．AL 要核查这个命名与跟在相应的 END 之后的字符串(如果有的话)，如果两个字符串不匹配，那么将报告一个错误信息．

BEGIN "block name" S; S; S; S END "block name"

4.1.2 变量

一个变量名以字母开头，是一个由字母数字字符和"_"组成的字符串．变量必须在使用前加以说明．AL 遵循正常的变量辖域规则：变量只能在其被说明的块或嵌套在此块内的块内被引用．相同名字的变量可以在几个块内说明，在这种情况下，在引用任何变量时，都要以包围该引用的最里层的说明为准．

4.1.3 注释

注释是插在程序中的、使其更加可读的文字．注释可以有两种形式．编译程序将跳过保留字 COMMENT 和遇到的下一个分号之间的所有文字．注释也可以用花括弧"{ }"包围．

4.2 数据类型和表达式

4.2.1 代数数据类型：SCALAR, VECTOR, ROT, FRAME, TRANS

AL 中基本的数据类型的选择是为了方便处理真实世界中三维问题．标量与其它计算机语言中的实数一样，是浮点数．向量是给出值的三元组 (x, y, z)，它表示对应于某坐标系的平移、速度和位置之类的量． 旋转是表示一个姿态或绕一个轴的旋转的 3 × 3 矩阵．一个旋转 rot 由一个给出旋转轴的向量和一个给出旋转角度的标量构成． (固联)坐标系用来表示局部的坐标系统．它们由一个给出原点位置的向量和一个给出轴的方向的旋转组成．变换用来把固联坐标系和向量从一个坐标系统变换到另一个坐标系统，它也是由一个向量和一个旋转构成．

4.2.2 标号和事件

标号和事件是用与代数数据类型同样方式说明的数据类型．标号有两种：语句标号和条件监控标号．给条件监控加标号是为了用 ENABLE 和 DISABLE 引用(参见 4.4.5.2 节)．给语句加标号是为了在调试时使用．一个标号由跟着一个冒号的标识符组成．目前，标号必须在使用前加以说明．

事件与语句 SIGNAL、WAIT 结合使用(见 4.5.4 节)可使并行过程同步化．

4.2.3 数组

AL 中允许使用多维数组．它们可以是任何代数数据类型或事件类型．数组的范围可以表示为标量常数、变量或表达式，可以是正整数或负整数．唯一的限制是下界要小于上界．在运行时，要作一次检查，看每个数组下标是否落在指定维数的已知的下限和上限之内．下标越界将打印出一个错误信息．

在进入定义数组的块时给数组分配空间，在退出该块时收回空间．

4.2.4 量纲

AL 允许使用与变量相联系的量纲．已知的量纲是：TIME，DISTANCE，ANGLE，FORCE，TORQUE，VELOCITY ANGULAR-VELOCITY 和 DIMENSIONLESS．如果需要的话，可以用 DIMENSION 定义新的量纲：

DIMENSION〈新量纲〉＝〈量纲表达式〉

其中在〈量纲表达式〉中定义的运算符是：(，)，*，/和 INV．INV 取其参数之逆，如 INV (TIME) = 1/TIME．

除了要乘以适当的保留字（SEC、CM、DEG、GM、INCHES、OZ 和 LBS，或者 SECONDS、INCH、OUNCES、DEGREES 和 RADIANS）之外，带量纲的量与一般的量一样．例如：

```
VELOCITY VECTOR v;
  v ← xhat * inches/sec
```

其它单位可用宏命令定义(参见 4.5.8 节)，例如：

```
DEFINE  feet = ⊂〈12 * inches〉⊃
```

AL 要检查带有量纲的量的使用是否一致，加法和减法，还有坐标系运算、变换运算和旋转运算需要精确的量纲匹配，而标量和向量的乘法和除法将产生一个具有新量纲的量．

4.2.5 说明

说明语句用来定义程序中使用的每个变量的数据类型和量纲，其形式是

〈量纲〉〈数据类型〉〈变量表〉

其中〈量纲〉是一个 AL 中已经定义的量纲（TIME，DISTANCE，ANGLE，FORCE，TORQUE，VELOCITY，ANGULAR，VELOCITY），或者是用户已经定义的一个量纲。〈数据类型〉是以下数据类型中的一个：SCALAR，VECTORY，ROT，FRAME，TRANS，EVENT 和 LABEL。只有代数数据类型 SCALAR，VECTOR，TRANS 可以有一个相关的量纲。除非已经说明，否则认为标量和向量是无量纲的，而且认为变换具有长度量纲（参见 3.1.1.5 节）。

数组说明的形式如下：

〈量纲〉〈数据类型〉ARRAY〈变量表〉

变量表中的每个变量都有命名，命名后面跟着包围在方括号"[ ]"中的上下限对。例如：

"name [L1: U1，L2: U2，···]"。

4.2.6 算术表达式

下面是可以使用的算术运算符的一览表，它们按结果值的数据类型分组。使用的缩写有：'s' = 标量，'v' = 向量，'r' = 旋转，'f' = 坐标系，'t' = 变换。

标量运算符

s + s	标量加法
s — s	标量减法
s * s	标量乘法

s / s	标量除法
s ↑ s	标量幂
s MAX s	最大值
s MIN s	最小值
INT (s)	s 的整数部分
s DIV s	每个参量经 INT 作用后的整数商
s MOD s	每个参量经 INT 作用后的整数余数
v · v	两个向量的点积
|s|	一个标量的绝对值
|v|	向量的大小(向量的模)
|r|	求取旋转的转角
INSCALAR	从终端读一标量

标量函数

SQRT (s)	平方根
SIN (s)	正弦(所有的三角函数都用度)
COS (s)	余弦
TAN (s)	正切
ASIN (s)	反正弦
ACOS (s)	反余弦
ATAN2 (s, s)	s/s 的反正切
LOG (s)	自然对数
EXP (s)	e 的 s 次幂

布尔运算符

s〈关系〉s	若关系满足则回送真，否则为假；可能的关系是：$<$, $\leqslant$, $=$, $\geqslant$, $>$, $\neq$
s ∧ s	逻辑与

s ∨ s	逻辑或
s ⊗ s	逻辑异或
s ≡ s	逻辑等价
¬s	逻辑非
QUERY	从终端读一布尔量(参见 4.5.7 节)

向量运算符

VECTOR(s,s,s)	给定 (x, y, z) 分量构造向量
s * v	向量的扩张
v / s	向量的收缩
v + v	向量的加法
v — v	向量的减法
v * v	向量的叉积
r * v	向量的旋转
t * v	向量的变换
f * v	向量的变换——(station → f) * v 的缩写
v WRT f	基准坐标系中与在 f 坐标系中的 v 指向相同的一个向量 v WRT f ≡ ORIENT (f) * v ≡ (f * v) — POS(f)
UNIT (v)	与 v 同向的单位向量
POS (f)	坐标系或变换的向量位置
AXIS (r)	旋转的轴

旋转运算符

ROT (v, s)	构造绕 v 旋转 s 度的旋转
ORIENT (f)	一个坐标系或变换的姿态
r * r	两个旋转的合成(先作用右边的旋转)

坐标系运算符

FRAME (r, v) 在 v 处方向为 r 的坐标系

CONSTRUCT(v,v,v) 构造一个坐标系：第一个向量给出位置，第二个是 x 轴上的一点，第三个是 xy 平面上的一点·

f + v	一个坐标系的平移
f — v	一个坐标系的平移
t * f	一个坐标系的变换
f * f	一个坐标系的变换——(station → f)* f 的缩写

变换的运算符

TRANS (r, v)	构造一个旋转 r 平移 v 的变换
f → f	把第一个坐标系映射到第二个坐标系的变换
t * t	两个变换的合成(先作用右边的变换)
INV (t)	取 t 的逆

AL 中的运算符一般遵循正常的优先级规则，即首先计算函数，接着是指数运算，乘法或除法，加法或减法。在适当的地方加入括号可以改变运算的顺序。在几个优先级相同的运算符出现在相同级的表达式中，则从左到右进行运算

优先级表

函数、( )、||、NOT

WRT、→、↑

*、/、MAX、MIN、DIV、MOD

+、—

=、≠、<、>、≤、≥

∧

V、⊗
═

4.2.7 预先说明过的常量

PI = 3.14159…	(也可以写成 π)
STATION	基准坐标系
BARM	蓝色机械手的位置
YARM	黄色机械手的位置
BHAND	蓝机械手手指间的距离
YHAND	黄机械手手指间的距离
BPARK	蓝机械手的停放位置 =FRAME (ROT (yhat, 180*degrees), VECTOR(43.53, 56.86, 9.96)*inches)
YPARK	黄机械手的停放位置 =FRAME (ROT (yhat, 180*degrees), VECTOR (40, 14, 9)*inches)
TRUE 和 FALSE	TRUE = 1, FALSE = 0
XHAT	VECTOR (1, 0, 0)
YHAT	VECTOR (0, 1, 0)
ZHAT	VECTOR (0, 0, 1)
NILVECT	VECTOR (0, 0, 0)
NILROT	ROT(zhat, 0*DEG)
NILTRANS	TRANS (nilrot, nilvect)
CRLF	打印一个回车跟换行的字符串常量

4.2.8 一些例子

```
DISTANCE VECTOR v1, v2;     {某些说明}
ANGLE SCALAR theta;
```

```
SCALAR ARRAY s1[1:5], s2[-3:3, 1:2];
FRAME f1, f2;
EVENT ready;
ROT (zhat, 90*deg)*v1      {v1 绕基准Z轴旋转90°}
v1 · yhat                  {v1 的Y分量}
f1*xhat                    {f1 在基准坐标系中的X轴}
3*s1 [2]                   {数组 s1 的第二个元素乘以 3}
```

4.3 "附着": AFFIX 和 UNFIX

一个物体不同位置、姿态之间的关系，以及不同物体间的关系可以用 AFFIX 语句进行模型化. AFFIX 语句的一般形式是

AFFIX f1 TO f2 BY t AT 〈表达式〉〈附着类型〉

上面语句的作用是建立一个表达 f1 和 f2 间关系的变换. 如果有〈BY t〉项，那么结果变换将与变量 t 有关，从而使附着关系可由用户修改，否则将产生一个内部变量. 变换的初值由语句的〈AT 表达式〉部分给出. 若没有给出，则用 f1 和 f2 的当前值产生一个从 f2 到 f1 的变换 (f2 → f1). 附着语句可有两种型式，〈附着类型〉给出是否按"刚性地"或"非刚性地"完成附着. 刚性的附着是对称的，当给其中一个坐标系以新值时，另一个将被修改以维持它们之间的关系. 非刚性的附着是不对称的，当改变 f2 时，f1 的值被修改；反之，当修改 f1 时，将重新计算描述 f1 和 f2 之间关系的变换以表达它们之间新的关系. 非刚性附着语句的一个例子可以是托盘上的一个盘子，盘子随托盘移动，但反之则不然. 如果没有给出〈附着类型〉，那么将假设是刚性的附着.

一个附着关系可以用 UNFIX 语句解除：

UNFIX f1 FROM f2

4.4 运动

4.4.1 编译时间和运行时间的考虑

在目前的 AL 系统中，轨迹的计算是在编译时间内完成的. 当编译程序遇到一条运动语句时，它计算出尽可能快地完成这一运动的一条轨迹（当然这要受到电机所能产生的最大加速度和力矩的限制），对于所有描述期望运动的表达式，都使用编译时间规划值. 在实际执行时，这些表达式可能有不同的值，因此运行系统在执行运动语句之前要立即修改轨迹. 在运行时间内所能纠正的差异的大小是有限制的. 如果规划值偏差很严重，那么作出紧急纠正的努力有可能使机械手超载运行，导致运动的中止. 解决这个问题的一种简单的方法是：告诉编译程序给运动较长的时间. 目前我们正在研究避免这种情况的实时计算路径的实现问题.

如果要求一个非法的运动，例如试图把机械手移动到它无法达到的位置，或者要求的运动时间过短以致于不可能实现，这种情况下，编译时间轨迹计算程序将会给出错误信息. 应该注意的还有，修饰运动子句的很多参数都必须是常数.

4.4.2 基本的 MOVE 语句

基本的 MOVE 语句具有如下形式：

MOVE〈可控制的坐标系〉TO〈目的地〉〈修饰子句〉

这条语句使给定的机械手移动，从而使其与目标坐标系表达式〈目的地〉具有相同的位置和姿态. 符号"⊗"可用在〈目的地〉中，以表示运动执行时〈可控制的坐标系〉的当前位置. 〈可控制的坐标系〉可以是一个实际的机械手（barm 或 yarm），也可以是已经"附着"到一台机械手上的某个坐标系. 对于后一种情况，将使用由连接坐标系与机械手的附着链来描述的它们之间的物理关系，从而

使运动结果到达要移到的〈目的地〉坐标系。通过使用下面将要介绍的各种〈修饰子句〉，可以有很多不同的方式来修饰运动。

4.4.3 中间点：VIA, DEPARTURE, APPROACH

当运动必须通过一系列中间点时（例如为了躲避障碍物），可以用 VIA 子句给出中间坐标系，如：

VIA f1, f2, f3, f4, f5

其中 f1, ···, f5 是坐标系表达式。运动将按给定顺序通过这些点。也可以给出机械手在某一中间点的速度，以及从最后的给定点到该中间点的运动的持续时间。这种完整的 VIA 子句如下所示：

VIA f WHERE VELOCITY = 〈v〉, DURATION = 〈n〉

其中 v 是速度向量，n 是时间标量。可以有一个或两个修饰子句，顺序任意。注意，与第一次提到的形式不同，在这种格式中只许给定一个坐标系。如果轨迹计算程序认为此段运动需要的时间多于 n 秒，那么将产生一个错误信息。速度和持续时间的值都必须是编译时间常量。

还可以给出接近/远离点。这些点是与机械手从其当前位置离开有关的点，或是与其接近目标位置有关的点。与中间点不同，接近/远离点用初始坐标系或目标坐标系来表示。子句如下：

WITH DEPARTURE = 〈表达式〉

WITH APPROACH = 〈表达式〉

句中的〈表达式〉如下所示（〈fr〉代表目标坐标系或当前位置，这取决于使用的是 APPROACH 还是 DEPARTURE 子句）：

〈表达式〉类型	基准坐标系中的接近/远离点
坐标系	〈fr〉*〈表达式〉
向量	〈fr〉+〈表达式〉 WRT 〈fr〉
标量	〈fr〉+(〈表达式〉*zhat) WRT 〈fr〉

也可以通过规定〈表达式〉为 NILDEPROACH 来说明没有使用接近/远离点，或者通过函数 DEPROACH (〈坐标系标识符〉)来使用与其它某个坐标系相联的接近/远离点。

接近/远离点可以隐含地给出。语句：

DEPROACH (〈坐标系标识符〉)←〈表达式〉

将使〈表达式〉与变量〈坐标系标识符〉代表的接近/远离点联系起来。如果在运动语句中没有明确地给出接近点，那么与目标坐标系相连的接近/远离点就用作接近点。如果目标坐标系没有与之相关的接近/远离点，那么编译程序将逐一搜索附着于它的那些坐标系，直到找到一个接近/远离点。如果没有发现，那么就使用基准坐标 3*zhat*inches 作为接近/远离点。如果目标是一个坐标系表达式，那么用 NILDEPROACH 作为所用接近点的默认值。如果没有给出远离点，那么就用上次运动的接近点作为远离点。

AL 预先说明过的宏命令 DIRECTLY 扩张成两个子句：

```
WITH  DEPARTURE = NILDEPROACH
WITH APPROACH = NILDEPROACH
```

4.4.4 力和柔顺性

我们可以使机械手施加一些指定的力和力矩，或者敏感一些力和力矩（对于力的传感将在下面的 4.4.5 节讨论）。为了避免不合适的要求，力的分量必须永远是正交的。要保证这一点，必须给出一个力的坐标系，而且所施加的力和力矩的方向必须与当前这个力的坐标系的一个基轴成一直线，还要说明轴的方向是否随手的移动而变化，即所定义的力的坐标系是相对于机械手还是工作台坐标系。完成所有这些要求的语句如下：

```
WITH FORCE = 〈sval〉 ALONG 〈axis-vector〉 OF
                    〈frame〉 IN〈coord sys〉
WITH TORQUE = 〈sval〉 ABOUT 〈axis-vector〉 OF
```

〈frame〉 IN 〈coord sys〉

或者:

WITH FORCE — FRAME = 〈frame〉 IN 〈coord sys〉
WITH FORCE = 〈sval〉 ALONG 〈axis-vector〉
WITH TORQUE = 〈sval〉 ABOUT 〈axis-vector〉

或者:

WITH FORCE-FRAME = 〈frame〉 IN 〈coord sys〉
WITH FORCE (〈axis-vector〉) = 〈sval〉
WITH TORQUE (〈axis-vector〉) = 〈sval〉

其中:

〈axis-vector〉 = xhat, yhat 或 zhat
〈coord sys〉 = HAND 或 WORLD (默认值=WORLD)
〈sval〉 = 力的大小
〈frame〉 = 力坐标系的轴的方向

在第一种形式中,所有子句给出的力的坐标系必须是相同的。如果没有给出 IN〈coord sys〉,那么就假设是 WORLD;同样,若省略 OF 〈frame〉,则假设是 STATION。注意,每次运动只能指定一个力的坐标系。施加一个大小为零的力意味着机械手将是柔性的,即可沿任何外部的力的方向移动。

对于那些只需施加或感知一个力而不是两个力的运动,可用缩写形式。例如:

WITH FORCE = 〈sval〉 ALONG 〈vect〉 OF 〈frame〉
IN 〈coord sys〉

或者:

WITH FORCE (〈vect〉) = 〈sval〉

上面这些形式可以显然地推广到力矩和力的感知的情况。如果没有给出〈frame〉和〈coord sys〉,那么将在环境坐标系统中自动生成一个X轴与〈vect〉在一条直线上的力的坐标系。否则,使

用给定的坐标系统，并且产生一个X轴与〈vect〉WRT〈frame〉在一条直线上的力的坐标系。

4.4.5 条件监控

4.4.5.1 类型：力、持续时间、事件和布尔量 在一台机械手运动期间，一般希望监视某个条件或者某些条件。如果条件满足就执行一个动作。条件监控子句就是用于这个目的。其一般形式如下：

ON〈条件〉DO〈动作〉

目前可以监视的条件包括力的感知、持续时间、事件和各种变量的布尔表达式。〈动作〉可以是任何合法的 AL 语句或块。唯一的限制是，如果〈动作〉中只有一条运动语句，那么必须用 BEGIN 和 END 包围，以避免歧义。

监视过程随运动的开始而开始，并持续到运动的结束。如果监控条件被触发，那么在完成其动作之后，监视过程暂时停止，并停止检查其条件。这种过程可以用后面要描述的（参见 4.4.5.2 节）ENABLE 和 DISABLE 语句进行修改。

当感觉到力和力矩时，使用下列语句：

ON FORCE〈rel〉〈sval〉ALONG〈axis-vector〉
OF〈frame〉IN〈coord sys〉DO〈动作〉
ON TORQUE〈rel〉〈sval〉ABOUT〈axis-vector〉
OF〈frame〉IN〈coord sys〉DO〈动作〉

或者：

WITH FORCE_FRAME = 〈frame〉IN〈coord sys〉
ON FORCE〈rel〉〈sval〉ALONG〈axis-vector〉
DO〈动作〉
ON TORQUE〈rel〉〈sval〉ABOUT〈axis-vector〉
DO〈动作〉

或者：

WITH FORCE_FRAME = 〈frame〉 IN 〈coord sys〉
ON FORCE (〈axis-vector〉) 〈rel〉 〈sval〉 DO 〈动作〉
ON TORQUE (〈axis-vector〉) 〈rel〉 〈sval〉 DO 〈动作〉

其中 〈axis-vector〉，〈coord sys〉，〈sval〉 和 〈frame〉 与前面 4.4.4 节所用的相同，〈rel〉 是≥或者≤，当力或力矩超过或低于对应的给定值时，条件监视被触发。与施加力时一样，当仅有一个力被感知或施加时，有缩写形式：

ON FORCE 〈rel〉 〈sval〉 ALONG 〈vect〉
OF 〈frame〉 IN 〈coord sys〉 DO 〈动作〉

或者：

ON FORCE (〈vect〉) 〈rel〉 〈sval〉 DO 〈动作〉

条件监控子句：

ON DURATION ≥ n * seconds DO 〈动作〉

将在运动开始有效 n 秒之后触发〈动作〉。

ON 〈事件〉 DO 〈动作〉

意味着如果有信号发给〈事件〉，就去执行〈动作〉(信号由其它监控条件或别的某个并行过程发)。

ON 〈布尔表达式〉 DO 〈动作〉

上句的作用是计算由代数变量组成的布尔表达式的值，如果为真(非零)，则完成希望的动作。如果表达式的值为假，那么在重新对表达式进行计算和检查之前，对条件的监视暂停一会儿（目前是100ms)。

4.4.5.2 ENABLE 和 DISABLE——给条件监视加标号 条件监视有两种状态：有效和失效。在有效状态，当出现正在检测的条件时，结论将被触发。在失效状态，对条件的监视是不起作用的。如上所述，当运动开始时，条件监视是有效的；当运动结束时，变为失效的。一旦条件监视触发后，它就处于失效状态，除非使其

重新成为有效的。利用放在条件监视前面的 ENABLE 语句，可以完成"重新有效"工作。

用 ENABLE 和 DISABLE 语句可以改变已经命名的任意一个条件监视的状态(通过在保留字 ON 前放一个标号，可以给条件监视命名)。这些语句的句法如下：

ENABLE〈条件监视〉

和

DISABLE〈条件监视〉

在条件监视前放置保留字 DEFER 将使其在开始时是失效的。可以在以后明确地使其有效。下面是一个例子，一个条件监视在开始时是失效的，经过 3s 后变成有效的。

```
MOVE barm TO dest
  test:DEFER ON FORCE (zhat) ≥ 10 * oz DO STOP
       ON DURATION ≥ 3 * sec DO ENABLE test
```

4.4.6 其它子句：DURATION、SPEED-FACTOR、NULLING 和 WOBBLE

这里是能用来修饰运动的其它一些子句。应注意的是这些子句的参数是编译时间常量。子句

WITH DURATION = 〈sval〉

使结果运动在给定时间〈sval〉(应具有 TIME 量纲)内完成。如果轨迹计算程序认为需要更多的时间，那么 AL 将发出一个警告信息。子句

WITH SPEED_FACTOR = 〈sval〉

的作用是减慢运动速度。AL 计算出的运动的最短时间用〈sval〉(应 ≥ 1)去乘，其积用作运动的时间。

运动的默认速度因子是 1，因此运动占用尽可能少的时间。用正常的赋值语句给预先说明的变量 SPEED_FACTOR 赋予希望

的默认乘数，这样就可以改变速度因子默认值：

SPEED_FACTOR ←〈新的默认速度因子〉

还有两个预先定义过的宏命令：CAUTIOUS 和 SLOW，它们相应地设置默认速度因子 2 和 3。子句。

WITH NULLING

告诉运行系统在该运动结束时不给出错误信息。当前默认的是 WITH NO_NULLING 子句。有两条可以达到同样结果的宏命令 PRECISELY 和 APPROXIMATELY. 子句

WITH WOBBLE ═〈sval〉

在机械手最后三个关节上增加一个小的正弦运动，以使其稍有抖动。这对消除小的摩擦力影响和放置部件是很有用的。〈sval〉是一个具有 ANGLE 量纲的小的编译时间常量，一般约为 2° 或 3°。

4.4.7 控制手指：OPEN、CLOSE 和 CENTER

手指可以有好几种控制方式：

OPEN 〈hand〉 TO 〈sval〉

和 CLOSE 〈hand〉 TO 〈sval〉

这两条语句使手指张开或合拢后相距〈sval〉。〈sval〉是具有 DISTANCE 量纲的任意的标量表达式。目前，OPEN 语句和 CLOSE 语句之间没有区别。以后，如果两个接触传感器被触发，那么 CLOSE 将停止手指的运动。子句

CENTER 〈arm〉

合拢指定机械手的手指，直到两个接触传感器指示已经接触上为止。进一步，如果一个手指在另一个之前接触到物体，那么 CENTER 语句将使机械手自身移动，以使得要抓的物体不被手指推走。OPEN 和 CLOSE 只是移动手指，如果被抓的物体不是位于两手指中间，那么物体将被移动，或者如果它固定在某处，手指可能用过大的力。因此，应该使运动中止。

4.4.8 STOP 和 ABORT

在运动完成之前，有两种方式使其停止：

STOP〈设备〉

和

ABORT (〈打印表〉)

STOP 语句使指定设备停止工作。〈设备〉可以是一个实际的机械手，或者是附着到某台机械手上的一个坐标系。如果没有给出〈设备〉，而且 STOP 语句出现在一条运动语句中，那么执行运动的机械手就是要停止的机械手。ABORT 语句用在更紧急的情况。它将停止所有设备的运动，打印出〈打印表〉中的元素(见下面 4.5.7 节 PRINT 语句的介绍)，并转到 11DDT 控制。用户可以打入〈alt〉P 到 11DDT 使程序继续执行。这些语句通常出现在条件监视的主体中，虽然它们可以出现在程序的任何地方。

4.4.9 其它设备——OPERATE 语句

OPERATE 语句用于控制接到 AL 系统的其它设备。其句法与 MOVE 语句的句法类似：

OPERATE〈设备〉〈修饰子句〉

其中〈设备〉是要控制的设备，〈修饰子句〉描述设备将要完成的动作。例如：

OPERATE vise WITH OPENING = 4 * inches

目前，除了机械手外没有其它设备可用。一个"改锥"和"虎钳"将在不久使用。

4.5 非运动语句

4.5.1 赋值语句

赋值语句

〈变量〉←〈表达式〉

使由〈表达式〉表示的值赋给出现在赋值符号左边的变量。赋值符号右边的表达式的数据类型和物理量纲必须与左边变量的数据类型和量纲相同。

还有另一种形式，即规划时间赋值：

〈变量〉←←〈表达式〉

其中〈变量〉和〈表达式〉与上面相同。规划时间赋值语句提供了把某些变量的值传递给环境模拟程序的一种方法，否则的话，这些变量的值只能在运行时间内知道（例如 barm, yarm, bhand 和 yhand）。规划时间赋值语句不产生可执行码，它们只在程序编译期间起作用。

有的情况下必须使用规划时间赋值。例如，在一条要较早停止运动语句（如停在接触点）之后，用规划时间赋值语句把运动结束时机械手希望的位置传送给轨迹计算程序，这样可计算出一条较好的轨迹。在运行期间，机械手的实际值由实际环境决定，并且该值用于修改已计算的轨迹。

4.5.2 传统的控制结构——IF, FOR, WHILE, UNTIL, CASE

AL 具有很多传统 ALGOL 的控制结构。

IF 语句形式为

IF〈布尔表达式〉THEN〈语句〉ELSE〈语句〉

ELSE 部分是可选择的。如果〈布尔表达式〉为真（非零），那么执行跟在 THEN 后的语句，否则执行 ELSE 后面的语句（如果有的话）。

FOR 循环形式为

FOR〈s变量〉←〈s表达式〉STEP〈s表达式〉
UNTIL〈s表达式〉DO〈语句〉

其中〈s变量〉是一个标量变量，〈s表达式〉是具有相同量纲的标量表达式。变量的初值是第一个表达式的值，每次执行语句，其值按第二个表达式的值增加，重复该过程直到其值超过第三个表达式的值。

WHILE 循环如下：

WHILE〈布尔表达式〉DO〈语句〉

检查布尔表达式，若为真则执行语句。重复这个过程直到条件为假。

UNTIL 语句如下：

DO〈语句〉UNTIL〈布尔表达式〉

这里，重复执行语句直到条件为真。这一语句与上面介绍的 WHILE 语句很相似，只是 WHILE 的循环是发生于条件为真时，而 UNTIL 的循环则是直到条件为真时为止。

CASE 语句有两种形式。正规的 CASE 语句的形式为：

CASE 索引 OF BEGIN S0; S1; S2;···Sn END;

计算索引的值，根据其值的整数部分，执行其中的一条语句。如果索引为零，那么选择 S0 语句，若索引为 1 则选 S1，等等，直到 n。如果索引为负或者大于语句的数目，那么将报告一个错误。任何语句都可以是空语句。例如，“S1; ; S3”，在这种情况下，若索引为 2 则不执行任何语句。

还有一种带标号的 CASE 语句：

CASE 索引 OF BEGIN [C0]S; [C1] [C2]S;···
[Cn]S; ELSE S END

其中每条语句有一个或多个非负标量常量作为标号。计算索引表达式，若结果与 Ci 中的某个相同，那么就执行标有此标号的语句。若没有与索引匹配的常量，则什么都不作，除非有一个 ELSE 语句，此时执行 ELSE 语句。如果索引是负的或大于最大的 Ci，那么就会出现错误，除非有一个 ELSE 语句。注意，ELSE 语句

可以出现在语句表中的任意地方，不必在最后。

4.5.3 过程

过程的定义如下：

〈类型〉PROCEDURE〈名字〉〈参数〉;〈语句〉;

其中语句在每次调用过程时执行。只有那些回送结果的过程才需要给出类型。参数的数据类型可用保留字 VALUE 和 REFERENCE 修饰。REFERENCE 是默认的。由于 AL 编译程序要完成环境的模拟，因此当定义过程时，必须说明作为参数使用的数组的维数。例如：

PROCEDURE foo (FRAME ARRAY pnts [1:4, 1:3]);

我们必须给定义在与过程相同的程序块内的形式参数或变量赋予规划时间，这样当模拟过程时，才能得到这些规划时间值(参见 4.6 节)。

用 RETURN 语句可使过程回送一个结果，其形式如下：

RETURN (返回值)

这条语句回送过程的结果。RETURN 语句不可能出现在条件监视中或 COBEGIN-COEND 块中。

调用过程要用过程名的正式形式，后面跟以参数表：name (参数表)。它们可能出现在表达式可能出现的任何地方，或者单独作为一条过程语句。如果一个标有类型的过程在一条过程语句中出现，那么它的回送结果就要被放弃。

4.5.4 并行控制：COBEGIN-COEND, SIGNAL, WAIT

除了按正常顺序执行在一对 BEGIN-END 之内的语句外，AL 还可以通过把要执行的代码块放在 COBEGIN-COEND 块内，允许它们并行执行。在进入 COBEGIN 块时，控制被分解到要同时执行的不同的进程之中。在所有这些进程结束时，控制被传送

到 COEND 后面的程序部分.用户的职责是：保证要并行执行的代码具有足够的独立性（如两个进程不在同一时间内使用同一台机械手），并且不发生死锁。

应该注意的是，COBEGIN 结构的目的是允许同时对独立的机械手进行控制。虽然在一台机械手运动的同时进行计算可以节省时间，但是并行执行纯属计算的代码并不特别有用.所用的调度算法启动并执行一个进程直到它被阻塞，到那时将运行另一进程.等待一个事件、暂停、执行 *I/O* 操作或运动的初始化，都可以造成一个进程的阻塞。

用明确的事件和 SIGNAL 及 WAIT 语句可以实现并行进程的同步。每一事件都有一个与之相关的值：向它发信号的次数。最初，该值为零，即没有信号出现，从而不等待任何进程。语句

```
SIGNAL  e1
```

使与事件 e1 相关的计数值加 1，如果结果是零或正数，那么等待 e1 的进程之一就从等待状态中释放出来，并准备执行。语句

```
WAIT  e1
```

使与事件 e1 相关的计数值减 1，如果结果为负数，那么发布 WAIT 命令的进程就不能继续执行而被阻塞，一直到另一个进程发信号给 e1。如果计数值是零或正数，那么就没有任何等待。

4.5.5 语句条件监视

条件监视除了修饰运动之外，还可以作为语句出现。4.4.5 节中的描述也适用于语句条件监视。当执行所定义的语句时，语句条件监视成为有效的。当条件触发即明确地失效时（必须标明这种情况的发生），或局部阻塞解脱时，条件监视将成为无效的。保留字 DEFER 仍使条件监视在开始时被定义为失效状态。

辖域规则在条件监视有效或失效时开始起作用。一个使条件

监视有效或无效的语句只是对与它自己定义在同一块内或包括它的块内的条件监视起作用。

4.5.6 PAUSE 语句

语句

PAUSE 〈sval〉

使程序进入"睡眠"状态，时间由〈sval〉给定，它应具有 TIME 量纲。

4.5.7 *I/O*

在运行时间内，字符串和变量的值可以用 PRINT 语句打印出来：

PRINT (〈参数 1〉,〈参数 2〉,…,〈参数 n〉)

其中〈参数〉可以是代数表达式或变量，也可以是字符串常量。字符串用双引号标明，CRLF 是一个预先定义的字符串，它表示一个回车跟一个换行。

语句

PROMPT (〈打印表〉)

在语法上很像 PRINT 和 ABORT 语句。当遇到一条 PROMPT 语句时，AL 运行系统打印出打印表中的所有项，然后还打印信息：

"Type P to proceed"（打入 P 以继续执行）

上述语句等待一个要打入的 P。与 ABORT 语句不同，控制不交给 DDT，因此任何并行进程(如 ALAID 和 COBEGIN) 将继续被执行。例如：

```
PROMPT ("Move barm to work station origin");
                                org ← barm;
```

有两个算术运算符可以从 VT05 终端上读入一个值。 INS-

CALAR 读入一个标量，并给用户提示："SCALAR, please:"；QUERY 读入一个布尔量，它类似于 PROMPT，也可以有一个打印表。在打入打印表后，用户被问以 "Type Y or N:"。例如：

```
PRINT ("How tall is casting?");
height ← INSCALAR;
WHILE QUERY ("More to do?")DO···
```

ALAID 是 AL 的一个调试程序，可以在 AL 和其他程序之间完成 *I/O* 操作(通常在 PDP-10 上完成视觉任务)。(参见 7.6.3.1 节)

4.5.8 宏命令

AL 有一通用文本宏命令工具。宏定义的句法是：

DEFINE〈宏命令标识符〉〈参数〉＝⊂〈宏本体〉⊃

其中〈宏命令标识符〉是宏命令的名，〈宏本体〉是在程序中遇到〈宏命令标识符〉时要被取代的文本，〈参数〉(若存在的话)是由逗号分隔、括号包围的宏命令的一个参数表。只有未说明的标识符可用作宏命令参数。当宏命令被展开时，实际的参数将代替出现在宏本体中的参数。如果该值不是一个简单的符号，它必须用界限符⊂⊃围住。宏本体也用⊂⊃定界。

这里有两个使用宏命令的例子：

```
DEFINE feet = ⊂12 * inches⊃;
DEFINE grasp (frob) = ⊂MOVE barm TO frob;
                      CENTER barm;
                      AFFIX frob TO barm RIGIDLY⊃;
size ← 10.4 * feet; {扩展为 10.4*12*inches}
grasp (handle);   {扩展为: MOVE barm TO handle;
                           CENTER barm;
                           AFFIX handle TO barm
```

RIGIDLY;}

4.5.9 REQUIRE 语句

REQUIRE 语句允许用户或用户程序与 AL 编译程序通讯. REQUIRE 语句不生成任何代码，它的效果是全局性的，一直要持续到退出请求使用它的程序块之后。为了解除或停止这种效果，需要另一条 REQUIRE 语句或其他某些终止条件. REQUIRE 语句的形式如下：

REQUIRE SOURCE_FILE "〈文件名〉"

指定的文件将一直是未来输入的源文件，等到遇到一个文件结束符时，才读入 REQUIRE 后面的代码。源文件被认为是磁盘文件，除非在其名字前用 "TTY:"，以表明它为电传打字机文件.

一个电传打字机文件不需要命名，但如果有的话，就用指定的文件名和默认的扩展名 TTY 存在一个磁盘文件上.对电传打字机输入的句法分析工作在每次打入回车后开始. 打入一个〈control〉〈meta〉〈linefeed〉使文件关闭。目前的操作系统一次只允许打开一个电传打字机文件。

文件名可以有多种形式：

"NAME"

"NAME. EXT"

"NAME [P, PN]"

"NAME. EXT [P, PN]"

其中 P 和 PN 分别表示某项工作的命名和程序编制者的名字。

REQUIRE MESSAGE "〈信息〉"

双引号内出现的所有信息都将在用户终端上打印出来。

REQUIRE ERROR_MODES"〈模式标志〉"

在程序编译期间，当 AL 句法分析程序要求用户对错误做出反应时，可以用 REQUIRE ERROR_MODES 语句设置某些标志，以便

对错误的标准处理方式预先定义。包括在引号内的有关字母用来设定标志，代码字母及其前面的减号用来清除标志。可使用的标志如下：

L——如果有错误的话就记入一个带有扩展名 LOG 的文件。

A——在打印出每个错误信息之后，编译将自动地继续进行。

M——系统给用户只提示可修改的错误。

F——对赋值语句、条件监视等不作严格的量纲核查，对无量纲的变量将根据它们出现的上下文强行赋予量纲。只有在量纲的使用不一致时才产生错误信息。

REQUIRE COMPILER-SWITCHES "〈编译开关〉"

所有用在命令行中的开关(见第V部分)可在此给出。这是规定命令行中开关的另一种办法。引号内只许有字母(不许有空格)。

4.5.10 调试工具：NOTE 和 DUMP

为了便于跟踪 AL 编译程序所报告的错误，可以使用下面两条语句。它们只有编译时起作用，而且没有为它们生成代码。

NOTE 语句的结果是将在编译期间输出某些信息。NOTE1 将在世界模拟阶段输出信息，NOTE2 将在代码发送和轨迹计算阶段输出信息，而 NOTE 则在这两个阶段输出信息。

句法如下：

NOTE ("信息")

NOTE1 ("信息")

NOTE2 ("信息")

在程序中遇到 DUMP 命令的地方，将打印希望变量的规划值。DUMP 命令如下：

DUMP 〈id list〉

其中〈id list〉是由逗号分隔的标识符表。

4.6 环境模拟

前面曾提到过(4.4.1 节),目前的 AL 系统是在编译时完成轨迹的计算.为了实现这一点,编译程序必须有一个关于"环境"的模型.它包括机械手的位置,描述运动的变量的期望值以及关于"附着"结构的知识.这些值在程序的不同地方是不同的.因此,编译程序必须对程序的执行进行一次仿真模拟,以便获得轨迹计算程序所需要的信息.在编译期间,一个 AL 程序要经过几个阶段:第一个阶段是对程序进行分析并建立其内部表示,然后对程序进行仿真模拟(在环境模拟阶段),最后计算轨迹并发送代码.由于在编译时间无法使用传感器信息,因此在环境模拟时会产生某些问题.在循环和并行过程的模拟中还采用了很多折衷方案.

在世界模拟时,每条语句都有一个与之相关的"输入环境"和一个"输出环境".."输入环境"给出机械手的当前位置、每个变量的值以及在该语句执行前必须立即说明的"附着"结构,而"输出环境"则反映该语句的执行效果.大多数语句(如赋值语句、附着语句)很容易模拟.与条件监视无关的运动语句也容易处理.避免碰撞问题现在尚未处理.然而,如果存在条件监视,那么环境模拟程序就无法决定条件是否已被触发,因而也就无法判断其对运动的影响.例如,如果机械手正拿着一个盒子,我们希望把盒子放到桌子上,程序可以是:

```
MOVE box TO table — 4 * zhat * inches
    ON FORCE (zhat) ≥ 8 * oz DO STOP;
```

我们希望当盒子碰到桌子时运动马上停止,但是,即使知道这一点,模拟程序也不能决定条件监视触发时盒子的位置.鉴于这种情况,目前的系统忽略了与运动有关的任何条件监视的影响,并假设运动正常停止.由于我们的目的是规划轨迹,这种方法带来的

不精确性一般并不重要。当事先知道条件监视的影响很关键时，应使用规划时间赋值语句来告诉环境模拟程序。

给环境模拟程序带来麻烦的另一类语句是条件从句。一种很"慷慨"的方法认为，在两个条件分枝的"环境输出"（除了有矛盾的）中有一个为真，那么就认为是真。目前，AL 采取一种比较保守的方法，即只有在不论哪路控制都为真时才按真处理。这种做法的后果可通过下面这条语句来分析：

IF j > 5 THEN j ← 1 ELSE j ← j + 1

在执行完后，变量 j 依赖于哪个条件被执行会有两个不同的值。因此，模拟程序就没有变量 j 的规划值。如果以后的程序需要 j 的值，那么在条件句后面应该跟有一条规划时间赋值语句，以便给 j 赋予一个期望值。CASE 语句也按类似形式处理。

循环也带来一些难题。环境模拟程序将展开循环一次的情况。注意，对循环中的每条 MOVE 语句只计算一条轨迹。如果其中某一步运动的目标呈现差别很大的值时，这就会带来问题。在这种情况下，用户必须在程序中附加一些代码，以便目标值在几个移动语句中进行选择，或者给运动分配更多的时间，以便使机械手有足够的时间去完成最长的运动。

可以设想，COBEGIN 的结构对环境模拟程序也是难以处理的。AL 处理并行问题的方法是综合考虑各个分枝引起的所有的变化，排除掉诸如同一变量由不同的分枝赋予不同的值这样明显的不一致情况，然后把所有这些事实放在一起，从而形成 COBEGIN 块的"环境世界"。

过程也带来一些问题。一个过程的本体只在定义过程的块的入口处被模拟一次。必须给所有的与过程定义在同一块内的形式参数或变量进行规划时间赋值，以便在过程被模拟时其值可以使用。还应注意的是，在循环中使用 MOVE 语句出现的问题，对过程也同样存在，即在编译时只计算一条轨迹。如果在运行时实际

运动与"规划的"运动差距甚大，那么将导致失败。过程的调用是被忽略的，它们对环境模拟没有影响。在环境模拟中，标明类型的过程回送一个零值 (nilvect, nilrot, ···).

环境模拟程序要对程序中第一条语句的"输入环境"初始化，使得机械手都位于其停放位置，手指离开两英寸，运动的速度因子为 1. 如果最后一条语句的"输出环境"中机械手不是重新位于其停放位置，那么环境模拟程序将发出一个警告信息。

V. AL 的使用

这一部分将介绍在没有错误的情况下，编译及执行一个 AL 程序的步骤。

5.1 用户程序的编译

下面介绍编译产生 PDP-11 的二进制装入模块的步骤：

1. 建立一个包含有用户程序在内的文件 FOO. AL，FOO 可以是任意的命名。

2. 使作业进入监控状态并键入 "COMPIL E FOO".

2a. 接受象 COMPILE 这样命令的系统程序 SNAIL 将给出信息：

```
Swapping to SYS: AL. DMP
```

然后启动分析程序 parser，parser 将给出：

```
AL:FOO
```

当 parser 处理到文件中一页的边界时，将得出"1"或任何将要开始读的页的页数。

2b. 当 parser 分析结束后，它将调用 AL 编译程序 AL compiler，它显示“ALC”.

2c. 当编译程序完成了它的世间模拟、轨迹计算以及代码生成后，它将调用 PDP-11 的交叉汇编程序 PALX. 在用户的终端上显示出“PA LX n”，其中 n 是 PALX 编译程序的版本号.

2d. PALX 汇编程序调用 ALSOAP，它将通过删除在编译 AL 程序过程中产生的中间文件来清除用户空间. 所删除的文件为具有扩展名 .ALP、 .ALV、.ALT 及 .SEX 的文件.

2e. 作业回到监控状态.

如果有拼写错误，那么 SNAIL 将给出:

File not found: FOO

其中 FOO 为拼错的名字.

5.1.1 有开关的编译

如果需要，可以将选择开关加入圆括号内，使编译具有选择的功能，例如：“COMPILE FOO. AL (KS)”.不同的选择开关的功能如下:

K	保留中间文件 (.ALP，.ALV，.ALT)
S	禁止 .SEX 文件的删除
L	生成一个 PALX 汇编清单

5.2 装入及执行 AL 程序

当用户程序“FOO. AL”已经顺利通过了 ALSOAP 后，就可以准备在 PDP-11 上执行这一程序.

1. 安排好机械手制动控制盒及紧急保险开关键的位置.在 AL 程序执行的任何时间内，用户的手要保持在紧急保险开关键的上方，而且准备着当未预料到的或破坏性的事件发生时，立即按下此

键．扯开围绕着桌子的黄色电线便可使机械手断电．这一措施也可以在紧急情况下使用．请注意放置好电线，以免发生意外事故．

2．键入“DO AL［AL，HE］”，接着按一个回车键，这样，一系列指令（将在5.3中叙述）就被初始化．当看到

```
f =
```

时，键入用户程序名“FOO”后按一个回车．这时将会看到有若干行信息打印出来．其中最后一行是

```
DDT STARTED AT 130000
```

3．在VT05终端屏幕上可看到一个星号“*”和一个闪动的光标．用户确认一下手指是否按在制动键上后，键入

```
*START ⟨alt⟩ ⟨alt⟩ G
```

后，程序开始执行．注意，只有⟨alt⟩G也可起同样作用．AL将在VT05上打印出

```
AL RUNTIME SYSTEM
```

在机械手每次运动之前，VT05将发出嘟嘟声．信息及数据将适时地打印出来．在程序执行结束后，将显示下列信息：

```
ALL DONE NOW. SEE YOU AROUND!
NO ACTIVE PROCESSES LEFT. YOU'RE IN DDT.
*
```

在VT05上键入“⟨alt⟩G”便可从头重新执行这一程序．

5.3 完整的运行步骤

下面介绍在二进制文件生成后，装入及执行一个AL程序的操作步骤，相应于用户键入DO AL［AL，HE］时，有错误的情况．

1．键入“A ELF”，把ELF（PDP-11的接口）指派给用户作业，这样在执行用户程序时，其他的作业就不会占用该接口了．

2. 键入 "R 11TTY"，执行装入程序，它将使用户程序被装入 PDP-11。11TTY 的响应是

```
CORE SIZE = 28K
VERSION USING 〈device〉
TYPE ? FOR HELP
*
```

这里的〈device〉或是 VT05 或是 TERMINAL。星号 * 是 11TTY 提示用户输入的标志。

两个设备之间可以变换，方法是，在星号后面跟着键入 "V"，11TTY 将自动添补命令行的其他内容，并且要求下一条命令：

```
*VERSION USING 〈other device〉
*
```

另一种变换设备的方法是使用扩展命令，即键入 "A"，然后接着键入 "VT05" 或 "TERM"，以表明所期望的设备。

```
*AN EXTENDED COMMAND VT05
*
```

选择设备 VT05 是比较合理的，因为这样在运行开始后，可以完全不依赖 PDP-10（如果用户正在运行 ALAID，那么就需要这样做）。

3. 键入 "Z" 来清除内存，紧跟着的是内存容量，通常为 500000，然后键入回车键以确认这条命令。11TTY 将作出响应：

```
*ZERO CORE [CONFIRM] 500000 〈cr〉
*
```

4. 键入 "G"，以得到内存映像二进制文件，再跟着键入文件名 AL [AL, HE] 及回车，这样，AL 解释程序及运行系统就被装入。

```
*GET SAV FILE-AL [AL, HE] 〈cr〉
*
```

5. 键入“O”，以便可以进行内存覆盖，再跟着键入文件名及回车，用户的 AL 程序的二进制文件便可装入。

```
*OVERLAY BIN FILE-FOO 〈cr〉
*
```

6. 顺序键入“S”、“D”和回车，使程序起动。

```
*START AT (1000) (D FOR DDT)-D 〈cr〉
  DDT STARTED AT 130000
*
```

7. 检查是否已经作好随时按下制动键的准备，在 VT05 终端上键入：

```
*START 〈alt〉 〈alt〉G
```

AL 将在 VT05 上打印出：

```
AL RUNTIME SYSTEM
```

此后，任何输入输出信息都将在 VT05 上显示出来。

当程序执行完毕后，VT05 上显示下列信息：

```
ALL DONE NOW. SEE YOU AROUND!
NO ACTIVE PROCESSES LEFT. YOU'RE IN DDT.
*
```

到此，在 VT05 上键入“〈alt〉G”将可以从头开始重新执行这一程序。

VI. POINTY

6.1 POINTY 的描述

6.1.1 引言

AL 中坐标系 FRAME 这一数据结构及其"附着"，可以形成一个物体的模型．现在，读者应该对这一概念有一个比较清楚的了解了．形成坐标系的附着，尤其对于不同姿态的坐标系的相互附着，是一件非常重要的任务．对于实际的物体，用户需要测量它的距离、角度和位置，并且通过对坐标系作某种旋转变换，才能确定坐标系之间的关系．除了最简单的情况外，这样一个过程是冗长而且容易出错的．

给定一个物体，我们需要一种产生"附着"结构的方法．在理想的情况下，我们可以将实际物体或其设计图由视觉等方式输入给计算机，使系统建立起附着结构．然而，我们所关心的物体特征取决于装配过程的性质，也许与部件的形状无关．生成附着结构的一种方法就是依靠人的帮助．操作人员指出感兴趣的物体特征，而包括各种特征间直接联系的各种记录则由系统来管理．

交互式地为 AL 程序建立世间模型的描述，可利用 POINTY 来实现．POINTY 是在 SAIL 开发并实现的系统．它使我们能够通过读机械手的位置来定义物体上感兴趣的点．

用户用手移动机械手并读其位置，就可记录物体上的各个点．然后，用户告知系统各点间的位置关系．这样，一旦知道一个点的方位和姿态，所需要的物体上所有的特征都可知道，从而物体的模

型就生成了.

POINTY 还具有执行某些运动语句的能力. 这使得我们可以先对这些运动语句进行试验后再写入 AL 程序,可以允许重新确定机械手的姿态,可以允许机械手以同一姿态作不同的运动.

6.1.2 用机械手定点

6.1.2.1 坐标系的隐含说明 Scheinman 机械手有六个自由度,它可达到任意的位置和姿态. 坐标系也有六个自由度,三个平移分量,三个旋转分量. 如果用机械手的一组位置及姿态值来显式地标定一个坐标系,那么将没有自由度的赢余. 缺乏赢余自由度将使机械手很难准确定位,因为所有的运动,无论"精"、"粗",都需要相同数量的动作;另外,那样还会限制对障碍的避绕.

用手把机械手引导到一个便于抓握的位置,使它把一个部件从夹具或托板中挑选出来,做到这点并不困难. 但是,如果想用手引导,使机械手具有较好的姿态,从而当机械手要将一个部件移开时,不会受到阻力,这却相当困难. 因为角度上的任何微小偏差,都将因为 AL 程序中机械手的力臂而被扩大,所以,要求角度非常精确,将使得用角度来定义世界模型变得很困难.

为了避免这一困难,我们采用多点标定的方法,隐式地定义每个坐标系往往是比较方便的. 例如,可以用第一个点定义坐标系的原点,第二个点定义坐标系的一个轴,第三个点定义坐标系的一个平面. 这样,每个点仅需定义其位置,而无需准确的姿态.

当姿态与一个已知的坐标系(例如基准坐标系或其他已定义过的坐标系)的姿态平行时,我们可以简化标定操作. 这时,姿态坐标可由已知的坐标系定义,而位置则通过一次标定得到.

6.1.2.2 探针 机械手的末端必须配有一个较尖的探针作为精确的量测工具. 探针的形状必须适当,以使其能够达到一些难以到达的位置,例如螺丝孔的内部、盒子的内部等. 为了能使探针

具有适应于各种无法预料的障碍，探针必须设计得能由用户弯成任意形状．这样一个特殊的装置，我们把它叫作柔性探针．

无论何时，只要用户需要都可以把柔性探针弯成任意的新的形状，以使其能够适应于下一步操作．探针形状改变之后，用户必须重新校准新的末端位置．这可以通过使探针指向一个标准的基准点来实现．由基准点坐标系及手爪的坐标系，系统便可推知由手爪坐标系到柔性探针坐标系的变换．如果不用柔性探针，也可以有别的方法：准备一套常用形状的刚性探针，而且非常便于装卸．无论使用哪种方法，这些探针都必须足够坚硬，以免在工作中发生变形．

6.1.3 系统结构

POINTY 在两台计算机上运行——在 PDP-10 上做算术运算及其他运算，而 PDP-11 则负责读机械手的位置并控制机械手的动作．

PDP-10 由如下模块组成：附着编辑程序，算术运算子程序，机械手接口，文件输入/输出设备，显示子程序，行命令扫描程序(句法分析程序)以及用户接口．

AL 语句及表达式的一个子集被 POINTY 所接受．行命令扫描程序(句法分析程序)利用星号“*”提示用户输入新的命令语句．若等待续写语句或表达式，则提示“＊＊＊＊＞＞＞”．对当前输入行的分析，在用户键入回车键后开始．输入行的第一个记号与内部记号表进行比较．若能匹配上，就根据这一记号进行一系列固定的分析．若不匹配，句法分析程序将通过检查这一记号左端是否是左箭头“←”，以此来判断它是否是一个赋值语句左端的变量．当句法分析程序遇到了不认识的符号，或当用户试图访问没有赋值的变量时，用户接口将通过给出错误信息来与用户进行通讯．

每当反映附着编辑程序、算术运算子程序以及机械手接口当前状态的数据发生变化时，显示子程序将修改用户终端上的数据。显示子程序还可以在必要的时候关闭所有显示。

附着编辑程序包括坐标系的生成及其相互间关系的描述这两部分。

当句法分析程序识别出一个表达式或赋值语句后，它便调用算术运算子程序，其中包含对 SCALARS、VECTORS、ROTS、FRAMES 以及 TRANSES 的一组运算。

文件输入/输出模块中的子程序用于往 AL 的变量说明文件中存入变量和数据值，或者根据这一文件来恢复变量和数据值。这些 AL 变量的说明和赋值可为 AL 程序直接使用；如果用户需要对一个已知的世间状态重新初始化时，也可由 POINTY 读入变量说明和赋值。

机械手接口提供了 POINTY 与机械手之间的通讯。它使得我们可以与 PDP-11 上的运行系统进行通讯。机械手接口有两个主要功能：把从机械手的关节和腕上读取的信息从 PDP-11 传给 PDP-10；把运动命令从 PDP-10 传给 PDP-11。

POINTY 系统在 PDP-11 的部分主要由 AL 机械手代码和一般程序组成。这段程序的功能是：读入机械手的位置；不断地在 VT05 上显示这些位置；当 PDP-10 部分需要机械手运动时，执行这些运动；打印出机械手运动企图的结果。

6.2 执行 POINTY

6.2.1 执行命令的缩写形式及显示形式

执行 POINTY 的最简单的方法是键入命令（6.2.2 节将给出完整的命令表）：

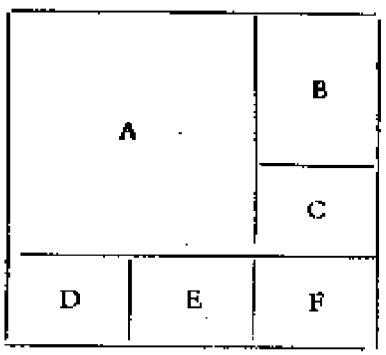
```
DO POINTY [PNT, HE]
```

后跟回车键．这一命令首先将 POINTY 的运行系统装入 PDP-11 并启动，不断地读入机械手的关节位置而且显示于 VT05 的屏幕．无论什么时候，如果屏幕上的数据修正停止了，那么只要在 VT05 的键盘键入〈alt〉G 便可重新启动 POINTY．然后，PDP-10 部分将装入并启动．POINTY 将在屏幕上显示信息并且随着越来越多的指令的执行，不断地修改显示．下图为在执行完几条指令后的显示状态．

```
STATION (NILROT, NILVECT)                    BHAND   1.20
  -BASE (NILROT, (15.0, 12.0, .500))         YHAND   .000
  -HANDLE (NILROT, (35.0, 32.0, .500))       OFFSET  3.00
    *HANDLE_TOP((Y,180.)*(Z,90.0),(2.10,.340,5.05))
    *HANDLE_REF (NILROT, (1.10, 2.30, .100))
  +YARM (NILROT, NILVECT)
  +BARM ((Y, 180.)*(Z, .002), (43.5, 56.8, 10.9))
    *BGRASP ((Y, 180.)*(Z, -180.), NILVECT)
                                               MOVE
                                               BARM

*D DECLAR .AL   NILROT(Z, .000)          NILVECT (.000,.000,.000)
                RT_AP(Y,180.)*(Z,-90.0)  APPR (3.00,.000,.000)
SAVED. TTY
```

上面的框图以后将用如下字母代替：



A：附着树，坐标系和变换
B：标量
C：默认的运动
D：输出文件
E：旋转
F：向量

首先，POINTY 想知道在何处存储终端对话信息，它的提问

是

file for TTY output = ⟨file name⟩⟨cr⟩ {文件名可省略}

如果给出文件名，在终端对话期间的 POINTY 命令表将存入这一文件，否则，终端输出将不存储．文件名显示在框图D的底部．注意，对大多数指令来说，回车键是这些指令的启动字符；还有，AL 的两种注释形式在 POINTY 中都是合法的．

然后，POINTY 将准备接受命令，并且以星号＊提示，如同每当等待一个新命令输入时，都以星号＊作为提示一样．单独一条指令由回车键或分号加上回车键来作为结束，然后 POINTY 将开始执行这一指令．同一行中的多条命令必须用分号隔开，而且在最后跟一个回车键．一遇到回车键，POINTY 将开始执行有意义的指令，对于意义不完整的指令，POINTY 将用"＊＊＊＊>>>"作为提示等待用户进一步输入以及下一个回车键．例如：

```
a1 ← 3 ⟨cr⟩          {POINTY 将把 3 赋给变量 a1}
a1 ← 3 + ⟨cr⟩        {POINTY 将等待进一步输入}
```

在显示的起始状态，框图A显示出 POINTY 已知的三个坐标系：station、barm 和 yarm．由于黄机械手目前没有联上，所以 yarm 的坐标全为零．框图B中的 bhand 和 yhand 分别表示蓝机械手和黄机械手手爪的开度．在框图E和F中，包含有预先定义过的旋转 nilrot 和向量 nilvect．

6.2.2 运行 POINTY 的完整指令

下面是在 PDP-11 和 PDP-10 上装入 POINTY 运行系统的完整操作步骤，它相应于用户键入 DO POINTY [PNT, HE] 后，存在有错误的情况．

1. 键入 "A ELF"，以便将 ELF (PDP-11 接口) 赋给用户使用，从而使得在执行用户程序时，其他作业不会将其占用．

2. 键入 "R 11TTY" 以执行用户程序，这将使用户程序装入

PDP-11，11TTY 的响应是：

CORE SIZE = 28K

VERSION USING ⟨device⟩

TYPE ? FOR HELP

*

这里的⟨device⟩是指 VT05 或 TERMINAL．星号＊是 11TTY 提示用户输入的标志．

两个设备之间可以变换，方法是在星号＊后面跟着键入“V”，11TTY 将自动添补命令行的其他内容，并且要求询问下一条命令：

＊VERSION USING ⟨other device⟩

*

选择设备 VT05 是比较合理的，因为这样在运行开始后，关节角度及机械手的其他信息将在 VT05 上显示出来．

3．键入“Z”来清除内存，后跟回车以确认该条命令．11TTY 的响应是

＊ZERO CORE [CONFIRM] 500000 ⟨cr⟩

*

4．键入“G”以得到内存映像二进制文件，再跟着键入文件名 POINTY [PNT，HE]，这样就可以装入 POINTY 运行系统．

＊GET SAV FILE-POINTY [PNT，HE] ⟨cr⟩

*

5．顺序键入“S”、“D”及回车键便可以启动程序．

＊START AT (462452) (D FOR DDT)-D ⟨cr⟩

DDT STARTED AT 130000

*

6．检查是否作好随时按下制动键的准备，然后在 VT05 上键入：

*⟨alt⟩ G

用户将在 VT05 上看到不断地扫描，显示关节角和其他信息的修改情况.

7. 键入"X"退出 11TTY

* X

8. 现在处于监控状态，为了在 PDP-10 上运行 POINTY，键入下面的命令：

R POINTY ⟨cr⟩

POINTY 将开始运行并且将给出如上一节所述的显示信息.

6.3 POINTY 的指令

POINTY 的指令可分成如下几类.

6.3.1 赋值语句

同 AL 中一样，POINTY 中的赋值语句也是计算式子右边表达式的值，并把它赋于左边的变量. 若左边的变量没有定义，赋值语句隐含地把这一变量定义成表达式计算结果的类型. 若一个变量已经定义过，而且它的类型与右边表达式计算结果的类型不同，则给出错误信息. 例如：

```
s4 → 2*3;          {s4 说明成常量，显示于框 B 中}
v4 ← zhat + yhat;  {v4 说明成向量，显示于框 F 中}
r5 ← nilrot;       {r5 说明成旋转，显示于框 E 中}
f5 ← bpark;        {f5 说明成坐标系，显示于框 A 中}
```

6.3.2 说明语句

SCALAR、VECTOR、ROT、FRAME 以及 TRANS 数据类型的显式说明与 AL 中的相同.

POINTY 可识别下列 AL 中预先说明过的变量和常量——常量：bhand、yhand；向量：nilvect、xhat、yhat、zhat；旋转：nilrot；坐标系：station、barm、yarm、bpark、ypark；变换：niltrans；单位常量：inch、inches、deg、degree、degrees。这里的 barm、yarm 或附着于一个机械手的坐标系经常出现在表达式中，所使用的是根据机械手当前位置计算出来的当前值。

在说明显式数据类型方面，POINTY 比 AL 有更大的灵活性。特别是对于不同的数据类型，参数的数据类型和数量并不相同（FRAME 与 TRANS 是一样的）；因此，它们的说明可不加限制词。如果 POINTY 无法分清一个说明是坐标系还是变换时，就先假设为变换。当与之相联系的变量用于附加语句时，再将它转换成坐标系类型。注意，显示时为了节省空间，保留字 VECTOR、ROT、TRANS、FRAME 都被略去，而且 xhat、yhat 及 zhat 被简写为 x、y、z，这样仍然是很清楚的。

向量：　VECTOR (〈标量〉,〈标量〉,〈标量〉)或(〈标量〉,〈标量〉,〈标量〉)

旋转：　ROT (〈向量〉,〈标量〉)或(〈向量〉,〈标量〉)

坐标系：　FRAME (〈旋转〉,〈向量〉)或(〈旋转〉,〈向量〉)

变换：　TRANS (〈旋转〉,〈向量〉)或(〈旋转〉,〈向量〉)

例：

合法标量：　a, a_b, +.01, 3.001

合法向量：　VECTOR (0, +5, −0.3)
(a_b, a, (.05 − a))
(1, +5, .01)

合法旋转：　ROT (xhat, 180)
(zhat, 90)
(yhat, a)

合法坐标系：　FRAME (r1 * r2, vec1)

FRAME (ROT (yhat, 90), (1, 1, 1))
(r1, VECTOR (2.3, a, —.3))

合法变换: TRANS (r1 * r2, vec1)
TRANS (ROT (yhat, 90), (1, 1, 1))
(r1, VECTOR (2.3, a, —.3))

6.3.3 删除语句

变量可由删除语句删掉. 若所删变量是坐标系变量, 则所有以它为根的子树也被删掉. 例如:

```
DELETE    s1, s2, s3, v1, v2, f1, f2;
QDELETE   s1, s2, s3, v1, v2, f1, f2;
DELETE    ALL;
QDELETE   ALL;
```

使用 ALL 将删掉用户说明的所有变量. 如果没有给定参数, 那么假定为 ALL, 但是在 DELETE 情况下, 用户将被要求确认是否确实想删掉所有的变量. 相应的变量将在相应的显示框图中消失. 如果 POINTY 遇到一个用户未曾说明的变量, POINTY 将假设这是一个拼写错误, 并要求用户进行修正. 如果用户不希望 POINTY 告诉他变量不存在, 那么应该使用 QDELETE 来代替 DELETE. 当从文件中读来的宏指令或标识符与已经说明的符号表中的相同时, QDELETE 是非常有用的.

6.3.4 函数及宏指令

POINTY 中可以使用宏指令, 其句法与 AL 中的相同. 例如可作这样的定义:

```
DEFINE  ARM = ⊂barm⊃;
DEFINE  V1 (A, B, C) = ⊂VECTOR (A, B, C)⊃;
```

注意宏定义体两边的定义符. 在宏定义中, 参数名必须是至

今尚未说明过的变量名．这样，那些名字用于其他目的时将不会影响宏定义．

所有宏定义体可以合法存在的地方，都可以用宏指令名来替代．下面的例子是合法的：

```
MOVE ARM TO BPARK;
VECT1 ← V1 (0, 0, 1);
VECT2 ← V1 (⊂2*3⊃, 1, 4);
```

当被替代的参数不是一个单一的记号，而是一个表达式或一系列记号时，注意使用定义符．

当一个复杂的表达式需要多次使用时，可定义成函数．虽然一个复杂的表达式可以作为宏指令来存储，但以函数的形式为好，因为这种内部存储形式使得 POINTY 能够检查表达式的合法性以及参数运算的正确性．函数定义的句法如下：

```
〈类型〉 FUNCTION 〈函数名〉 ═
        〈合法的 POINTY 表达式〉
〈类型〉 FUNCTION 〈函数名〉(〈类型〉〈参数表〉) ═
        〈合法的 POINTY 表达式〉
〈类型〉 FUNCTION 〈函数名〉(〈类型〉〈参数表〉;
                          〈类型〉〈参数表〉;…
                          〈类型〉〈参数表〉) ═
        〈合法的 POINTY 表达式〉
```

下面是使用函数的例子：

```
SCALAR FUNCTION F1 = 2*3;
SCALAR FUNCTION SQUARE (SCALAR X1) ═
        X1*X1;
SCALAR FUNCTION DOTPROD (VECTOR V1,
        V2) = V1.V2;
VECTOR FUNCTION CHANGEXCOMP (VECTOR V1;
```

SCALAR S1) = VECTOR (S1, V1. YHAT, V1. ZHAT);

函数与 AL 中标明类型的过程相似，只是其函数体仅由回送值计算表达式组成. 例如，与上面例举的函数等价的 AL 过程如下：

```
SCALAR PROCEDURE F1;
    RETURN (2*3);
SCALAR PROCEDURE SQUARE (SCALAR X1);
    RETURN (X1*X1);
SCALAR PROCEDURE DOTPROD (VECTOR V1, V2);
    RETURN (V1. V2);
VECTOR PROCEDURE CHANGEXCOMP(VECTOR V1;
SCALAR S1);
    RETURN (VECTOR (S1, V1. YHAT, V1. ZHAT);
```

函数可引用用户定义的标识符，但当这一标识符被删掉后，函数将成为非法的；当该标识符又重新定义后，函数又将成为合法的.

全程变量的当前值，或由全程变量组成的表达式，或一个算术运算表达式，可通过在函数定义体中使用 EVAL 函数而得到其结果.

考虑下列情况：

```
V1 ← VECTOR(2, 0, 0)
FUNCTION F1 = 2*V1
FUNCTION F2 = EVAL (2*V1)
V1 ← VECTOR (0, 0, 2)
```

如果现在调用 F1，那么得到 VECTOR (0, 0, 4)，而调用 F2 则得到 VECTOR (4, 0, 0).

6.3.5 表达式

POINTY 可接受除布尔表达式之外的 AL 句法分析程序所

能处理的所有其他代数表达式。POINTY 不作量纲相容性检查。下面总结了所有合法的运算，其意义与 AL 中的相同，不再详述。

标量：s + s, s − s, s*s, s/s, s↑s, v.v, |s|, |v|, |r|, s MAX s, s MIN s, s DIV s, s MOD s, INT(s), SIN(s), COS(s), TAN(s), SQRT(s), ASIN(s), ACOS(s), ATAN2(s, s), LOG(s), EXP(s)

向量：s*v, v*s, v/s, v + v, v − v, v*v, r*v, POS(f), f*v, v WRT f, UNIT(v), AXIS(r), t*v, v REL f ≡ f*v {v 是坐标系 f 中的一个向量。该式给出了基坐标系中该向量的等价向量}

旋转：ORIENT(f), r*r

坐标系：f + v, f − v, t*f, f*t

f1 REL f2 ≡ f2*f1 {f1 是坐标系 f2 中的一个坐标系。该式给出其在基坐标系中的表示}

CONSTRUCT (v, v, v), CONSTRUCT (f, f, f) {利用三个坐标系的位置部分构造一个坐标系，或者利用三个向量来构造：第一个定义原点的位置，第二个定义 X 轴，第三个定义 X-Y 平面。这样便可以不必把机械手精确地引导到某个期望的角度}

↑f, ↓f, $f, αf {将给出与 f 位置相同但姿态不同的坐标系。↑给出姿态的垂直分量，即若 ORIENT(f) = rot(zhat, a)*rot(yhat, b)*rot(zhat, c)，则 ↑f = FRAME (rot(zhat, c), POS(f)); ↓ 给出 bpark 的姿态，即 rot (yhat, 180); $给出基准姿态，即 nilrot; α 给出机械手处于停放位置时 bgrasp 的姿态，即 rot (zhat, 180)}

变换：　　f→f，t*t，INV(t)

6.3.6 附着树的操作

6.3.6.1 AFFIX 和 UNFIX　附着指令与 AL 中的类似，但允许将刚性附着（RIGIDLY）及非刚性附着（NONRIGIDLY），分别简记为“*”和“+”：

AFFIX f1 TO f2;
AFFIX f1 TO f2 RIGIDLY;
AFFIX f1 TO f2 NONRIGIDLY;
AFFIX f1 TO f2 *;
AFFIX f1 TO f2 +;

这里的坐标系 f1 附着于 f2。除非另有说明，否则附着都将是刚性附着。每个新定义的坐标系都将相对于基准坐标系显示（在显示中以“—”指示）。附着树将按其结构显示于框格 A 中。两个坐标系还可依其间的变换关系进行附着，方法如下：

AFFIX〈标识符〉TO f3 AT（〈旋转〉,〈向量〉）;
AFFIX〈标识符〉TO f3 AT（〈旋转〉,〈向量〉）RIGIDLY;
AFFIX〈标识符〉TO f3 AT（〈旋转〉,〈向量〉）NONRIGIDLY;

若〈标识符〉不是一个坐标系，则在附着前，它将作为一个新的坐标系而被定义。这一指令主要用于读入前一个 POINTY 对话期间产生的 AL 指令。

附着解除指令与 AL 中的相同，或可简写成下面例子中的形式。frame_1 被解除与 frame_2 的附着，从而独立地附着于基准坐标系 station。

UNFIX frame_1;
UNFIX frame_1 FROM frame_2;

6.3.6.2 COPY　COPY 指令用于将一个坐标系及其子树的

复本附着于另一个坐标系．句法如下：

COPY 〈frame_1〉 INTO 〈frame_2〉;

POINTY 将〈frame_2〉的前半部分(在有下横线的情况下为横线前的部分，或在名字少于5个字符时，取整个名字，否则取前三个字符)或用户定义的前缀，作为〈frame_1〉子树中诸坐标系的前缀，具有下横线的〈frame_1〉的子树中的任何坐标系，将使得横线后部分作为由〈frame_2〉确定的前缀的后缀．若这一过程产生了重名，POINTY 允许用户定义一个新名字．

如果只将一个坐标系的子树拷贝到另一子树，可利用下列语句：

COPY SUBTREE (〈frame_1〉) INTO 〈frame_2〉;

下面给出一个将 COPY 命令用于两个以上 base 和 handle 为根的附着树的例子：

```
station (nilrot, nilvect)
  —base (nilrot, (15.0, 12.0, .500))
  —handle (nilrot, (35.0, 32.0, .500))
    * handle_top ((Y, 180.)*(Z, 90.0), (2.10, .340,
      5.05))
    * handle_ref (nilrot, (1.10,2.30, .100))
```

指令： COPY SUBTREE (handle) INTO base 产生如下结果：

```
station (nilrot, nilvect)
  —base (nilrot, (15.0, 12.0, .500))
    * base_top ((Y, 180.)*(Z, 90.0), (2.10, .340, 5.05))
    * base_ref (nilrot, 1.10, 2.30, .100))
  —handle (nilrot, (35.0, 32.0, .500))
    * handle_top ((Y, 180.)*(Z, 90.0), (2.10, .340,
      5.05))
      * handle_ref (nilrot, (1.10, 2.30, .100))
```

新坐标系的名字是按照如前所述的名字生成的，作为接受一方的坐标系 base，将作为新名字的前缀，而且它是取自横线的前半部分．坐标系 handle 的子树被复制成 base 的子树．

指令 COPY handle INTO base 将产生如下结果：

```
station (nilrot, nilvect)
  —base (nilrot, (15.0, 12.0, .500))
    *base_handle (nilrot, (35.0, 32.0, .500))
      *base_top((Y, 180.)*(Z, 90.0), (2.10, .340,
        5.05))
      *base_ref (nilrot, (1.10, 2.30, .100))
  —handle (nilrot, (35.0, 32.0, .500))
    *handle_top((Y, 180.)*(Z, 90.0), (2.10, .340,
      5.05))
    *handle_ref (nilrot, (1.10, 2.30, .100))
```

handle 及其子树都被复制为 base 的子树．利用前述规则产生了新的名字：base_handle，因为 handle 没有下横线．

6.3.7 机械手交互作用命令

PDP-11 接口用于与机械手的交互通讯．机械手的当前位置要送回 PDP-10，而指挥机械手运动的指令要送入 PDP-11．当机械手的这种交互作用发生时，一个联锁软件将阻止下一条 POINTY 指令的执行，直到读完机械手的位置，或运动已经结束，或已被查明运动失败．

6.3.7.1 机械手读指令 每当一个表达式计算完毕，机械手的位置将被读入．一旦直接涉及到机械手，就需要知道其当前位置，而且它的当前位置还要用于计算任何附着坐标系的值．用户不必对坐标系 barm 和 yarm 赋值，

两个特殊的赋值语句用于系统初始化。机械手运动到基准点(一个任意的参考点,其位置由下列语句决定),语句如下:

FIDUCIAL ← ⟨arm⟩;

为寻找探针的位置,需把探针握于机械手上且指向基准点。语句如下:

POINTER ← FIDUCIAL;

然后利用 AFFIX 语句将机械手刚性地附着于相应的机械手。

指令 FCONSTRUCT 用于由三个输入点构造一个坐标系的情况,如下例所示:

FCONSTRUCT f;

这里的 f 是未定义的标识符。 POINTY 询问使用哪一个设备 (barm, yarm 或 pointer),并通过机械手指向三个点来说明三个点的含义。第一个通常为该坐标系的原点,第二个点用于用户指定的一个轴 (xhat, yhat 或 zhat),第三个点在由该轴及用户指定的另一个轴构成的平面上。这三个点的位置将用于计算所期望的坐标系。

6.3.7.2 MOVE 命令 机械手的运动可用绝对、相对两种方式来指定。绝对运动的格式与 AL 中的基本 MOVE 指令类似:

MOVE f1 TO ⟨fr_exp⟩

这是一条一般的运动指令。 f1 (假定它已附着于一个机械手) 向指定的目标运动。目标可通过运动开始时 f1 的位置来确定。

POINTY 目前还无法识别接近点和远离点、力的感受、条件控制或 VIA 点。然而,若不采用一个坐标系表达式,而代之以一系列坐标系表达式,我们可指定一个由若干个轨迹段组成的运动,如下所示:

MOVE f1 TO ⟨fr_exp_1⟩, ⟨fr_exp_2⟩, ···⟨fr_exp_n⟩

这里最多允许有9个表达式。

通过使用 BY 来代替 TO、使用向量代替坐标系表达式，我们可以规定相对运动（在 AL 中无法直接实现，但可通过宏定义 ⊂TO ⊗+⊃ 间接地实现）。如下例所示：

```
MOVE f1 BY 〈向量〉
MOVE f1 BY 〈向量〉 WRT f2
```

这两条指令等价于：

```
MOVE f1 TO ⊗+〈向量〉
MOVE f1 TO ⊗+〈向量〉 WRT f2
```

平行于基准坐标系的 X、Y 或 Z 轴的相对运动可通过下列指令实现：

```
MOVEX f1 BY 〈标量〉;
MOVEY f1 BY 〈标量〉;
MOVEZ f1 BY 〈标量〉;
```

这些指令与下列 AL 指令等价：

```
MOVE f1 TO ⊗+〈标量〉*〈轴〉
```

为了避免书写重复，对于与当前一个已执行的运动指令（显示于框格C中）相似的指令，可只输入指令的后半部分。因而如下的指令是合法的：

```
TO f1;
TO f1 +〈向量〉;
TO f1 +〈向量〉 WRT f2;
BY 〈向量〉
BY 〈向量〉 WRT f2;
BY 〈标量〉;
```

最后一句只能用在沿 xhat、yhat 或 zhat 轴的相对运动语句之后。

由于使机械手回归停放位置的运动经常使用，它可简写成如

下形式：

PARK BARM; {与 MOVE BARM TO BPARK; 相同}

PARK YARM; {与 MOVE YARM TO YPARK; 相同}

PARK; {使两个机械手归位}

6.3.7.3 CENTER 命令 CENTER 的句法及使用都与 AL 中相同。语句

CENTER〈机械手〉; {机械手可省略}

缓缓地合拢机械手的手指，同时移动机械手以便调节到使物体位于两手指中间的位置。如果省略〈机械手〉，将默认上次运动的机械手。

6.3.7.4 OPEN 和 CLOSE 命令 手指运动语句的句法与 AL 中的相似，但是 POINTY 还允许定义相对运动：

OPEN〈手〉TO〈标量〉;

CLOSE〈手〉TO〈标量〉; {绝对地张开或合拢}

OPEN〈手〉BY〈标量〉;

CLOSE〈手〉BY〈标量〉; {相对地张开或合拢}

如果下一条运动语句是张开或合拢同一机械手，那么指令可简写成

TO〈标量〉;

或 BY〈标量〉;

6.3.7.5 DRIVE 命令 POINTY 允许单个关节的运动（在 AL 中是不允许的）。句法如下：

DRIVE BJT (〈关节号〉) TO〈标量〉;

DRIVE YJT (〈关节号〉) TO〈标量〉;

DRIVE BJT (〈关节号〉) BY〈标量〉;

DRIVE YJT (〈关节号〉) BY〈标量〉;

被指定的 barm 或 yarm 的关节将运动到〈标量〉之处或以〈标量〉为运动量。〈关节号〉是代表关节的一个整数。关节 7 是手爪，而

关节 1 到 6 是手臂关节. 驱动关节 7 与 OPEN 或 CLOSE 指令等价. 〈标量〉表示关节 1、2、4、5、6 的角度,以度为单位,而对关节 3 和关节 7 则为长度,以 in 为单位.

同样,在适当的场合也可简写为

TO 〈标量〉;

或 BY 〈标量〉;

6.3.8 显示子程序

标准显示已在 6.2.1 节中讨论过,通过省略象 VECTOR、TRANS 等保留字,简写 XHAT、YHAT、ZHAT 为 X、Y、Z,以及不显示POINTY定义的常量值等方法,可以显示尽可能多的有用信息. 一个可移动的箭头可用来特别地标明一些感兴趣的变量.

显示共有三种方式. POINTY 在初始状态下采用表格显示方式,在这种情况下,标量、向量、变换、坐标系、旋转、未执行的运动语句以及用于存储当前信息的文件都将显示出来. 由于显示屏的空间有限,在这种方式下,宏指令及函数表达式没有显示. 类型显示方式可使用户看到所有说明的数据类型的当前定义:

DISPLAY 〈数据类型〉 {其中〈数据类型〉指 SCALAR, VECTOR, ROT, TRANS, FRAME, MACRO, FUNCTION}

这种显示方式所允许显示的一个类型的变量数目要比标准显示方式多,而且可以显示出宏指令及函数. 当用户希望得知他到目前为止所键入的命令时,最有效的显示方式为非显示方式,用如下命令可请求这种方式:

NODISPLAY

这将消除所有显示,从而仅仅显示出用户键入的一系列命令. 我们可使用重新显示命令,再回到表格显示方式:

REDISPLAY

6.3.9 文件输入/输出

文件的输入/输出，对于为 AL 指令生成附着树是非常必要的，同时，它对于存储当前 POINTY 对话期间产生的结果及利用以前的结果也是非常必要的。

6.3.9.1 存储当前状态——WRITE，CLOSE WRITE 指令用于将 AL 指令所需的变量说明语句、赋值语句及附着语句写到指定的文件内，从而可以定义变量、保存当前的世间状态．格式如下：

```
WRITE;                          {写入上次所写的文件}
WRITE〈标识符〉;                 {写入上次所写的文件}
WRITE INTO〈文件名〉;{将一切内容写入文件〈文件名〉}
WRITE〈标识符〉INTO〈文件名〉;
```

如果省略〈标识符〉，那么所有变量(除 station、fiducial、pointer、yarm barm 及其他一些预先定义过的变量)都将写入文件；否则，仅将所指定的坐标系及以它为根的子树，或这一标识符写入文件．由于一个坐标系是根据它与另一个坐标系之间的变换关系附着其上的，所以所要存储的坐标系还应该独自附着于基准坐标系 station，以得到其绝对位置。

POINTY 允许将信息写入不同的文件中。 如果指名的文件不存在，则在写入用户所指定的信息之前，先建立这一文件，并将当时的日期及时间写入；如果文件存在而且已经打开，在此之前也写完了一些信息且文件尚未关闭，那么当前的信息将加到上次信息的后面；如果文件存在但还没有信息写入过，那么就在新的一页上先写入当时的日期及时间，然后再写入所需的信息。如果一直没有信息输入，或在上次关闭文件后再没有新的信息写入，而且没有指定文件名，信息将写入文件 DECLAR．AL，它将占用用户的当前空间；否则将使用上次所写的文件。

在结束 POINTY 之前，当再无信息写入文件时，文件应该关闭，其句法如下：

```
CLOSE;                {关闭前一个文件}
CLOSE〈文件名〉;{关闭文件〈文件名〉；新的信息将写入
                      下一页}
CLOSE_FILES;{在用户确认后，将包括存储终端输出的
    文件在内的所有已打开的文件关闭。之后，POINTY
    询问存储新的终端输出的新文件名}
```

为了不使这一 CLOSE 指令与合拢机械手手指的指令相混淆，用户应当避免使用文件名 BHAND 或 YHAND.

如果用户没有明确地关闭文件，则在正常退出 POINTY 后，所有文件将被关闭。

6.3.9.2 得到一个给定的世间状态——READ 和 QREAD

READ 和 QREAD 命令用于读入指定的文件，从而将 POINTY 的世界处于一个已知的状态，或恢复成前次操作结束时所保存下来的状态，以使得 POINTY 所产生的文件不但可以作为 AL 的输入文件，而且可以用于存储建立坐标系树结构所必需的指令并给变量赋值。

```
READ;              {从文件 DECLAR.AL 中读入}
READ〈文件名〉;
QREAD;
QREAD〈文件名〉;
```

由于运动指令有可能出现在输入文件中，用户应该谨慎，要看这些命令是否会引起破坏性的运动。READ 命令在读入文件时要将文件打印出来。QREAD 命令执行速度较快，因为它不打印所读入的文件。

6.3.10 其他命令

EDIT〈变量〉;

给行编辑程序装入变量的值，并且允许用户对它进行编辑。当用户希望改变一个变换的旋转部分而不改变其向量部分时，这一命令是非常有用的。它还可用于改变宏命令及函数的定义。

PRINT〈表达式〉;

显示出算术表达式的值。

SPRINT "〈任何文本〉";

在用户终端上显示〈任何文本〉。

PROMPT

等待用户键入 "P" 及回车键后，继续进行别的操作。

RENAME〈变量〉;

改变变量的名字。

EXIT;

退出 POINTY。

键入问号:

?

用户将得到关于可用指令及其意义的信息。 在出现句法错误时，该指令能给出语句正确的句法的信息。

↑,↓,n↑, 或 n↓

使显示箭头上下移动。n 决定箭头的移动距离。

还有一些错误恢复过程可供使用。无论何时，在 POINTY 需要一个已定义的变量或其值而用户却使用了一个未定义的变量后，POINTY 将不断地询问正确的变量名，直到给出可使其继续工作的信息，或由用户键入〈control〉C 来退出疑问循环。

〈ESCAPE〉|

键入终端后，将在当前输入行或语句结束后，结束整个程序的执

行。所有键入信息将被清除掉，若 POINTY 正在读一个文件，那也将停止。以后可接收的输入将来自键盘。

6.4 使用 POINTY 须知

6.4.1 操作步骤

在初始化 POINTY 后，我们建议用户采用下述操作步骤使用 POINTY。

(1) 使机械手抓住基准点并键入指令：

FIDUCIAL ← BARM;

(2) 用机械手紧紧抓住探针，并使探针尖端指向基准点，键入指令：

POINTER ← FIDUCIAL;

(3) 将探针附着于机械手：

AFFIX POINTER TO BARM;

(4) 对于任何物体，我们都希望为参考坐标系找到一个参考点。为了能够迅速地给物体定位，我们希望它的姿态平行于基准坐标系的姿态。因而，探针应当指向参考坐标系，键入下面这一命令：

origin ← $POINTER;

(5) 用于表示其他感兴趣的特征的坐标系，可通过使用 barm，pointer，CONSTRUCT 或 FCONSTRUCT 来得到。设这些新的坐标系分别为 f1，f2，f3；

(6) 这些坐标系应当用下列指令附着于原点：

AFFIX f1 TO origin RIGIDLY;

AFFIX f2 TO origin RIGIDLY;

AFFIX f3 TO origin RIGIDLY;

(7) 在退出 POINTY 前，不要忘记把你感兴趣的坐标系树存

储起来。

6.4.2 须知

(1) 通过对变量进行编辑，我们可将变量的值记录下来。这些值将记录在存储终端输出的文件中。

(2) 若探针已经从机械手上取下，用户不必担心解除附着 UNFIX。对于 POINTY 来说，机械手上仍有一个假设的探针。若用户日后能将探针仍装回原来的位置，则所有的数据仍可使用。为处理特殊的可达位置，可使用柔性探针(见 6.1.2.2 节)，但在每次弯曲后必须重新定义。

(3) 有些位置很容易通过移动机械手到位并且抓握而读入，因此不必使用探针。在这种情况下，将要用到 barm 的值。

(4) 在移动机械手之前，将一个坐标系的当前值存入另一个变量是一种非常好的想法。这样，在日后决定回溯时，所有数据仍能使用。

(5) 虽然物体可以有任意的方向，但当坐标系的轴与基准坐标系的轴平行或正交时，使用 POINTY 是非常容易的。

主要参考文献

[1] John J. Craig, Introduction to Robotics: Mechanics & Control, Addison-Wesley Publishing Company, 1986.

[2] Richard P. Paul, Robot Manipulators: Mathematics, Programming, and Control, The MIT Press, 1981.

[3] 辻三郎、江尻正員，ロボット工学とその応用，三美印刷株式会社，1984.

[4] 若松清司、佐藤知正，知能ロボット，オーム社，1984.

[5] 合田周平、木下源一郎著，王棣棠译，机器人技术，科学出版社，1983.

[6] 高井宏幸等著，第一机械工业部技术情报所编译，工业机械人的结构与应用，机械工业出版社，1977.

[7] V. Daniel Hunt, Smart Robots, New York London Chapman and Hall, 1985.

[8] Paul R. Cohen, Edward A. Feigenbaum, The Handbook of Artificial Intelligence, Vol. III, William Kaufman, Inc., 1982.

[9] Patrick Henry Winston, The Psychology of Computer Vision, McGraw-Hill, Inc., 1975.

[10] Tomás Lozano-Pérez, Spatial Planning: A Configuration Space Approach, IEEE Transcations on Computers, Vol. C-32, No. 2, February 1983.

[11] R. T. Chien, Ling Zhang, Bo Zhang, Planning Collision-Free Paths for Robotic Arm Among Obstacles, IEEE Transcations on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 1, January 1984.

[12] Alan Paugh, Robot Sensors, Vol. 2, IFS (Publication) Ltd, UK Spring-Verlag Berlin Heidelberg New York Tokyo, 1986.

[13] Shahid Mujtable & Ron Goldman, AL User's Manual, Computer Science Department, Stanford University, January 1979.