摘　要

随着航空、汽车等现代工业的发展，对于复杂曲面的研究日益深入。传统的曲线曲面造型方法，设计制造周期长，制造精度低，互换协调性差，而不能适应现代航空、汽车等工业的发展。并且对蜡模进行试切，校验数控程序的正确性，调试周期长，成本高。计算机性能的飞速提高，计算机图形学和图像处理技术的进步，形成了虚拟制造的新热点，而成形制造的模拟仿真又成为成形制造领域的新热点。虚拟制造是实际制造过程在计算机上的本质体现，采用计算机仿真与虚拟显示技术，使设计人员可以直观的从 CRT 上观察到刀具加工轨迹及加工效果。

本文在全面归纳、总结国内外虚拟制造的基础上，着重对复杂曲面的虚拟仿真加工进行了研究。对复杂曲面的建模，Visual C++6.0 环境下 OpenGL（Open Graphics Library，开放图形库）的应用进行了大体的介绍。并在此基础上，分别从毛坯建模、刀具建模、NC 代码编译等方面进行论述，从而建立了复杂曲面的虚拟仿真系统，进一步验证 NC 代码的正确性，减少了试制时间成本，提高效率，有效的避免了过切、欠切和干涉等现象。

•分析了复杂曲面的描述方法、仿真技术的重要性以及目前在国内外的发展现状。

•提出了不同的复杂曲面的建模方法，着重对 NURBS（Non-Uniform Rational B-Spline,非均匀有理 B 样条）曲面建模进行了分析。

•运用了 OpenGL 绘出工件毛坯和刀具模型，并用其双缓存技术实现动画演示，结合 GAPT(Graphic Automatically Programmed Tools,图形自动编程器)的 NC 代码自动编译，实现复杂曲面的虚拟仿真加工。

关键词：复杂曲面；NURBS；OpenGL；数控编译；干涉

Abstract

As the development of modern industry such as aerospace and automobile industry, the research on complex curved surface is going in greater depth day by day. The modeling of tradtional curved surface is not fit for development of modern industry because of their long design and manufacturing circle, low precision of manufacture, inferior coordination. Trying to cut with the wax model, and verifying NC program make the debugging so long and the cost too high. But computer performance rapid enhancing, computer graphics and picture processing technology become the new domain. Furthermore, the simulation of modeling manufacture has formed the new domain of modeling manufacture. Virtual Manufacturing is the essential embodiment of the actual manufacturing process with the simulation and virtual display technology of computer, and it makes designers observe the processing path and effect directly from CRT.

This thesis pays the emphases on the research of the virtual simulation of complex curved surface process on the basis of concluding and summarizing the status quo of domestic and overseas virtual manufacturing, and introduces the modeling of complex curved surface and the application of OpenGL (Open Graphics Library) under the environment of Visual C++6.0. On these conditions, this article discusses the roughcast modeling, cutting tool modeling and NC program translating and editing. And it also establishes the system of the virtual simulation of complex curved surface process in order to vertify the correctness of NC program, reduce the cost and time of trial-manufacture, enhance the efficiency , and avoid many phenomenon such as cutting too lot or few, interference and so on.

·To analyze the description of complex curved surface, the essentiality of simulation technology and its domestic and overseas status quo.

·To propose the different complex curved surface modeling method, emphatically to analyze the NURBS (Non-Uniform Rational B-Spline) curved surface modeling.

·To realize the virtual simulation of complex curved surface process with the roughcast and cut modeling by OpenGL, the demonstration by double buffers technology and the automatically translation and edit of NC program by GAPT(Graphic Automatically Programmed Tools).

Key words: Complex curved surface; NURBS; OpenGL; NC translating and editing; Interference

插图索引

附表索引

兰州理工大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：王俊鹏　日期：08年 5月 9日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权兰州理工大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。

作者签名：王俊鹏　日期：08年 5月 9日

导师签名：　日期：08年 5月 9日

第1章 绪 论

1.1 复杂曲面的定义与描述方法

随着航空、汽车等现代工业的发展，对于复杂曲面的研究日益深入。传统上采用模线样板法表示和传递自由型曲线曲面的形状的方法，因其所表示与传递的几何形状因人而异，要求设计与制造人员付出繁重的体力劳动，设计制造周期长，制造精度低，互换协调性差，而不能适应现代航空、汽车等工业的发展。随着计算机的出现，依据定义自由型曲线曲面形状的几何信息，可以建立相应的曲线曲面方程（数学模型），并通过在计算机上执行计算和处理程序，计算出曲线曲面上大量的点及其他信息，从而可了解所定义形状具有的局部和整体的几何特征，为所有的后置程序（如数控加工）提供了必要的条件。

工业产品的形状大致上可分为两类：一类是仅由初等解析曲面（例如平面、圆柱面、圆锥面、球面、圆环面等）组成，大多数机械零件属于这一类，可以用画法几何与机械制图的方法完全清楚表达和传递所包含的全部形状信息；第二类是不能由初等解析曲面组成，而以复杂方式自由变化的曲线曲面即所谓的自由型曲线曲面组成，例如飞机、汽车、船舶的外形零件。

自由曲面可以表示为 Ferguson 曲面、Coons 曲面、Bézier 曲面、均匀 B 样条曲面、非均匀 B 样条（NURBS）曲面、三角曲面、散乱数据插值曲面等。NURBS 曲面具有以下优点[1][2]：

（1）可用一个统一的表达式同时精确表示标准的解析形体和自由曲线、曲面。

（2）为了修改曲线曲面的形状，既可借助调整控制顶点，又可利用权因子，因而具有较大灵活性。

（3）与多项式 B 样条一样，NURBS 方法的计算也是稳定的。

（4）NURBS 曲线曲面在线性变换下是几何不变的。

（5）已具有功能完善的几何计算工具，其中包括节点插入与删除、节点加密、升阶、分割等的算法与程序。

NURBS 曲面在 CAD/CAM 领域中获得了广泛的应用，所以本文在对曲面进行数学建模时也采用 NURBS 曲面法。

1.2 虚拟制造

1.2.1 虚拟制造系统构成与体系结构

随着计算机性能的飞速提高，计算机图形学和图象处理技术的进步，形成了

虚拟制造的新热点。而成形制造的模拟仿真又成为成形制造领域的新热点[3]。虚拟制造是实际制造过程在计算机上的本质实现，即采用计算机仿真与虚拟现实技术，在计算机上群组协同工作，实现产品的设计、工艺规划、加工制造、性能分析、质量检验，以及企业各级过程的管理与控制等产品制造的本质过程，以增强制造过程各级的决策与控制能力。

虚拟制造系统是现实制造系统在虚拟环境下的映射。根据二者之间的对应关系，可将其分为虚拟物理系统（Virtual Physical System, VPS）和虚拟信息系统（Virtual Information System, VIS）[4~11]。

VPS 由现实世界中的物质实体在虚拟空间的映射组成，是现实实体的抽象模型。这些实体可以使材料、零部件、产品、机床、夹具、机器人、传感器、控制器等。当制造系统运行时，这些实体以特有的行为和作用相互影响。VPS 中的抽象模型与现实实体一一对应，具有与真实实体相同的性能、行为和功能。

VIS 涉及到的是关于信息的知识，包括制造过程中的相关信息、信息处理和决策活动，如设计、规划、调度、控制、评估等信息。这些信息中既有静态信息，也有动态信息。图 1.1 表达了虚拟制造与真实制造之间的关系[12]。

从企业的生产角度，虚拟制造可划分为虚拟加工、虚拟生产、虚拟企业三个层次，分别具有如下的体系结构[13][14]：

（1）虚拟加工平台 该平台进行产品的可加工性分析，是众多仿真分析软件的集成，具有以下研究环境：①产品设计与分析，除几何造型和特征造型外，还包括运动学、动力学、热力学分析环境；②基于仿真的零部件制造设计与分析，包括工艺规程优化、工具设计优化、刀位轨迹优化、控制代码优化等；③基于仿真的制造过程碰撞干涉检验和运动轨迹检验；④材料加工成形仿真，包括温度场、应力场、流动场分析、工艺优化等；⑤产品虚拟装配，通过人机交互方式利用三维模型真实地模拟装配过程，检验产品的可装配性。

（2）虚拟生产平台 该平台支持生产环境的布局设计及设备集成、远程虚拟测试、企业生产计划及调度优化，进行可生产性分析。包括虚拟生产环境布局、虚拟设备集成及虚拟计划与调度等问题。

（3）虚拟企业平台 该平台以虚拟企业的形式为敏捷制造提供可合作性分析支持，其中的虚拟企业系统工作环境支持异地设计、装配与调试，而虚拟企业动态联盟及支持运行环境用于 INTERNET 与 INTRANET 下的系统集成与任务协调。

（4）基于 PDM 的虚拟制造平台集成 基于产品数据管理（Product Data Management, PDM）的虚拟制造集成技术提供 PDM 环境下以上三个平台的集成研究环境及其相互接口技术和过程管理技术，实现虚拟制造环境下产品开发全生命周期的过程集成。

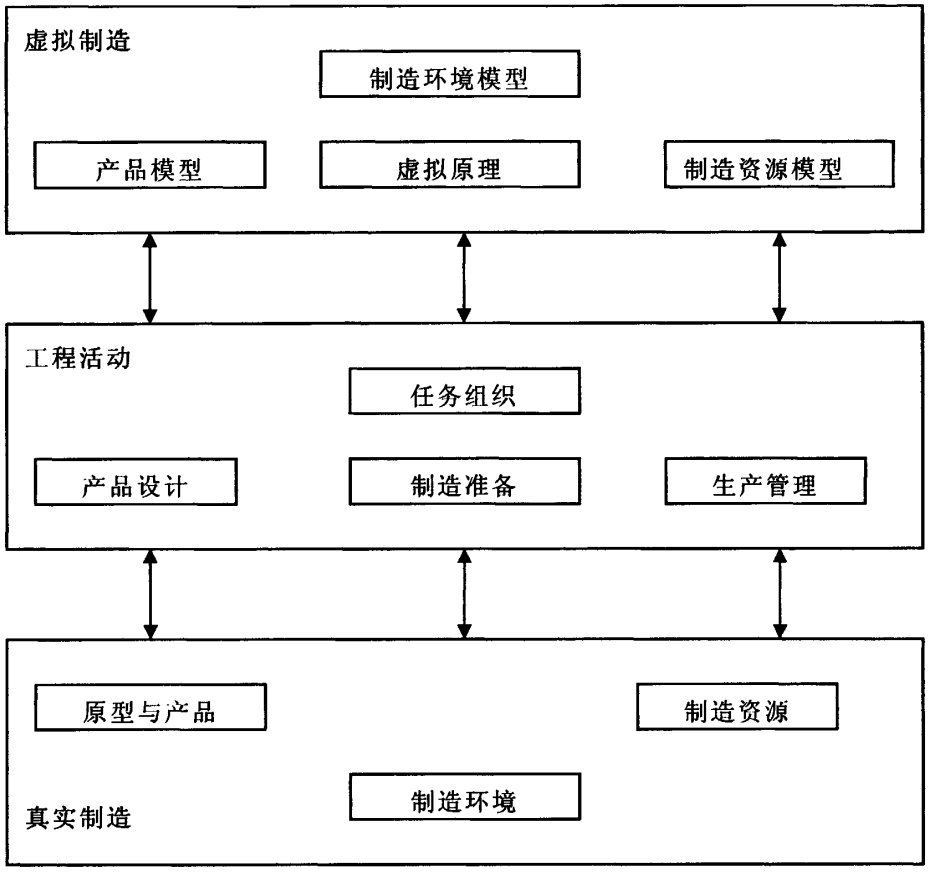

图 1.1 虚拟制造与真实制造

1.2.2 虚拟制造系统国内外研究情况

虚拟制造最早出现在西方发达国家，为适应现代制造业高智能化、高效及高自动化的要求而出现。迄今美、日等工业发达国家已投入了大量的人力、物力进行虚拟制造系统的研究与开发。

我国在虚拟制造技术方面的研究只是刚刚起步，其研究也多数是在原先的CAD/CAE/CAM 和仿真技术等基础上进行的，目前主要集中在虚拟制造技术的理论研究和实施技术准备阶段，系统的研究尚处于国外虚拟制造技术的消化和与国内环境的结合上。由于我国受到 CAD/CAE/CAM 基础软件、仿真软件、建模技术的制约，阻碍了虚拟制造的发展。但这几年，我国对虚拟制造的研究和应用也已经开展起来，主要集中在三个方面：产品虚拟设计技术、产品虚拟制造技术、虚拟制造系统。国家自然科学基金和国家高技术研究发展计划（863 计划）等都有专门的研究课题，清华大学国家 CIMS 工程研究中心正在建立支持产品生产全过程的 VM 平台。浙江大学也正在开展体元和面元混合建模的 CAD 系统研究和面向产品创新设计的新一代 CAD 系统开发。据不完全的调查统计，国内进行虚拟制造技术

研究的单位达到了100家，已经取得了一些可喜的进展。在虚拟现实技术、建模技术、仿真技术、信息技术、应用网络技术等单元技术等方向的研究都很活跃。但研究的进展和研究的深度还属于初期阶段，与国际的研究水平尚有很大的差距，除了三维建模已经有了4种商业软件外，其他方面还没有形成产业化，我国的研究多集中于高等院校和少量的研究院所，企业和公司介入的较少。

在国外，虚拟制造技术VMT首先在飞机、汽车等领域获得成功的应用。典型的例子有波音777，其整机设计，部件测试，整机装配以及各种环境下的试飞均是在计算机上完成的，整个设计制造周期从8年缩短到5年。克莱斯勒公司从1994年开始对汽车车体进行无纸化设计，1998年汽车发动机也全部实现了无纸数字化设计。产品开发周期从36个月缩短到24个月。福特汽车公司的先进车辆技术组应用虚拟制造技术于装配仿真和虚拟成形，以提高空气动力学，人机工程学和表面建模的效果。德国宝马汽车公司为车门的装配操作设计了一个虚拟装配系统，能识别语音输入，完成相应的操作，当发生干涉碰撞时，能发出声音报警。美国Boneing公司设计的一架VS-X虚拟飞机，可用头盔显示器和数据手套进行观察与控制，使飞机设计人员身临其境地观察飞机设计的结果，并对其外观，内部机构及使用性能进行观察。日本Matsushita公司开发的虚拟厨房设备制造系统，允许消费者在购买商品前，在虚拟的厨房环境中体验不同设备的功能，按自己的喜好评价、选择和重组这些设备，他们的选择将被存储并通过网络送至生产部门进行生产。

虚拟制造是面向21世纪的企业发展战略和模式。虚拟企业（或企业动态联盟）是实现虚拟制造的主要途径。虚拟企业强调充分利用社会上已有的设计、制造资源，通过组织动态联盟，快速响应市场变化，把握市场机遇。它克服了传统企业的封闭性、局限性和设计、制造能力的不完备性，减少了资源的重复投入，缩短了生产准备周期，提高了产品从设计、制造到销售全过程的整体柔性和敏捷性，增强了企业（群体）的竞争能力。目前有关虚拟制造、虚拟企业的研究与实践活动已在世界范围内蓬勃展开，现代企业向虚拟企业、虚拟制造发展已是历史的必然。

1.3 仿真技术在制造业中的应用

1.3.1 数控仿真的发展现状与体系结构

数控加工一般包括以下几个过程：

（1）对图样进行分析，确定需要数控加工的部位；

（2）利用图形软件对需要数控加工的部分进行几何造型；

（3）根据加工条件，选择合适的加工参数，生成刀具轨迹；

（4）仿真检验；

（5）生成NC代码文件并传给机床。

由此可见，上述工作需要人与计算机相互配合、共同完成。其中需要大量的计算和重复性的工作，如刀具轨迹计算、仿真检验、NC代码生成等，基本上可由计算机完成，而人只需指定加工部位与工艺条件。计算机仿真数控加工软件可以让用户方便地建立起工件的几何模型（曲面与实体模型），同时只要用户在系统的引导下输入少量数据（工艺参数等），就可以迅速地完成相关的加工编程工作，而且系统具有相当的柔性，可以适应不同类型的情况，对切削加工过程进行仿真，快速检验NC程序，避免发生碰撞和干涉。

目前，流行的计算机数控加工仿真系统主要有以下种:UniGraphics是高档CAM软件的代表，其加工方式完备，计算准确，实用性强，是航空、汽车、造船行业的首选CAM软件。CIMAIRON90是中档CAM软件的代表，该软件产自以色列，其实用性强，也是航空、汽车、电子、模具行业广泛应用的CAM软件。MasterCAM是低档CAM软件的代表，主要应用在中小企业的模具行业。CAXA-ME 是国内CAM软件的代表，主要面向中小企业。由于市场的国际化，全球竞争要求产品的制造过程具有高速度和低成本。产品更新的速度越来越快，市场需求朝着小批量、个性化方向发展。传统的小而全的企业模式已越来越丧失竞争力，各种形式的合作开发、生产和销售方式应运而生。因此，异地设计、异地编程、异地加工越来越被众多企业采用，虚拟制造技术也应运而生。

数控加工仿真软件的主要特点是具有CAD/CAM的系统集成性，比较成熟的CAM系统主要以两种形式实现CAD/CAM系统集成:一体化的CAD/CAM系统(UGII, Euclid, Pro/ENGINEER等)和相对独立的CAM系统(Masterearn, Surfcam等)。前者以内部统一的数据格式直接从CAD系统获取产品几何模型，而后者主要通过中性文件从其它CAD系统获取产品几何模型。

由于机械加工过程仿真还处于起步阶段。目前存在以下几方面的问题:

（1）仿真的加工形式少，研究范围窄[15][16] 在切削加工众多的种类与形式中，目前的仿真加工主要集中于车削、铣削和磨削等。同时，这些加工方法的仿真也局限在很窄的范围内。如铣削仿真多是仿真立铣刀与端铣刀，而这种仿真系统对其他种类的铣刀如加工成形表面用的成形铣刀就无能为力。一方面是因为铣削加工种类繁多，存在着铣平面、铣外圆、铣外形、铣型腔、铣螺旋槽、铣齿轮等多种铣削形式；另一方面是因为铣削加工理论复杂，不同的加工方法、刀具形状的加工模型有较大差别。目前的仿真系统大多数只能进行几何仿真，即刀位轨迹仿真、工件与刀具的干涉校验等，有人称之为NC校验。但在机械加工过程中，几何校验只是前提条件，更为重要的是切削力、刀具振动及刀具磨损等在切削过程中起决定因素的各物理量。

（2）物理仿真考虑理想状态，与实际有较大差距 在目前的仿真系统中预

先设定了大量的假设因素，如设定工艺系统刚性满足要求、无振动，加工材料结构统一、无硬点等缺陷，刀具无磨损，切削要素不发生变化等。这种假定的理想状态不能将切削过程中的随机干扰如工件硬点造成的材质变化、振动造成的切深变化等因素考虑进去，使仿真系统不能真实地反映实际切削过程。

1.3.2 虚拟机械加工过程仿真

虚拟加工过程仿真是虚拟制造技术的底层关键技术，同时也是构成虚拟加工平台的核心内容，目前的研究通常分为几何仿真与物理仿真两大方向。

加工过程几何仿真从形式上表现为在计算机上虚拟执行加工过程，实现机床—刀具—工件构成的工艺系统在视觉上进行的切削加工活动，
在计算机上反映机床的外形、控制面板、操控方法、运动方式，工件的运动与装夹，刀具的运动轨迹等。几何仿真主要用于验证NC程序的正确性（NC Verification）[17]。即可检验NC程序控制的刀位轨迹是否符合加工要求，有无过切或欠切，又可检验干涉和碰撞，避免了耗时、费力的试切过程。

加工过程的物理仿真是将切削过程中的各物理因素的变化映射到虚拟制造系统中。随着先进制造业的飞速发展，生产过程高效率、高智能化及自动化发展的特点，单纯的虚拟加工过程几何仿真已经不能满足要求，在实际加工过程之前对切削过程进行全面的仿真、预测与分析，
是提高加工效率、加工质量和生产率的有效手段。物理仿真在实际加工过程进行之前分析与预测各参数的变化及干扰因素对加工过程的影响，揭示加工过程的实质，分析工件加工质量，辅助在线检测与在线控制，分析具体工艺参数下的工艺规程质量，进行工艺规程的优化.物理仿真
的主要内容包括加工过程中实际切削力的变化规律、整个工艺系统的动态变化特点、刀具磨损、工件的加工质量、工艺参数对加工质量的影响及危险、异常情况如切削颤振等的预测等方面。

1.4 本课题研究的内容和意义

1.4.1 本课题研究的内容

目前，零件造型越来越复杂，尤其是复杂曲面的应用越来越广泛。为缩短产品的试制周期、降低成本、提高数控加工效率，本论文开发了一种复杂曲面仿真加工软件，在加工试制之前，让设计人员可以直观的在计算机屏幕上看到将要加工成型的复杂曲面及其毛坯模型，并可以快速、直观、形象逼真地在毛坯实体模型上进行实时切除过程，不断的对所编的数控程序进行修改，保证工件加工正确，防止加工过程的过切、欠切及干涉现象的出现。

本课题在大量研究的基础上，利用 OpenGL 的计算机仿真和动画技术。在

Visual C++6.0 的开发环境下，采用基于数控代码的仿真方法，研究了复杂曲面数控加工过程的三维动态几何仿真。本课题主要完成以下内容：

（1）研究计算机图形学，对复杂曲面进行建模分析。

（2）研究计算机仿真、建模方法，实现对刀具，毛坯的建模仿真。

（3）学习 NC 文件的自动编程，设计 NC 文件编译器，检查解释器，并引用到本论文中。

（4）设计 OpenGL 与 Visual C++6.0 间的接口，在 Visual C++6.0 环境下实现复杂曲面的三维动画加工场景。

1.4.2 本课题研究的意义

通过对复杂曲面及刀具的建模，读取 NC 文件代码，不仅可以清楚地表达加工刀具的位置和进给的快慢和深度，而且可以实时校验数控程序的正确性（如出现过切、欠切、加工路线不佳或非加工区的干涉现象等）。通过加工过程的仿真，即可十分方便地进行错误修改，NC 代码调试和检验等工作，直到获得合格的数控加工程序。

加工过程仿真是虚拟制造技术的基础。复杂曲面的仿真加工为复杂曲面的数控加工程序提供了简单适用高效直观的编制检验过程。虚拟制造[18~20]是应用计算机技术，对产品的设计、加工、装配等工序的统一建模，形成虚拟的生产过程，从而产生了虚拟的产品、企业。虚拟制造技术使厂家可以在不同的城市甚至不同的国家通过 Internet/Intranet 进行设计、加工，共享同一产品模型，从而大大提高生产效率，降低成本。虚拟制造的概念，集中体现了仿真技术应用的分布、交互和集成化趋势，是计算机仿真在制造业中应用的展望。本课题对复杂曲面的仿真加工的研究，对以后研究虚拟加工中心对复杂曲面仿真加工奠定了基础。

1.5 本章小结

随着航空、汽车等工业的飞速发展，对于复杂曲面的研究日益深入。传统的表示复杂曲面形状的方法已经不能满足现代工业的告诉发展，同时，传统的零件试切法既浪费成本，试制时间又长。鉴于此，本章主要论述了国内外虚拟仿真加工的现状，并提出了本课题所研究的内容及课题研究意义。

第 2 章　复杂曲面的建模

2.1 常用几何建模方式

（1）线框建模 线框建模是 CAD/CAM 中开发应用最早的建模方法，它用顶点和边棱线的有限集合来表示和建立物体的计算机内部模型。线框模型数据结构的关键在于正确地描述每一线框的棱边，点表描述每个顶点的编号和坐标，边表说明每一棱边起点和终点的编号。实际上，物体是边表和点表相应的三维映射。线框建模具有数据结构简单、运算速度快，很好的交互作图功能等特点，但它也存在一些问题：对于形状复杂的零件，常常会导致加工轨迹数量增大，刀具轨迹非常拥挤，仿真中无法分辨刀具位置。线框模型无法实现对结构体的消隐处理，既不可能渲染得到具有真实感的产品图像；无法进行碰撞等干涉检验。

（2）表面建模 表面建模是将物体分解为组成物体的表面、边线和顶点，用顶点、边线和表面的有限集合来表示和建立物体的计算机内部模型。表面模型的数据结构是在线框模型数据结构的基础上增加面的有关信息与连接指针，其中还有表面特征码，各条棱边除了给出连接指针外，还给出方向、可见不可见信息等。表面模型中的几何形体表面可以由若干面片组成，这些面片可以是平面、解析曲面、参数曲面。利用表面模型，可以对物体做剖面、消隐、着色、表面积计算、曲面求交、NC 刀具轨迹生成、获得 NC 加工所需要的表面信息等，有助于对零件进行渲染等处理，有助于 CAM 系统直接提取有关面的信息生成数控机床的加工指令。表面模型虽然比线框模型具有较丰富的形体信息，可以描述任何复杂的结构体，但它并未指出该物体是实心还是空心，无法区别面的哪一侧是体内、哪一侧是体外，因此，表面模型仅适用于描述物体的外壳，不宜用作表示零件的一般方法。

（3）实体建模 实体建模的研究重点是如何运用简单几何体构造复杂组合实体，如何方便地定义形状简单的几何体，如何经过适当的布尔集合运算构造出所需的复杂集合体，并最终在图形设备上输出各种视图。实体建模系统对结构体的几何表达克服了线框建模存在的二义性（即一个线框模型可能被解释为若干个有效的几何体）以及表面建模容易丢失面信息等缺陷，从而可以自动进行真实感图像的生成和物体间的干涉检查，具有很大的优点，所以在设计与制造中广为应用，尤其在运动学分析、干涉检验等方面已成为不可缺少的工具。但由于实体构造法在仿真过程中全部进行的是实体间的布尔与运算，需要进行大量的曲面与曲面，实体与实体间的求交运算，所以计算量非常惊人，其时间复杂度为 $O(N^4)$，其中 N 是刀具运动段的数量。实际加工中，一个复杂零件的加工过程通常会有几万、几十万甚至更多的加工轨迹，刀位点的数量就大，造成的计算量是巨大的。此外，

工件的形状改变则需要重新对刀具扫描体与工件进行求交计算，这样也大大降低了该法的实时性。

（4）基于图像空间建模 图像空间建模方法是使用类似图形消隐的Z-buffer思想，将工件和刀具按屏幕的像素离散为Z-buffer结构，切削过程简化为沿视线方向上的一维布尔运算。它根据平行透视原理，将视线方向与屏幕垂直，沿视线方向将毛坯和刀具离散，使计算机屏幕上的每一个像素点都对应唯一的一条视线。刀具切削毛坯的过程中，从屏幕上发出的某一条视线与工件体或刀具体如果存在交点，则利用这些交点将工件体和刀具体沿视线方向离散，由于将工件和刀具赋予了不同的颜色，这些交点在屏幕上显示的就是这个像素点处的颜色。

(5)离散矢量建模 该方法是将零件表面按照一定的方式以一定精度进行离散，用这些离散点来代替原曲面，计算每一个离散点在原曲面处的法矢，从该点沿法矢方向的直线与所定义的毛坯边界或与零件别的表面相交，交点与原离散点之间的距离的最小值为该离散点法矢的初始长度。仿真计算时，从该离散点出发并沿该点法矢方向的直线与刀具运动形成的刀具包络体相交，如果交点到离散点的距离小于原来的法矢长度，则用交点距离代替原来的法矢长度，否则保留原来的发矢长度值。这样重复这个求交过程直到刀具切削加工完成，通过这些离散点的矢量值不断减少来模拟仿真加工过程中刀具切削毛坯体的材料去除过程。

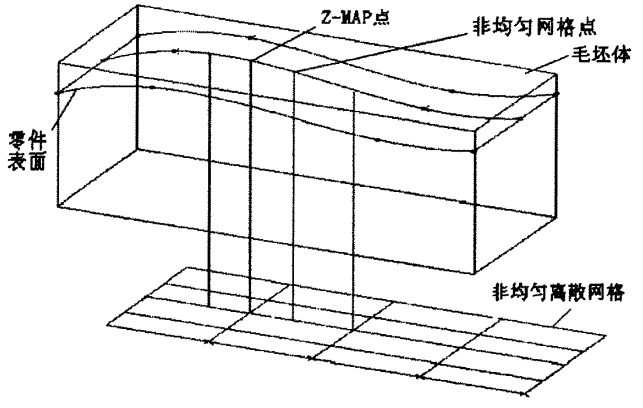

图 2.1 均匀离散矢量模型

2.2 NURBS 曲面建模

2.2.1NURBS 的提出及优缺点

在飞机外形设计与绝大多数机械零件中经常遇到许多由二次曲线与二次曲面表示的形状，机械零件、塑料制品中圆柱面、圆锥面、圆环面等二次曲面及平

面构成的形状比比皆是。还有一些形状如叶轮，既包含自由型曲面也包含二次曲面，这些形状在设计上都由图纸明确无误的给出，在制造上往往又要求较高的精度，而 B 样条曲面包括其特例的贝齐尔曲面都不能精确表示除抛物面的二次曲面，而只能给出近似表示。近似表示将带来处理上的麻烦，使本来简单的问题复杂化，还带来原来不存在的设计误差问题。因此由皮格尔[21]和蒂勒[22][23]提出并深入研究了 NURBS 方法。

NURBS 方法之所以在 CAD/CAM 与计算机图形学领域获得越来越广泛，是因为其有以下优点[24]:

（1）既为标准解析形状也为自由型曲面的精确表示与设计提供了一个公共的数学形式。

（2）有操纵控制顶点及权因子为各种形状设计提供了充分的灵活性。

（3）计算稳定且速度相当快。

（4）NURBS 有明显的几何解释，使得它对有良好的几何知识尤其是画法几何知识的设计人员特别有用。

（5）NURBS 有强有力的几何配套技术（包括插入节点/细分/消去、升阶、分裂等），能用于设计、分析与处理等各个环节。

（6）NURBS 在比例、旋转、平移、剪切以及平行和透视投影变换下是不变的。

然而，NURBS 也存在一些缺点：

（1）需要额外的存储以定义传统的曲线和曲面。

（2）权因子的不合适应用可能导致很坏的参数化。

（3）某些基本算法（如反求曲线曲面上点的参数值）存在数值不稳定问题。

NURBS 方法是建立在非有理贝齐尔方法与非有理 B 样条方法基础上的，因其具有统一表达自由曲线曲面和解析曲线曲面的能力，当几何形体同时存在自由曲线曲面和解析曲线曲面时，应用 NURBS 方法最为有效。

2.2.2 NURBS 曲面建模

（1）曲面方程的建立

一张 p×q 次 NURBS 曲面的有理分式可以表示为如下形式：

$$S(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}\omega_{i,j}d_{i,j}N_{i,p}(u)N_{j,q}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}\omega_{i,j}N_{i,p}(u)N_{j,q}(v)} \qquad 2.1$$

其中 $d_{i,j}$ (i=0, 1, …, m; j=0, 1, …, n)为控制顶点，呈拓扑矩形阵列，形成一个控制网格，见图 2.2。$\omega_{i,j}$是与顶点 $d_{i,j}$联系的权因子，规定四角顶点处用正权因子

即 $\omega_{0,0}$、$\omega_{m,0}$、$\omega_{0,n}$、$\omega_{m,n}>0$，其余 $\omega_{i,j}\geq 0$ 且顺序 p×q 个权因子不同时为零。$N_{i,p}(u)$ (i=0, 1, …, m)和 $N_{j,q}(v)$ (j=0, 1, …, n)分别为 u 方向 p 次和 v 方向 q 次有理基函数，其节点矢量分别为：

$$U=\{0,0,\cdots,0,u_{p+1},\cdots,u_{r-p-1},1,1,\cdots,1\}$$

$$V=\{0,0,\cdots,0,v_{q+1},\cdots,v_{s-q-1},1,1,\cdots,1\} \qquad 2.2$$

开始和结束的节点值分别重复 p+1 和 q+1 次，r=n+p+1, s=m+q+1。

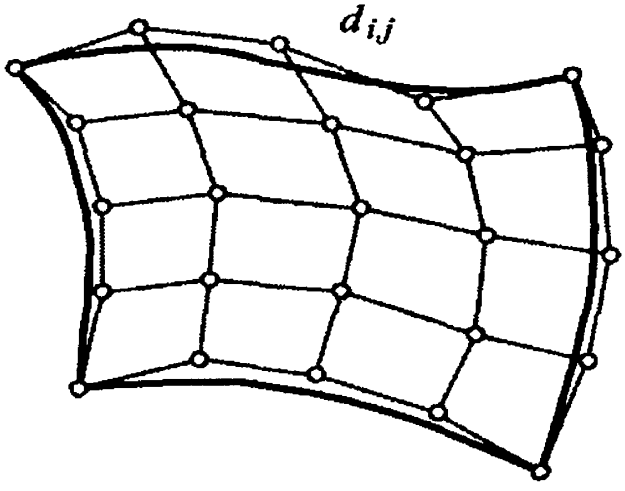

图 2.2 NURBS 曲面

$N_{i,p}(u)$可以由下式确定[25]：

$$N_{i,0}(u)=\begin{cases}1,\text{若 } u_i\leq u\leq u_{i+1}\\0,\text{其 他}\end{cases}$$

$$N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u) \qquad 2.3$$

$N_{j,q}(v)$可以由下式确定：

$$N_{j,0}(v)=\begin{cases}1,\text{若 } v_j\leq v\leq v_{j+1}\\0,\text{其 他}\end{cases}$$

$$N_{j,q}(v)=\frac{v-v_j}{v_{j+q}-v_j}N_{j,q-1}(v)+\frac{v_{j+q+1}-v}{v_{j+q+1}-v_{j+1}}N_{j+1,q-1}(v) \qquad 2.4$$

（2）曲面的性质[2]

①NURBS 曲线曲面既覆盖了多项式的也覆盖了有理的 B 样条曲线曲面。

②线性变换的不变性，对控制顶点网格的逼近具有局部性。

2.2.3 NURBS 曲面的计算

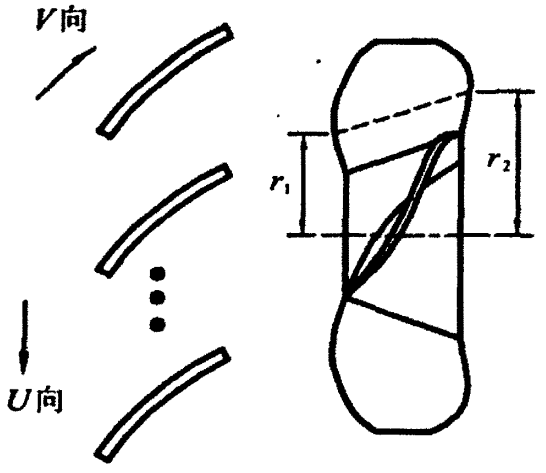

图2.3　叶片截面图

以复杂扭变叶片为例，介绍NURBS曲面的计算。复杂扭变叶片有其自身特点，如图2.3所示，叶片在u 方向上被不同半径的圆锥面分割为n 个截面，每个空间截面在v 方向上展平后得到m 个二维型值点，这些点不能直接用于设计，在NURBS运算前，应首先进行坐标转换，依次将每条截面的二维型值点恢复成原始的三维数据点，形成n ×m个曲面空间点阵列。

在此基础上，插值反求曲面控制点阵列，然后进行曲面正求运算，得到叶片曲面上一系列离散点的信息。同时，为了增加叶片数据与其他三维软件(如Pro/ E)的互换性，将叶片曲面离散化为三角片序列，生成通用数据交换标准STL 文件，为叶片后续处理工作奠定了基础。整个流程如图2.4 所示。

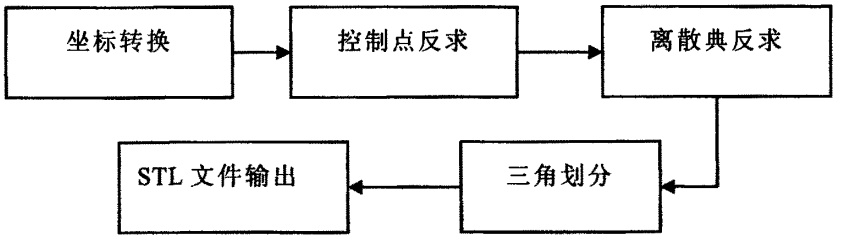

图2.4　设计流程图

（1）曲面控制点反求

为分析方便，坐标转换后输入点假设为n×m个，定义叶片为u，v 双三次曲面，控制点权因子为1。

① 曲线控制点反求

以v向为例，通过截面数据点 q_i （i = 0, 1, …, m）的一条3次NURBS 曲线由控制点 d_i （i=0, 1, …, m + 2）与节点矢量U=[u_0, u_1, …, u_{m+6}]来决定。节点矢量U取规范定义域[0, 1]，其中 u_0=…= u_3=0, u_{m+3}=…= u_{m+6}=1，内节点按积累弦长参数化法计算得到。一般叶片截面为圆弧状或机翼型，因此反求曲线控制点时，可按开曲线处理。定义曲线方程组时，要增加边界条件，可按自由端点条件处理，最终确定的矩阵形式方程组如下：

$$\begin{bmatrix} a_1 & b_1 & c_1 & & \\ a_2 & b_2 & c_2 & & \\ \cdots & & \cdots & & \\ & & a_m & b_m & c_m \\ & & a_{m+1} & b_{m+1} & c_{m+1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \cdots \\ d_m \\ d_{m+1} \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \cdots \\ e_m \\ e_{m+1} \end{bmatrix} \qquad 2.5$$

通过列主元消去法求解以上方程组，即可求得该曲线的控制点 d_i （i = 0, 1, …, m+2）。

②曲面控制点反求

在上述方法的基础上，依次求得每条三次截面曲线的控制顶点 $D_{i,j}$ （i=0, 1, …, n, j=0, 1, …, m+2），然后固定v向，以求得的控制点 $D_{0,j}$, $D_{1,j}$, …, $D_{n,j}$为“数据点”，二次反求，最终得到整个叶片曲面的控制点 $d_{i,j}$ （i =0, 1, …, n+2, j=0, 1, …, m+2），同时按照“节点矢量中所有同下标的节点取算术平均值”的方法，确定u, v两个方向的统一节点矢量U = [u_0, u_1, …, u_{n+6}]，V=[v_0, v_1, …, v_{m+6}]，为曲面正算作好准备。

（2）曲面离散点正求

一般，称求取曲面数据点的过程为正算，它与反算过程相反，建立在德布尔递推公式基础之上。

①德布尔递推公式[1]

德布尔算法是一种非常快捷实用的算法，它通过递推控制顶点，最终求得曲线上点的坐标，如下：

$$p(u) = \sum_{j=0}^{n} d_j N_{j,k}(u) = \sum_{j=i-k}^{i-1} d_j^l N_{j,k-1}(u) = \cdots = d_{i-k}^k$$

$$u \in [u_i, u_{i+1}] \subset [u_k, u_{k+1}] \qquad 2.6$$

在这里，k=3，因此将影响参数u 所在节点区间的4个控制顶点迭代3 次，即可求出p(u) 。

②曲面任意点求值

首先利用德布尔递推公式，根据参数v求得各截面曲线上的数据点 $p_i(v)$ （i = 0,1,…,n+ 2），然后把这些数据点看作控制点，以输入参数u 再次得布尔递推，最终

得到曲面点。若循环计算,则可以指定求得n×m个曲面离散点。

在计算机辅助几何设计（CAGD）里，参数曲面描述被广泛地分成形状表示与形状设计的两大类别中。对于形状表示，因已经存在一个原型及描述它的数值信息。在相当程度上，造型系统或程序能在合适的允差内自动的进行处理，但是对于形状设计,一个特别重要的问题是怎样使曲面定义与交互的维数从三维的物体空间减少到二维的屏幕显示平面。通过蒙面（Skinning）法设计的曲面，其插值不再是通过给定的数据点阵，而是直接顺序通过一族曲线。蒙面法通常被考虑为最适合于交互 CAD 应用的。目前市场上每个 CAD 系统实际上都采用类似的曲面定义，是当今曲面造型实践中广发应用的技术[1]。

2.2.4 用蒙面（skinning）法设计 NURBS 曲面

蒙面就是拟合一张曲面通过一组有序的称为截面曲线的空间曲线。可形象地看成为给一族截面曲线构成的骨架蒙上一张光滑的皮,所生成的曲面就称为蒙皮曲面[1]。

用蒙面法设计一般 NURBS 曲面可按下述步骤进行：

（1）初始地生成形状符合要求的截面曲线，都分别用 NURBS 表示。它们可能具有不同的次数与节点矢量。

（2）统一次数，使所有较低次数的截面曲线都升阶到其中的最高次数。

（3）插入节点，每一截面曲线的节点矢量插入其他截面曲线节点矢量中相异的节点值。插入节点的结果使得所有截面曲线都具有统一的节点矢量，即所求曲面沿 u 向的节点矢量。

（4）反算得到各截面曲线在 u 向的新的控制顶点。

（5）计算 v 向节点矢量。

（6）以步骤(4)产生的各截面曲线的控制顶点为型值点，在 v 向反求顶点，所得到的控制顶点即为蒙面法设计 NURBS 曲面的控制顶点。

2.2.5 蒙面法设计举例

（1）给定两条一般的 NURBS 曲线

$$C_1(u)=\frac{\sum_{i=0}^{m_1}\omega_i d_i^1 N_{i,k_1}(u)}{\sum_{i=0}^{m_1}\omega_i N_{i,k_1}(u)} \qquad 2.7$$

$$C_2(u)=\frac{\sum_{i=0}^{m_2}\omega_i d_i^2 N_{i,k_2}(u)}{\sum_{i=0}^{m_2}\omega_i N_{i,k_2}(u)} \qquad 2.8$$

其中 0≤u≤1。

（2）统一次数，若 $k_1<k_2$，则可使 $k=k_2$，然后，可将 $C_1(u)$ 的次数从 k_1 升阶到 k；若 $k_1>k_2$，则可使 $k=k_1$，然后，可将 $C_2(u)$ 的次数从 k_2 升阶到 k。升阶的结果，部分老控制顶点及其权因子被新控制顶点及其权因子所替代，现在两条曲线具有了相同的次数 k。

（3）插入节点，升阶过程改变老节点矢量为新节点矢量，新节点矢量很可能与另一未升阶曲线的节点矢量是不一样的，这时，可取两者的并集作为公共的节点矢量 U。同时也使两条曲线有了相同数量 m+1 个新控制顶点与新权因子。

（4）欲在该两曲面之间生成一张直纹面——该曲面的两族等参数线 u 线和 v 线中有一族是直线，这里设 v 向节点矢量 V=[0，0，1，1]。根据（2）、（3）中所求的阶数 k、u 向节点矢量，新控制顶点与权因子。可求得

$$S(u,0)=\frac{\sum_{i=0}^{m}\omega_{i,0}d_{i,0}N_{i,k}(u)}{\sum_{i=0}^{m}\omega_{i,0}N_{i,k}(u)}=C_1(u) \qquad 2.9$$

$$S(u,1)=\frac{\sum_{i=0}^{m}\omega_{i,1}d_{i,1}N_{i,k}(u)}{\sum_{i=0}^{m}\omega_{i,1}N_{i,k}(u)}=C_2(u) \qquad 2.10$$

综上步骤，得到了该两条 NURBS 曲线间形成的直纹面

$$S(u,v)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{1}\omega_{i,j}d_{i,j}N_{i,p}(u)N_{j,q}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{1}\omega_{i,j}N_{i,p}(u)N_{j,q}(v)} \qquad 2.11$$

2.3 本章小结

本章主要介绍了线框建模、表面建模、实体建模、基于图像空间建模及离散矢量建模等几种常见的建模方式，重点介绍了 NURBS(Non-Uniform Rational B-Spline,非均匀有理 B 样条)曲面的建模及计算，并讨论了用蒙面（Skinning）法设计 NURBS 曲面。

第 3 章 OpenGL 三维图形设计

3.1 OpenGL 简介[26][27]

OpenGL 是 SGI 公司（Silicon Graphics Incorporated）在他们的图形工作站上开发出的高质量的图像接口。最近几年它已成为一个非常优秀的开放式三维图形接口。实际上它是图形软件和硬件的接口，包括有 120 多个图形函数。微软在 Visual C++6.0 中已经提供了三个 OpenGL 图形库（glu32.lib,glau.lib 或 glut.h,OpenGL32.lib）,可以使我们方便地编程，简单、快速地生成美观漂亮的图形。OpenGL 通过集成大量的渲染、纹理映射、特殊效果和其他强大的可视化函数，使得其应用程序更加新颖，并大幅度加速了图形应用程序的开发。OpenGL 具有以下特点：

（1）高质量、高性能的可视化

任何要求高性能图形的应用程序，从 3D 动画、CAD 到可视化仿真，都可以利用 OpenGL 高质量、高性能的特点来进行开发，开发者可以在不同的领域绘制并显示让人难以想象的 3D 图形，如 CAD/CAM/CAE 等。

（2）开发方便

从渲染一个简单的几何点、线或填充多边形到生成最复杂的光照和纹理映射的 NURBS 曲面，OpenGL 简化了图形软件的开发。软件开发者可以获取几何图像图元、显示列表、模型变换、光照和纹理、反走样、混合和其它一些特征。

不同平台的 OpenGL 实现程序包括了 OpenGL 函数的完全实现。OpenGL 标准已经与 C、C++、Fortran、Java 语言捆绑。利用 OpenGL 函数编写的程序很容易移植到其他平台，使编程效率大大提高，缩短了产品的开发周期。所有的 OpenGL 状态元素，甚至纹理内存和帧缓存的内容，都可以通过其他应用程序获得。

（3）与平台无关

OpenGL 是 SGI 努力提高 IRISGL 的可移植性的结果。新的图形 API 可提供 GL 的性能，但它却是“开放的”，具有与硬件无关的特性，容易适应其他硬件平台和操作系统。因此，为了实现这一跨平台性能，OpenGL 没有一条函数或命令与窗口或屏幕管理有关，而且也没有用于键盘输入或鼠标交互的函数。创建和打开窗口在不同的平台上是以不同的方式实现的。

OpenGL 支持所有 UNIX 工作站和 WindowsNT 和 Windows95/98PC,是适用硬件平台和软件环境最广泛的图形编程 API。OpenGL 可以在所有主要的操作系统上运行，如 WindowsNT、Linux、UNIX 等。另外，也适用于所有主要的窗口

系统，包括 Win32、X/Windows 等系统。OpenGL 可以在 C，C++，Fortran 和 Java 语言中调用，而且与网络协议与布局无关。

（4）灵活的结构

尽管 OpenGL 规范定义了特定的图形处理流程，平台经销商可以删除某些 OpenGL 程序来满足特定系统及其性能的要求。在专用硬件上可以执行单个调用，在标准的 CPU 系统上作为一个标准的例程运行，或者作为专用硬件和软件例程的合并执行单个调用。这种实现上的灵活行意味着从简单的渲染到整个几何体，从低性能的 PC 到高性能的工作站和超级计算机，都可以实现 OpenGL 硬件加速。在不同的 OpenGL 的开发平台上，应用程序开发者可以保证显示结果的一致性。使用 OpenGL 扩展机制，硬件开发者通过开发其扩展来开发不同的产品，从而允许软件开发者获得额外的性能和技术更新。

（5）可以建立高级 API

高级软件开发者可以利用 OpenGL 强大的渲染库创建 2D 和 3D 图形的高级 API。开发者利用 OpenGL 提供的广泛支持为市场提出解决方案。例如，OpenInventor 提供了一个跨平台的用户接口和灵活的场景，使创建 OpenGL 应用程序更容易。

（6）不断创新

OpenGL 标准是不断升级的，周期性地进行正式的修改，通过不断地进行开发，允许应用程序开发者通过 OpenGL 获得最新的硬件更新扩展。当扩展被广泛接受后，即可考虑将它包含到 OpenGL 标准的内核中。这使得 OpenGL 以一种创新的方式升级。

OpenGL 还具有可扩展性、可靠性、稳定性、文档丰富等特点，并由 OpenGLARB 来负责管理其规范性，而且硬件平台经销商经过 OpenGLARB 的授权，可以获得 OpenGLARB 同意的 OpenGL 规范和原码。而对于终端用户、独立的软件经销商，其他基于 OpenGL 编写的代码不需要授权。

OpenGL 可用于各种用途，包括 CAD 工程和建筑应用，创建电影中计算机上生成恐龙的建模程序等等。把工业标准的 3DAPI 引入市场巨大的操作系统，比如 Microsoft Windows，这会产生某些激动人心的相互作用。随着硬件加速和快速 PC 机处理器变得逐渐平常，3D 图形不久就会成为个人消费品和商业应用品，而不仅仅是游戏和科学应用的典型组件。

3.2 OpenGL 的主要图形功能

OpenGL 的主要图形功能如下：

（1）累积缓存

在累积缓存中，多个渲染的帧组合产生单个的图像。累积缓存用于产生各

种效果，如域的深度、移动模糊和反走样等。

（2）alpha 混合

alpha 混合是提供生成透明实体的一种方法。使用 alpha 可以定义从完全透明到完全不透明的实体。

（3）反走样

反走样是一种用于绘制光滑线条和曲线的方法。该方法将与线条相近的像素平均分配颜色。它减少了线条上和与线条相近的像素的平移，从而看起来更加光滑。

（4）颜色索引模式

颜色索引模式是颜色缓存存储颜色的索引值，而不是红、绿、蓝和 alpha 值。

（5）显示列表

显示列表是一组 OpenGL 命令的命名清单。显示列表的内容经过了预先处理，因而执行起来比在即时模式下运行相同的命令更加迅速。

（6）双缓存

双缓存用于实体的光滑动画。运动实体的每个连续的场景可以在后缓存中构造，然后显示。它只允许完整的图像显示在屏幕上。

（7）反馈

反馈是一种 OpenGL 将处理过的几何信息（颜色、像素位置登）返回给应用程序的模式，是与直接绘图到帧缓存对比而言的。

（8）Gouraud 阴影

Gouraud 阴影是穿过一个多边形或线段的光滑插值。颜色在顶点赋值，并穿过图元进行线性插值，以便产生一个较为光滑的颜色变化。

（9）即时模式

即时模式是有 OpenGL 命令在调用时执行，而不是通过显示列表执行。

（10）材质光照与阴影

该功能根据表面的材质特性，准确计算任意点的颜色。

（11）像素操作

像素操作包括存储、变换、映射、放缩等操作。

（12）多项式求值器

多项式求值器支持非均匀有理 B 样条（NURBS）。

（13）图元

图元是点、线、多边形、位图或图像。光栅图元有位图和像素矩形。

（14）RGBA 模式

RGBA 模式指颜色缓存存储红、绿、蓝和 alpha 值，而不是颜色的索引值。

（15）选择与拾取

该功能决定用户制定的某个图元是否绘制在希望的帧缓存区。

（16）模板平面

模板平面用于掩盖颜色帧缓存中单个像素的缓存。

（17）纹理映射

纹理映射是将纹理应用到图元上的过程。这个技术用于产生真是的图像。

（18）变换

该功能实现在 3D 坐标空间旋转实体，改变实体大小及透视变换等。

（19）Z 缓存

Z 缓存用于确定实体的某一部分比另一部分与观察者是否更近的情况。在消除隐藏面时 Z 缓存非常重要。

3.3 OpenGL 的函数库

OpenGL 提供了功能强大、操作简单的一组渲染命令，所有的高级绘图必须通过这些命令来完成。因此，为了简化特定的程序任务的编写与执行，程序员可以在 OpenGL 的顶层编写自己的函数库。同样，程序员还可以编写一些子程序，使 OpenGL 程序更具有针对性，容易在自己的窗口系统中得以运行。事实上，这类可以提供特殊性的函数库和子程序，有一些已经编制在 OpenGL 函数库中了。

（1）OpenGL 实用库（GLU）

OpenGL 实用库（GLU）中的函数通过底层 OpenGL 命令来完成这样一些任务，如为特定的观察方位和投影设置矩阵、完成多边形的网格化，以及渲染表面等。这个库是作为 OpenGL 实用工具的一部分提供的。GLU 子程序使用前缀 glu 来标志。

（2）OpenGL 扩展库（GLX）

OpenGL 对 X 窗口的扩展库（GLX）提供了 OpenGL 的内容表，并使其与计算机（该计算机使用 X 窗口系统）上的可画窗口联系的一种手段。GLX 是 OpenGL 的一个附件。GLX 库函数使用的前缀是 glx。

（3）OpenGL 辅助库（GLAUX）

尽管 OpenGL 包括渲染命令，但却是独立于任何窗口系统或操作系统而设计出来的。因此，OpenGL 并不包括用来打开窗口以及从键盘或鼠标读取事件的命令。只要编写一个完整的图形程序，就必须打开一个窗口，要不然就无法进行任何操作。许多程序都要求一些用户输入，或者需要操作系统或窗口系统提供其他的一些服务。OpenGL 辅助库便可以用来简化诸如打开窗口、检测输入等命令。

由于 OpenGL 的绘图命令仅限于生成简单的几何图元（点、线和多边形），因此，辅助库包括了一些用于创建更复杂的三维物体，如圆球、圆环及茶壶等物

体的子程序。这样，不但使程序的编制变得轻松，而且将增加不少乐趣。OpenGL发展至今，辅助库已经作为一个部分几乎集成在所有的编译平台中。辅助库使用的前缀是 aux。

（4）使用 GLUT

在 OpenGL 的初期，使用的是辅助库（AUX）。创建 AUX 是为了帮助学习和编写 OpenGL 程序，而不会被任何特定环境的细节分散注意力，不管它是 UNIX,Windows,还是别的什么。在使用 AUX 时编写的不是“最终”代码，它更象一个用于测试创意的预备基础阶段。缺少基本的 GUI（Graphics User Interface，图形用户接口）功能限制了把这个库用于建立有用的应用程序。

仅仅在两三年前，流行的大多数 OpenGL 都是用 AUX 库编写的。AUX 库的 Windows 实现有很多错误，因此，很容易导致频繁的崩溃。缺少 GUI 功能在当今面向 GUI 的世界中无论如何都是另一个缺陷。

在跨平台的编程示例和演示中，AUX 在很大程度上已经被 GLUT 库取代。GLUT 代表 OpenGL utility toolkit(OpenGL 实用程序软件包)。GLUT 是 Mark Kilgard 在 SGI 时编写的，它作为 AUX 库的功能更强的替代品，包括了一些 GUI 功能，至少在 X Windows 下可以生成更有用的示例程序。这个替代品包括了使用弹出菜单、管理其他窗口，甚至可以提供游戏杆支持！GLUT 并不位于公众域中，但它是免费的，而且可以自由分发。使用 GLUT 库所需要的头文件是 glut.h。最好是把这个头文件放到操作系统保存 gl.h 和 glu.h 的同一目录下。

3.4 OpenGL 在 Windows 下的工作机制和基本操作

（1）OpenGL 工作机制

SGI 和微软共同开发的新版 Windows 中嵌入了 OpenGL，使用 Visual C++等集成开发环境。同 GDI（Graphics Device Interface，图形设备接口）的工作方式相似，OpenGL 是通过容器/服务器的方式实现的。OpenGL 客户模式同 OpenGL 服务模块通讯并发出 OpenGL 命令，服务模块调用设备驱动接口来使用驱动设备。由于在 Windows 中 OpenGL 环境为软件实现，并同硬件捆绑，所以可以通过在特定的数据结构中设置和使用标志位来达到利用硬件和驱动程序提供的加速功能。即 OpenGL 所有的命令都是通过绘图描述表所表达的着色环境输出到设备描述表所对应的环境。设备描述表包含 GDI 信息，要在 GDI 函数调用时说明，GDI 函数通过 GDI 服务器直接操作设备描述表（包含虚拟设备描述表）。绘图描述表包含的信息，在函数调用时隐含说明。因此，创建绘图描述表前要设定设备描述表的像素格式，使创建的 OpenGL 绘图描述表适合于设备。图 3.1 给出了 OpenGL 工作顺序。

OpenGL 的像素格式确定绘制图形的特性。如支持单缓冲区还是双缓冲区，

像素数据是 RGBA 格式还是颜色索引格式。OpenGL 在支持单缓冲区的情况下能够使用部分 GDI 函数。

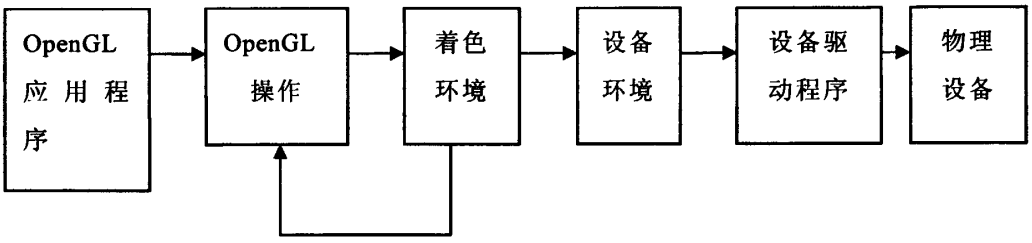

图 3.1 OpenGL 工作顺序

（2）OpenGL 的基本操作

OpenGL 是一种过程性而不是描述性的图形 API。它并不是描述场景及其外观，程序员其实规定了现实某种特定外观或效果所需要的步骤。这些“步骤”牵涉到对这种高度可移植性 API 的调用。它包括了超过 200 条命令和函数。这些命令可用于在三维空间中绘制图元，比如点、线和多边形。另外，OpenGL 支持光照和阴影、纹理贴图、混合、透明度、动画以及许多其他特殊效果和功能。

OpenGL 不包括任何窗口管理、用户交互或文件 I/O 函数。每个主机环境（比如 Microsofe Windows）在这些方面都有自己的函数，由这些函数负责实现某些方法，以便把窗口或位图的绘制控制权移交给 OpenGL。

不存在模型或虚拟环境所用的“OpenGL 文件格式”。这些环境由程序员根据自己的高级需求自行构造，再使用低级的 OpenGL 命令精心编制。其实现有通用实现和硬件实现两种方式。

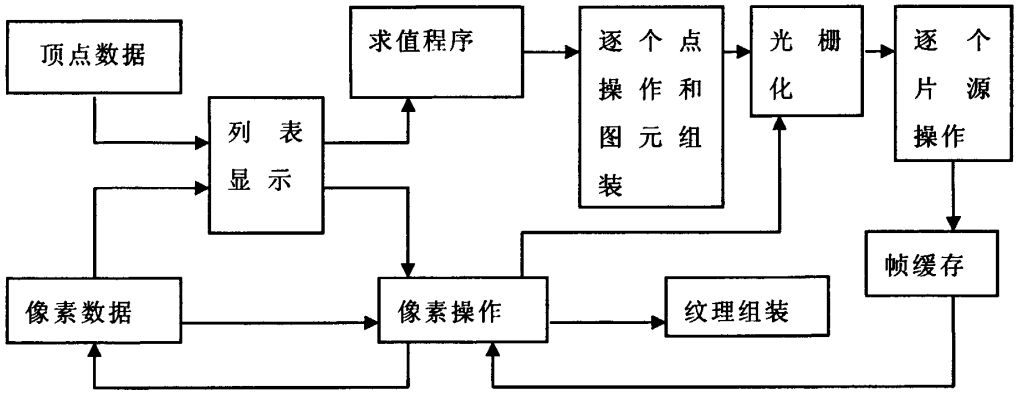

图 3.2 OpenGL 模块框图

图 3.2 给出了 OpenGL 处理数据的高层模块简要框图。图中，函数从左侧进入，通过一系列类似管道的处理过程。一些命令制定要绘制的几何物体，另一些

在不同的操作阶段控制对物体的处理。如图 3.2 所示，OpenGL 并不让每个函数立即通过处理通道，而是把它们累积在一个显示列表中，稍后一次性进行处理。处理管道的求值程序，提供了一种高效的方法来近似绘制几何曲线和曲面。下一阶段进行对每个顶点的操作和图元组合。对顶点进行变换和光照处理，对图元进行裁剪以适合视区大小，这样，便为下一阶段的处理做好了准备。

在光栅操作中，通过点、线段和多边形的二维描述，产生一系列的帧缓存地址和相关数值。这样，产生的每一片原都传入最后的处理阶段——“逐个片原操作”，对数据进行最后的处理，然后将它们作为像素存储在帧缓存中。这些操作包括：基于输入和先前存储的 Z 值对帧缓存（Z 缓存）进行有条件的更新，混合输入像素颜色与存储颜色、屏蔽等其他对像素值的逻辑操作。

输入数据也可以是像素形式，而不是顶点形式。这类数据（可以描述纹理映射中的一幅图像）将跳过上述处理的第一阶段——“求值程序”，作为像素值直接在像素操作阶段进行处理。这一阶段的处理结果，或者存储与纹理内存以用于光栅操作阶段，或者采取同几何数据一样的形式进行光栅化，并将结果片原融合入帧缓存中。所有 OpenGL 状态的要素，包括纹理内存的内容表，甚至帧缓存，都可以在 OpenGL 管理运行过程中获得。

3.5 OpenGL 3D 图形变换

（1）三维投影中的坐标系

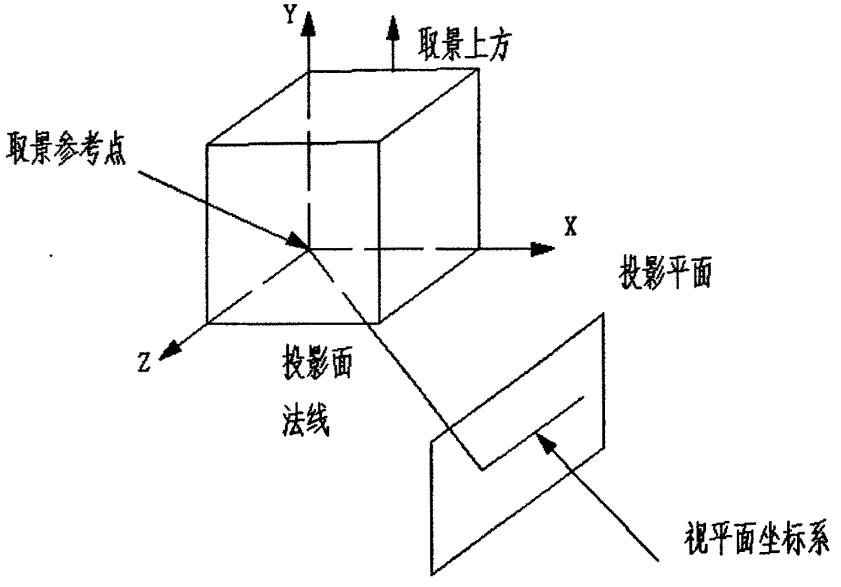

图 3.3 三维投影中的坐标系

OpenGL 物体坐标系采用如图 3.3 所示的左手坐标系。投影三维物体的二维平面称为投影平面，视平面坐标系附在投影平面上。物体坐标系描述物体的模型，

取景参考点为物体建模参考点，一般将物体坐标系原点取为建模参考点。视平面坐标系原点为取景参考点在投影平面上的投影点，即平行于投影平面法线且通过取景参考点的直线与投影平面的交点。确定视平面坐标系向上方向的向量称为取景上方向量。取景上方向量确定视平面坐标系绕取景参考点与视平面坐标系原点连线旋转的角度。观察点或视点是观察者眼睛所在的位置。

（2）OpenGL 绘制物体模型与画几何体的顺序

利用 OpenGL 图元功能如线、多边形等功能绘制物体模型，所有图元最终都以一组顶点坐标来描述。OpenGL 使用 glVertex 命令定义顶点坐标，使用 glColor 命令定义顶点颜色。定义点的定义顺序是按照用户画几何体的顺序。定义立方体的点顺序如图 3.4 所示。定义物体模型可以在 glBegin 和 glEnd 命令对之间给出顶点的坐标。glBegin 命令的参数决定如何绘制几何图元。其参数值的含义如下所示

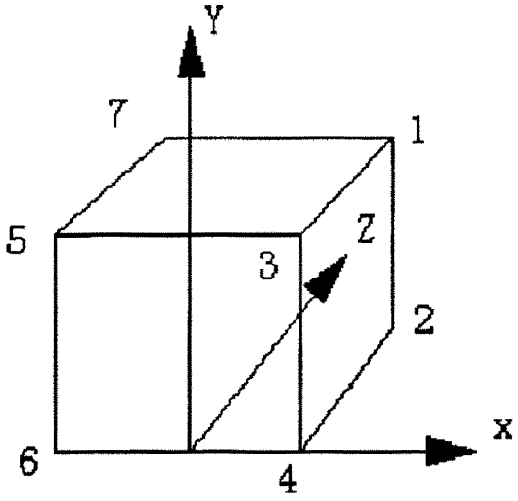

图 3.4 定义点的定义顺序

GL_POINTS：单独的点；

GL_LINES：一队顶点组成一单独的线段；

GL_POLYGON：一简单凸多边形的边界轮廓；

GL_TRIANGLES：三个顶点组成三角形；

GL_QUADS:四个顶点组成四边形；

GL_LINE_STRIP：一组首尾相连的线段；

GL_LINE_LOOP:一组首尾相连的线段，且起始点相连；

GL_TRIANGLE_STRIP:相连的三角形；

GL_TRIANGLE_FAN:扇形三角形；

GL_QUAD_STRIP:相连的四边形。

3.6 在 MFC 中实现 OpenGL 图形编程

（1）OpenGL 在 MFC 中的使用

OpenGL 是一个功能强大的 3D 图形库，它包括将近 120 个绘制点、线和多边形等 3D 图形原语的绘制命令，并且可在这些原语的基础上，构造更复杂的 3D 物体。此外，OpenGL 还支持阴影、纹理映射、反走样、光照以及动画等效果。这足以使 WindowsGDI 相形见绌，因为它只输出 2D 图像。显然，将 OpenGL 引入 Windows 具有很多优点，例如，OpenGL 将简化 3D 应用程序的开发，Windows 则可为应用程序添加 3D 效果，windows 可以开发更生动的 3D 游戏，OpenGL 可作为 Windows 的 3D 图形库等。

然而，从 Windows GDI 转向 OpenGL 并非易事。OpenGL 直接使用绘制环境，只是间接使用设备环境，因此设备环境中的当前画笔、画刷、颜色、字体等均对 OpenGL 绘制无任何影响。甚至 GDI 映射模式都对 OpenGL 不起作用。所以，如果需要使用 MFC 开发应用程序并采用 OpenGL 进行绘制的话，则必须遵循固定的规范，先建立 OpenGL 绘制环境，然后才能使用 OpenGL 命令。

在这里，按照各相关消息句柄被触发的先后顺序，阐述在 Visual C++中构造 OpenGL 绘制环境的必要步骤。此过程共涉及七个消息句柄，依次为 :PreCreateWindow()，OnCreate()，OnSize()，OnEraseBkgnd()，OnInitialUpdate()，OnDraw()，以及 OnDestroy()。在 OnDraw()函数中 OpenGL 绘制环境已经构造完毕并且被设置为当前绘制环境，可使用 OpenGL 命令进行 3D 场景绘制.

①准备工作

为了能够使用 OpenGL 命令，首先需要在预编译头文件 stdafx.h 中添加:

#include “gl/gl.h”

#include “gl/glu.h”

这样预编译头文件才能提供对 OpenGL 库和用户库的支持。此外，如果还需要辅助库中所定义的函数，则还要在预编译头文件 stdafx.h 中添加:

#include “gl/glaux.h”(或#include “glut.h”)

相应地，还应当通过 Project 菜单下的 Settings 选项在 Link 选项卡中链接以下库:OpenGL32.lib, glu32.lib 以及 glaux.lib(如果需要辅助库)。如果是使用 GLUT 库而不是使用辅助库，则应链接 glut.lib。

②消息句柄

i） PreCreateWindow()

为了进行 OpenGL 绘制，必须先在有关窗口的客户区中进行 OpenGL 初始化。由于 OpenGL 不能在子窗口或兄弟窗口中绘制，所以需要重载消息句柄

PreCreateWindow()并将窗口风格规定为：

```
cs.style|=WS_CLIPSIBLTNGS|WS_CLIPCHILDREN;
```

ii) OnCreate()

为了使 OpenGL 能在绘图表面（窗口或位图）上绘制图像，必须先对绘图表面进行初始化，即通过对像素格式的描述（分配并填充 PIXELFORMATDESCRIPTOR 结构）、选择（通过 ChoosePixelFormat()函数）和设置（通过 SetPixelFormat()函数），规定绘图表面的某些属性。此外，只有在 OpenGL 绘制环境中，OpenGL 命令才能被接受并执行，所以必须创建 OpenGL 绘制环境（由 wglCreateContext()函数完成）。在必要的情况下，还将进一步创建调色板。选择视图类消息句柄 OnCreate()作为完成这些工作的恰当场所。

像素格式的描述通过分配并填充 PIX_ELFORMATDESCRIPTOR 结构完成，例如：

```
CClientDC dc(this);
PIXELFORMATDESCRIPTOR pfd;
memset(&pfd, 0, sizeof(PIXELFORMAT_DESCRIPTOR));
pfd.nSize=sizeof(PIXELFORMAT_DESCRIPTOR));
pfd.nVersion=1;
pfd.dwFlags=PFD_DOUBLEBUFFER|PFD_SUPPORT_OPENGL|
            PFD_DRAW_TO_WINDOW;
pfd.iPixel|Type=PFD_TYPE_RGBA;
pfd.cColorBits=24;
pfd.cDepthBits=32;
pfd.iLayerType=PFD_MAIN_PLANE;
```

dwFlags 字段值的含义是使用 OpenGL(PFD_SUPPORT_OPENGL)，在窗口的客户区上绘制(PFD_DRAW_TO_WINDOW)，支持双缓存图像(PFD_DOUBLE_BUFFER)。iPixelType 字段被设置为 PFD_TYPE_RGBA，表示 OpenGL 将用红绿蓝(RGB)分量模式而不是颜色索引模式来规定颜色。cColorBits 字段表示每个像素的颜色数据所占用的位数，这里设置为 24，如果系统支持，则将有最佳显示效果。cDepthBits 字段则表示每个像素的深度数据所占用的位数。在 OpenGL 中，深度缓存负责存储深度数据并提供隐藏面消隐机制。因为对每个像素来说，深度缓存包含着像素与观察者的距离。当 OpenGL 绘制一个物体时，它将每个新像素的位置与深度缓存中所存储的位置相比较。如果新像素距离观察者较近，它将被放置在屏幕上，并且深度缓存被更新。反之则不被写到屏幕上。

像素格式的选择则由 Win32ChoosePixelFormat()函数完成，如下所示：

```
int nPixelFormat=ChoosePixelFormat(dc.m_hDC,&pfd);
```

ChoosePixelFormat()函数将 Windows 所支持设备像素格式与 pfd 所描述的像素格式进行比较，并返回最佳匹配的像素格式的可能索引值。此索引值并非唯一，并且可根据当前显示模式改变。

可以自由地检查 ChoosePixelFormat()所推荐的像素格式，并且可以在不喜欢时重新选择。当选定了 ChoosePixelFormat()所推荐的像素格式之后，就可以使用 SetPixelFormat()函数进行设置，如下所示：

```
BOOL bResult=SetPixelFormat(dc,m_hDC,nPixelFormat,&pfd);
```

可以看到，SetPixelFormat()函数将设备环境的句柄作为第一个参数，所以能将与设备环境相关的窗口设置成选中的像素格式。

已经为设备环境设置了像素格式，接下来，就是要依据设备环境创建 OpenGL 绘制环境(GLRC)。显然，两个环境具有同样的像素格式，但设备环境与绘制环境是不同的：设备环境包含 GDI 信息，而绘制环境包含 OpenGL 信息。一个程序可以创建多个绘制环境。wglCreateContext()函数负责创建绘制环境并返回一个绘制环境句柄 HGLRC。例如：

```
m_hc=WglCreateContext(dc,m_hDC);
```

如果从 ChoosePixelFormat()所返回的像素格式中 dwFlags 中的 PFD_NEED_PALETTE 标志被设置，那么需要创建一个调色板。对于类属 OPenGL 像素格式来说，必须是 3-3-2 调色板，即 8 个数位被分成 3 位红，3 位绿，以及 2 位蓝。

iii) OnSize()

窗口大小变动时会触发消息句柄 Onsize()。在此函数中，目的是建立 3DOpenGL 坐标与 2D 屏幕坐标之间的映射，体现为做三件事:获取当前绘制环境；设置映射方式；屏蔽当前绘制环境。

一个程序可以不止有一个绘制环境，所以 OpenGL 采用当前绘制环境的概念，只向当前绘制环境进行写操作。如果某个绘制环境未被设置成当前的，则 OpenGL 调用不做任何事情，wglMakeCurrent()函数被用于设置当前绘制环境，如下所示：

```
BOOL bResult=wglMakeCurrent(dc,m_hDC,m_hrc);
```

OpenGL 大量使用矩阵运算，因为场景到屏幕的变换，以及 3D 图形的 3D 旋转、平移和缩放都是采用矩阵变换实现的。相应地 OpenGL 支持两个用于矩阵变换的矩阵栈。一个是投影栈，用于建立 3D 坐标到 2D 屏幕的映射模式；另一个是物体栈，用于场景中物体的变换，将在 OnDraw()函数中使用。

在 OnSize()函数中，使用投影栈来设置观察物体的方法时，总共用到如下所示的四个函数：

```
Gldouble gldAspect=(GLdouble)cx/(GLdouble)cy;
glMatrixMode(GL_PROJECTION);
```

```
glLoadIdentity();
gluPerspective(30,0,gldAspect,1.0,10.0);
glViewport(0, 0, cx, cy);
```

其中 glMatrixMode()命令用来指定所使用的矩阵栈，这里参数 GL_PROJECTION 指出将使用投影栈。glLoadIdentity()命令通过装载单位矩阵来清空矩阵栈，这是因为任何新添加到栈上的变换都与前面的变换进行合成，所以在适当时需要栈的清空操作。用户库函数 gluPerspective()用于设定用户的可见区域，它的四个参数分别是:视角、视口纵横比、最近剪裁平面以及最远剪裁平面。在上面的例子中，视角设为 30°，纵横比则根据窗口客户区的大小来调整，较近的剪裁平面设置为距离视点 1 单位，较远的剪裁平面则设置为距离视点 10 单位。glViewport 用来设置在客户区上的绘制区域，在上面的例子中，设置为在整个客户区上绘制。

在使用多个绘制环境的情况下，即将离开 OnSize 时，应调用 wglMakeCurrent()来屏蔽当前绘制环境：

```
wglMakeCurrent (NULL, NULL);
```

iv) OnEraseBkgnd()

重载 OnEraseBkgnd()并且返回 TRUE，可以消除额外的屏幕闪烁，因为此举将防止程序在 OpenGL 绘制屏幕之前将屏幕刷成白色.

v)OnInitialUpdate()

如果需要利用由若干 OpenGL 命令组成的 OpenGL 显示列表,则应重载消息句柄 OnInitialUpdate()。使用 OpenGL 显示列表可以提高显示速度，因为它可以存储列表中各 OpenGL 命令累计变换的最终结果,而不必在每次调用时重新计算。

vi) OnDraw()

此函数真正负责绘制 3D 图像，包括以下步骤:

在 DC 中选择并实现调色板：调用 wglMakeCurrent()使绘制环境成为当前；调用 OpenGL 命令绘制场景；如果使用双缓存像素格式，交换绘制缓存；重新为 DC 选择原始的调色板。

因为 wglMakeCurrent()基于当前的逻辑调色板初始化绘制环境，所以应在调用 wglMakeCurrent()之前选择调色板。此外，缓存的交换使用 Win32GDI 函数 SwapBuffers()； glMatrixMode(GL_MODELVIEW)命令指出将使用物体栈。具体实现如下所示:

```
CPalette *pPalOld=pDC->SelectPalette(&m_Pal,0); //选择调色板
PDC->RealizePalette();  //实现调色板
BOOL bResult=
   wglMakeCurrent(pDC->m_hDC, m_hrc); //使 HGLRC 成为当前的
```

… //调用 OpenGL 命令允许光照计算，深度计算缓存

… //调用 OpenGL 命令清除颜色及深度缓存

… //调用 OpenGL 命令设置颜色、材质属性等

glMatrixMode(GL_MODELVIEW)；//转向物体栈

…//调用 OpenGL 命令绘制场景

SwapBuffers(pDC->m_hDC)； //交换缓存

If(pPalOld) pDC->SelectPalette(pPalOld,0)；//恢复原调色板

wglMakeCurrent(NULL,NULL)；

vii) OnDestroy

此消息句柄负责清除在 OnCreate()中创建的绘制环境。

wglMakeCurrent(NULL, NULL)；

if (m_hrc)

{wglDeleteContext(m_hrc)； m_hrc=NULL； }[28]

（2）建立 OpenGL 与系统的接口类 CGL

C++类通过隐藏复杂性使编程变得容易。例如，一个 C++类能用 Create 成员函数建立正确的调色板。在仿真系统的诸多应用中，如:颜色索引模式、优化和在位图中绘图等，都要用到 OpenGL 初始化程序。如果我们将 OpenGL 代码设计在 C++类中，将会对我们的程序编制带来极大的方便。于是，我们建立一个 CGL 通用类，以免在不同的地方修改同一错误。OpenGL 类的建立，将简化 OpenGL 应用程序的编写。CGL 不必隐藏或改变标准的 OpenGL 代码，但应隐藏那些非标准的 Windows 9x/NT 实现细节。CGL 封装了所有 OpenGL 程序所需的资源描述表、设备描述表和调色板。因此，CGL 成为唯一包含 Wmdows9x/NT OpenGL 实现细节的类，标准的 OpenGL 代码放在另外的类中[27]。

设计 OpenGL 类时，常用两种结构：函数封装和结构封装。这两种结构并非互相排斥，因为函数封装可以是结构封装的一部分。

①函数封装

函数封装比结构封装更象 MFC。在函数封装中，OpenGL 函数变成 OpenGL 类的成员函数，这些函数通过 OpenGL 类来调用。函数封装有以下优点:

i)与 CDC 类类似

函数封装使得 OpenGL 命令看起来象 MFC 所包括的 CDC 成员函数。

ii) OpenGL 命令面向对象化

C++用户喜欢通过面向对象调用函数。在函数封装中，OpenGL 命令变成 C++类的成员函数。OpenGL 命令通过对象调用，看起来更面向对象。

iii)通过函数重载减少命令的类型

OpenGL 命令接受多种参数类型。如 glColor()有多种形式：glColor3b()，

glColor3d() ,glColor3i(), glColor4f()和 glColor4uiv()。通过 C++函数重载，就能使函数形式为一种形式:glColor()或 Color()。

iv）封装 float 型函数，可接受 double 型参数

许多 OpenGL 命令接受 float 型参数。缺省情况下，Visual C++编译器区别对待 float 型和 double 型。若将 double 型参数赋给一个 float 型参数，将会产生一个警告。为避免警告，就必须在数字的后面加上“f”。因此，可以建立接受 double 型的函数来代替接受 float 型的函数。

v）提供跟踪错误的方法

函数封装提供了跟踪错误的方法。运行 OpenGL 命令的调试版本，就可以打印遇到的错误信息。

②结构封装

在结构封装中，OpenGL 程序的结构被封装在类中，隐藏了 Windows9x/NT 的实现细节。OpenGL 命令不作为 OpenGL 类的成员函数，而函数封装把 OpenGL 命令作为其成员函数。

在仿真程序中，程序的 OpenGL 代码包含在 OnCreate(), OnSize(), OnInitialUpdate(), OnDraw()函数中。因为，封装了 OpenGL 的程序结构，在 OpenGL 类中提供对应于这些函数的成员函数是很显然的。CGL 类的简单定义如下:

```
class CGL
{public:
   BOOL Create();  BOOL Init();  BOOL Size();
   BOOL Render();
   void Destroy(); }
```

成员函数通过视类中相应的消息句柄来调用，如图 3.5 所示。

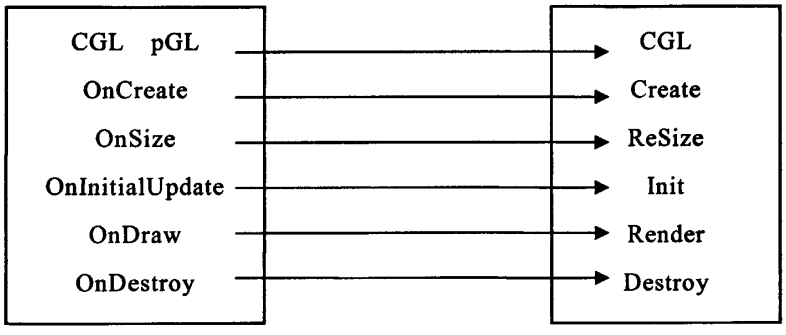

图 3.5 CMyView 的消息句柄调用 CGL 的成员函数

在设计中，CGL::Create()建立绘图描述表、设备描述表和调色板。CGL::Destroy()执行相关的清理任务。CGL::ReSize()实现图形到屏幕的投影。CGL::Init()初始化 OpenGL 参数和建立显示列表。CGL::Render()绘制图形。

在仿真程序中，OnSize()隶属于 CMyView 类，其函数代码如下所示：

```
void CMyView::OnSize(UINT nType, int cx, int cy)
{
  CMyDoc* pDoc = GetDocument();
  CView::OnSize(nType, cx, cy);
  cx1=cx;
  cy1=cy;
  if ((cx <= 0)||(cy <= 0) )
   return;
  else
   m_pGL->Resize(cx,cy);
}
```

在仿真程序中，CGL 类的 ReSize()函数中的代码如下：

```
void CGL::ReSize(int cx,int cy)
{
  if(cy==0)
   cy=1;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glViewport(0,0,cx,cy);
  glOrtho(-cx/5*m_Zoom,4*cx/5*m_Zoom                    cy*2/3*m_
          Zoom,cy/3*m_Zoom,-500,500);
  glViewport(0,0,cx,cy);
  glMatrixMode(GL_MODELVIEW);
 }
```

CGL 类封装了 wglMakeCurrent()函数调用。在调用任何应用程序专有的 OpenGL 代码前，CGL 必须保证程序的绘图描述表被激活。

至此，构造好了 CGL 类，程序可以利用 OpenGL 进行画图。在程序中产生了一个 RC，自始自终都使用它。这与大多数的 GDI 程序不同。在 GDI 程序中，DC 在需要时才产生，并且是画完立刻释放掉。实际上，RC 也可以这样做；但是，产生一个 RC 需要很多处理器时间。因此，要想获得高性能流畅的图像和图形，最好只产生 RC 一次，并始终用它，直到程序结束。

CreateRC 产生 RC 并使之成为当前 RC。wglCreateContext()返回一个 RC 的句柄。在调用 CreateViewGLContext()之前，必须用 SetWindowPixelFormat()将与设备相关的像素格式设置好。wglMakeCurrent()将 RC 设置成当前 RC。传入此函数的 DC 不一定就是产生 RC 的那个 DC,但二者的设备句柄(Device Context)和像素格式必须一致。假如在调用 wglMakeforCurrent()之前已经有另外一个 RC 存在，wglMakeforCutrent()就会把旧的 RC 冲掉，并将新 RC 设置为当前 RC。另外，可以用 wglMakeCurent(NULL, NULL)来消除当前 RC。

3.7 用 OpenGL 对 NURBS 建模

下面是以对称的丘陵状渲染的一个 NURBS 曲面,采用了从-6 到+6 的控制点。

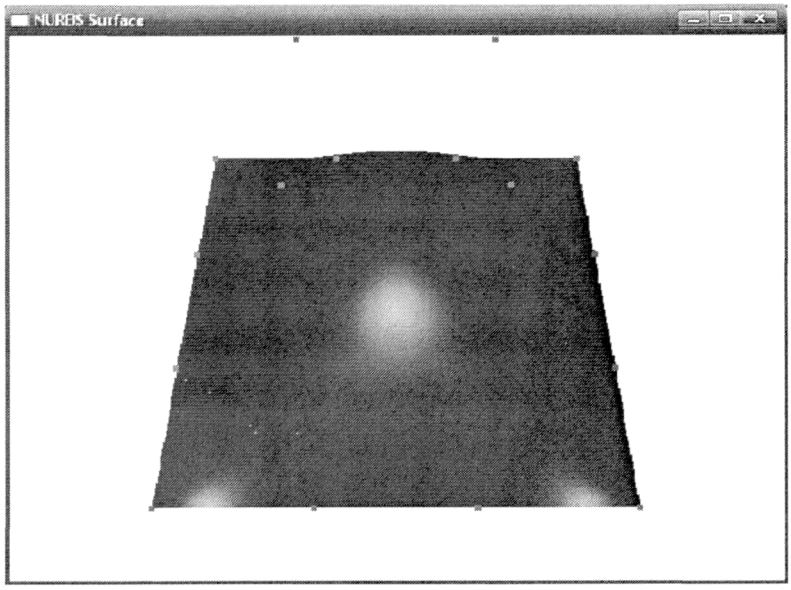

图 3.6 OpenGL 绘制的 NURBS 曲面

利用 VC++6.0 中的 OpenGL 库绘制 NURBS 曲面的主要程序段：

```
// 开始 NURBS 曲面定义
gluBeginSurface(pNurb);
//曲面求值
gluNurbsSurface(pNurb, // NURBS 着色指示器
8, Knots,  //节点数量和节点 u 向矩阵
8, Knots,  //节点数量和节点 v 向矩阵
4 * 3,     //u 向上两个控制点间的距离
3,     //  v 向上两个控制点间的距离
  &ctrlPoints[0][0][0], // 控制点
        4, 4,    // u 向和 v 向曲面表面节点的次序
  GL_MAP2_VERTEX_3);//曲面的二维求值程序，表示无理的控制点
// 曲面设置结束
```

gluEndSurface(pNurb);

3.8 本章小结

本章介绍了基于 Visual C++6.0 的 OpenGL（Open Graphics Library，开放图形库）的三维图形设计，OpenGL 作为一个优秀的开放式三维图形接口，为三维图形设计提供了便利。本章主要论述了 OpenGL 的函数库及在 MFC 中实现 OpenGL 图形编程，并通过实例，编译了 OpenGL 对 NURBS 曲面的绘制程序。

第 4 章　加工仿真系统建立

4.1 引言

计算机辅助设计和制造（CAD/CAM）工业是当今社会全球经济发展最迅速的领域之一，越来越多的制造公司以复杂的图形为基础，进行数字快速原型和全球范围内的 CAD/CAM。

数控加工是 CAD/CAM 中的一个重要组成部分。但是实现 CAD 和 CAM 的集成是一种非常复杂的工作，所以在实际的制造系统中，经过 CAD/CAM/NC 的零件，在正式加工之前，一般要经过试切这一阶段。试切的过程也就是对 CAD/CAM/NC 系统生成的 NC 程序的检验过程。一个相对复杂的工件需要大量的时间和费用在机床上试切来检验刀具路径。而且随着 NC 编程的复杂化，NC 代码的错误率也越来越高。如果 NC 程序生成不正确，就会造成过切、欠切，或加工出来废品，甚至发生零件与刀具、刀具与卡具、刀具与工作台的干涉和碰撞。传统的试切采用塑模、蜡模或木模在专用设备上进行，这不但浪费人力物力，而且延缓了生产周期，增加了产品开发成本，降低了生产效率，极大影响了系统性能。如果在计算机显示屏上仿真加工，检验数控加工程序代码，具有直观、快速、且不需要额外费用的优点，对缩短产品的试制周期、降低成本、提高数控加工效率，具有十分重要的意义。

OpenGL 渲染图形库是一个工业标准的三维计算机图形软件接口。用户可以方便地利用这个图形库,创建出接近光线追踪的高质量静止或动画的三维彩色图像。特别是 Visual C++等集成开发环境的出现，是的在计算机上用 OpenGL 实现高品质、交互式的三维图形更加方便。

4.2 仿真系统的原理及基本框架

4.2.1 系统设计的原则

为了缩短零件从加工到设计的开发周期，提高加工质量，减少制造费用，在设计仿真系统时遵守了以下原则:

(1)实用性:即设计的仿真系统应该具有实际的应用前景。

(2)先进性:即设计的系统在主要技术上应具有一定的先进性。

(3)可靠性:即系统能够准确地模拟数控加工过程，仿真结果可靠.

(4)可移植性:为了使研究成果进一步推广应用,系统采用标准的编程语言和图形软件进行编码实现，尽量减少对硬件的依赖性。

4.2.2 仿真系统的基本框架

基于第二章的分析，本课题采用直接实体造型法建立加工过程的仿真模型。为了便于分析和研究，将复杂的仿真功能模块大致分为两大核心部分：工件造型和仿真显示。工件造型主要以基本体素为直接的操作对象，用来对基本体素进行布尔（Boolean）运算和对毛坯进行离散化处理以及记录体素之间的相互关系。仿真显示主要提供三维图形显示界面，负责动画处理和图形显示。真实感仿真显示不仅在于它显示三维立体图形，而且在于它具有可设置的材料属性，即所加工的零件具有随毛坯材料变化的视觉效果。这些功能都是由动画处理完成。但是两大模块并不是相互独立的，在进行干涉碰撞检验时，必须两模块结合，共同完成，如图 4.1 所示。

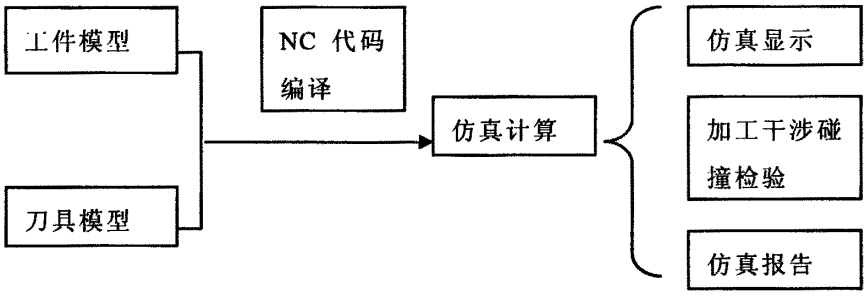

图 4.1 仿真系统构成

从上图可以构造出仿真模块的基本思想：

（1）对工件进行建模，将工件毛坯进行整体离散，采用将工件划分为很小的长方体体素来构造工件模型，并用链表（List）表示。

（2）根据铣刀半径的大小，用同样的方法，将铣刀离散成一系列的小长方体，从而建立铣刀模型，并同样用链表（List）表示。

（3）遍历毛坯链表，根据刀具的坐标，判断该长方体基本体素是否被加工，若刀具坐标（Z 坐标）小于该长方体基本体素 Z 坐标，则该体素被加工，仿真显示部分得到的即是工件毛坯在刀具按照刀位轨迹（NC 代码编译）走完一刀后的中间结果，然后显示中间结果的真实感图形。

4.3 仿真系统中实体模型的建立

4.3.1 毛坯实体模型的建立

对于零件的复杂曲面我们可以根据图 2.1 均匀离散模型的方法，对零件进行

毛坯构造，即选择复杂曲面的最高点（max Z）与零件的边界构造一长方体毛坯。

现实中，三轴联动数控铣床加工的特点[29-36]为①毛坯的上表面是唯一加工表面；②刀轴方向上的射线与加工表面的交点唯一；③加工工件的上表面可通过离散体素的不同高度来表达。鉴于三轴联动数控铣床的以上特点，可以将毛坯在机床坐标 Z 轴方向上离散。在实体造型的基础上，采用将毛坯划分为很小的长方体体素来实现这种如图 4.2 所示模型。当刀具在毛坯上运动的时候，只需要按照刀具路径修改毛坯上表面 Z 轴的坐标，就可以达到显示加工过程的效果。

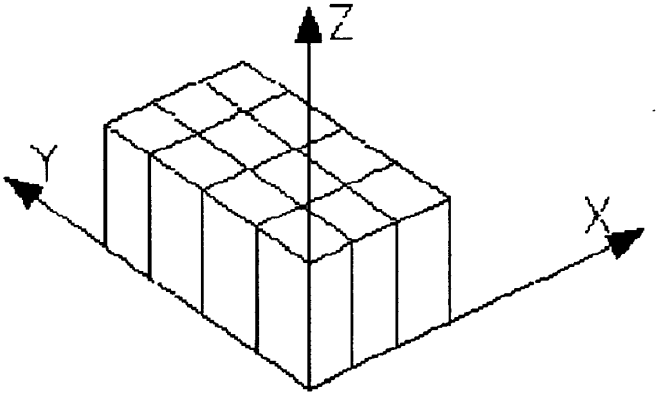

图 4.2 基于小长方体的毛坯模型

基于这个思想，本论文采用 M×N 阶的矩阵来表示工件毛坯，矩阵中每一个元素代表一个小长方体，元素的下标代表小长方体的位置，元素的值代表小长方体的高度，即：

$$B=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix} \qquad 4.1$$

举例说明，可以用 400×300 的矩阵来代表高度为 80 的长方体毛坯，即毛坯的长、宽、高分别为 400mm，300mm，80mm，那么工件毛坯矩阵为：

$$B=\begin{bmatrix} 80_{11} & 80_{12} & \cdots & 80_{1,300} \\ 80_{21} & 80_{22} & \cdots & 80_{2,300} \\ \cdots & \cdots & \cdots & \cdots \\ 80_{400,1} & 80_{400,2} & \cdots & 80_{400,300} \end{bmatrix} \qquad 4.2$$

这种基于小长方体的基本体素离散法，在程序中通过标准的链表结构来实现。表示这种结构的链表如下所示[37]：

```
typedef CTypedPtrList<CObList,CCube> CCubeList;
```

其中 CobList 用来记录个基本体素在工件毛坯矩阵中的相互位置关系，CCube 用来记录各基本体素的几何信息，其中基本体素类定义如下：

```
class CCube:public CObject
{
 public:
      CCube();
      CCube(double X,double Y,double t=0,int side=0,
            bool Origin=true);
      virtual~CCube();
      void ChangeHeight(double height);
      private:
      double m_dMax_X;      //最大的 X 坐标
      double m_dMax_Y;      //最大的 Y 坐标
      double m_dHeight;     //高度 Z
      int m_iSide;          //标志为可见面还是隐藏面
      bool m_bOrigin;       //标志是否被切削
};
```

4.3.2 刀具模型建立

根据上节对工件毛坯模型的建立原理，我们可以根据刀具的大小，用不同的 N×N 矩阵（4.3）来表示切削刀具，从而与毛坯进行比较，在切削过程仿真中用来进行材料去除判断。

以一把直径为 6 的铣刀为例，来说明铣刀模型的建立过程。

①将立铣刀离散成一系列的小长方体，用 7×7 的矩阵来代替刀具，如式 4.4 所示。

②为了建立刀具矩阵中的元素与刀具底面离散点的对应关系，建立一个 6×6 的方格，如图 4.3 所示。

$$T=\begin{bmatrix} t_{-n/2,-n/2} & t_{-n/2,-n/2-1} & \cdots & t_{-n/2,0} & t_{-n/2,1} & \cdots & t_{-n/2,n/2} \\ t_{-n/2-1,-n/2} & t_{-n/2-1,-n/2-1} & \cdots & t_{-n/2-1,0} & t_{-n/2-1,1} & \cdots & t_{-n/2-1,n/2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ t_{0,-n/2} & t_{0,-n/2-1} & \cdots & t_{0,0} & t_{0,1} & \cdots & t_{0,n/2} \\ t_{1,-n/2} & t_{1,-n/2-1} & \cdots & t_{1,0} & t_{1,1} & \cdots & t_{1,n/2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ t_{n/2,-n/2} & t_{n/2,-n/2-1} & \cdots & t_{n/2,0} & t_{n/2,1} & \cdots & t_{n/2,n/2} \end{bmatrix} \quad 4.3$$

$$T=\begin{bmatrix} t_{-3,-3} & t_{-3,-2} & t_{-3,-1} & t_{-3,0} & t_{-3,1} & t_{-3,2} & t_{-3,3} \\ t_{-2,-3} & t_{-2,-2} & t_{-2,-1} & t_{-2,0} & t_{-2,1} & t_{-2,2} & t_{-2,3} \\ t_{-1,-3} & t_{-1,-2} & t_{-1,-1} & t_{-1,0} & t_{-1,1} & t_{-1,2} & t_{-1,3} \\ t_{0,-3} & t_{0,-2} & t_{0,-1} & t_{0,0} & t_{0,1} & t_{0,2} & t_{0,3} \\ t_{1,-3} & t_{1,-2} & t_{1,-1} & t_{1,0} & t_{1,1} & t_{1,2} & t_{1,3} \\ t_{2,-3} & t_{2,-2} & t_{2,-1} & t_{2,0} & t_{2,1} & t_{2,2} & t_{2,3} \\ t_{3,-3} & t_{3,-2} & t_{3,-1} & t_{3,0} & t_{3,1} & t_{3,2} & t_{3,3} \end{bmatrix} \quad 4.4$$

每个方格的节点代表刀具上的一个小长方体，图中的圆可看成刀具的直径，圆内的每一个节点对应刀具头上的某一个小长方体，对于底面为平面的刀具，可将圆内对应的节点的元素值赋为 0，圆外各节点对应的元素赋为不可能出现的值。对于球头刀具，可根据刀具类型调用相应的子程序计算出对应圆内各点的元素值。

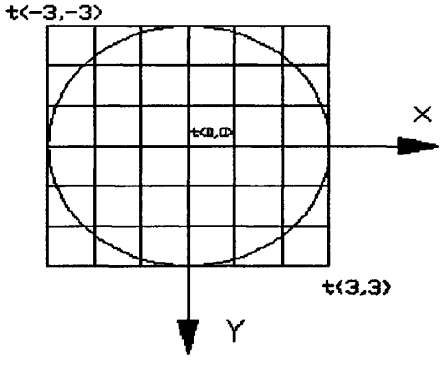

图 4.3 刀具离散方格图

这种基于小长方体的刀具切削模式在程序中也通过标准的链表结构来实现。刀具体的链表同工件毛坯的链表结构一样，如下所示[37]：

```
typedef CTypedPtrList<CObList,CCube> CCubeList;
```

不同的是还需要一个链表结构来表示刀具底面，这个刀具底面具有与刀具离散体相对应的离散点。刀具底面的链表结构如下所示：

```
CObList ToolDotList;
```

这个链表结构存储的数据实体是属于类 CmaxCor 的对象，类 CmaxCor 的结构如下所示：

```
class CMaxCor:public CObject
{
 public:
      CMaxCor();
      CMaxCor(double X,double Y);
      virtual~CMaxCor();
      double X;      //离散点的 X 坐标
      double Y;      //离散点的 Y 坐标
};
```

4.4 NC 代码编译

NC 代码编译是一个较为复杂的过程，同时也是数控加工仿真的重要部分，目前已有的数控仿真系统较多的是利用刀位数据作为仿真系统模拟运动的驱动代码[38]。目前数控系统种类繁多，一些性能良好的却大多说在国际标准出台之前就早已形成了自己的一套数控代码，尽管多数依据 ISO 和 EIA 标准，一般均有扩充，使得各系统的代码千差万别。为使仿真系统能够适应多种数控系统，并且能够真实地反映实际的加工状态，对数控代码的计算机识别分析方法、能力及准确程度都提出了较高的要求。对各系统的数控代码综合分析后发现，尽管在一些代码上存在着功能的差异，但有以下共同点：①数控程序段为典型的上下文无关文法，即语法单位可完全独立于其可能出现的环境；②数控代码语法规则简单，数量较少。基于以上分析和数控仿真目的，只需提炼与仿真系统运动部件有关的运动与状态信息，而对那些无法体现在仿真系统里的运动及状态信息，没有必要刻意的去分析，只要计算机能够识别并进行词法的检验就可以[39]。

数控代码的编辑属于文本编辑，可以用以下方法编辑数控程序[40]：

（1）基于 Windows 环境用 Visual C++6.0 在图形状态下开发全屏幕编辑器。其功能与 Windows 记事本类似，具有数控加工程序输入、输出、编辑、修改、存盘和删除等功能；

（2）用 GAPT 图形自动编程编辑。图形自动编程编辑器 GAPT 是以计算机辅助设计软件为基础，利用 CAD 软件的图形编辑功能将几何零件图形绘制到计

算机屏幕上，形成零件的图形文件，然后调用数控编程模块，采用人机交互的方式在计算机屏幕上选择被加工部件，在输入相应的加工参数，计算机便可以进行必要的数学处理，并绘制出数控加工程序，同时在屏幕上动态地显示出刀具的加工轨迹。

（3）用 Windows 系统的文本编辑器编辑。

编译程序完成从源程序到目标程序的翻译工作，其整个工作分阶段进行，每个阶段将源程序的一种表示形式转换成另一种表示形式，各阶段进行的操作在逻辑上紧密连接[41][42]，如图 4.4。.

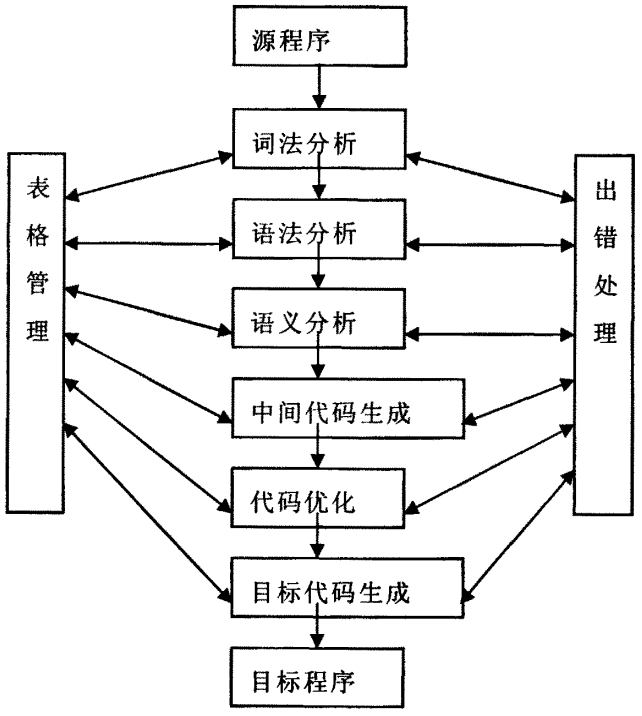

图 4.4 编译过程结构

4.4.1 数控预处理模块

预处理任务负责对零件处理的扫描与词法、语法识别，并将结果放入缓存区[43]，如图 4.5。

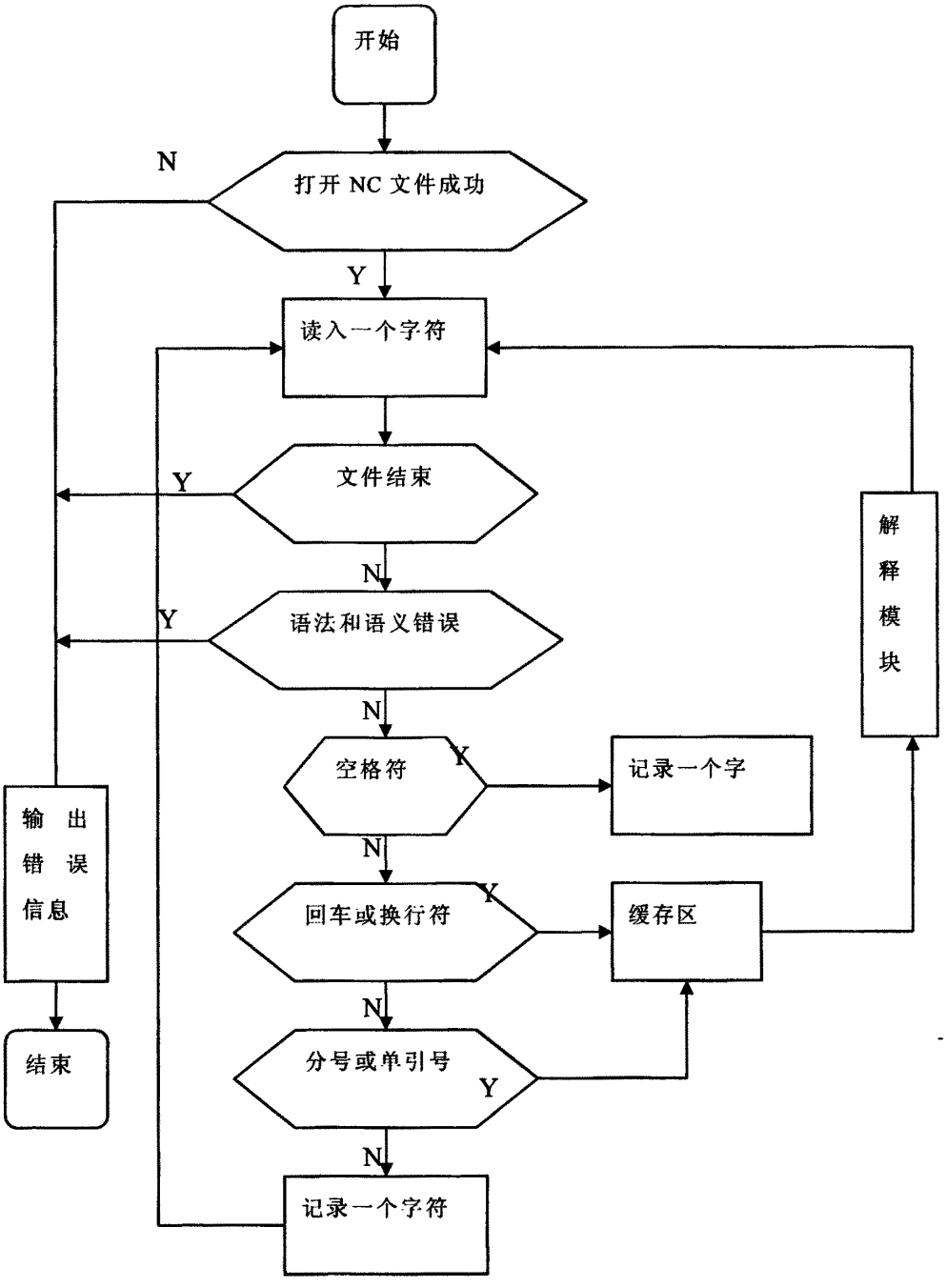

图 4.5 数控预处理模块

预处理模块首先对输入的数控程序采用单向链表结构进行组织管理，然后对保存在链表中的程序段进行识别技术处理，去除不必要的注释及回车符，形成仅含有功能代码字的标准程序段，在按照地址符转入形影的词法、语法判别检验处理分析。

词法分析是先按照标识符类型记录其后面的表达是字符串，再按各赋值方式进行分析。对数控加工程序进行语法语义等错误检查，可以保证后面的编译、仿真、加工进行顺利。如果有错误，则将错误代码位置记录到错误信息文件中，以待改正，最后将改正的程序存入缓存区。

4.4.2 数控解释模块

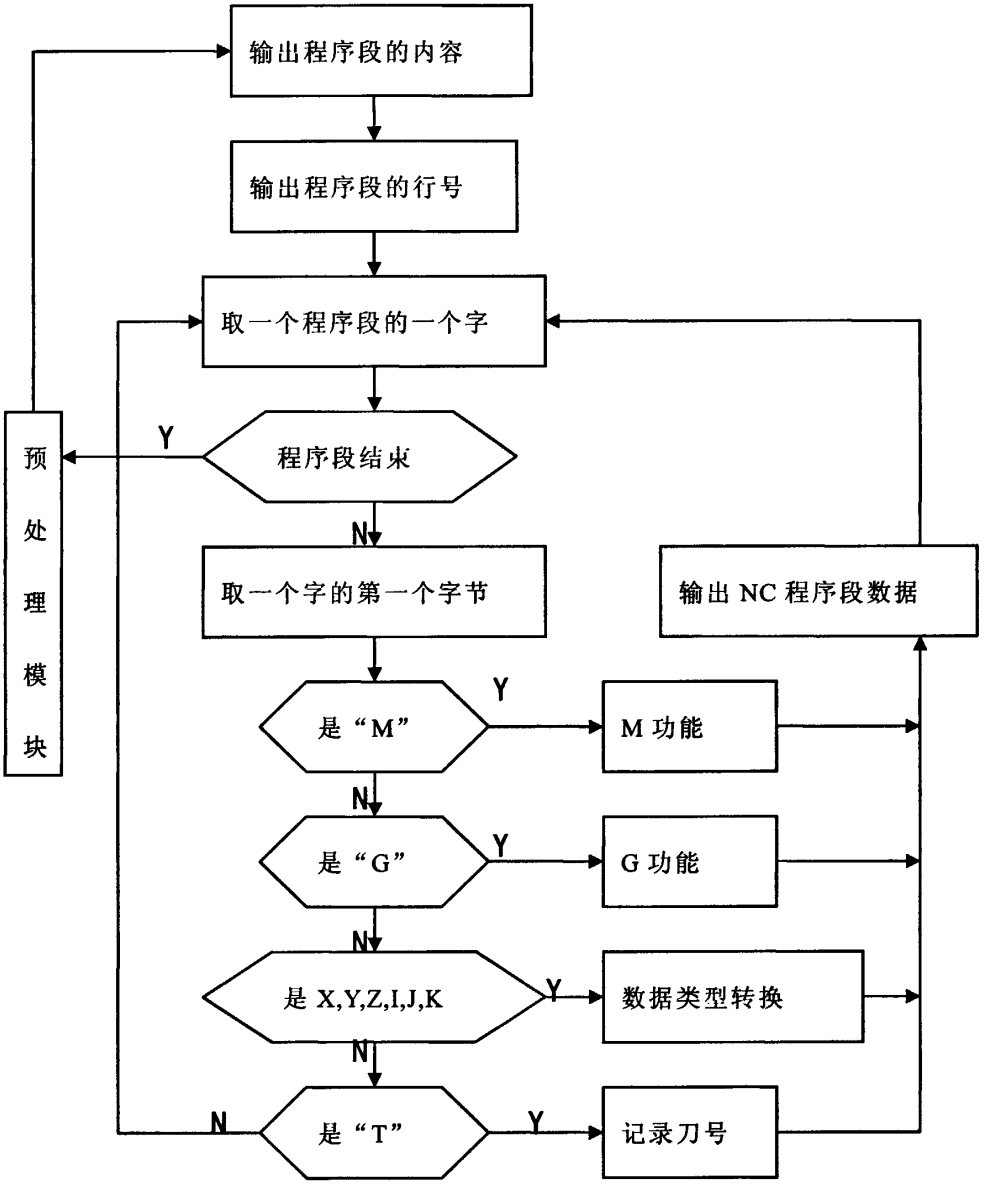

图 4.6 数控解释模块

数控解释模块完成对缓存区数据的扫描提取与分析，从而形成仿真驱动文件，保证了方针的及时性与高效性[43~45]，如图 4.6 所示。

解释模块负责提取有关命令动作和状态信息，并将运动的数据按位移和速度的变化划分成时间片段，从而驱动仿真系统模型的运动。通过对已经存入缓存区的数据结构进行分析扫描，提取与仿真有关的动作及状态信息并转换数据格式。

4.5 仿真动画显示原理

在 OpenGL 中的所有数据包括几何顶点数据和图像数据都可以被存储在显示列表中，或者立即可以得到处理。OpenGL 中，显示列表技术是一项重要的技术。OpenGL 要求把所有的几何图形单元都用顶点来描述，这样运算器和逐个点计算操作都可以针对每个顶点进行计算和操作，然后进行光栅化形成图形碎片；对于图像数据，象素操作结果被存储在纹理组装的内存中，再与几何顶点操作一样光栅化形成图形片元。图形片元都要进行一系列的逐个片元操作，这样最后的象素值送入帧缓冲器实现图像显示。

动画技术是利用计算机生成一系列可供动态演播的连续图像的计算机图形技术。人眼对观察到的对象所产生的视觉残留时间为 1/25s，动画技术正是利用视觉残留的特点使不连续画面看起来好像是连续的[27]。计算机动画的实现方式有 3 种：逐帧动画，位块动画和实时动画。本论文采用实时动画方式，首先生成读入毛坯链表，然后根据机械加工时的材料去除规律生成动画的每一帧。

另外，OpenGL 中实现动画使用了双缓存技术。在绘图前先分配前后 2 个缓存区，绘制时先将图形绘制到后台缓冲区中，然后通过交换前后缓存区（auxSwapBuffers），将后台缓存区中已经绘制好的图形直接送到前台缓存区，有显示设备完成图像的屏幕显示；此时应用程序已经在后台缓存区中绘制下一幅图像了。如此反复，屏幕上总可以显示已经绘制好的图像，而看不到绘制的过程。

整个加工仿真动画演示的步骤为仿真模块首先定时地从 NC 代码编译器的处理结果中取出下一步的插补运动位置，重新进行工件和刀具的切削运算，根据运算结果更新工件实体模型数据链表中的信息，然后调用实体显示程序，获取链表中的数据并将工件的模型重新生成在后缓存区，最后通过交换前后缓存区将新的工件实体显示在屏幕上。要想让模型连续地按预定轨迹运动起来，首先需要一个脉冲发生器，让一个个脉冲来驱动模型不断地运动，采用 TTimer 控件做脉冲发生器，每触发一个 OnTimer 事件，就让模型读取一行数据，然后根据数据的运动，以达到仿真加工运动的效果。模型读取的数据，保存在 NC 程序处理类 TMachiningProcess 生成的临时数据文件中，该文件按一定格式写入，然后按一定格式读出。当刀具运动一步后，判断零件是否有可能被切割，如果被切割，就更新零件曲面上对应离散点的坐标值，实现仿真零件切削加工的效果。

4.6 复杂曲面数控加工中干涉检验

干涉现象一般发生在加工曲面曲率骤变，切削不连续和表面存在间隙的情况下[46~49]。根据产生干涉的被加工曲面几何特性，可将干涉大致分为以下三类：

（1）单面干涉 主要有“曲率干涉”和“曲面干涉”两种情况，它们都是发生于单面元素内的干涉。“曲率干涉”是由曲面局部曲率特性引起的，当刀触点出的曲面曲率半径小于刀具的有效切削半径时，这种干涉就比较容易出现。“曲面干涉”是指非球面刀底部刀刃切入该点附近曲面区域，是非球面刀加工的一种特有的干涉。

（2）面间干涉 一种由于加工曲面不连续和表面存在间隙所引起的组合曲面元素之间的干涉，主要发生在加工相邻曲面在凹向拼接处或间隙拼接处。例如，曲面不连续发生于组合曲面 S1 和 S2 的拼接处。在加工曲面 S1 时，刀具轨迹对该曲面本身是无干涉的，但刀具位置对 S2 曲面来说却发生了干涉。

（3）运动干涉 这种干涉又称“凸干涉”，是在加工凸曲面时用于刀具作直线插补运动引起的。当刀具在凸曲面上两相邻刀位点之间作直线插补运动时，刀具会因直线运动误差过大而过切加工曲面，这种干涉一般发生于凸曲面高曲率骤变或凸曲面不连续的情况下。它们产生的加工误差即为相应的走到运动误差。

干涉检验通过将刀具简化为圆柱，根据刀具半径和刀具长度来判断是否产生干涉[50]。如图 4.7 所示，切削刃的长度是 h_e,它限制了侧面切削的深度 h_1。当 h_1 值大于最大切削深度 h_{max} 时，就会发生干涉。最大切削深度 h_{max} 及未切削面与刀心之间在（i,j）处的切削深度的计算公式如下：

$$h_{max} = h_e - R$$

$$h_1 = Z_{ij} - Z_c \qquad 4.5$$

式中 R——刀具半径

Z_{ij}——(i,j)位置的 Z 值

Z_c——(i,j)位置刀心的 Z 值

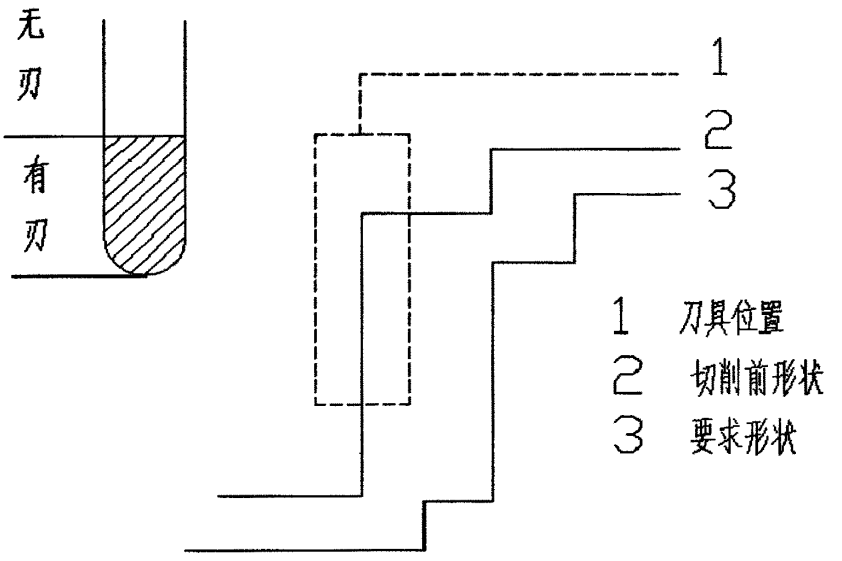

图 4.7 干涉检验

当数控加工发生干涉时，仿真动画演示读取 NC 代码自动跳出，仿真加工停止如图 4.8。

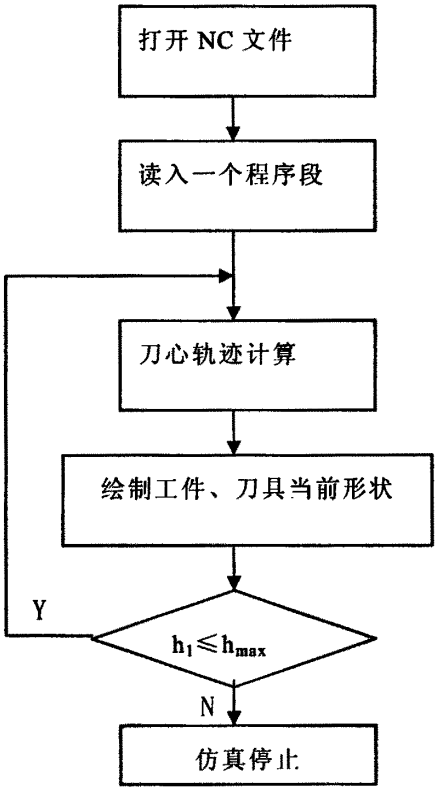

图 4.8 干涉时，仿真流程图

为了避免干涉，实现有效的复杂曲面加工，提高复杂曲面的数控加工效率，

可以将复杂曲面划分为不易被加工的关键区域和可加工区域[51~54]。通过此方法修改刀具路径，使用较大曲率的刀具跳过不易加工的关键区域加工可加工区域以避免过切发生，之后选择合适的小表面曲率刀具来加工关键区域。

下面确定干涉区域，设任意自由曲面的方程为[55]

$$S(u,v)=[x(u,v),y(u,v),z(u,v)] \qquad 4.6$$

式中 u, v 是参数。

则自由曲面 $S(u,v)$ 上任意一点 (u,v) 处的法向量为

$$n=\frac{\frac{\partial S}{\partial u}\times\frac{\partial S}{\partial v}}{\left|\frac{\partial S}{\partial u}\times\frac{\partial S}{\partial v}\right|} \qquad 4.7$$

平面曲率 H 的表达式为

$$H=\frac{EN-2FM+GL}{2(EG-F^2)} \qquad 4.8$$

高斯曲率 K 为

$$K=\frac{LN-M^2}{EG-F^2} \qquad 4.9$$

其中 E、F、G 是任意自由曲面一次矩阵 A 的组成元素

$$A=\begin{bmatrix}\frac{\partial S}{\partial u}\frac{\partial S}{\partial u} & \frac{\partial S}{\partial u}\frac{\partial S}{\partial v}\\ \frac{\partial S}{\partial v}\frac{\partial S}{\partial u} & \frac{\partial S}{\partial v}\frac{\partial S}{\partial u}\end{bmatrix}=\begin{bmatrix}E & F\\ F & G\end{bmatrix} \qquad 4.10$$

L、M、N 为曲面二次矩阵 B 的组成元素

$$B=\begin{bmatrix}\vec{n}\frac{\partial^2 S}{\partial u^2} & \vec{n}\frac{\partial^2 S}{\partial u\partial v}\\ \vec{n}\frac{\partial^2 S}{\partial v\partial u} & \vec{n}\frac{\partial^2 S}{\partial v^2}\end{bmatrix}=\begin{bmatrix}L & M\\ M & N\end{bmatrix} \qquad 4.11$$

从而可以计算取复杂曲面任意点的曲率：

$$K_{min}=H-\sqrt{H^2-K} \qquad 4.12$$

$$K_{max} = H + \sqrt{H^2 - K} \qquad 4.13$$

根据高斯曲率 K 和平均曲率 H 的值可以判断曲面曲率变化较大区域的类型（见图 4.9），而 K、H 与曲面形状的关系见表 4.1。

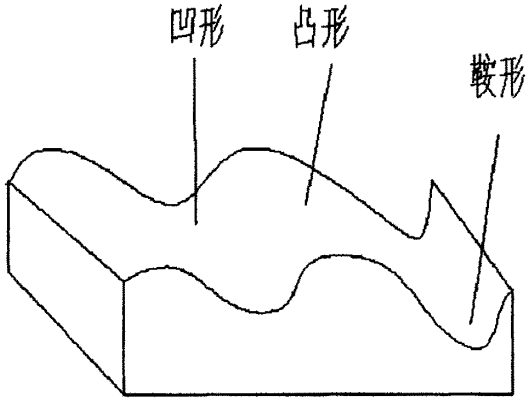

图 4.9 具有凹形、凸形、鞍形的曲面

表 4.1 K、H 与曲面形状的关系

K 值	H 值	表面形状
K>0	H>0	凹形
	H<0	凸形
K<0		鞍形
K=0	H>0	凹形
	H≤0	凸形

在干涉检验过程中，首先在曲面上选取一系列点，并根据式 4.11 计算出这些点的 L、M、N 的值。。由于普通表面（$EG-F^2$）> 0[55]，由式 4.9 可知，K 的符号与($LN-M^2$)的符号一致，则可以用 $K' = (LN - M^2)$ 的值判断表面形状。因此，由表 4.1 可知，如果 $K' < 0$，则 $K < 0$，所取样点处为鞍形。曲面上 $K' = 0$ 的样点轨迹把曲面分成不同类型的区域。在由 $K' = 0$ 样点形成的边界包围具体区域上，通过计算H 值来判断是否为凸形或凹形区域。这里特别指出的是，如果 $K' = 0$ 的样点形成的不是曲线而是一个区域，则这个区域需要根据 H 值来划分成不同的区域类型，接着计算各凹形、凸形和鞍形区域样点处的曲率，并找出主曲率值（曲面最大曲率 Kmax 和最小曲率 Kmin），最后通过比较零件表面主曲率值和刀具

主曲率值来判断是否为干涉区域，如表 4.2 所示[2][56][57]。

图 4.2 干涉和主曲率之间的关系

零件表面主曲率(K)	刀具表面主曲率(C)	干涉条件	干涉可能性
Kmin≤Kmax≤0	Cmin<Cmax≤0	无	不可能
Kmax>0≥Kmin	|Cmin|≥|Cmax|≥0	|Cmax|>Kmax	不可能
Kmax≥Kmin>0	|Cmin|≥|Cmax|≥0	|Cmax|>Kmax	不可能
		Cmin>|Kmin|	肯定

4.7 仿真实例

为了验证仿真系统的正确性，本文以整体叶轮叶片（如图 4.10 所示）为例，研究了叶片曲面的建模，刀具的建模，及仿真加工的效果。

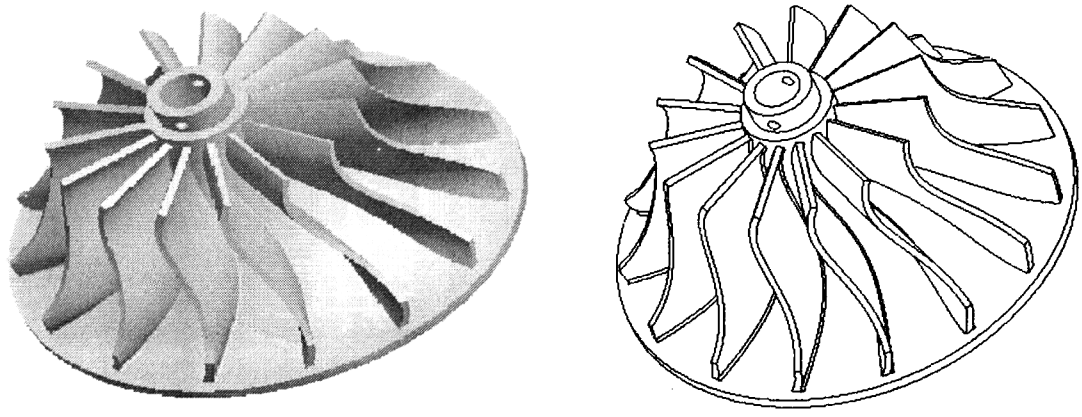

图 4.10 整体叶轮叶片

4.7.1 叶片曲面构型的原理及方法

在没有零件图纸的情况下，我们采用逆向工程，运用三坐标测量仪测量出叶片中性面上两条空间曲线的离散点坐标和各点的厚度值[58][59]。在已知数据的基础上，研究其构型方法。用 B 样条曲线对测出的离散点进行插值计算。

假设给定中性面轴盘曲线上 n+1 个数据点 $p_i=(x_i,y_i,z_i,h_i)$，（i=0,1,2,…,n），其中，x_i,y_i,z_i 为 p_i 点的坐标值，h_i 为该点对应的叶片厚度的一半。则三次 B 样条曲线方程

$$r_{hub} = p(u) = \sum_{j=i-3}^{i} d_j N_{j,3}(u) \qquad u \in [u_i, u_{i+1}] \subset [u_3, u_{n+3}] \qquad 4.14$$

其中，d_j j=0, 1, 2, …， n+2 为控制顶点

$N_{j,3}(u)$ j=0, 1, 2, …， n+2 为 3 次规范 B 样条基函数。

运用累加弦长参数化方法进行参数化，令控制多边形的各边边长依次为 $l_i=\|p_i-p_{i-1}\|_2$，i=1,2,…,n。总的边长为 $L=\sum_{i=1}^{n} l_i$，则三次 B 样条曲线的节点矢量为 $U=\left[0,0,0,0,\frac{l_1}{L},\frac{l_1+l_2}{L},\cdots,\frac{\sum_{i=1}^{n-1} l_i}{L},1,1,1,1\right]_{n+7}$，将与数据点 p_i 相对应的参数值 u_{3+i},i=0,1,…,n 代入式 4.14，同时又有两个切矢边界条件：

$$d_1=p_0+\frac{u_4-u_3}{3}(p_1-p_0) \qquad 4.15$$

$$d_{n+1}=p_n+\frac{u_{n+3}-u_{n+2}}{3}(p_n-p_{n-1}) \qquad 4.16$$

则联立式（4.14）（4.15）（4.16），可求得 n+3 个控制点，至此则可给出插值 n+1 个数据点的轴盘曲线 r_{hub}。同理，可求得叶轮的盖盘曲线 r_{shroud}，如图 4.11 所示。

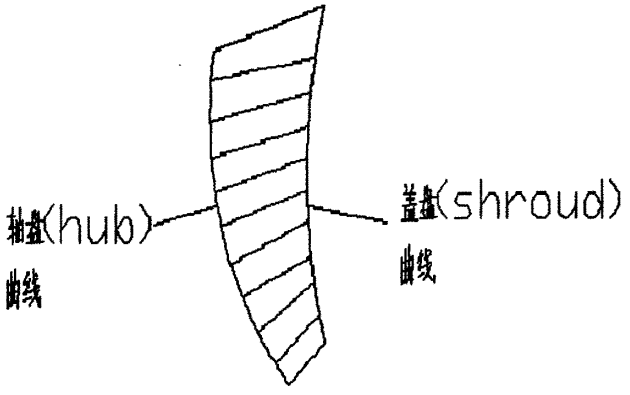

图 4.11 叶片中性面

将轴盘曲线和盖盘曲线上的对应点相连，则形成叶片中性面，它为一扭曲的非可展直纹面，其方程为

$$r_{中性面}=r_{hub}+(r_{shroud}-r_{hub})v \qquad 4.17$$

式中：$v\in[0,1]$ 为直母线参数。

在对盖盘和轴盘上离散点进行 B 样条曲线插值的基础上，根据叶片沿叶轮径向厚度的变化规律，设计出叶片的左、右廓面。由于叶片沿叶轮的径向和叶片的高度方向其厚度均在变化，在构造叶片曲面时，分别做叶片中性面上盖盘曲线和轴盘曲线的法线，并在此法线上截取相应的叶片厚度的一半，则可形成盖盘曲线和轴盘曲线的变距等距曲线，将两条曲线上的对应点相连则形成叶片曲面，如图 4.12 所示。

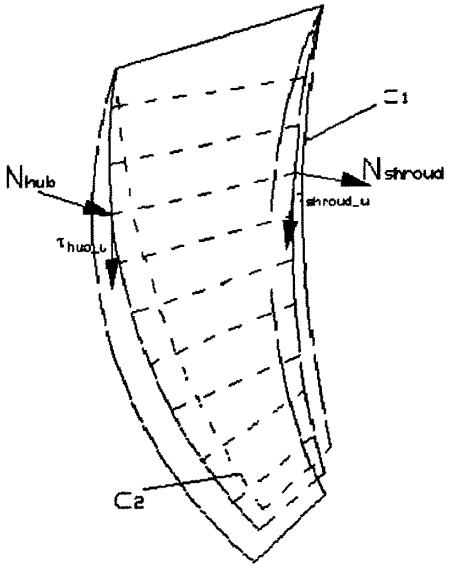

图 4.12 叶片曲面

其中 C_1, C_2 为中性面上盖盘曲线和轴盘曲线的变距等距曲线，τ_{shroud_u}，τ_{hub_u} 分别为中性面上盖盘曲线和轴盘曲线的切矢，而盖盘曲线和轴盘曲线的法矢为

$$N_{shroud}=\tau_{shroud\_u}\times(r_{shroud}-r_{hub}) \qquad 4.18$$

$$N_{hub}=\tau_{hub\_u}\times(r_{shroud}-r_{hub}) \qquad 4.19$$

则盖盘曲线和轴盘曲线的单位法矢可表示为

$$n_{shroud}=\frac{N_{shroud}}{\|N_{shroud}\|_2} \qquad 4.20$$

$$n_{hub}=\frac{N_{hub}}{\|N_{hub}\|_2} \qquad 4.21$$

则有

$$C_1 = r_{shroud} \pm h_{shroud} \bullet n_{shroud} \qquad 4.22$$

$$C_2 = r_{hub} \pm h_{hub} \bullet n_{hub} \qquad 4.23$$

式中，h_{shroud}，h_{hub}分别为叶片顶部和根部曲线对应点的厚度值的一半，其值分别为两曲线参数 u, u_1 的函数，式中的正负号分别对应叶片曲面的左右轮廓。

综上所述，叶片曲面的方程可表示为

$$r_{叶片} = C_2 + (C_1 - C_2)v \qquad v \in [0,1] \qquad 4.24$$

综上，利用 OpenGL 绘出叶片模型，如图 4.13 所示。

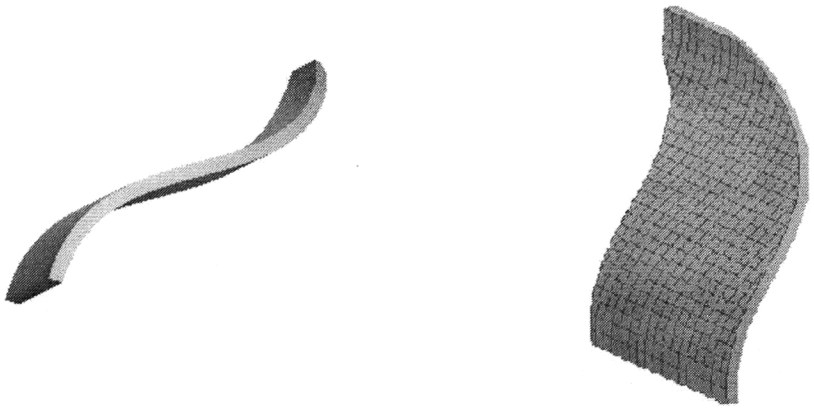

图 4.13 叶片模型及网格划分

4.7.2 刀具的建模方法

（1）刀具半径的计算

用三坐标数控铣床加工自由曲面，一般用球头刀采用行切法进行。用球头铣刀加工的优点是加工时对曲面的法矢有自适应的能力，能够满足一定的加工需要，编程简单，计算量也相对减少。但这种加工方法的切削不是很有效，切削速度随刀具与工件接触点的变化而变化，越接近球头刀的底部，切削速度越趋于零，切削条件越差；同时被加工出来的表面不平度较大，需要增加手工打磨工序，虽然可以增加截面的数目以提高加工表面质量，但生产效率也就相应降低。

设工件曲率半径为 r，行距为 S，残留断面高度为 H。如图 4.14 所示。

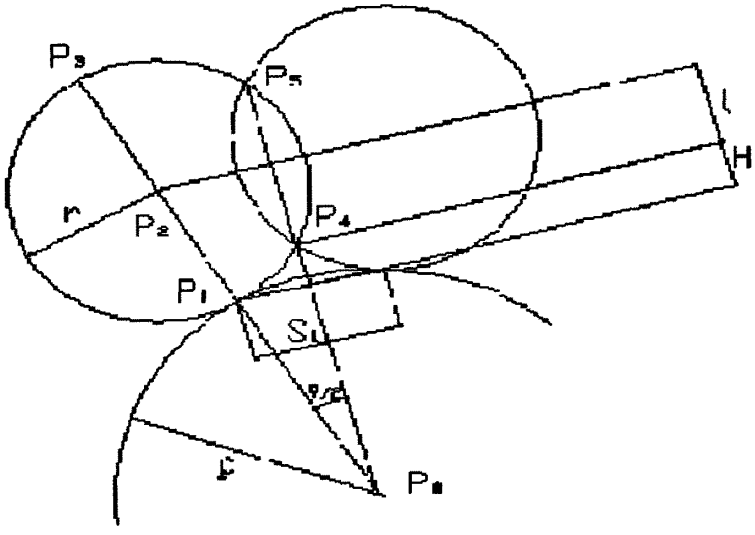

图 4.14 残留断面高度计算

从上图可知：

$$P_0P_1 \bullet P_0P_3 = P_0P_4 \bullet P_0P_5 \qquad 4.25$$

即：
$$\rho(\rho+2r)=(\rho+H)(\rho+H+2l) \qquad 4.26$$

又：
$$\cos\frac{\phi}{2}=\frac{\rho+H+l}{\rho+r} \qquad 4.27$$

将式 4.26 和式 4.27 联立可知：

$$\rho(\rho+2r)=(\rho+H)[2(\rho+r)\cos\frac{\phi}{2}-\rho-H] \qquad 4.28$$

从而：
$$\cos\frac{\phi}{2}=\frac{\rho^2+\rho r+\rho H+H^2/2}{(\rho+H)(\rho+r)} \qquad 4.29$$

又因：
$$\sin\frac{\phi}{2}=\frac{s}{2\rho} \qquad 4.30$$

则有：
$$(\frac{\rho^2+\rho r+\rho H+H^2/2}{(\rho+H)(\rho+r)})^2+\frac{s^2}{4\rho^2}=1 \qquad 4.31$$

将式 4.31 展开，略去 H^2 及 H^4 项得：

$$H=s^2(\frac{1}{8r}+\frac{1}{8\rho}+\frac{H}{8\rho r}+\frac{H}{4\rho^2}) \qquad 4.32$$

考虑到：$H \ll \rho$ ，　$H \ll r$

有：
$$H=s^2(\frac{1}{8r}+\frac{1}{8\rho}) \qquad 4.33$$

从而：
$$r=\frac{1}{\frac{8H}{s^2}-\frac{1}{\rho}} \qquad 4.34$$

考虑到加工表面的质量和生产效率，设允许残留高度为 H_1，行距为 S_1，则由式 4.33 和式 4.34 得：

$$H=s_1^2(\frac{1}{8r}+\frac{1}{8\rho})\le H_1 \qquad 4.35$$

即
$$r\ge\frac{1}{\frac{8H_1}{s_1^2}-\frac{1}{\rho}} \qquad 4.36$$

根据三坐标测量，可以计算出工件曲率半径最大为 8cm，设行距为 5cm，残留断面高度为 1cm，带入式 4.36 得：r≥5.1282,即球头铣刀刀具半径 r≥5.1282cm，取 r=6.0cm。

（2）刀具的建模

刀具的刀杆、刀片和夹具的构造比较复杂，不易建立完善的几何模型进行描述，所以利用 OpenGL 函数来实现刀具的可视化是比较困难的。为了避免在 OpenGL 中进行复杂的绘图工作，利用 Pro/ENGINEER 软件完善的三维绘图功能，完成刀具的绘制，并将其输出为*.wrl 的文件类型。为了把刀具三维模型数据引入到 OpenGL 仿真系统里，借助三维图形解释器，输出为*.cpp 文件类型，完成刀具三维模型数据的读入，见图 4.15。

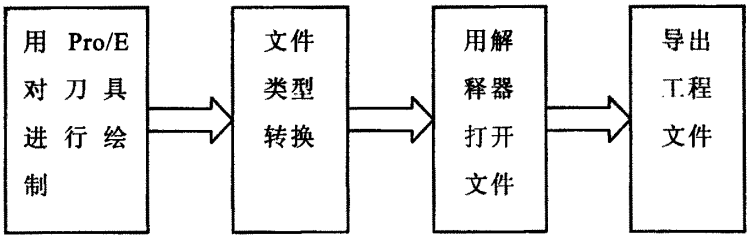

图 4.15 刀具三维模型的文件格式转换

4.7.3 叶片的仿真加工

数控加工仿真过程实际上是将刀具的运动包络体和零件毛坯体进行连续的布尔减运算，得到每一步加工后的工件数据，将这些数据在屏幕上显示出来，形成一幅幅画面，通过 OpenGL 的双缓存技术，使画面连续地快速显示，从而可以得到动态的加工过程。通过前两节对毛坯和刀具的建模，利用 Pro/E 建模，之后利用解释器将文件格式转换到 Visual C++可识别文件，就可以对叶片进行仿真加工，见图 4.16 所示。

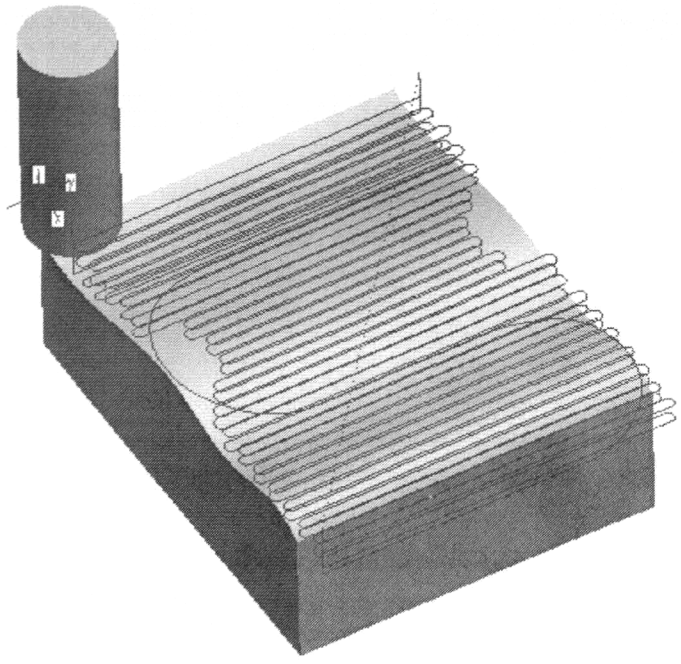

图 4.16 叶片仿真加工示意图

当由于人员编程问题，NC 代码产生错误时，导致零件的欠切（如图 4.17）、过切、干涉等现象，此时，虚拟仿真技术的应用可以使设计人员直观的在电脑屏幕上看出编程的错误，不至于在现实加工中产生不必要的经济损失及加工时间。

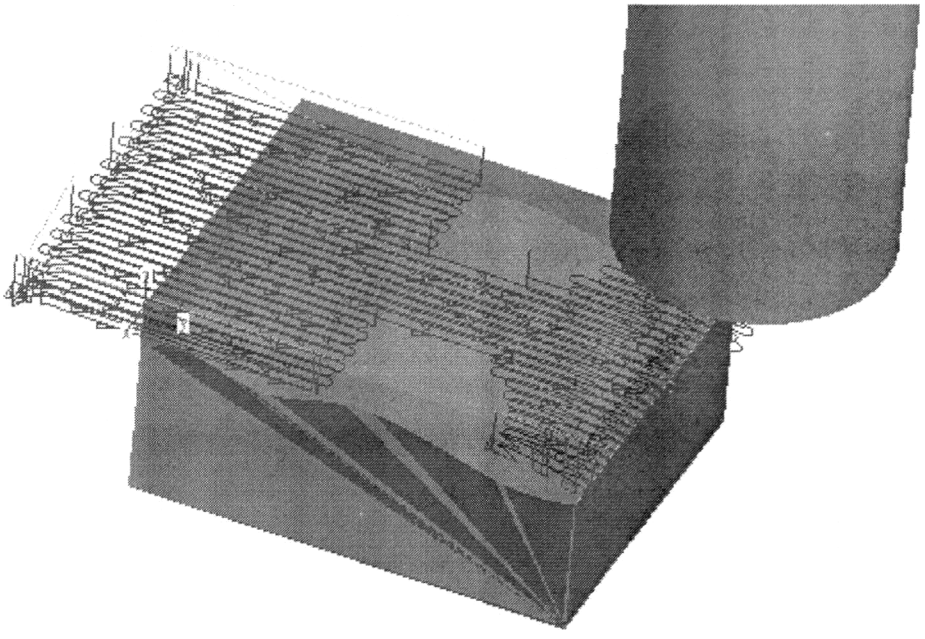

图 4.17 曲面铣削加工中欠切状态

图 4.17 所示，刀具轨迹（图中直线所示）并未遍历整个加工曲面，说明工程人员在编译 NC 代码是出现错误，造成零件的欠切削。虚拟仿真技术在零件设计加工中的运用，为设计人员提供了便利，减少了试制成本，提高了工作效率，

并且在一定程度上弥补了工程人员在编程中产生的漏洞。通过 OpenGL 的双缓存技术，可以使仿真加工更加形象具体，使工程人员在 CRT 上直接看到加工效果，及时纠正错误。

4.8 本章小结

本章在前两章的基础上，主要论述了零件毛坯和刀具模型的建立，阐述了仿真加工系统的原理及结构，并引用了 GAPT（图形自动编程编译器），通过叶片仿真加工的实例，验证了该系统的实用性。

结　论

虚拟制造作为一个很有发展前途的新领域，正在日益受到人们的重视。在本文中，对复杂曲面的仿真加工系统进行了建立，对仿真加工中的前期毛皮模型和刀具模型进行了研究，重点研究了 OpenGL 建模动画演示和数控自动编程两个部分。虚拟仿真技术是一项综合多项技术的成熟的设计理论，在实际应用中仍存在着一定的问题，如对复杂曲面仿真加工的研究较少，复杂曲面建模比较复杂。因此，有必要对其进行改进，缩短复杂曲面仿真加工的整体时间，提高产品试制的效率，使其在市场上更加具有竞争性。

在本文中，大体介绍了复杂曲面的建模方法及 NURBS 曲面的建模；研究了 OpenGL 的图像绘制、双缓存动画演示及 NC 代码自动编译器等问题，应用这些方法不仅有利于复杂曲面的建模，并且可以更加直观时时的显示出复杂曲面加工图像；尝试了 GAPT 方法进行数控的自动编程，并通过仿真检验，降低了 NC 代码的错误率，从虚拟中直接避免了实际加工中的过切、欠切及干涉等现象。

虚拟制造涉及到多个领域的知识，系统中有多个问题有待解决。由于时间关系，本论文中只能对其一部分进行研究。鉴于虚拟制造在实际应用中遇到的难题，今后应重点加强在以下三个方面的研究：

(1)必须加强对 Visual C++6.0 环境下 OpenGL 的研究，尤其在渲染方面的研究，使仿真更加逼真，减少建模编程所需的时间，并同时能够满足复杂曲面造型需要。

(2)加强对 NC 代码自动编译技术研究，NC 代码自动编译环节即为虚拟制造中模型和仿真加工两个流程的连接部分，同时也关系到实际加工中零件的质量，对整个制造来讲十分重要，因此，在今后的研究中，会考虑到虚拟制造中加工精度的控制实现。

(3)因为时间及知识面的关系，未能对整个制造系统（如加工中心）进行虚拟建模，从而没有对加工过程中碰撞现象进行检验，希望在今后的研究中，能够得以实现。

参考文献

[1] 施法中.计算机辅助集合设计与非均匀有理B样条[M].北京:高等教育出版社,2001.8

[2] 朱心雄.自由曲线曲面造型技术[M].北京：科学出版社，2000

[3] 陈定方，罗亚波.虚拟设计[M].北京：机械工业出版社，2002

[4] 黄雪梅.虚拟数控车削加工物理仿真系统研究[D]：[东北大学博士学位论文].沈阳：东北大学机械工程学院，2001

[5] 史永芳,田启华.虚拟制造中数控加工仿真的研究与开发[J].三峡大学学报（自然科学版），2006，28（1）：65~67

[6] 王占礼，杜爽，王尧.虚拟制造中的数控车削仿真加工[J].电子机械工程，2006，22（2）：47~49

[7] 陶亦亦，黄炜，姜左.虚拟制造的研究与发展[J].机械制造与自动化，2006，35（1）：4~6

[8] 周杰韩，吴波，杨叔子.虚拟制造系统综述[J].中国科学基金，2000（5）：279~283

[9] 王启义，葛研军，施志辉等。数控车削虚拟加工环境全景仿真技术[J].先进制造技术，2001，30（1）：39~40

[10] 马正元，朱岩峰，邹世革.虚拟制造系统中的数控加工仿真[J].沈阳工业大学学报，2002，24（6）：451~453

[11] M.Onosato, K.Iwata. Development of a virtual manufacturing system by integrating product models and factory models[J]. Annals of the CIRP,1993,42(1):475~478

[12] 朱恒,何汉武,熊有伦等.虚拟制造系统建模与仿真[J].中国机械工程,1996,7(3):28~32

[13] 柳卓之,李圣怡.面向产品生命周期的虚拟制造系统结构[J].机械工业自动化,1999,21(2):11~13

[14] 肖田元,韩向利,王新龙.虚拟制造的定义与关键技术[J].清华大学学报(自然科学版),1998,38(10):102~106

[15] 黄雪梅、高国利、王启义.拟实制造的机械加工过程仿真[J].机械设计与制造，1999，6

[16] 梁宏宝，曹喜承，梁宏山.虚拟加工中心加工过程算法研究[J].计算机仿真，2004，21（11）：217~218

[17] 姬舒平，刘卡林，邹继明，马玉林.NC Verification的研究现状及展望[J].

组合机床与自动化加工技术,1997(5):42~45
[18] 肖田元.虚拟制造[M].北京：清华大学出版社，2004
[19] 王振忠，郭隐彪.虚拟制造技术浅析[J].制造及自动化，140~144
[20] 周宝华，姬慧勇，盛显涛，裴平.浅谈现代机械制造技术的新发展[J].煤矿机械，2006，27（3）：366~368
[21] 唐荣锡.计算机辅助飞机制造[M].北京：国防工业出版社，1985
[22] Piegl L,Tiller W.Curve and surface constructions using rational B-splines[J].CAD,19,9,1987
[23] Piegl L.On NURBS: A Survey[J].IEEE CG&A,1,1991
[24] Piegl L. Modifying the shape of radional B-splines,part2: surfaces[J].CAD,21,9,1989
[25] 李刚，刘华明.叶轮、叶片类复杂零件造型方法研究[J].机械设计与制造，2000，（3）：59~60
[26] 薛惠锋，吴慧欣，解丹蕊.OpenGL 图形程序开发实务[M].西安：西北工业大学出版社，2005
[27] 江早，王洪成.OpenGL VC/VB 图形编程[M].北京：科学出版社，2001
[28] 刘崟.在 MFC 中使用 OpenGL[J].电脑与信息技术，1999，4
[29] 冯裕强，雷宝珍.NC 铣削加工的计算机三维仿真系统[J].机械设计与制造工程，1999，28（5）：36~38
[30] Hsu P H, Yang W T. Real-time 3D simulation of 3-axis milling using isometric projection[J]. Computer-Aid Design, 1993,25(4):215~224
[31] 吴修彬，曹西京.虚拟数控机床技术的发展与应用[J].机械制造与自动化，2006，35（1）：7~8，12
[32] 熊家伟，黄明吉，贾志新.虚拟数控车床仿真系统的研究与开发[J].机械制造与自动化，2006，35（3）：95~96，99
[33] 黄明吉.虚拟数控技术及应用[M].北京：化学工业出版社，2005
[34] 朱照梁，黄云龙，杨陈华.数控铣削过程仿真技术的研究[J].机械工程师，2003，9：50~52
[35] 范牧昌.零件的三维显示和数控加工仿真[J].现代机械，2001，4：6~8
[36] 戴同，王启付，黄正东.NC 铣削加工切削过程仿真[J].机械工业自动化，1996，18（2）：27~30
[37] 刘斌，王忠.面向对象程序设计——Visual C++[M].北京：清华大学出版社，2003
[38] 方沂.数控机床编程与操作[M].北京：国防工业出版社，1999
[39] 李福生.数控机床编程与操作[M].北京：机械工业出版社，1982

[40] 张国彬，林亨.车削中心数控加工仿真系统程序检查模块的开发[J].机械设计与制造，2001，2

[41] 沙智华.基于拟实体数控车削加工仿真研究[D]：[大连交通大学博士学位论文].大连：大连交通大学机械工程学院，2005，90~101

[42] 史永芳，田启华.基于OpenGL的虚拟数控系统的研究与开发[J].沈阳工程学院学报（自然科学版），2006，2（1）：86~88

[43] 余斌.基于 OpenGL 的数控加工仿真系统的研究与开发[D]：[四川大学硕士学位论文].成都：四川大学制造学院，2002

[44] 龙伟.数控机床及加工编程[M].成都：成都科技大学出版社，1994

[45] 孔凡新，周伯荣.面向虚拟制造的三维数控仿真系统研究[J].模具制造，2005，(12)：8~12

[46] 戴同，吴明华.加工过程仿真器及其关键技术[J].华中理工大学学报，1996，24（3）：10~12

[47] 高军，司马中文，王小椿.数控加工碰撞干涉检验的新算法[J].山东工程学院学报，2001，15（3），22~24

[48] 穆塔里夫·阿赫迈德，张年松，郑力.加工中心虚拟装配建模及装配干涉研究[J].现代制造工程，2002，9：14~16

[49] 谭光宇，袁哲俊，李建广. 数控加工碰撞干涉检验的新方法[J].制造技术与机床，1999，1：37~39

[50] 李建广，袁哲俊，王新龙.基于微机的三轴数控铣削仿真系统的研究[J].制造技术与机床，1997，2：21~23

[51] Tao Chen, Peiqing Ye. A tool path generation strategy for sculptured surfaces machining[J]. Journal of Materials Processing Technology, 127, 2002:369~373

[52] J H Bobrow. NC machine tool path generation from CSG part representations[J]. Computer Aided Design, 21(6), 1989:371~ 378

[53] Yuan-Shin Lee, Yawei Ma, George Jegadesh. Rolling ball method and contour marching approach to identifying critical regions for complex surface machining[J]. Computers in Industry, 441, 2000:163~180

[54] J H Oliver, D A Wyscoki, E D Goodman. Gouge detection algorithms for sculptured surfaced NC generation[J]. Journal of Engineer, Industry, 115, 1993:139~144

[55] 姚运萍，闫赟.复杂曲面加工中干涉问题的研究[J].工具技术，2006，40(4)：37~39

[56] X M Ding, J Y H Fuh, K S Lee. Interference detection for 3-axis mold machining[J]. Computer-Aided Design, 33, 2001:561~569

[57] Zezhong Chen, Zuomin Dong, Geoffrey W Vickers. Automated surface subdivision and tool path generation for 3-axis CNC machining of sculptured parts [J]. Computers in Industry, 50, 2003: 319~331

[58] 王霄.逆向工程技术及其应用[M].北京：化学工业出版社，2004

[59] 赵杰.涡轮叶片离散数据点曲面重构技术研究[D]：[西北工业大学硕士学位论文].西安：西北工业大学航空宇航制造工程，2004

致　谢

值此论文完稿之际，谨向所有关心、支持和帮助过我的老师和同学表示真诚地感谢，正是有了他们的帮助和鼓励，才使我能够顺利地完成课题的全部工作。

特别感谢我的导师阎树田教授，他严谨、务实的治学态度和积极、负责的敬业精神无时不给我极大的鼓舞，让我终生受益。三年来，无论是在学业上还是在生活上，阎老师都给了我莫大的鼓舞和帮助，让我不断地丰富自己，充实自己，从他那里我学到了作为一个科技工作者应具有的品德和修养，也使我更加有信心面对人生道路上的任何挑战。三年的时光虽然短暂，但足以使我一生受益。

感谢机电研究所的各位老师和同学，感谢他们对我的悉心教导和无私帮助！

感谢我的家人对我学业的支持和鼓励，正是他们的默默支持和帮助，使我能够顺利地完成学业！

附录 A　攻读硕士学位期间发表的学术论文

[1] 阎树田，王俊鹏，易湘斌. 加工中心的虚拟建模仿真研究. 科学技术与工程. 2007，9