五轴四联动玻璃雕刻机离线编程方法研究

摘 要

玻璃刻花加工中心是一种集数控、机器人技术、图形学、算法设计和机械加工技术于一体的高科技产品，亦称为数控玻璃雕刻机器人，其加工工具为砂轮，应用于平板玻璃刻花加工领域。玻璃刻花加工中心系统由上位控制计算机、控制器、执行体三部分组成。用户在上位机输入图形并进行排版设计，计算砂轮运动轨迹，并转换为雕刻指令传送给控制器；控制器根据雕刻指令控制执行体工作；执行体则负责玻璃刻花的最终实现。本课题对玻璃刻花加工中心离线自动编程系统进行研究，并加以编码实现，程序名称为 Glass Sculptor。

本文首先在对玻璃刻花加工中心与国外进口设备和现有通用电脑雕刻机进行横向比较的基础上，规划了项目的设计思路和开发原则，并明确了课题的研究难点和创新点。然后分析与玻璃刻花加工密切相关的重要对象的数学模型并建立标准的数据结构，进行程序描述。为了便于程序的开发和升级维护，本课题引入了模块化的思想，采用动态链接库技术对具体模块进行实现，并使用标准建模语言（UML）对程序模块进行规划，建立软件模型，使得模块间的接口关系清晰明朗。在这些研究的基础上，本文按照数据流程对玻璃刻花加工中心从数据输入到数据处理直至结果数据的发送进行了重点研究，并详细介绍了产生加工轨迹、砂轮切痕轮廓线和雕刻包络曲线三个核心算法，以及圆弧拟和、最优加工路径求取等两个优化处理的设计思路。

最后，本文对 Glass Sculptor 的实践和应用也进行了介绍，并在附录中展示了本课题研制的玻璃刻花加工中心的雕刻实物照片和本课题通过国家八六三计划项目验收的相关资料。

关键词： 玻璃刻花，自动编程系统，UML，DLL，串口通讯，模块化，算法

STUDY ON OFF-LINE PROGRAMMING METHOD FOR GALSS ENGRAVING MACHINE CENTER WITH MULTI-COORDINATED AXES

ABSTRACT

Glass Engraving Machine Center (GEMC) is a high-tech production integrated with several subjects including CNC, Robotics, Graphics, Arithmetics, Mechanics, and so on. Glass Engraving Machine Center is also named CNC Glass Engraving Robot. The Glass Engraving Machine Center, which uses diamond grinding wheel to engrave designed picture on flat glass, is nowadays widely used in the field of flat glass processing. GEMC includes at lease 3 parts: Control Computer, Controller and Executive Machine. This article is mainly focus on the off-line programming system that is run on the control computer with the application name of "Glass Sculptor.exe".

Based on the comparisons between GEMC and imported equipment and all-purpose engraving machine, we firstly discuss designate principles, difficulties and innovations of our GEMC. Then we come to analyze the important objects in glass engraving, calculate the mathematical model and design the standard data structures for these objects. In order to make the developing and upgrading of GEMC easier, we use modularization theory in this task and realize every function as an absolute DLL file, and as a result, we use Unified Modeling Language (UML) to describe the relationships and interfaces between those modules. We mainly research the data flow sequence of GEMC from input, calculation to transmission, and describe 3 key arithmetic subprograms and 2 optimization processes in details.

At the end of this article, we simply introduce the usage and program interface of Glass Sculptor, and show the photograph of engraving outcome of our GEMC in the appendix.

KEY WORDS: Glass Engraving, GEMC, off-line programming system, UML, DLL, serial port communication, modularization, arithmetic

第一章 课题的提出及设计思路

1.1 本课题的研发背景及意义

玻璃是一种传统材料，广泛地应用于建筑、汽车、装璜等行业。随着生活水平的提高，人们已不再满足于将玻璃简单应用，玻璃深加工的市场需求愈来愈大，工艺要求也愈来愈高。

目前最常见的玻璃深加工方法有：异形切割、异形磨边、钻孔、玻璃雕刻等，其中，玻璃雕刻升值潜力大，发展前景非常广阔，然而另一方面它的技术含量最高，开发难度也最大，在国内仍是一个技术空白点。

1.1.1 国内玻璃深加工的现状

近几年来，国内在玻璃加工方面迅速与国外接轨，制作工艺与加工精度日益进步。然而，经过本课题前期的市场调研和分析，我们发现国内目前在玻璃加工领域与国外发达国家相比还存在一下几点问题急需解决：

- 市场需求极大。随着我国建筑、装潢等行业的飞速发展，对于加工过的玻璃的需求大大增加。据了解，江苏省江阴市某玻璃装饰厂进口一台玻璃雕刻机 24 小时运转加工仍然供不应求，经济效益极高；同时，本课题在研究过程中就曾有多家国内玻璃厂商与我们进行联系，可见国内市场对玻璃雕刻加工设备的需求很大。
- 起步晚。国外一些在玻璃深加工领域占主导地位的几个公司在这个领域已有 80 多年的历史，已积累了大量的加工经验和试验数据，产品日趋完善。而国内在玻璃深加工领域的研究仅仅只有十年的历史，仍有大量的技术空白点；
- 工艺落后，加工方法亟待提高。目前国内采用较多的传统玻璃雕刻方法主要有三种：喷沙雕刻，化学腐蚀和手工雕刻。这些方法污染严重，不符合绿色生产的要求，并且效率低下，难以扩大生产规模；
- 加工附加升值量低，据统计，与国外发达国家的玻璃深加工相比，我国的平板玻璃制品销售值中，玻璃深加工所带来的附加产值仅仅只占 20～25%，而在某些发达国家，这一数值可达 75%，可见，我国的玻璃深加工仍有很大的发展潜力。
- 玻璃雕刻设备完全依赖进口。国内的玻璃机械近年来发展迅速，已能独立生产出玻璃异形切割、异形磨边、钻孔等机器，然而玻璃雕刻领域仍然是空白点。

解决上述不足的最好途径便是使用玻璃雕刻机。

1.1.2 进口设备不是推动国内玻璃产业发展的根本之道

目前世界有技术实力生产玻璃雕刻机的公司和国家并不多，主要分布在意大利，其中 Z.Bavellion 的 KAM 系列和 InterMAC 公司的 Master Groove 系列玻璃雕刻机功能较为全面，是目前国内进口最多的两种玻璃雕刻加工设备。

但是，进口设备价格昂贵，一台中等型号的玻璃雕刻机的售价大约在 160～180 万人民币左右，而且，由于受到采购量的限制，定货期很长，往往需要三个月到半年的时间。此外，

更重要的是，由于这些公司在国内只有代理商而没有设立分公司，因此售后服务和维修难以实现。

不可否认，进口国外设备确实可以促进国内的玻璃工业，但从长远角度来看，并不是推动国内在玻璃深加工领域发展的根本之道。另一方面，周边邻国在玻璃雕刻领域同样存在的技术空白给我们提供了巨大的商业前景，因此，自行研制玻璃雕刻机，替代昂贵的进口设备填补国内空白，有着极其深远的意义。

1.1.3 课题来源

数控系统是国家重点科技攻关项目，国家在这方面的投入逐年加大，重视程度越来越高。另一方面，上海交大海泰科技发展有限公司是国家科技部高科技成果的孵化基地，多年来致力于机电一体化产品的开发，并积极推进科技成果的产业化。海泰公司在通用电脑雕刻机有着多年的研发经验，积累了大量的试验数据和客户反馈信息。该公司一直在产品范围上进行积极拓展，在玻璃深加工方面已进行了大量的市场研究和方案分析。

本课题为国家 863 高科技资助项目，项目承担方为上海交大海泰公司。

1.2 玻璃雕刻机简介

玻璃雕刻机（Glass Machine Center），又称为数控玻璃雕刻机器人，顾名思义是一种以玻璃为加工对象的智能加工设备，国外某公司就是这样描述他们生产的玻璃雕刻设备的："4 axes numerically controlled work centres for glass grooving……"，可见，玻璃雕刻机是以数控和机器人技术为基础，并融合了图形处理、算法设计、计算机控制和机械等多学科技术的机电一体化产品。

1.2.1 玻璃雕刻机的机械本体和集成数控系统简介

如图 1.1，玻璃雕刻机的机械本体采用直角龙门式结构，共有五个传动轴，其中 X、Y、Z、C 轴为联动轴，分别控制砂轮的 X、Y、Z 空间位置和砂轮的偏转角度，另一轴为砂轮自转主轴，控制砂轮的自身旋转运动。四个联动轴通过计算机控制实现同步运动，砂轮自转主轴控制砂轮自转转速，一般在速度优化及针对不同加工材质进行选择时对其加以设置。如图 1 所示。X、Y、Z 轴分别由交流伺服电机直接连接滚珠丝杠驱动，C 轴由交流伺服电机经谐波齿轮减速器减速来驱动。

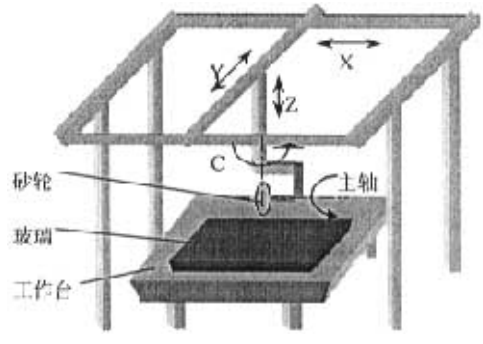


图 1.1 机械结构简图

玻璃雕刻机的集成控制系统采用分层体系结构。可分为路径规划层、控制层和执行机构，如图 1.2 所示。路径规划层采用自行开发玻璃雕刻控制程序（上位机），控制层采用基于 SERCOS 伺服现场总线的数控系统（下位机），执行层包括各轴的伺服驱动器、伺服电机、各轴的限位开关、水路控制系统以及控制面板。上下位机之间通过 RS-232 串行通讯的方式实时交换数据。在加工前，操作者先在离线编程系统中作出需要雕刻的图形，编程系统自动将图形转换成数控系统能够接收的数据文件。数

控系统根据接收的图形数据，驱动各轴的伺服电机，并在开始加工前自动打开冷却水，在加工结束时自动关闭冷却水。

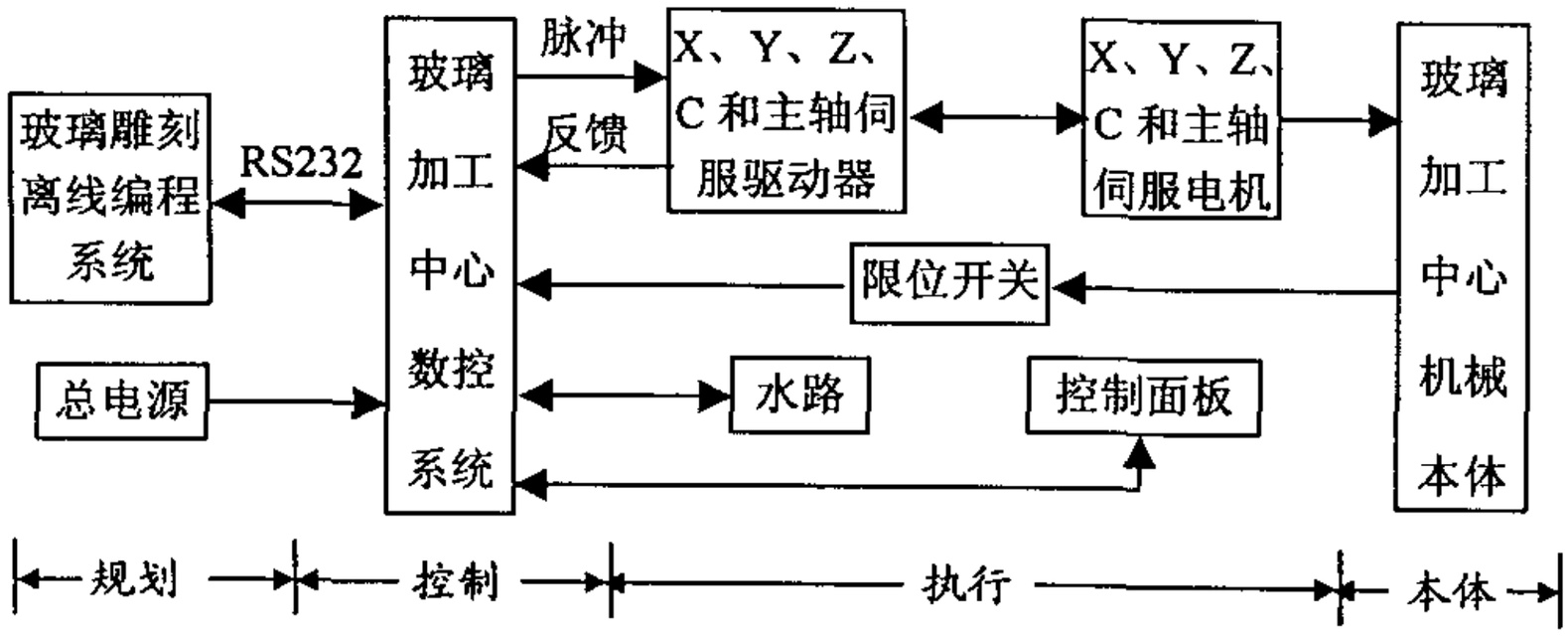


图 1.3 玻璃雕刻机的集成数控系统

1.2.2 玻璃雕刻机的工作过程

一个玻璃雕刻机控制系统实际上就是一个 CAD/CAM 和 CNC 的集成系统。用户在 CAD 系统上进行雕刻内容设计，通过 CAM 系统产生必要的加工控制信息，最后由 CNC 系统完

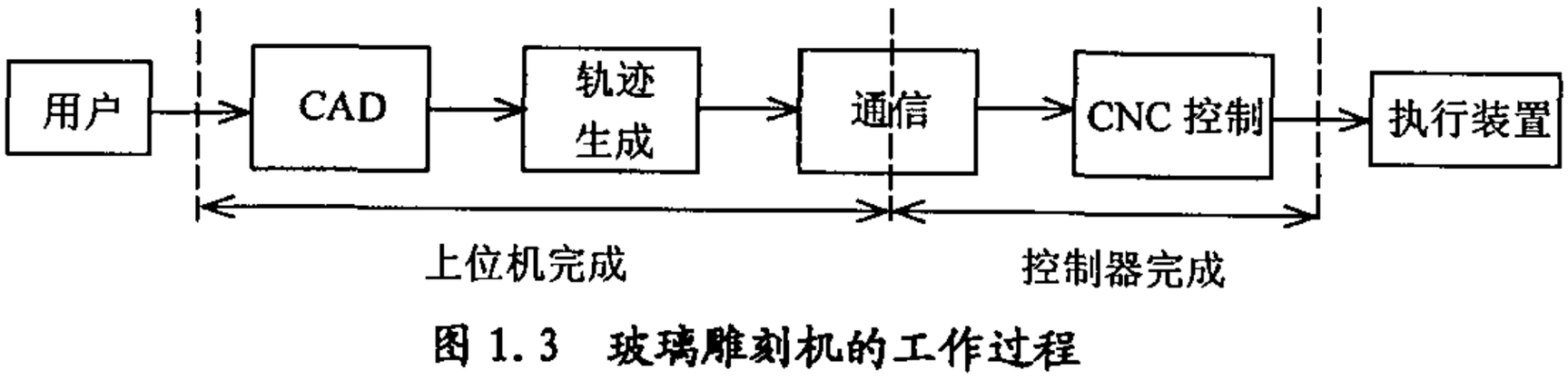


图 1.3 玻璃雕刻机的工作过程

成整个玻璃的加工。其工作原理如图 1.3 所示。

在整个加工中心系统中，上位机和控制器分别完成一部分工作。上层软件安装在上位机中，主要负责从用户获取雕刻内容并生成运动控制指令，应该具有排版编辑功能、数据计算处理、雕刻效果预览、上下位机间的通信等。下层软件主要从上层软件接受指令并控制执行装置部分完成多轴协调运动，即 CNC 实时控制功能。

显然，上层软件和控制器是玻璃雕刻机开发任务中最重要的两部分，根据任务的分工，本课题的重点是开发完成玻璃雕刻机离线控制系统，即上层软件，软件名称为 Glass Sculptor。

1.2.3 玻璃雕刻机与普通电脑雕刻机的比较

玻璃雕刻机与普通电脑雕刻机相比，有许多相同点，也有许多不同点。

相同点是两者的工作过程基本相同，都是在上位机进行编辑排版，生成走刀轨迹，通过通信模块将运动控制指令传送给控制器，控制器接受到指令后控制机械本体完成指定的运动，过程如图 1.1 所示。

玻璃雕刻机与普通电脑雕刻机的不同点有：

- 加工对象的区别。普通电脑雕刻机的加工材料是有机玻璃、木材等，使用的加工刀具为普通刀头；而玻璃雕刻机的加工对象是玻璃，由于玻璃质脆且硬度很大，因此必须以砂轮作为加工刀具。可以说，加工对象的不同导致了加工刀具的不同。

- 自由度数目不同。玻璃雕刻机有四个自由度，即 X、Y、Z 三个平移自由度和一个绕 Z 轴转动的转动自由度，通过砂轮的转动，调节加工图形的宽度。也就是说，玻璃雕刻机的控制器需要控制四个伺服电机协同工作，再加上砂轮轴的伺服调速系统，系统的控制机制比较复杂，而普通雕刻机只有 X、Y、Z 三个平移自由度。
- 上层软件功能不尽相同。由于普通雕刻机的加工刀具的刀头较尖，而且是中心对称的，因此可以简单的视为一个质点。它能够雕出比较复杂的图形，如汉字、花纹等；而砂轮切割痕迹为梭形（见图 1.4（a）），由一个个梭形图案组合成复杂的加工的图形的难度远远高于普通电脑雕刻机。
- 精度要求不同。在玻璃进行雕刻，一个微小的机械振动都能造成雕刻实物在视觉上的强烈反差，因此玻璃雕刻机的精度要求高于普通雕刻机。

可以这么说，玻璃雕刻机与普通电脑雕刻机的不同点都来源于加工对象的不同：不同的加工对象决定了不同的加工工具和加工精度，而不同的加工工具又导致机械结构和上层软件必须以不同的方法实现。

同时，上述的不同点也正是本课题研发过程中的难点和重点所在。

1.2.4 本课题的难点和技术创新点

基于上一节对玻璃雕刻机和普通电脑雕刻机之间异同的对比，不难总结出本课题的难点：

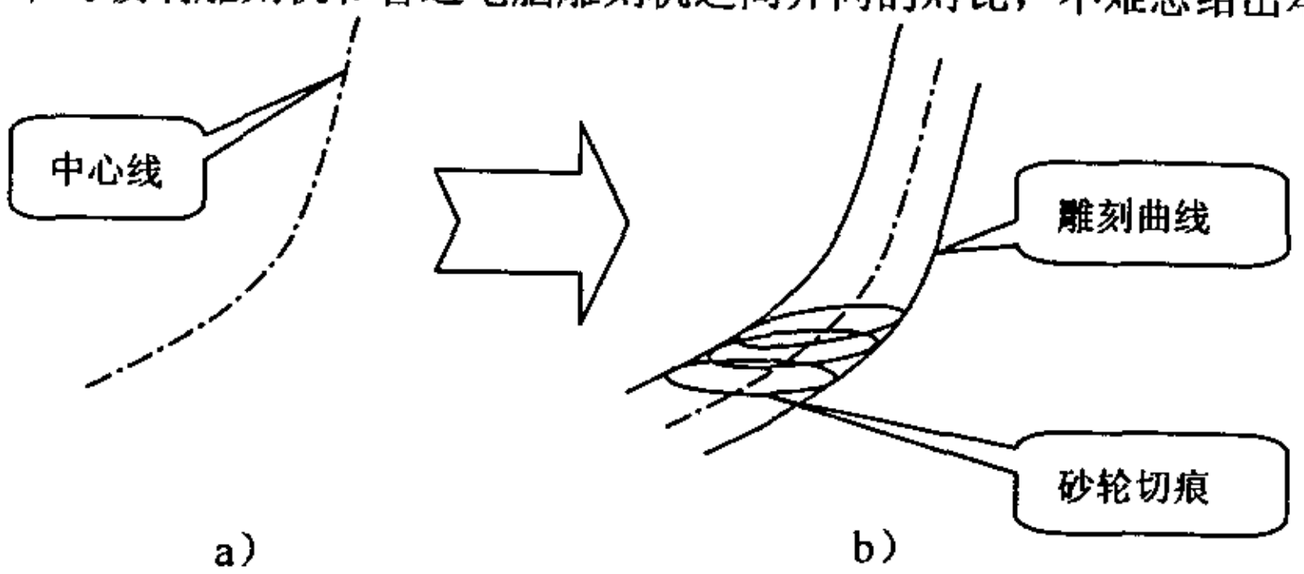


图 1.4 由输入中心线计算雕刻指令的过程示意图

Figure1.4 From Input Graph To Groove Command

如图所示，输入的待加工图形为平面图形，只具有 x 和 y 两个轴的坐标值，而要确定每一点处的砂轮空间姿态必须有四维数据：x，y，z 和砂轮转角 θ，如何设计合适的算法来获得冗余数据，并且由如图 1.4b）中所示的一个个离散的砂轮切痕轮廓线组合成光滑平整的雕刻曲线，是本课题的主要难点所在，本文的随后几章也将以此问题为中心进行展开。

在介绍本课题的创新点之前，我们先分析一下国外进口玻璃加工中心的工作方式。通过分析，我们发现进口玻璃雕刻机的工作方式有以下特点：

- 加工图形以线条为主；
- 待加工图形以中心线的形式输入；
- 通过各种简单线条的组合，实现复杂精美的雕刻效果；
- 不同宽度的线条由不同的砂轮实现。

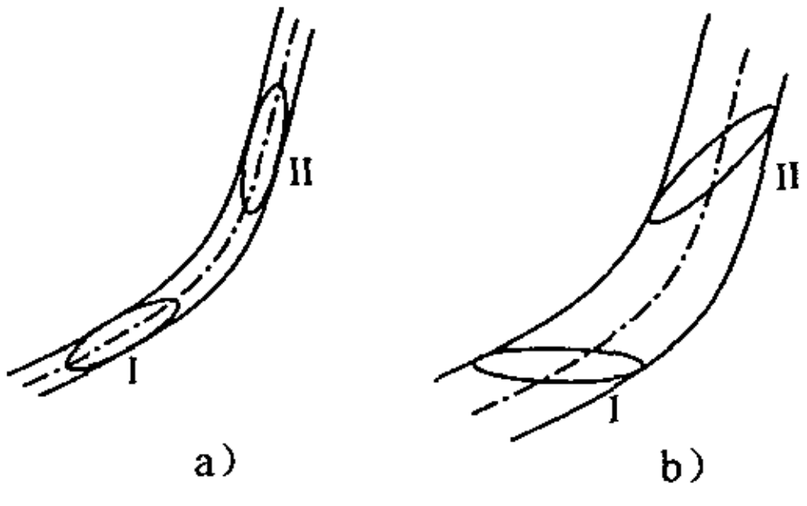


图 1.5 与进口加工中心的比较

Figure1.5 Differences Compare

众所周知，一个科技成果要在产业化道路上成功，必须做到“人无我有，人有我优”，借鉴现有技术固然必不可少，但只停留在简单的模仿上是不可取的，必须在功能和工艺上有所创新，有所突破。同样，本课题所开发的玻璃雕刻机与进口设备相比，不能只具备价格和维护方便上的优势，因此本课题在开发时，在对一些优秀的进口设备进行了分析研究的基础上，在功能上对玻璃雕刻机进行了一些技术创新：

- 进口玻璃加工中心在雕刻时，砂轮对称轴与中心线的前向切线夹角为 0°，待加工线条的宽度只能是当前砂轮的宽度，无法进行进一步的调整，如图 1.5a）所示。而本课题中，砂轮可以相对加工中心线进行转动，这样雕刻出来的线条更富有变化，加工范围更广，如图 1.5b）所示；
- 添加了雕刻预览和走刀模拟模块。国外进口设备一般没有配置预览功能，其实，在正式雕刻前对加工图案进行预览是有必要的，因为用户可以通过预览对加工图案进行进一步修改，从而提高成品率，减少加工成本；
- 采用纯中文界面，用户不必具有英语基础便可进行操作；
- 为将来引入汉字字库和雕刻图库保留了程序接口。

1.2.5 设计原则

一般来说，一个课题的项目背景和课题来源往往决定了课题的实现途径和设计原则。玻璃雕刻机也不例外，前文曾提及，本课题研究目的是推广自主知识产权的玻璃雕刻机，从而在根本上推动国内玻璃深加工领域的发展。作为国家 863 高技术研究发展计划资助项目，其研究成果将直接投入应用生产并实现产业化，为此研究开发必须遵循以下原则：

- 经济实用　　所谓“经济”，是指与国外进口设备相比，本课题所研究开发的玻璃雕刻机必须具有价格优势，否则，“推广自主知识产权的玻璃雕刻机，从而在根本上推动国内玻璃深加工领域的发展”将无从谈起；所谓“实用”，是指开发的产品能胜任使用者的加工需求，不但能雕刻出国外进口设备所能加工的图案，而且在精度方面必须达到甚至超过进口设备的技术指标。
- 简单易用。　我们不能苛求玻璃雕刻机的操作者具有高超的手工技艺和知识背景，而且，使用复杂晦涩的产品往往难以推广普及，因此必须精心设计人机界面，尤其上位程序应一目了然。这样，用户在购买玻璃加工中心后，只需一个短期的学习就能投入生产。
- 技术上必须有所创新。尽管目前有技术实力生产玻璃雕刻机的公司并不是很多，市场竞争并不激烈，但随着中国正式进入 WTO，本课题所研究玻璃雕刻机要在产业化方面取得成功并在国际市场上获得进一步发展，必须相对国外先进设备有自己的技术优势，因此必须进行创新。（详见 1.2.4 节）
- 便于升级更新。任何一项技术都有一个从不成熟到成熟，不完善到完善的逐渐进步的过程，如何让改进后的产品为此前购买产品的用户服务，即实现产品升级更新，是一个产品化的课题必须考虑的问题。

确定了设计原则后，我们就可以进行课题设计思路的制定和开发工具的选取了。

1.2.6 设计思路

正如前面所提到的，由于加工工具的特殊性，玻璃雕刻机不可能加工非常复杂的图形，因此本课题的开发目标是由用户从外部程序输入中心线，并可以编辑和移动中心线，砂轮将

沿着中心线运动，加工出来的图形要达到用户预先设定的目标，并能够进行加工模拟和预览，便于用户及时修改输入的数据以达到满意的效果。

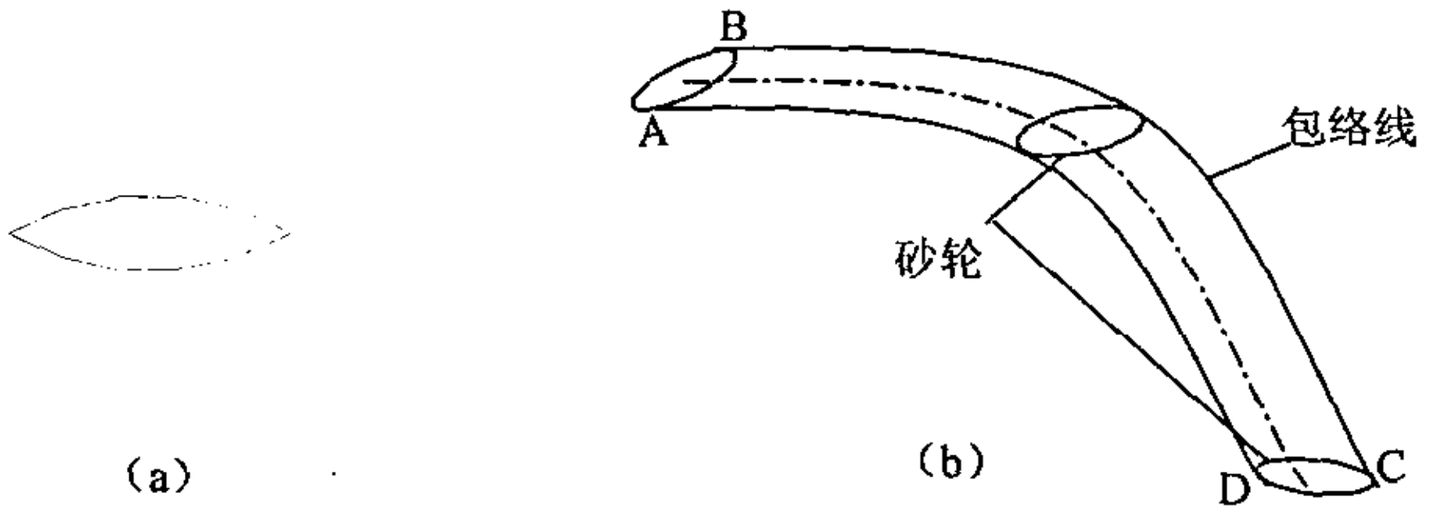


图 1.6 玻璃雕刻机的工作示意图
（a）砂轮在某一个位置产生的切痕轮廓线
（b）砂轮沿中心线运动时切割产生的包络线

可以通过以下步骤来实现：

首先建立对各种重要的对象进行分析，设计数据结构实现程序描述。此外，由于本系统规模较大，应从整体上进行把握，用模块化的思想使项目的开发和维护易于实现。

在数据处理方面，由于输入的待加工图形是由一系列平面上的点来描述的，每一点的坐标只有 x 和 y，而要描述砂轮运动到每一点时的空间姿态，仅仅只有 x 和 y 是不够的，还必须加上吃刀深度 z 和砂轮绕 z 轴的转角 θ。因此，必须设计合理的算法，以实现冗余数据的获取，并结合用户输入的参数来确定砂轮的走刀轨迹。对于砂轮切痕轮廓线，我们可以确定其几何方程，但要在计算机中描述它，还必须将图 1.6（a） 用一系列的点来描述，即所确定的方程进行离散化。然后还应该找出计算包络线的方法，求出图 1.6（b）所示的包络线 ABCD，只有求出包络线，才能实现雕刻预览功能。

最后还应对砂轮走刀轨迹进一步进行处理，转换为雕刻指令传送给控制器，实现对执行机构的控制。

1.2.7 开发工具选用

鉴于以上分析，我们在玻璃雕刻机离线编程系统的上位程序 Glass Sculptor 的研究过程中对开发工具进行了选择。

我们首先使用标准建模语言 UML 建立了软件模型，然后在 Windows 9X/NT/2000 平台上，使用 Microsoft Visual C++开发 Glass Sculptor。为了缩短开发周期，我们引用了 MFC（Microsoft Foundation Class）库。

此外项目实现过程中还涉及了一些动态链接库，多线程及串口通讯理论，本文将在后面的章节详细探讨。

1.3 本文的主要内容

本文共分六章，主要内容及章节安排如下：

第一章为课题的综述，主要对玻璃雕刻机的研究背景、课题来源、项目概况、设计原则及研究方案进行介绍。

第二章针对玻璃雕刻机编程系统的特点，在分析数学模型的基础上，为所涉及的重要对象，如加工刀具、基本绘图单元、基本加工对象、砂轮切割轮廓线和包络曲线设计数据结构，实现程序描述。

第三章从整体上设计了 Glass Sculptor 模块结构，并采用软件建模的方法对这些模块进行规划。并结合 Glass Sculptor 的开发实例，论述了动态链接库的设计思路和标准建模语言 UML 建立软件模型的实现方法。

第四章是按照 Glass Sculptor 中对数据进行加工处理的顺序对各模块的具体实现进行描述，并着重探讨了由输入的二维数据计算砂轮运动轨迹的主要算法和一些优化处理，本章的最后对 Glass Sculptor 与控制器间通讯部分设计过程中所涉及的技术进行了专门的研究。

第五章是前几章所探讨的理论的具体实践，从程序界面、雕刻实物和试验数据的角度展示玻璃雕刻机的最终实现。

第六章总结和展望部分，在总结本课题研发过程的基础上，对玻璃雕刻机离线编程系统的进一步发展和完善进行前景展望。

沿着中心线运动，加工出来的图形要达到用户预先设定的目标，并能够进行加工模拟和预览，便于用户及时修改输入的数据以达到满意的效果。

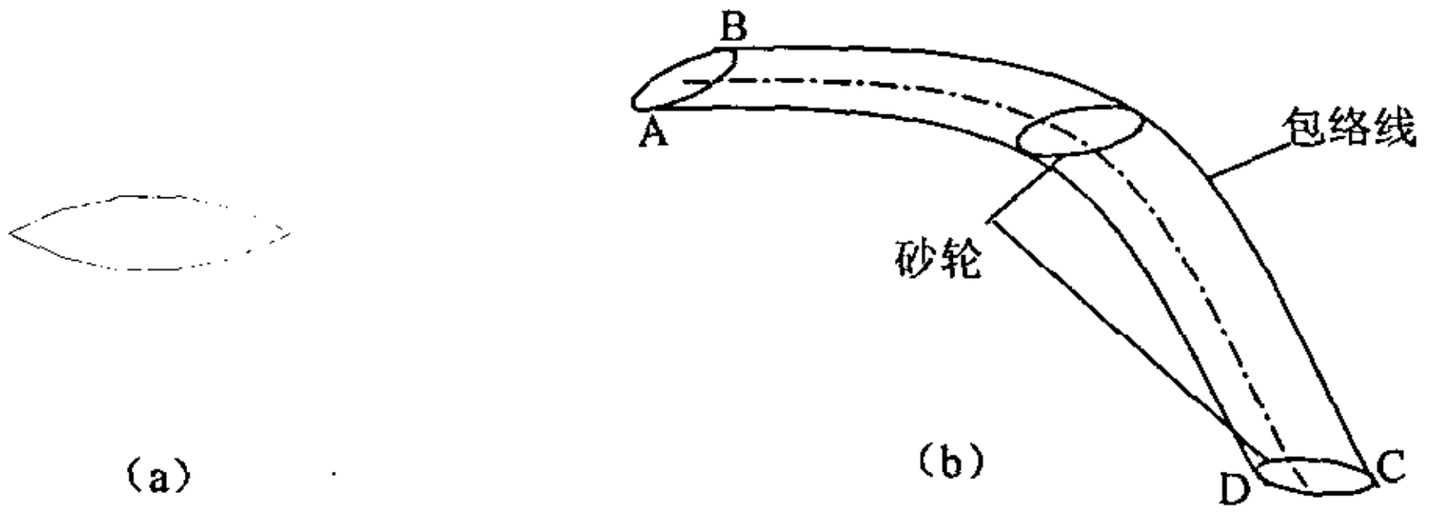


图 1.6 玻璃雕刻机的工作示意图
（a）砂轮在某一个位置产生的切痕轮廓线
（b）砂轮沿中心线运动时切割产生的包络线

可以通过以下步骤来实现：

首先建立对各种重要的对象进行分析，设计数据结构实现程序描述。此外，由于本系统规模较大，应从整体上进行把握，用模块化的思想使项目的开发和维护易于实现。

在数据处理方面，由于输入的待加工图形是由一系列平面上的点来描述的，每一点的坐标只有 x 和 y，而要描述砂轮运动到每一点时的空间姿态，仅仅只有 x 和 y 是不够的，还必须加上吃刀深度 z 和砂轮绕 z 轴的转角 θ。因此，必须设计合理的算法，以实现冗余数据的获取，并结合用户输入的参数来确定砂轮的走刀轨迹。对于砂轮切痕轮廓线，我们可以确定其几何方程，但要在计算机中描述它，还必须将图 1.6（a） 用一系列的点来描述，即所确定的方程进行离散化。然后还应该找出计算包络线的方法，求出图 1.6（b）所示的包络线 ABCD，只有求出包络线，才能实现雕刻预览功能。

最后还应对砂轮走刀轨迹进一步进行处理，转换为雕刻指令传送给控制器，实现对执行机构的控制。

1.2.7 开发工具选用

鉴于以上分析，我们在玻璃雕刻机离线编程系统的上位程序 Glass Sculptor 的研究过程中对开发工具进行了选择。

我们首先使用标准建模语言 UML 建立了软件模型，然后在 Windows 9X/NT/2000 平台上，使用 Microsoft Visual C++开发 Glass Sculptor。为了缩短开发周期，我们引用了 MFC（Microsoft Foundation Class）库。

此外项目实现过程中还涉及了一些动态链接库，多线程及串口通讯理论，本文将在后面的章节详细探讨。

1.3 本文的主要内容

本文共分六章，主要内容及章节安排如下：

第一章为课题的综述，主要对玻璃雕刻机的研究背景、课题来源、项目概况、设计原则及研究方案进行介绍。

第二章针对玻璃雕刻机编程系统的特点，在分析数学模型的基础上，为所涉及的重要对象，如加工刀具、基本绘图单元、基本加工对象、砂轮切割轮廓线和包络曲线设计数据结构，实现程序描述。

第三章从整体上设计了 Glass Sculptor 模块结构，并采用软件建模的方法对这些模块进行规划。并结合 Glass Sculptor 的开发实例，论述了动态链接库的设计思路和标准建模语言 UML 建立软件模型的实现方法。

第四章是按照 Glass Sculptor 中对数据进行加工处理的顺序对各模块的具体实现进行描述，并着重探讨了由输入的二维数据计算砂轮运动轨迹的主要算法和一些优化处理，本章的最后对 Glass Sculptor 与控制器间通讯部分设计过程中所涉及的技术进行了专门的研究。

第五章是前几章所探讨的理论的具体实践，从程序界面、雕刻实物和试验数据的角度展示玻璃雕刻机的最终实现。

第六章总结和展望部分，在总结本课题研发过程的基础上，对玻璃雕刻机离线编程系统的进一步发展和完善进行前景展望。

第二章 重要对象的程序描述

对于涉及软件开发的课题而言，数学模型和数据结构的确立是项目开发过程中最先进行也是最关键的部分之一，它们设计的好坏，将决定着整个课题的走向和难易程度——计算准确的数学模型和设计合理的数据结构，可以有效地降低项目的开发难度，从而使开发周期大大缩短，开发成本大大降低。

对于玻璃雕刻机而言，需要对很多对象进行描述，使得控制程序可以识别，从而实现从输入图形得到我们所期待的加工指令以传递给下位控制器。此外，本课题中出于将来的升级的考虑，我们将采用了模块化的实现方法，因此必须要考虑到各模块之间的接口，所以，对于本课题而言数据结构的设计更为重要。

2.1 加工刀具的程序描述

2.1.1 加工刀具的数学模型

第一章中我们已分析过，玻璃雕刻机与通用雕刻机相比，最大的不同在于玻璃雕刻必须以金刚石砂轮为雕刻刀具，而且，通过使用各种不同的刀具可以实现在玻璃上雕刻出各种效果迥异的图案，刀具库中砂轮的种类决定了玻璃雕刻机的加工能力。所以，对于雕刻加工中心而言，金刚石刀具是最重要的对象，对刀具库中的各种刀具进行程序描述并管理是玻璃雕刻机的重要任务之一。

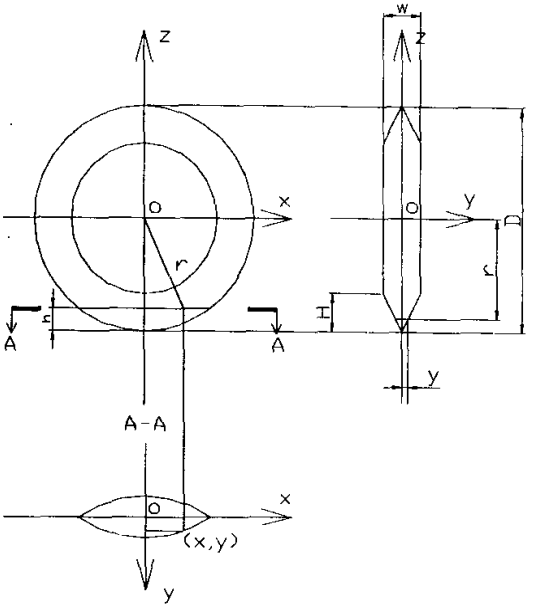


图 2.1 尖角砂轮

Figure 2.1 The Mathematical Model of Grinding Wheel

根据加工边缘形状的不同，我们可以对砂轮进行分类，其中最常见的为尖角砂轮和圆角砂轮，下面我们以这两种砂轮为例，具体分析一下砂轮的截面方程，以对其在玻璃上的切割痕迹进行描述。

2.1.1.1 尖角砂轮

尖角砂轮如图 2.1 所示：

上图中 D 为砂轮外径，W 为砂轮厚度，H 为砂轮的最大有效加工深度，且均为已知，h 为吃刀深度，雕刻机上通常认为向下为负，故 h<0。 A—A 视图为吃刀深度为 h 处的截面图。

由上图，有：

$$r=\sqrt{(D/2+h)^2+x^2} \qquad (式\ 2—1)$$

由于砂轮是中心对称图形，因此半径为 r 在砂轮上的点在 Y 轴上的投影都是相等的。

$$\begin{aligned} y &= \pm(\tfrac{D}{2}-r)\times(\tfrac{W}{2}/H) \\ &= \pm\tfrac{D-2r}{2}\times\tfrac{W}{2H} \\ &= \pm\tfrac{W}{4H}(D-2r) \end{aligned} \qquad (式\ 2—2)$$

将（式 2—1）代入（式 3—2）中，即得到截面曲线的方程（式 2—3）：

$$y=\pm\tfrac{W}{4H}(D-2\sqrt{(D/2+h)^2+x^2}) \qquad (式\ 2—3)$$

由（式 2—3）可知，曲线方程是 x 和吃刀深度 h 的函数，当 h 一定，曲线也就一定。为了与一般的方程的表示形式一致，将 h 用 z 代替，即得（式 2—4）：

$$y=\pm\tfrac{W}{4H}(D-2\sqrt{(D/2+z)^2+x^2}) \qquad (式\ 2—4)$$

其中 $-\sqrt{-Dz-z^2}\le x\le\sqrt{-Dz-z^2}$

2.1.1.2 圆角砂轮

圆角砂轮如图 2.2 所示。

其中，R 为圆角部分的半径，是一个常量，且 h≤H≤R。

同（式 2—1），

$$r=\sqrt{(D/2+h)^2+x^2} \qquad (式\ 2—5)$$

由图 2.2 知，

$$y=\pm\sqrt{R^2-[r-(\frac{D}{2}-R)]^2} \qquad (式\ 2—6)$$

化简得，

$$y=\pm\sqrt{-r(r-D+2R)-D(\frac{D}{4}-R)} \qquad (式\ 2—7)$$

其中 $-\sqrt{-Dz-z^2}\le x\le\sqrt{-Dz-z^2}$

将（式 2—5）代入（式 2—7）中，即得到 y=f(x,h) 的方程，将 h 用 z 代替，即得 y=f(x,z)。

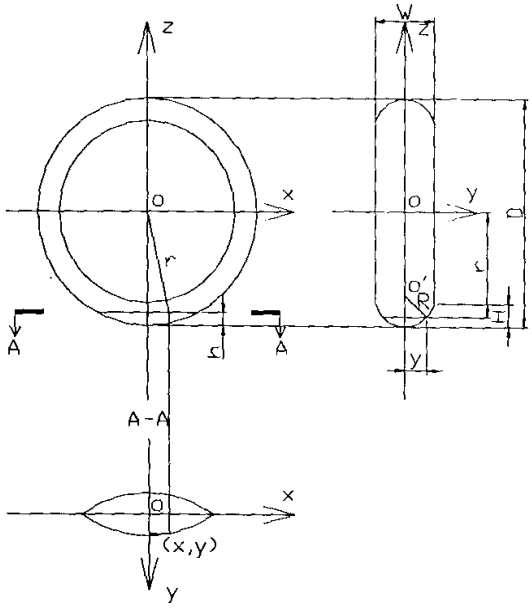


图 2.2　圆角砂轮

Figure 2.2 The Mathematical Model of Grinding Wheel

2.1.1.3 其他类型的砂轮

限于篇幅我们只计算了两种砂轮的截面方程，但所使用的方法具有一般性，上述对尖角砂轮和圆角砂轮截面方程的计算所使用的方法稍加修改便可以应用于其他类型砂轮截面方程的求取。

确立了砂轮的截面方程后，我们便可以通过砂轮轴心的当前坐标推算出砂轮的雕刻痕迹，并以此为计算依据，生成最终的雕刻指令、进行雕刻效果预览和走刀模拟，它的具体应用将在后面介绍。

2.1.2 加工工具的数据结构

在上面的分析过程中，我们

结构 TOOLF 是用来定义砂轮的类型、尺寸。如下所示：

```
typedef struct TOOLF
{
unsigned long Kind;
float D;
float W;
float H;
float r;
float dD;
}TOOLF;
```

在该结构中，Kind 用来描述砂轮的类型，如为圆角砂轮、尖角砂轮或者其他；D、W、H 均与图 2.1、2.2 所描述的一样，分别为砂轮的外径、厚度和最大有效加工深度；r 为圆角砂轮的圆角半径，当砂轮为尖角时是无效的，r 有效时，H<=r；考虑到砂轮在使用过一段时

间以后会有一定的磨损，因此定义 dD 为砂轮外径的磨损值，这样尽管砂轮的实际尺寸与初始值有一定误差，用户也可以自己对砂轮进行校准。砂轮的实际外径值为 D-dD。

2.2　加工对象的程序描述

一个功能完善的图形编程系统应支持用户采用多种方式来输入待加工图形，比如说，用户可以自己绘制待雕刻图形，也可以从别的图形程序（如一些专业的 CAD 软件）中导入数据，还应该支持由扫描仪支持的光栅图形数据。因此，我们必须对多种数据都加以考虑，然而通过上面三种方式得到的待加工图形有的是矢量化的数据，有的则是离散化的数据，我们必须在程序中设计相应的数据结构。

关于加工图形的上述的几种输入方式，将在本文的第四章详细讨论。

2.2.1 基本绘图单元的数据结构

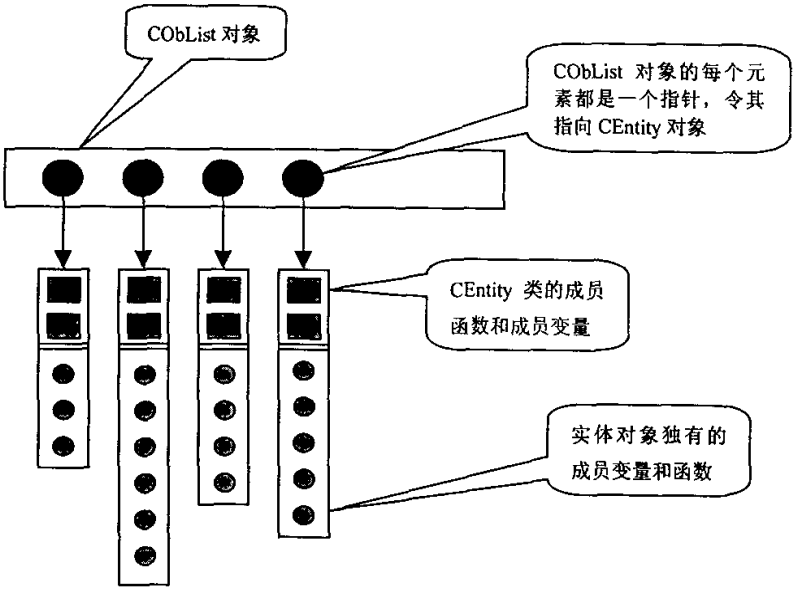


图 2.3　基本绘图单元的数据结构图

Figure 2.3 The Data Structure of Fundamental Drawing Unit

在几种原始数据的获取方式中，自绘图形的方法最灵活，并且获得的图形易于修改和编辑，但实现起来难度也最大。尽管在 Glass Sculptor 中，目前仍以从现有的 CAD 程序（如 AutoCAD）中获得待加工图形为主，但从长远角度来说，绘图单元必须进一步发展和完善，出于这个原因，本课题在设计过程中，对绘图部分也进行了相应的研究。

通常，一个平面 CAD 系统的基本绘图单元有点、直线、圆、圆弧、矩形框、样条曲线和文字等，这些基本绘图单元有各自的个性，同时也有共同的属性和操作，因此可以考虑用 C++类继承与派生的思想来实现这些绘图单元的数据结构。

先定义一个 CObject 类的派生类 CEntity，将它作为这些基本绘图单元的父类，因此，这些基本绘图单元共同拥有的特性可以在 CEntity 中描述，如线型以及序列化进行保存的操作等。然后，将每种基本单元自己独有的性质和操作实现在它们自己的派生类中，如表示直线的类 CLine 中，可以包含直线的起点和终点等信息。以直线和圆为例，代码如下：

```
class CEntity:public CObject
{
public:
    ......
    UINT m_nType;
public:
    Serialize(CArchive& ar);//序列化以进行保存
}

class CLine:public CEntity
{
public:
    ......
    CPoint m_pointStart;//起点和终点
    CPoint m_pointEnd;
public:
    ......//函数
    void Draw(CDC* pDC, CGlassSculptorView* pView) const;
    void Edit(CPoint pointNewStart, CPoint pointNewEnd);
}

class CCircle:public CEntity
{
public:
    ......
    CPoint m_pointCenter;//
    double m_dRecius;//
public:
    ......//函数
    void Draw(CDC* pDC, CGlassSculptorView* pView) const
    void Edit(CPoint pointNewCenter, CPoint dNewRadius);
}
```

最后，必须再设计一个数据结构将这些绘图单元有机的结合起来，由于 Glass Sculptor 需要对加工路径进行优化，因此会涉及大量的排序操作，因此选用链表结构最为合适。可以使用 MFC 库中的 CObList 类，它的每个元素都可以是一个 CObject 类的对象。这也是为什么要从 CObject 类派生绘图单元基类 CEntity 的原因。

图 2.3 所示即是上面介绍的基本绘图单元的存储结构。

2.2.2 基本加工对象的数据结构

2.2.2.1 玻璃雕刻机的基本加工图形

在第一章中提到过，玻璃玻璃雕刻机加工而成的玻璃图形不可能很复杂，而且，通过分析，大多数的镜面图案基本上都是由最简单的线条状图形组合而成。因此我们可以通过控制

砂轮沿着给定的中心线运动，以实现玻璃雕刻。玻璃雕刻机经常加工的图形见图 2.4 中的（a）、（b）、（c）：

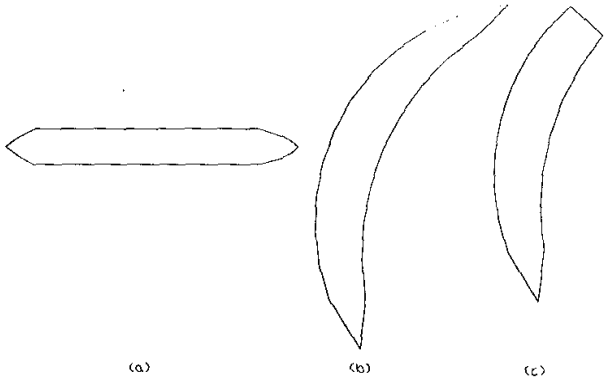


图 2.4　基本加工图形

Figure 2.4 The Basic Fundamental Processed Unit

在程序实现的过程中，有必要对所要加工的图形用统一的形式进行描述，以使得程序能识别所需加工的图形，并以此生成雕刻指令。如图 2.3 所示，需要加工的图形中可以是直线线条，也可以是曲线线条；可以有尖角，也可以无尖角。由于在玻璃雕刻机的控制系统中，雕刻对象的中心线和雕刻所生成的轨迹线等曲线是通过一组折线的形式来表示的，因此我们可以将曲线拓扑拉直，就可以用直线来描述它。

不妨设该曲线的长度为 L，并根据曲线的外观将其中心线分成三段，如图 2.5 所示。这三段分别为：L1 段（加工图形的起始过渡部分，该段逐渐变宽），等宽段（即中间段），以及 L2 段（加工图形的结尾过渡部分，该段逐渐变窄）。砂轮的起始雕刻角度和终止雕刻角度可以由用户调节（图 2.5 表示的起始角度和终止角度为 0）。

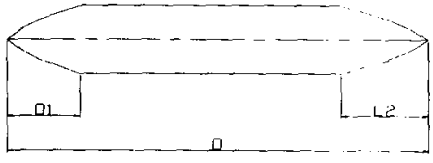


图 2.5　玻璃雕刻机加工图形的定义

Figure 2.5 The Definition of Processed Unit

当中心线为直线时，即为 2.4（a），当中心线为曲线时，即为 2.4（b），当 L2 为 0 时，即为 2.4（c）。除这几种图形外，还能加工出等宽度的直线和曲线等样式更多的图形。

这里用结构 CENTERLINEATTR 来定义待加工图形的属性：

```
typedef  struct  _CENTERLINEATTR
{
  unsigned char   Type;
  float           Theta1;
  float           Theta2;
  float           L1;
```

```
    float          L2;
    float          Width;
    float          Depth;
} CENTERLINEATTR;
```

其中，Type 用来定义过渡轮廓线的类型，目前采用的是二次曲线；Theta1 是砂轮在中心线起点处与其前进方向的夹角；Theta2 是砂轮在中心线结尾处与其前进方向的夹角；L1 是起始过渡轮廓线的中心线长度与中心线总长之比；L1 是结尾过渡轮廓线的中心线长度与中心线总长之比；Width 是轮廓线的最大宽度，由用户来给出；Depth 是轮廓线的最大深度，同样由用户来给定。

2.2.2.2 待加工图形中心线（原始数据）的描述

一个设计合理的玻璃雕刻机应当对通过扫描而得到的待加工图形加以支持，而通过扫描得到的光栅数据是一种离散形式的数据，为了简化问题，避免进行离散图形的矢量化转换，我们采用离散形式来描述待加工图形，即用一组折线来近似地表示每一条加工图形的中心线。也就是说，即使是待加工图形是以矢量形式得到的，我们也先用插点的方式对其进行离散化，统一成一致的数据格式。

首先，针对对于折线化的中心线，我们是定义了一个结构 POSITIONF 来描述每一个折点：

```
typedef  struct _POSITIONF
{
float          x;
float          y;
}POSITIONF;
```

其中，x，y 分别为该点的 X 坐标值和 Y 坐标值。

然后，在 POSITIONF 结构的基础上，定义另一个结构 LINESF 来描述折线形式的中心线：

```
typedef  struct _LINESF
{
unsigned short   Total;
POSITIONF    Vertex[1];
}LINESF;
```

Total 是折线的顶点数，Total>=2；Vertex 是一个 POSITIONF 类型的结构数组，用来确定折线的顶点，其下标在此仅是占位之用，数组的大小是根据 Total 的值而定的。此结构不可用 sizeof 来直接求其大小，而应用 sizeof(LINESF)+(Total-1)*sizeof(POSITIONF)来确定其大小。

LINESF 结构在内存中的存储结构见图 2.6。

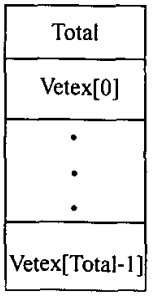


图 2.6 LINESF 结构在内存中的存储结构

Figure 2.6 The Storage Structure of LINESF

2.2.2.3 砂轮运动轨迹（输出数据）的描述

根据待加工图形的中心线，可以通过一些具体的算法计算出砂轮对应每一点的空间姿态，并且，这些空间姿态贯穿起来，就组成了玻璃雕刻机所需要最终数据——砂轮运动轨迹。将这些砂轮运动轨迹以控制指令的方式传送给下位控制器，玻璃雕刻机就可以进行玻璃雕刻了。（至于如何从二维数据生成四位轨迹，以及如何实现四轴联动，将在第四、五章中详细介绍。）

上一节中介绍的数据结构仅仅是二维数据，可以是用来描述待加工图形的中心线（输入数据），但是它只包含了运动中砂轮的 x、y 轴坐标，这对于四自由度的玻璃雕刻机来说是不够的。要明确表示砂轮某一时刻的空间姿态，还必须知道砂轮的吃刀深度（z 方向的位移）以及转角 θ，因此，必须重新设计数据结构来描述生成的砂轮运动轨迹，并以此为标准，在玻璃雕刻机的下位控制器程序也采用相同的数据结构，从而实现上下位机间的通信。

类似地，我们首先定义了一个结构 POSEF 来描述砂轮的空间姿态：

```
typedef  struct  POSEF
{
float          x;
float          y;
float          z;
float          Theta;
}POSEF;
```

其中 x、y、z 分别为砂轮 X 坐标、Y 坐标和吃刀深度，Theta 为砂轮与 X 轴的夹角，$-\pi<=Theta<\pi$。确定了以上四个参数，即可确定砂轮的空间位置。

砂轮沿着中心线运动，中心线上的每一点都对应着一个砂轮的空间姿态，与中心线的定义类似，定义了一个结构 PATHF 来描述一系列的砂轮空间姿态。

```
typedef  struct  PATHF
{
unsigned short   Total;
POSEF          Pose[1];
}PATHF;
```

Total 是沿着一条中心线运动的空间姿态总数；Pose 是一个空间姿态数组，其下标在此仅是做占位之用，数组的实际大小是根据 Total 的值而定的，其大小的计算方法应为 sizeof(PATHF)+(Total-1)*sizeof(POSEF)。PATHF 的存储方式与 LINESF 相似。

2.3 预览图形的程序描述

在第一章中，我们介绍本课题与国外进口玻璃雕刻机相比，创新点之一就是加入了雕刻预览功能，以提高成品率。可以根据砂轮运动轨迹（输出数据）中每一点砂轮空间姿态和砂轮的截面方程，如式（2—4），计算出砂轮在每一点的切削痕迹，贯穿起来就可以得到最后的雕刻效果预览。如何从一组离散的切削痕迹得到最后的雕刻效果预览图，就是包络曲线的获取。

包络现象广泛地存在于工程技术中。机加工的展成法用刀具去切削工件，可看作空间直线族的包络；成型轧制中孔型和轧件之间的关系，是一种空间曲线族的包络；在平面二次包络弧面蜗杆传动的加工中，用平面砂轮去磨削蜗杆，是平面族的包络。当电脑雕刻机的砂轮在玻璃上移动时，就会形成平面曲线族的包络。为了达到预览的效果，便于用户根据自己的

意愿来自由地修改包络线的宽度和样式，必须求出当砂轮的中心点沿着给定的中心线运动时产生的包络线。

其示意图如图 2.7 所示：

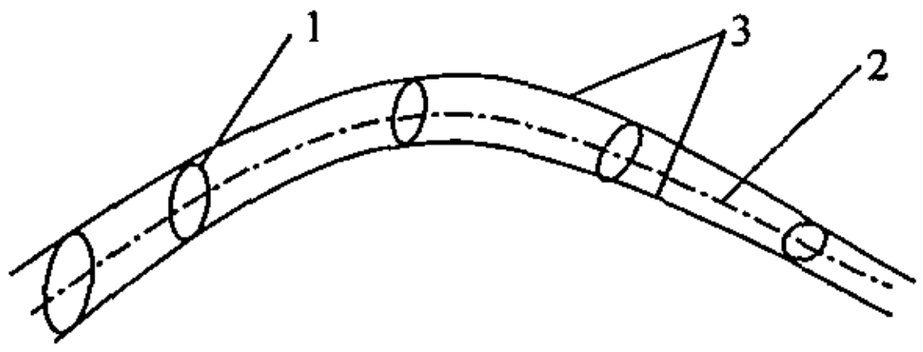


图 2.7　砂轮运动时形成包络线的示意图

Figure 2.7 The Creation of Engraving Outline

其中，1 为砂轮，2 为砂轮中心沿着运动的中心线，3 为砂轮切割玻璃时形成的包络线。

要求如图所示的包络线，通常有两种方法：解析法和离散法。解析法利用包络原理，用计算的方法求出如图所示的包络线 3 的解析式；离散法是采用近似的方法，在每一个位置求出砂轮的切痕轮廓线在前进方向上的最高点和最低点，分别将所有的最高点和最低点连起来，也就得到了包络线。

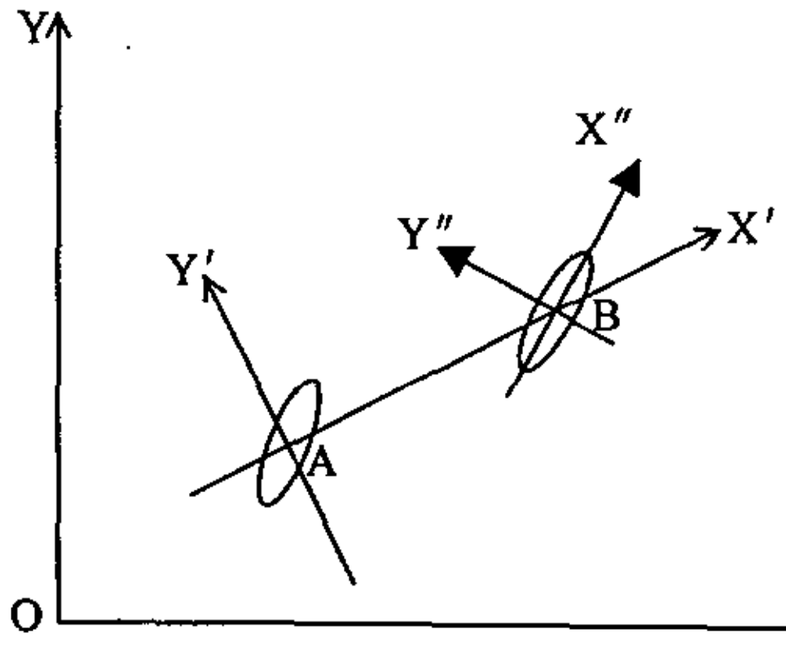


图 2.8　砂轮由 A 运动到 B 时三种坐标系的示意图

Figure 2.8 The Coordination Conversion From A to B

2.3.1 坐标系变换和图形变换

在本课题的开发过程中，多次涉及到坐标系变换和图形变换，其中在离散法求取包络线轨迹的过程中，就需要在三种坐标系之间进行转换（如图 2.8 所示）。此外，本课题的其他模块如图形输入和图形编辑部分也有不少地方需要对坐标系进行转换，因此对各种转换矩阵进行总结是有必要的。

2.3.1.1 坐标系变换

坐标系变换图形学和机器人运动学的基础内容之一，通过坐标系转换很多时候能使复杂问题简单化，往往在另一个坐标系中难以计算的问题，在另一个坐标系里能简单地得以解决。

由于本文的研究主题是平板玻璃雕刻，所涉及的转换大都是在平板玻璃平面上的各坐标系间进行，所以本部分讨论的转换矩阵以二维坐标系转换矩阵为主。

1）二维变换矩阵

不妨设两种坐标系下，某个点的坐标分别为[x,　y,　1]′ 和[x′ ,y′ ,1]′ ，它们的几何变换矩阵为 T_{2D}。

二维图形几何变换矩阵可以用下式表示：

$$T_{2D}=\left(\begin{array}{cc:c} a & b & c \\ d & e & f \\ \hdashline g & h & i \end{array}\right) \qquad \text{（式 2—8）}$$

从变换功能上可以把上述的变换矩阵分为四个子矩阵。其中$\begin{bmatrix} a & b \\ d & e \end{bmatrix}$是对图形进行缩放、旋转、对称、错切等变换；$\begin{bmatrix} c \\ f \end{bmatrix}$是对图形进行平移变换；$\begin{bmatrix} g & h \end{bmatrix}$对图形进行投影变换；而$[i]$则是对整个图形作伸缩变换。

2）平移变换矩阵

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}=\begin{pmatrix} 1 & 0 & x_A \\ 0 & 1 & y_A \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}=\begin{pmatrix} x'+x_A \\ y'+y_A \\ 1 \end{pmatrix} \qquad \text{（式 2—9）}$$

平移变换如图 2.9（a）所示。

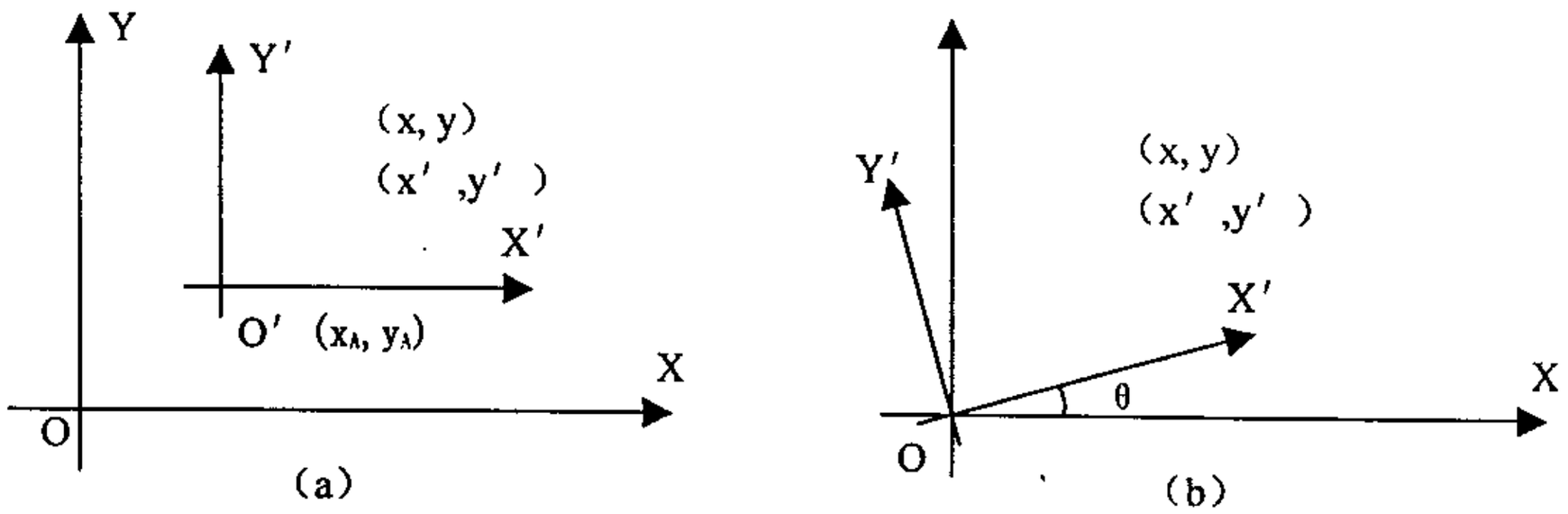


图 2.9　坐标系变换

Figure 2.9 The Conversion of Coordination

3）旋转变换矩阵

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}=\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}=\begin{pmatrix} x'\cdot\cos\theta-y\cdot\sin\theta \\ x'\cdot\sin\theta+y\cdot\cos\theta \\ 1 \end{pmatrix} \qquad \text{（式 2—10）}$$

平移变换如图 2.9（b）所示。

4）复合变换矩阵

复合变换是指坐标系作一次以上的几何变换，变换结果是多次变换矩阵相乘。比如在图 2.7 中，坐标系 X″ BY″ 到 XOY 的变换矩阵为：

$$\begin{aligned} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & x_B \\ 0 & 1 & y_B \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta & -\sin\theta & x_B \\ \sin\theta & \cos\theta & y_B \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \end{aligned} \qquad \text{（式 2—11）}$$

$$=\begin{pmatrix} x'\cdot\cos\theta - y\cdot\sin\theta + x_B \\ x'\cdot\sin\theta + y\cdot\cos\theta + y_B \\ 1 \end{pmatrix}$$

2.3.1.2 图形变换

与坐标系变换相似，图形变换也是图形学的基本内容，两者的区别在于图形变换是在同一个坐标系里对图形进行平移、放缩、对称、旋转与错切等操作，使曲线 F（x，y）变换为 F′（x，y)。玻璃雕刻机的图形绘制和编辑模块中，涉及了大量的图形变换操作。

由于基本图形变换矩阵、平移、旋转矩阵与上述（式 2—8)、(式 2—9）和（式 2—10）一样，故本节不再重复。

1） 比例变换矩阵

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S_x \cdot x' \\ S_y \cdot y' \\ 1 \end{pmatrix} \qquad \text{（式 2—12）}$$

其中：

a）当 $S_x=S_y=1$ 时，为很多比例变换，即图形不变；

b）当 $S_x=S_y>1$ 时，图形沿着两个坐标轴方向等比例放大；

c）当 $S_x=S_y<1$ 时，图形沿着两个坐标轴方向等比例缩小；

d）当 $S_x\neq S_y$ 时，图形沿着两个坐标轴方向作非均匀的比例变换。

2）对称变换矩阵

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ d & e & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} ax' + by' \\ dx' + ey' \\ 1 \end{pmatrix} \qquad \text{（式 2—13）}$$

特殊地：

a）当 b=d=0，a=－1，e=1 时，有 x=－x′ ，y=y′ ，产生以 y 轴为对称轴的对称变换图形；

b）当 b=d=0，a=1，e=－1 时，有 x=x′ ，y=－y′ ，产生以 x 轴为对称轴的对称变换图形；

c）当 b=d=0，a=e=－1 时，有 x=－x′ ，y=－y′ ，产生以原点为对称中心的中心对称变换图形；

d）当 b=d=1，a=e=0 时，有 x= y′ ，y= x′ ，产生以直线 y=x 为对称轴的对称变换图形；

e）当 b=d=－1，a=e=0 时，有 x= －y′ ，y=－x′ ，产生以直线 y=－x 为对称轴的对称变换图形。

3）错切变换矩阵

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & b & 0 \\ d & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x' + by' \\ dx' + y' \\ 1 \end{pmatrix} \qquad \text{（式 2—14）}$$

其中：

a）当 d＝0 时，x=x′ +by′ ，y=y′ ，此时，变换后的图形 y 坐标不变，x 坐标随

初值（x′ ,y′ ）及变换系数 b 而作线性变化；如 b>0，图形沿着 x 轴正方向作错切位移；如 b<0，图形沿着 x 轴负方向作错切位移；

b）当 b＝0 时，x=x′ ，y=d x′ +y′ ，此时，变换后的图形 x 坐标不变，y 坐标随初值（x′ ,y′ ）及变换系数 d 而作线性变化；如 d>0，图形沿着 y 轴正方向作错切位移；如 d<0，图形沿着 y 轴负方向作错切位移；

c）当 b≠0，d≠0 时，x=x′ +by′ ，y=d x′ +y′ ，此时，图形沿着 x，y 轴两个方向作错切位移。

2.3.2 解析法求解包络线

在用计算法求包络线之前，必须先给出单参数平面曲面族的包络原理，这是用解析法求解包络线的基础。

2.3.2.1单参数平面曲线族的包络

对于隐式给出的单参数平面曲线族，其方程为

$$\{C_\alpha\}: F(x,y,\alpha)=0$$

当 α 取某个实数值时，就得到一条确定的族中曲线 Cα 。

如果存在一条平面曲线 C，它的每一点至少和族中一条曲线相切，并且 C 与族中每一条曲线至少切于一点，则称 C 为曲线族{ Cα }的包络线，简称 C 是{ Cα }的包络。

据此定义，如果单参数平面曲线族{ Cα }的包络 C 存在，则对于任意的 M(x,y)∈C，必有族中曲线 Cα 与包络在 M 点相切，M 点的坐标 x 与 y 由 α 来确定，也就是说，包络 C 的方程可表示为：

$$C:\begin{cases} x=x(\alpha) \\ y=y(\alpha) \end{cases}$$

由于确定的 α 值所对应的点 M(x,y)，既在 C 上，也在 Cα 上，因此有恒等式

$$F(x(\alpha),y(\alpha),\alpha)\equiv 0$$

上式对 α 求偏导，有

$$F_x\cdot\frac{dx}{d\alpha}+F_y\cdot\frac{dy}{d\alpha}+F_\alpha=0$$

曲线 Cα 在 M 点的法向量为 N=Fx・i+Fy・j，而包络 C 在 M 点的切向量 T=dx/dα ・i+dy/dα ・j。由 T・N=0 可得到：

$$F_x\cdot\frac{dx}{d\alpha}+F_y\cdot\frac{dy}{d\alpha}=0$$

也就是包络 C 上的点 M 必须满足条件

$$F_\alpha(x,y,\alpha)=0 \qquad \text{（式 2—15）}$$

称式（2—15）为曲线族{ Cα }的包络条件，而包络 C 的方程可表示为

$$C:\begin{cases} F(x,y,\alpha)=0 \\ F_\alpha(x,y,\alpha)=0 \end{cases} \qquad \text{（式 2—16）}$$

上式是用方程组来表示包络 C 的，如果能从第二式解出 α = α (x, y)代入第一式，便得到 C 的隐式表示（不含参数 α ）：

$$\phi(x,y) = F[x,y,\alpha(x,y)] = 0 \qquad \text{（式 2—17）}$$

（式 2—16）还说明，如果单参数平面曲线族{ Cα }的包络 C 存在，则 C 必可用该式表示， Fα (x, y, α)=0 是包络存在的必要条件。

以下是曲线族包络存在的充分条件（详见参考文献[1]）。

包络存在的充分条件：如果在平面曲线族{ Cα }：F(x, y, α)=0 的族中曲线 α_0 上任一点 M_0（x_0, y_0）满足 $F(x_0, y_0, \alpha_0)=0$，且在 M_0 点的某个邻域内函数 F(x, y, α)对三个变元有一、二阶连续偏导数，在该点处的 J≠0 及 Fα α ≠0，则判别曲线（式 2—16）必为曲线族{ Cα }的包络。

其中 J 为雅可比行列式，

$$J = \frac{\partial(F, F_\alpha)}{\partial(x,y)} = \begin{vmatrix} F_x & F_y \\ F_{x\alpha} & F_{y\alpha} \end{vmatrix}$$

由以上的讨论，可知：

曲线族{ Cα }的包络必在判别曲线（式 2—16）中， 但判别曲线未必是都是包络。

2.3.2.2 求解步骤分析

由 3.1 的讨论可知，在玻璃雕刻机的刀具坐标系中，刀具的几何模型可描述为

$$F(x,y,z) = 0$$

但在世界坐标系中，由于砂轮会相对坐标原点平移和绕 Z 轴旋转，因此，通过平移变换和旋转变换，其方程可写成

$$F(x,y,z,x_0,y_0,\theta) = 0 \qquad \text{（式 2—18）}$$

其中 x0，y0 为砂轮中心在世界坐标系中的坐标， θ 为砂轮绕 Z 轴的转角。该式共有 z，x0，y0， θ 四个未知数，如果相互之间独立，则无法求出其包络线。因此，必须人为地建立联系，z， θ 均随着 x0，y0 的变化而相应地变化，即

$$z = f_z(x_0, y_0) \qquad \text{（式 2—19）}$$

$$\theta = f_\theta(x_0, y_0) \qquad \text{（式 2—20）}$$

将（式 2—19）和（式 2—20）代入（式 2—18），得到

$$F(x,y,x_0,y_0) = 0 \qquad \text{（式 2—21）}$$

又因为 y_0 是 x_0 的函数，即，

$$y_0 = f(x_0) \qquad \text{（式 2—22）}$$

将（式 2—22）代入（式 2—21），可得

$$F(x,y,x_0) = 0 \qquad \text{（式 2—23）}$$

（式 2—23）即为单参数平面曲线族的标准形式，由（式 2—16）即可求得砂轮运动的包络线。

设砂轮与玻璃相交曲线（即图 2.1、2.2 中的俯视图）上任一点的坐标为（x, y），在刀具坐标系中的坐标为（x´, y´），砂轮的转角与 x 轴正方向的夹角为 θ，砂轮的中心的坐标为（x_0, y_0），由坐标变换，有：

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & x_0 \\ \sin\theta & \cos\theta & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \qquad \text{（式 2—24）}$$

因此，

求出该逆矩阵，即得：

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & x_0 \\ \sin\theta & \cos\theta & y_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \text{（式 2—25）}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & -x_0\cos\theta - y_0\sin\theta \\ -\sin\theta & \cos\theta & x_0\sin\theta - y_0\cos\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \text{（式 2—26）}$$

由（式 2—26），有：

$$\begin{cases} x' = x\cos\theta + y\sin\theta - x_0\cos\theta - y_0\sin\theta \\ y' = -x\sin\theta + y\cos\theta + x_0\sin\theta - y_0\cos\theta \end{cases} \qquad \text{（式 2—27）}$$

由（式 2—4），有：

$$y' = \pm\frac{W}{4H}\left(D - 2\sqrt{(D/2+z)^2 + x'^2}\right) \qquad \text{（式 2—28）}$$

$$\theta = f(x_0, y_0) \qquad \text{（式 2—29）}$$

若

且已知

$$z = f(x_0, y_0) \qquad \text{（式 2—30）}$$

$$y_0 = f(x_0) \qquad \text{（式 2—31）}$$

将（式 2—27）、（式 2—29）、（式 2—30）与（式 2—31）代入（式 2—28），得到一个含有参数 x_0 的方程 θ

$$f(x, y, x_0) = 0 \qquad \text{（式 2—32）}$$

对（式 2—32）求 x_0 的偏导，并与（式 2—32）本身联立，消去 x_0，即得到所求的包络线方程。

2.3.3 离散法求解包络线

中心线是用一系列离散的点来描述的，任何两点之间的距离非常小。对应中心线上的每一点都有一个砂轮的空间姿态与之相对应，产生一个切痕轮廓线，一连串的切痕轮廓线的外

沿也就形成了包络线。因此，对于每一个切痕轮廓线，相对于其前进方向的运动坐标系（注意，不是世界坐标系）而言，都有一个最高点和最底点，将每一个切痕轮廓线的最高点和最底点连起来，即形成包络线的外沿，如果中心线上任何两点之间的距离足够小的话，从精度上是完全可以达到要求的。

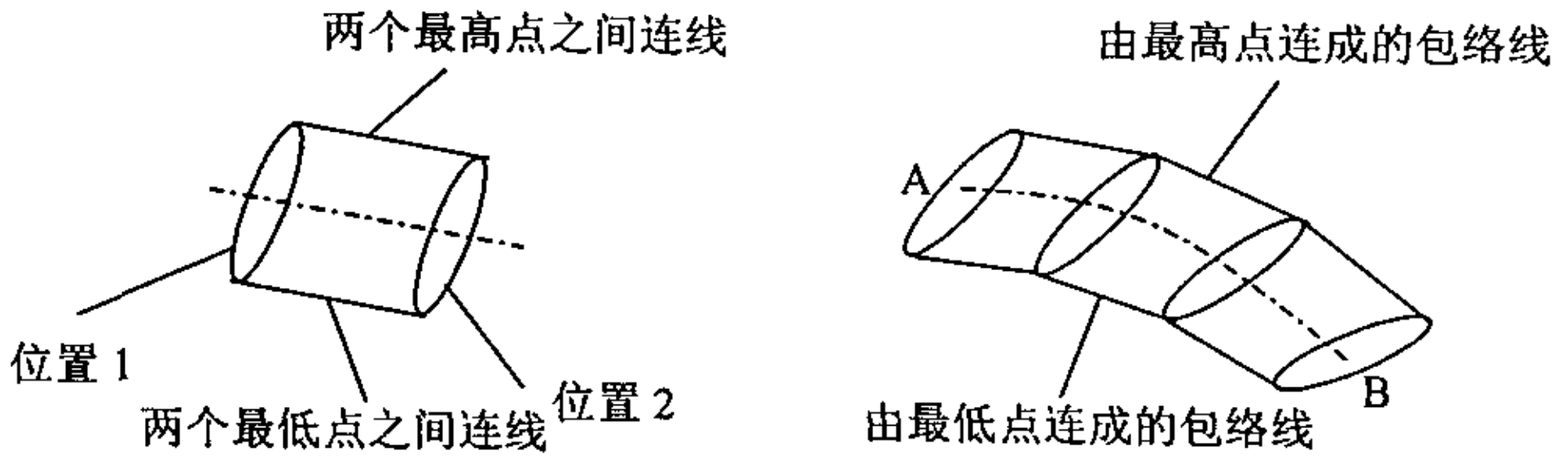


图 2.10　用离散法求解包络线的示意图

Figure 2.10 Calculation of Outline Discretely

图 2.10 表示了用离散法求解包络线的过程。如图 2.10（a），当砂轮由位置 1 运动到位置 2 时，分别求出位置 1 和位置 2 在前进方向上的最高点和最低点，将两个最高点和最低点连起来，即为砂轮在这两点的包络线。如图 2.10（b），当砂轮沿中心线由 A 运动到 B 时，将这两点之间所有位置在前进方向上的最高点和最低点连起来，就得到两条完整的包络线。

以下是砂轮在某个位置时最高点和最低点的求法。

如图 2.8 所示，XOY 为世界坐标系，X′A Y′为运动坐标系，其 x 轴即 A 到 B 连线，X″B Y″为刀具坐标系。为了便于计算，本章 2.1 节计算的切痕轮廓线方程（式 2—4）是相对刀具坐标系 X″B Y″而言的。同时，为了计算出运动坐标系中的最高点和最低点，需要运动坐标系中进行。因此，至少需要两次坐标变换，它们可以通过使用前面讨论过的坐标系变换矩阵来实现。

首先，先将曲线方程从刀具坐标系 X″B Y″转换到运动坐标系 X′A Y′中，找到单个切痕轮廓线的最高点与最低点，依次类推，对于中心线上的每一点所对应的切痕轮廓线，均可由该点与其下一点建立运动坐标系（最后一点除外，最后一点由该点与其前一点建立运动坐标系），来求取最高点与最低点。将所有的最高点连起来，即得到一条包络线，将所有的最低点连起来，得到另一条包络线。这些包络线的坐标都是在运动坐标系中得出的，最终还需要转化成世界坐标系，变换方法同上。

2.3.4 矢量法与离散法的比较及选取

显然，上述两种方法中，解析法解析法严谨、精确，但计算复杂，难以实现；相比之下，离散法则比较直观，易于程序实现。由于计算包络线的目的是为了进行雕刻轨迹预览和走刀模拟，因此这部分的精度要求比传送给控制器的实际雕刻数据的精度要求要低，而且另一方面，在离散化的过程中，只要选择合适的折点间距，精度能够达到要求。综合以上分析，我们在最终的程序中选用离散法来求取包络线轨迹。

第三章 模块划分及软件建模

3.1 模块化思想在Glass Sculptor中的应用

3.1.1 模块化思想简介

大型软件的复杂性往往来自于三个方面。其一，由于大型软件通常由多人合作完成，如何划分任务，估计、分配资源，掌握、控制、检查每一阶段的设计标准等，导致了大型软件在设计管理方面的复杂性；其二是来自于大量的系统状态。一个复杂的系统必须正确处理多种每个状态，组织系统的程序逻辑，验证系统的正确性，因此在实现方面也非常困难；其三，随着用户需求的增加、系统漏洞的发现，大型软件必须考虑升级的问题，然而如何将升级服务方便的提供给最终用户，导致了大型软件在维护方面的复杂性。

大型软件设计的首要目标是正确性，同时强调其强健性、易维性、可读性、可重新使用性等，已经超过了执行效率和系统开销方面的要求。基于大型程序的特点，要求人们在开发一个大型软件系统时，应对整个任务进行清晰的，无歧义的分划，因为这可以使得每个开发人员都能清楚地知道自己了解自己所处的位置，了解自己所要做的工作，以及与其他程序员所开发模块的接口，这样他们才能更好地设计、调试自己的模块。可以这么说，大型软件的特点决定了软件工业必然会走上模块化的道路。

然而，什么是模块化呢？

举个简单例子来说：做一把椅子，如果不能通过把它分割成几个部件来分别做，那只有用凿子从一整块木头里凿出一把椅子来了。显而易见，这样的方法的效率低下，而且除了技艺高深的雕塑家外，普通人很难运用这种方法雕刻出美观耐用的家具。在现实生活中，家具几乎都是以更合理的方式加工出来的——按部就班地做好椅子腿、椅子面、靠背和扶手，然后将这些部分组装起来便可以得到一个完整的椅子。这种方式具有更多的优越性：椅子的几个部分可以由几个人分工合作，同时进行加工，易于实现家具产业的大规模生产；更重要的是，以后如果椅子的某个部分遭到损坏，只需简单地更换相应的部分即可。

程序设计与上述过程很相似：早期的程序往往由一个人完成，在某种意义上就像艺术家的作品，完全是个人风格的产物。而模块化技术的出现大大推动了软件产业，使分工合作易于实现，以往不可想象的大系统可以被分割成一个个相对独立的小问题来分步实现，而后再组装成大的系统，软件的规模和质量得以提高。

把椅子腿和其他部分组装起来是靠木榫、钉子和木胶，如何把模块化的程序组装起来呢？通用的方法有动态连接库（DLL）和组件对象模型（COM）技术，在本课题的设计过程中，采用了 DLL 技术，本节将从理论到实践，逐步介绍 DLL 在 Glass Sculptor 中的应用，至于 COM 可以作为玻璃雕刻机将来进一步完善的考虑，限于篇幅，本文不作深入介绍。

3.1.2 动态连接库（DLL）概述

3.1.2.1 动态链接库（Dynamic Link Library，DLL）

动态链接库 DLL（Dynamic Link Library，也称为动态库或库模块）是一个程序模块，

它包含代码、数据或资源，可以被其他应用程序共享。它与操作系统的联系极其紧密，事实上从最初的版本开始，动态链接库就是 Microsoft Windows 最重要的组成要素之一，所有的 Win32 API 函数都包含在 DLL 中，Windows 系统中 3 个最重要的 DLL 是 Kernel32.dll，它由管理内存、进程和线程的函数组成；User32.dll，它由执行用户截面的认为（如创建窗口和发送消息）的函数构成；Gdi32.dll，它由绘图和显示文本的函数组成。Windows 的很多功能都是由一些 DLL 完成，他们含有执行某些专门任务的函数。如，Advapi32.dll 包含了有关对象安全、注册表管理和事件记录的函数，Comdlg32.dll 则处理常见的“打开文件”、“另存为……”等通用对话框。可以说，每当直接访问一些 Window API 函数时，其实是在访问相应的 DLL。

一个动态链接库在结构上与一个可执行文件（.exe)很相似，它们之间的主要差别在于 DLL 不能单独执行。动态链接库的文件扩展名一般是.dll，也有可能是.drv（设备驱动程序）、.sys（系统文件）和.fon（字体文件）。

所谓“动态链接”，是指 Windows 把一个模块中的函数调用链接到动态链接库文件中的实际函数上的过程。在程序开发中将各种目标模块（.OBJ）、运行库（.LIB）文件，以及经常是已编译的资源（.RES）文件链接在一起，以便创建 Windows 的.EXE 文件，这是的链接是静态链接，动态链接与此不同，它发生在运行时刻。

有些动态链接库（如字体文件等）被称为“纯资源”。它们只包含数据（通常是资源的形式），而不包含代码。由此可见，动态链接库的目的之一就是提供能被许多不同的应用程序使用的函数和资源。在常规的操作系统中，只有其本身才包含其他应用程序能够调用来完成某一作用的例程。在 Windows 中，一个模块调用另一个模块函数的过程被推广了。事实上，编写一个动态链接库，也就是在扩充 Windows。当然，也可以认为动态链接库（包括 Windows 本身的 DLL）是对用户程序的扩展。

3.1.2.2 为什么在Glass Sculptor中使用DLL?

使用动态链接库有很多优点，正是基于一下这些原因，我们在 Glass Sculptor 中采用了动态链接库：

- 有利于程序的模块化。当需要对应用程序进行修改时，只需要修改其中的一个模块，而不是整个应用程序。因此可以将系统中可能需要频繁升级的部分放在某个 DLL 里，升级市只需向用户提供新的 DLL 版本，代替当前的 DLL 文件。其实，这也是 Glass Sculptor 中将主要算法模块用 DLL 来实现的主要原因；
- 有利于共享代码、资源和数据。因此，应该将那些被应用程序的好几个部分都需要调用的函数和资源编写成 DLL 文件；
- 有助于节省内存。如果两个或多个线程要用到同一个 DLL，那么该 DLL 只要被载入内存一次即可使得所有的线程都可以调用它。C/C++运行库就是个很好的例子，许多应用程序都使用这个库，如果所有的应用程序都链接到这个静态库，那么 sprintf、strcpy 和 malloc 等常见函数的代码就要多次存在于内存中。但是实际上它们是以 DLL 的形式被链接的，因此它们的代码只需载入内存一次。采用 DLL，可以使 Glass Sculptor 的系统开销降低，内存的使用更加有效；
- 适合于复杂程序开发。当开发一个庞大而且需不断更新或改正错误的应用程序，可以将其划分为多个执行部分和 DLL，这样只对需要改变的部分进行操作，而不是对整个文件进行改动，保证了其他部分的完整性，减少了误操作导致问题的可能；
- 有助于解决平台差异。不同版本的 Widnows 版本的 API 函数内部实现是不一样的，开发人员常常想要调用新的函数，但是，如果源代码包含了对一个新函数的调用，并且应用程序将要在不提供该函数的 Windows 版本上运行，那么操作系统的加载程

序将拒绝运行这个进程，即使实际上从不调用该函数，情况也是这样。如果将这些函数保存在 DLL 中，那么应用程序就能够将它们加载到 Windows 的老版本上。当然，系统仍然可以成功地调用该函数。Glass Sculptor 是个面向用户的商业系统，必须考虑到用户各种可能的使用环境。

3.1.2.3 映射DLL到进程的地址空间

一个动态链接库是一组源代码模块或资源数据，编写这些 DLL 时应假设有应用程序（.EXE 文件）和别的 DLL 会调用它们。而一个应用程序（.EXE 文件）和别的 DLL 要调用某个 DLL 中的函数或需要使用该 DLL 的资源，就必须首先把 DLL 的文件映象映射到调用进程的地址空间中。一旦 DLL 文件映象被映射到调用进程的地址空间，DLL 中的函数和资源就能被运行在这个进程中的所有线程使用。实际上可以这么理解，DLL 几乎失去了它的 DLL 的身份，对于进程中的所有线程，DLL 的代码和数据就像偶尔出现在进程的地址空间中的额外的代码和数据。一旦线程调用了任一个 DLL 函数，DLL 函数查看线程的栈来得到它的传递函参数，一能使用线程的栈来分配任何它需要的局部变量。而且，DLL 函数中的代码创建的任何对象都归调用的线程或进程所有，DLL 在 Win32 中什么都不拥有。

把 DLL 文件映象映射到调用进程的地址空间的方法有两种：显式的运行时链接和隐式的加载时链接。

1. 隐式链接

隐式链接时将 DLL 的文件映象映射到进程的地址空间中最常用的方法，当链接一个应用程序时，必须指定要链接的一组 LIB 文件，每个 LIB 文件中包含了 DLL 文件允许应用程序或另一个 DLL 调用的函数的列表。当链接器看到应用程序调用了某个 DLL 的 LIB 文件中给出的函数时，它就在生成的 EXE 文件映象中加入了信息，指出了包含有函数的 DLL 文件的名称。当操作系统载入 EXE 文件时，系统查看 EXE 文件映象的内容来看要装入哪些 DLL，而后试图敬爱你跟需要的 DLL 文件映象映射到进程的地址空间中。当寻找 DLL 时，系统按照下列顺序寻找文件映象：

1) 包含EXE映象文件的目录；
2) 进程的当前目录；
3) Windows的系统目录；
4) Windows目录；
5) PATH 环境变量中列出的目录。

如果找不到 DLL 文件，系统将会显示一个消息框给出提示，然后立即中止整个进程。

当使用隐式链接时，映射到进程的地址空间中的所有的 DLL 到进程终结实才被解除映射。

2. 显式链接

当进程中的一个线程调用 LoadLibrary 或 LoadLbraryEx 函数时，DLL 文件能被显式地映射进程的地址空间。

```
HINSTANCE LoadLibrary (LPCTSTR pszDllPathName) ;

HINSTANCE LoadLibraryEx(
                        LPCTSTR pszDllPathName,
                        HANDLE hFile,
                        DWORD dwFlags);
```

这两个函数均用于找出用户系统上的文件映像，并设法将DLL的文件映像映射到调用进程的地址空间中。两个函数返回的HINSTANCE值用于标识文件映像映射到的虚拟内存地址。如果DLL不能被映射到进程的地址空间，则返回NULL。LoadLibraryEx函数有两个参数：hFile和dwFlags。参数hFile保留供将来使用，现在必须是NULL。对于参数dwFlags，必须将它设置为0，或者设置为DONT_RESOLVE_DLL_REFERENCES、LOAD_LIBRARY_AS_DATAFILE和LOAD_WITH_ALTERED_SEARCH_PATH等标志的一个组合。下面是这些标志的意义：

- DONT_RESOLVE_DLL_REFERENCES标志用于告诉系统将DLL映射到调用进程的地址空间中。通常情况下，当DLL被映射到进程的地址空间中时，系统要调用DLL中的入口函数，即DllMain，该函数用于对DLL进行初始化。DONT_RESOLVE_DLL_REFERENCES标志使系统不必调用DllMain函数就能映射文件映像。
- LOAD_LIBRARY_AS_DATAFILE标志告诉系统只是将DLL映射到进程的地址空间中，就像它是数据文件一样。系统并不花费额外的时间来准备执行文件中的任何代码。该标志在将那些“纯资源”DLL或者是另一个.exe文件映像映射到当前进程的地址空间时常常被用到，借助被映射文件的HINSTANCE值，就能够访问该文件中的资源。
- LOAD_WITH_ALTERED_SEARCH_PATH标志用于改变LoadLibraryEx用来查找特定的DLL文件时使用的搜索算法。通常情况下，LoadLibraryEx隐式载入的顺序进行文件的搜索。但是，如果设定了LOAD_WITH_ALTERED_SEARCH_PATH标志，那么LoadLibraryEx函数就按照下面的顺序来搜索文件：
  1) pszDLLPathName参数中设定的目录；
  2) 进程的当前目录；
  3) Windows的系统目录；
  4) Windows目录；
  5) PATH环境变量中列出的目录。

以Glass Sculptor为例来说，在其主要算法模块compute.dll中导出函数的代码如下：

```
extern "C" __declspec(dllexport)
unsigned long _GenerateToolPath(const TOOL* Tool,
                                const CENTERLINEATTR* CenterLineAttr,
                                const LINESF* CenterLine,
                                unsigned long CenterLineByteSize,
                                PATHF* ToolPath,
                                unsigned long ToolPathByteSize);
```

则在程序中可以用如下语句来对其进行显式载入：

```
HINSTANCE hInstance;
VERIFY(hInstance = ::LoadLibrary("c:\\winnt\\system32\\compute.dll"));
If(1==_GenerateToolPath(&m_Parameter,
                        pCenterlineAttr,
                        pCenterline,
                        sizeof(LINESF)
                        +(pCenterline->Total-1)*sizeof(PCSITIONF),
                        (PATHF*)m_pResultBuffer,
                        MAX_ALGORISM_BUFFER))
    {
        ……//调用DLL中函数后的相应处理语句
    }
```

3．隐式链接和显示链接的比较

隐式链接和显式链接是载入 DLL 的两种方式，它们的区别在于：

- 隐式链接所有的 DLL 都在客户被装载的时候被装载，但在显式链接的情况，可以决定什么时候装载和卸出；
- 隐式链接实现简单，较为常用，显式链接功能更全面，适用于复杂场合，比如，显式链接允许在运行时决定装载哪个 DLL，如果同时有两个语言版本的 DLL，一个带有英文的字符串资源，另一个带有中文字符串资源，那么应用程序可以根据自己的判断或用户的选择而选取不同的 DLL 来载入。

对于玻璃雕刻机的上层软件 Glass Sculptor 而言，由于使用隐式链接的 DLL 可以很好地实现相应的功能，因此目前我们选用隐式链接来载入几个以 DLL 方式来实现的模块，但是从发展的角度来看，隐式链接可以在将来对课题进一步完善时加以考虑。

3.1.2.4 DLL的入口函数——DllMain

通常情况下，连接器为 DLL 指定主入口点_DllMainCRTStartup。当 Windows 加载 DLL 时，就调用该函数，该函数首先调用全局对象的构造函数，然后调用全局函数 DllMain（如果编写了 DllMain 函数）。DllMain 不仅在 DLL 被链接到进程时被调用，而且在断开进程的连接时是也调用。下面是 DllMain 函数的框架：

```
extern "C" int APIENTRY
DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID lpReserved)
{
    UNREFERENCED_PARAMETER(lpReserved);
    if (dwReason == DLL_PROCESS_ATTACH)
    {
        if (!AfxInitExtensionModule(DllmfcextDLL, hInstance))
            return 0;
        new CDynLinkLibrary(DllmfcextDLL);
    }
    else if (dwReason == DLL_PROCESS_DETACH)
    {
        AfxTermExtensionModule(DllmfcextDLL);
    }
    return 1;
}
```

3.1.2.5 扩展DLL和正规DLL

通过使用 Microsoft Visual C++的 AppWizard，可以创建 MFC 库支持的两种 DLL：扩展的 DLL 和正规的 DLL。在决定使用那一种前，我们先比较一下二者的区别。

扩展 DLL 支持 C++接口，也就是说，该 DLL 可以导出整个类，客户程序可以构造这些类的对象或者对这些类进行派生。扩展 DLL 动态链接到 MFC 苦的 DLL 版本的代码，因此扩展 DLL 要求客户程序被动态的链接到 MFC 库，并且，客户程序和扩展 DLL 要一致连接 MFC DLL 的相同版本。所以说，扩展 DLL 的很小，一个简单的 DLL 大约只有 10KB 大小，它的转载也理所当然很快。

如果希望编写的 DLL 不仅是被由 VC 开发客户程序使用，而且也能被其他 Win32 编程环

境（如 VB，Delphi 等）使用，则应该选用正规 DLL，然而，正规 DLL 只能导出 C 风格的函数，但不能导出 C++类、成员函数或重载函数，不过可以在正规 DLL 内部使用 C++类。

可以选择两种方式创建正规 DLL：静态链接或动态链接到 MFC 库，如果选择了静态链接，DLL 将包括所有它需要的 MFC 库代码的拷贝，因此它可以独立于 MFC 库。一个典型的 Release 版本静态链接的正规 DLL 大约为 144KB 左右。如果选择动态连接，大小可以降为 17KB 左右，但前提是，相应的 MFC DLL 在目标机器上存在。

可见，动态链接的正规 DLL 同时具有两方面的优点：体积小，载入快，容易为其他程序共享，有利于将来的升级，因此我们在 Glass Sculptor 中选择采用动态链接的正规 DLL 来实现相应模块。

3.1.2.6 在DLL中实现函数的导入和导出

DLL 包含一个导出函数表，可以通过函数的符号化的名字和称为序号的数字（使用较少）来识别这些函数。函数表也包含了函数在 DLL 内的地址。当客户程序载入 DLL 时，它并不知道它将要调用的函数的地址，但它可以通过符号化的名字或者序号找到相应的函数地址。动态链接的进程然后建立一张表，把客户调用与 DLL 的函数地址连接起来。

在 DLL 代码中，必须显式地声明导出函数，代码如下：

```
__declspec(dllexport) unsigned long _GenerateToolPath(
                                                    ……//函数体
                                                    );
```

在客户程序方面，需要声明对应的导入函数，代码如下：

```
__declspec(dllimport) unsigned long _GenerateToolPath(……);
```

如果使用了 C++，编译器会为函数_GenerateToolPath 产生一个其他语言不能使用的修饰名，如果希望能在非 C++开发的程序中调用_GenerateToolPath，应相应地使用下面的函数声明：

```
extern "C" __declspec(dllexport) unsigned long _GenerateToolPath(……);
extern "C" __declspec(dllimport) unsigned long _GenerateToolPath(……);
```

前面介绍过，扩展 DLL 可以导出 C++类，实现方法是在 DLL 和客户程序的头文件中类的声明处，使用 AFX_EXT_CLASS 宏（该宏根据相应的条件产生不同的代码，在 DLL 中导出类，在客户程序中导入类），如下：

```
class AFX_EXT_CLASS class CEditTool : pub…c CObject
```

在正规 DLL 中对导出函数有自己特殊的要求，当 MFC 库文件 mfc42.dll 被载入市，它把数据存放在一些可靠的全局变量里，如果从扩展中调用函数，mfc42.dll 知道如何代表调用进程取设置这些全局变量，然而，如果是正规 DLL 则全局变量并不同步，其后果不可预知，因此，需要在正规 DLL 导出函数的开始处中插入相应的一段代码 AFX_MANAGE_STATE(AfxGetStaticModuleState())，以 Glass Sculptor 为例，三个算法函数代码如下：

```
extern "C" __declspec(dllexport)
unsigned long _GenerateToolPath(const TOOL* Tool,
                                const CENTERLINEATTR* CenterLineAll,
                                const LINESF* CenterLine,
                                unsigned long CenterLineBy…eSize,
                                PATHF* Too…Path,
                                unsigned long Too…PathBy…eSize)
{
```

```
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return GenerateToolPath(Tool,
                            CenterLineAttr,
                            CenterLine,
                            CenterLineByteSize,
                            ToolPath,
                            ToolPathByteSize);  //算法的主要内容
                                                //在该函数中实现
}

unsigned long _GenerateToolProfile(……)//参数略
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    Return GenerateToolProfile(Tool, Pose, ToolProfile,
                               ToolProfileByteSize);
}

unsigned long _GenerateCutOutline(……)//参数略
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return GenerateCutOutline(Tool, ToolPath, ToolPathByteSize,
                              CutOutline, CutOutlineByteSize);
}
```

3.1.3 DLL在Glass Sculptor中的应用

在 Glass Sculptor 中，出于将来对课题进一步完善和玻璃雕刻机商品化后的升级考虑，我们根据功能，对 Glass Sculptor 进行了模块划分，如图 3.1 所示：

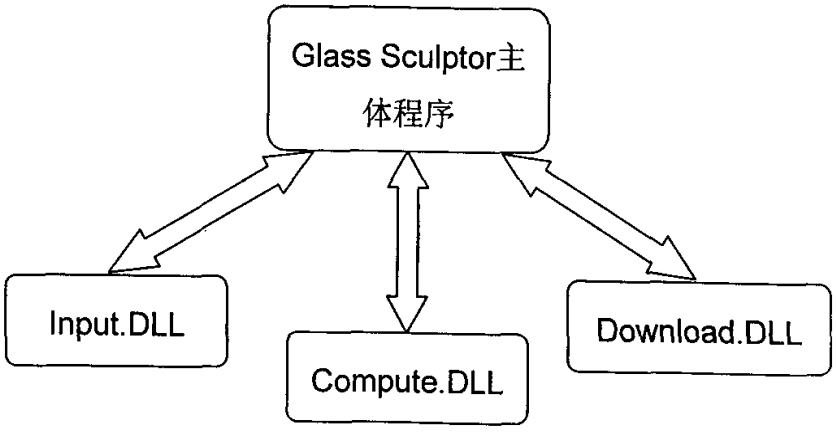


图 2.1 Glass Sculptor 的模块结构

Figure 2.1 The Module Structure of Glass Sculptor

由图可见，Glass Sculptor 由四个主要模块组成：主体程序及三个 DLL。上面几节中所

用的范例大部分是这三个 DLL 中的代码。Glass Sculptor 使用的是动态链接到 MFC 库的正规 DLL，并以隐式链接的形式被主体程序载入。

其中，Input.dll 的任务是提供接口，读取由其他图形应用程序绘制的待加工图形，并将其转换为 Glass Sculptor 自己的数据结构；Compute.dll 是 Glass Sculptor 中最重要的部分，主要的算法都在这个 DLL 中实现，包括由原始的待加工图形计算生成砂轮的运动轨迹，根据每个砂轮姿态计算相应点处的砂轮切痕，并且由同一条中心线上的切痕曲线族得到包络曲线，实现雕刻效果预览，此外，Compute.dll 还包括一些优化算法；Download.dll 根据砂轮的运动轨迹，生成雕刻指令，并通过串口与控制器通讯；主体程序则实现 Glass Sculptor 的界面、简单图形的绘制和修改、并调用这些 DLL 中的函数。

从另一个角度来看，Input.dll、Compute.dll、Download.dll 分别实现了 Glass Sculptor 中的原始数据输入、数据计算处理、结果数据传送，它们包含了玻璃雕刻机的数据流程：由二维数据（中心线坐标）生成四维数据（雕刻指令）。本文第四章将对这部分进行详细讨论。

3.2 玻璃雕刻机的上层软件建模

模型在数学上有着严格的定义和理论基础。而在另一方面，在人们的生活中，模型一词由来已久，并且应用十分广泛。无论是对一个实物的建造，还是一个复杂问题的求解，人们总是在真正实施之前，为其构造若干种模型。譬如，建筑模型用于向客户展示建筑物的整体结构，飞机模型用来进行风洞实验，画油画时先用铅笔勾画轮廓，机械零件的蓝图，广告的构思，写书前的大纲等等。同样，在对玻璃雕刻机离线编程系统进行编码实现之前，同样有必要对其上层软件 Glass Sculptor 建立软件模型。

本节将首先对目前使用较多的标准建模语言 UML 进行介绍，然后从使用实例、逻辑结构和并发特性三个视角着手，利用 UML 方法对 Glass Sculptor 建立用例图、类图、合作图和活动图模型，从而在达到整体上对其进行结构规划。

3.2.1 标准建模语言UML（Unified Modeling Language）

3.2.1.1 什么是标准建模语言UML?

国际上，软件工程领域在最近三年内（1995 年~1997 年）取得了前所未有的进展，其成果超过软件工程领域过去 10 至 15 年来的成就的总和。其中最重要的、具有划时代重大意义的成果之一，就是标准建模语言 UML（Unified Modeling Language）的出现。

UML 是由世界著名的面向对象技术专家 Grady Booch, James Rumbaugh, Jacobson 发起，在著名的 Booch 方法、OMT 方法和 OOSE 方法的基础上，广泛征求意见，集众家之长，几经修改而完成的。设计者们为 UML 设计的目标是：

- 运用面向对象概念来构造系统模型（不仅仅是针对软件）。
- 建立起从概念模型直至可执行体之间明显的对应关系。
- 着眼于那些有重大影响的问题。
- 创建一种对人和机器都适用的建模语言。

UML 采用了一整套成熟的建模技术，广泛适用于各种应用领域。它得到了工业界的广泛支持，人们普遍认为 UML 定将成为工业界广泛接受的语言。在美国，截止 1996 年 10 月，UML 已经获得工业界和科技界的广泛支持，有 700 多家公司表示将采用 UML 语言作为建模语言。UML1.1 版已经于 1997 年 11 月 17 日被 OMG（Object Management Group）批准作为标准，这

标志着近 15 年来面向对象技术中关于建模语言的争论暂时告一段落。

在世界范围内，至少在近 10 年，如同 80 年代的 C 语言和 90 年代的 C++语言那样，UML 将成为面向对象技术领域内占主导地位的标准建模语言。在我国， UML 也应用越来越广，已经发展成为通用的标准建模语言。

3.2.1.2 标准建模语言UML的主要特点

标准建模语言 UML 的主要特点可以归结为三点：

- UML 统一了 Booch、OMT 和 OOSE 等方法中的基本概念；
- UML 吸取了面向对象技术领域中其他流派的长处；
- 在演变过程中 UML 还提出了一些新的概念。

可以认为，UML 是一种先进实用的标准建模语言。当然，UML 也存在一个进化过程，其中某些概念尚待实践来验证，还可能需要提出一些新的概念。

3.2.1.3 为什么在Glass Sculptor中使用标准建模语言UML

Glass Sculptor 是一个模块化的系统，如何把各模块有机的联系起来，并将它们之间的接口和关系用合理的表达出来，提高可读性，使将来的升级和排错更为容易是玻璃雕刻机开发过程中必须面对的问题。

UML 的下列优点是我们最终选择它的主要原因：

- 和其他的建模方法相比，UML 具有表达力更强、更清晰和一致的优点。它不仅可以应用在更广泛的领域，而且也消除了不同方法在表示法和术语上的差异，避免了符号表示和理解上不必要的混乱。
- 标准建模语言 UML 适用于以面向对象来描述任何类型的系统，而且适用于系统开发的不同阶段，从需求规格描述直至系统完成后的测试和维护。
- 便于学习面向对象技术。对于大多数人来说，学习如何使用一种面向对象语言进行编程并不是很难，难的是如何充分利用面向对象语言所提供的优势。UML 技术有助于设计较好的面向对象程序。
- 从整体上把握全局。对于大型项目，常犯的错误是只见树木，不见森林，而 UML 所提供技术有助于得到系统的整体概念。
- 便于与同项目组其他成员、合作伙伴及客户交流。

3.2.1.4 标准建模语言UML的内容

标准建模语言 UML 提供的基本模型图包括：

- 用例图：展示系统外部的各类执行者与系统提供的各种用例之间的关系。
- 类图：展示系统中类的静态结构。
- 对象图：是类图的一种实例化图。
- 包图：是一种分组机制。在 UML1.1 版中，包图不再看作一种独立的模型图。
- 状态图：描述一类对象具有的所有可能的状态及其转移关系。
- 顺序图：展示对象之间的一种动态协作关系。
- 合作图：从另一个角度展示对象之间的动态协作关系。
- 活动图：展示系统中各种活动的执行流程。
- 构件图：展示程序代码的物理结构。
- 配置图：展示软件在硬件环境中（特别是在分布式及网络环境中）的配置关系。

UML 为人们提供了从不同的角度去观察和展示系统的各种特征的一种标准方法。在 UML

中，从任何一个角度对系统所做的抽象都可能需用几种模型图来描述，而这种来自不同角度的模型图最终组成了系统的完整图像。

一般而言，我们可以从以下几种常用的视角来描述一个系统：

- 系统的使用实例：从系统外部的操作者的角度描述系统的功能。
- 系统的逻辑结构：描述系统内部的静态结构和动态行为，即从内部描述如何设计实现系统功能。
- 系统的构成：描述系统由哪些程序构件所构成。
- 系统的并发特性：描述系统的并发性，强调并发系统中存在的各种通信和同步问题。
- 系统的配置：描述的系统的软件和各种硬件设备之间的配置关系。

显然，前两种视角对任何系统的开发是必需的，而后三种视角对于大多数的复杂系统，特别是分布式及并发系统而言，也是十分重要的。

构造多种不同的模型图是为了描述从不同视角观察到的系统特性。UML 提供十种基本的模型图，通过对这些图的综合运用来全面刻画整个系统的全貌。

3.2.2 Glass Sculptor的建模

上一节简单介绍了标准建模语言 UML，本节将进一步研究这一理论在开发玻璃上层控制软件——Glass Sculptor 中的应用。根据玻璃雕刻机的特点，我们将从系统的使用实例、系统的逻辑结构和系统的并发特性三个视角来描述本系统，即采用用例图（use-case diagram）、类图（class diagram）、合作图（collaboration diagram）和活动图（activity diagram）来对其进行描述。

本文中 Glass Sculptor 所使用到的 UML 表示符将在在附录中加以说明。

3.2.2.1 Glass Sculptor的用例图

在进行软件开发时，无论是采用面向对象方法还是传统方法，首先要做的就是了解需求。在实践中，分析典型用例是开发者准确迅速地了解用户要求和相关概念地最常用也是最有效的方法，是用户和开发者一起深入剖析系统功能需求的起点。过去人们只是在不自觉地这样工作， 而且分析用例的方法很不规范，通常只是口头交流，或者在随手找到的一张纸上画几张潦草的示意图，符号简单、含义模糊，只有少数人心领神会，更谈不上建立正式文档。而采用 UML 来建立用例图，则极大地改变了这一现象。

从本质上讲，一个用例是用户与计算机之间为达到某个目的的一次典型交互作用。一系列用例的集合，就组成了整个系统的需求。

根据海泰公司多年研究开发电脑雕刻机的经验和用户反馈来的信息，玻璃雕刻机的控制软件 Glass Sculptor 需要具有以下一些用例：

- 版面文件存取。主要是用户在排版完成以后生成的走刀轨迹数据的保存与打开。
- 中心线输入。用户应该能将在外部程序中画好的中心线输入到系统中。
- 刀具选择。包括砂轮的种类（尖角还是圆角）以及输入砂轮的各种参数。
- 中心线加工属性定义。包括设置最大宽度、最大深度、砂轮的起始角和终止角、起始过渡轮廓线长度、结尾过渡轮廓线长度、轮廓线类型等。
- 中心线几何尺寸编辑。包括中心线的移动、缩放等功能。
- 效果预览。让用户观察根据他们所输入的信息加工出来的图形，便于及时修改参数，使效果达到最佳。包括走刀模拟和生成包络线预览。
- 指令下载。将砂轮运动轨迹的一系列点的数据传送给控制器或其它外设。

结合以上对用例的分析，可画出如图 3.2 所示的用例图。

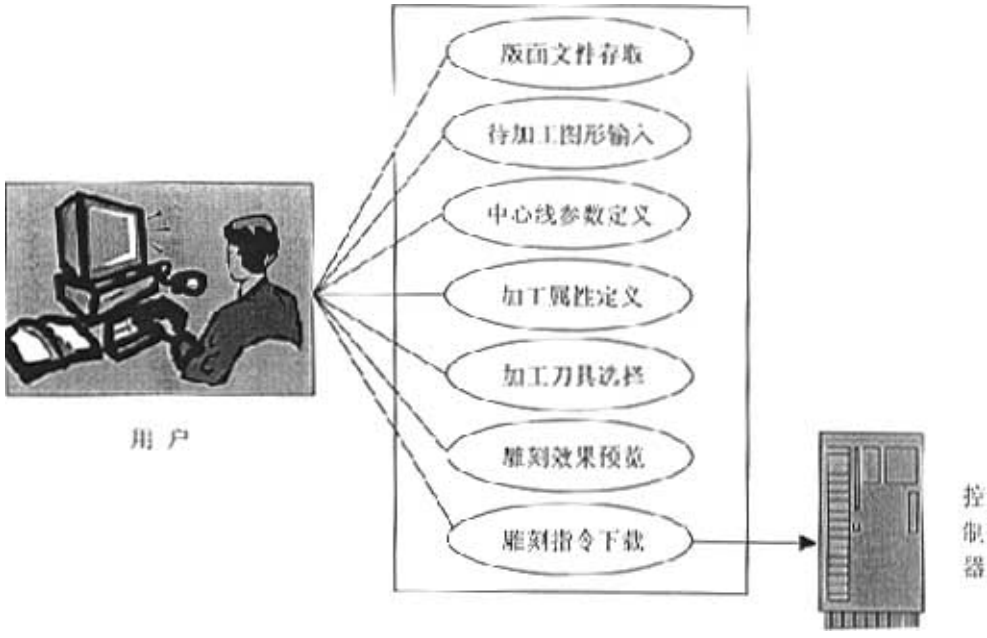


图 3.2　Glass Sculptor 的用例图

Figure 3.2 The Instance Graph of Glass Sculptor

3.2.2.2 Glass Sculptor的类图

类图技术是面向对象方法的核心技术。事实上，几乎任何一种面向对象方法都有与类图类似的技术。类图的应用非常广泛，其基本概念在许多地方都会经常用到，譬如类（class）、属性（attribute）、行为（behavior）或操作（operation）以及关联（association）、聚集（aggregation）和泛化（generalization）等。此外，类图中还定义了其他很多重要的概念，并提供了丰富的表示法，使得类图有着很强的表达能力。

在面向对象的建模技术中，类、对象和它们之间的关系是最基本的建模元素。对于一个想要描述的系统，其类模型、对象模型以及它们之间的关系揭示了系统的结构。建立类模型的过程，实际上是对现实世界的一个抽象过程，它把现实世界中与问题有关的各种现象及其相互之间的各种关系进行适当的抽象和分类描述。

对象是指与应用问题有一定关联的某个事物，更准确地说，是对某个事物地一种抽象描述。对象大多对应于我们真实世界中的某个客观实体，可以是一个事物或者是一个概念。对象总是以这样或那样的方式反映了我们对真实世界的理解。所有的对象都是独立的实体，都有其唯一的标识。对象之间的区别是由它们固有的存在性所决定的，而与它们的特征是否相同无关。

类是对一类具有相同特征的对象的描述。一般地讲，对象的基本特征可以归纳为两类，即对象的属性和行为，一个类描述了此类对象的属性和行为。任何对象都是某个类的实例。

类图描述了系统中的类及其相互之间的各种关系，其本质反映了系统中包含的各种对象的类型以及对象间的各种静态关系。这些类之间的关系主要有两种：

关联：例如，某位顾客可能购买了玻璃雕刻机；

子类型：例如，学生是一类人。

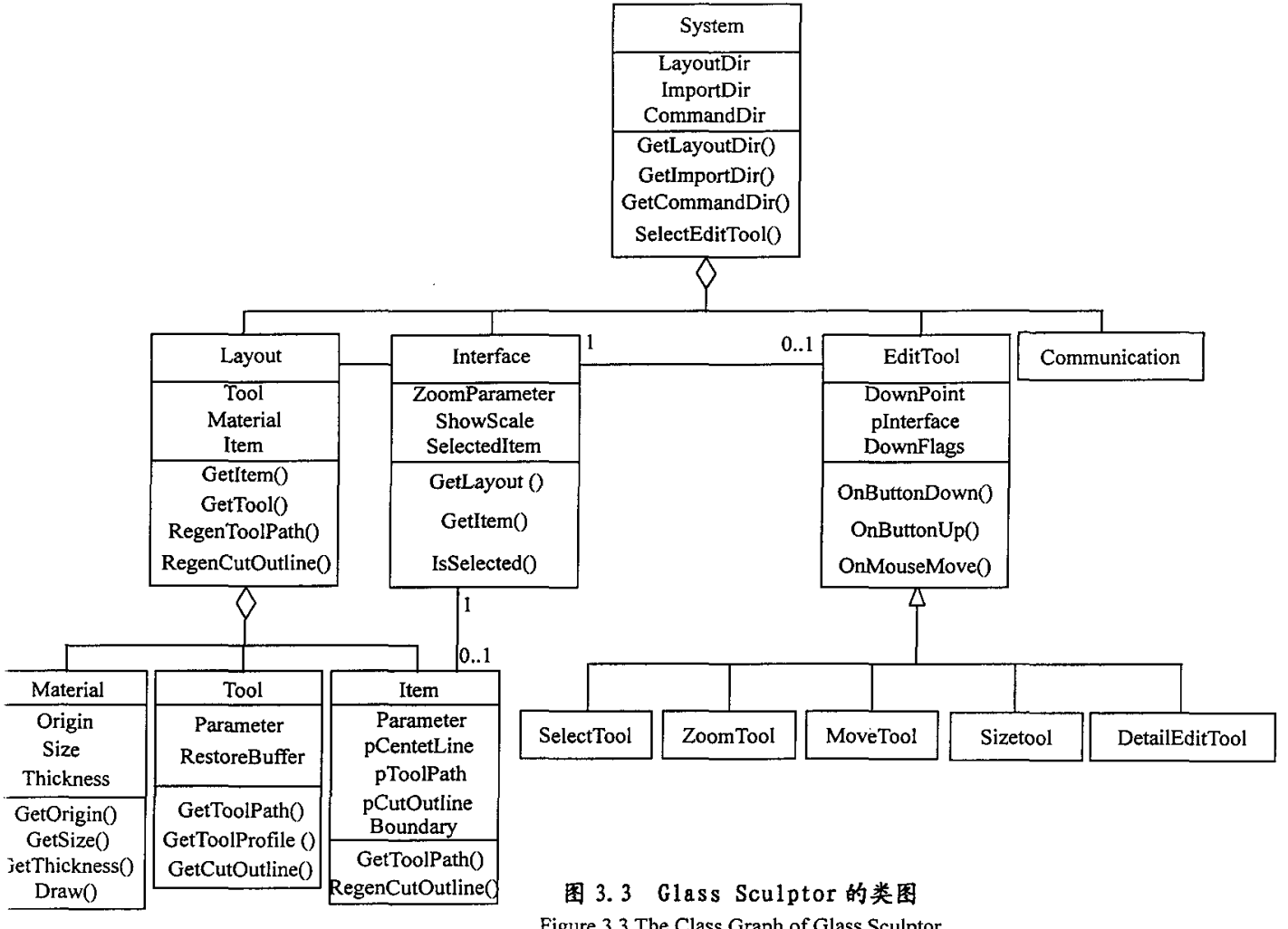


图 3.3 Glass Sculptor 的类图

Figure 3.3 The Class Graph of Glass Sculptor

另外，类图也可以描述类的属性和行为以及对模型中各种成分的约束。不同的面向对象方法对这些概念使用了不同的（甚至相互冲突的）术语，这样容易造成误解。使用 UML 来描述这些概念，有利于对各种术语的同一，从而有利于对问题的理解和交流。

Glass Sculptor 的类图如图 3.3 所示。

System 类作为本系统的代理与操作系统进行交互，同时与软件中的其他类直接或间接关联。

Layout 类负责存储数据，它与 Material 类、Tool 类和 Item 类是聚集关系。Material 类定义了材料的在机器坐标系中的位置、材料大小和厚度；Tool 类定义了砂轮的尺寸；Item 类定义了中心线对象。

Interface 类负责负责与用户的交互，是系统的界面。

SelectTool 类、ZoomTool 类、MoveTool 类、SizeTool 类和 DetailEditTool 类用来编辑 View 类中的对象，它们是由共同的基类 EditTool 继承而来。

3.2.2.3 Glass Sculptor的合作图

合作图是交互图的一种，用来展示对象之间的动态协作关系，侧重于说明那些对象之间有消息传递。对象的名称可表示为对象名：类名，其中对象名和类名中的任何一个都可以省略，但不能同时省略。如果省略对象名，则应保留冒号。

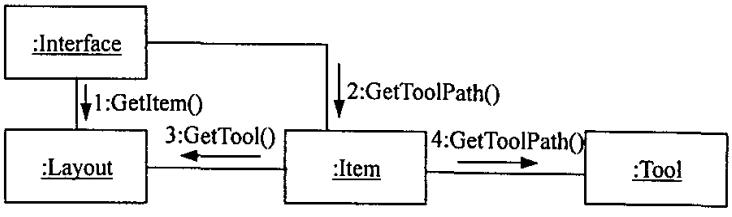


图 3.4 产生走刀轨迹时的合作图

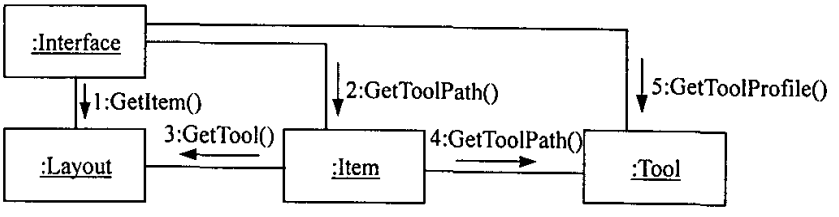


图 3.5 产生切痕轮廓线时的合作图

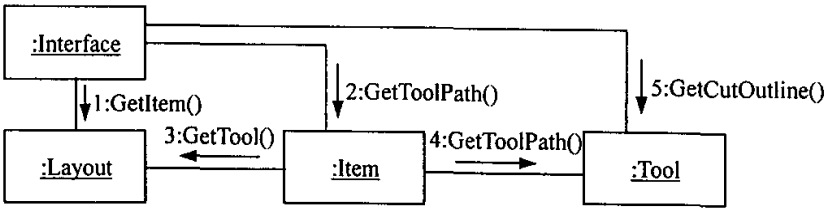


图 3.6 产生包络线时的合作图
Figure 3.6 The Cooperation Graph for Creating Outline

图 3.4 表示了产生走刀轨迹时的对象之间的合作图。

如图所示，要产生走刀轨迹，Interface 给 Layout 发一个 GetItem 的消息，知道是哪

一个Item要产生走刀轨迹，然后Interface向Item发GetToolPath消息，接着Item向Layout发出GetTool的消息，取得Tool的信息，最后Item向Tool发出GetToolPath消息，以取得所求的走刀轨迹。

产生切痕轮廓线和包络线的合作图与图3.4类似，如图3.5，3.6所示。

由图可知，Interfac类是与外界的接口，Layout类负责存储，而实际上所有的计算功能都在Tool类中完成。

3.2.2.4 Glass Sculptor的活动图

活动图来源于Jim Odell的事件图、SDL状态建模技术和Petri网技术，这些技术主要用于描述工作流和并行过程的行为，并且十分有效。

一个活动可以顺序地跟在另一个活动后执行，这是简单的顺序关系。从顺序关系上看，活动图与常用的程序流程图非常相似。活动图描述了需要做的活动以及执行这些活动的顺序。在用活动图表达并发过程时，活动图给予我们选择做事顺序的自由。换句话说，在抽象程度较高的层次上，活动图仅仅列出了那些需要做的事情（即活动）以及那些必须的、不得不遵守的工作顺序，而程序流程图对活动顺序的描述比较确定，这是活动图与程序流程图的根本区别。程序流程图一般用来表示串行过程，而活动图则常常用来表示并行过程。活动是并行的，是说活动的执行次序可以是随意的，执行顺序是不受限制的，交叉进行、同时进行都可以。

Glass Sculptor的活动图如图3.7所示。

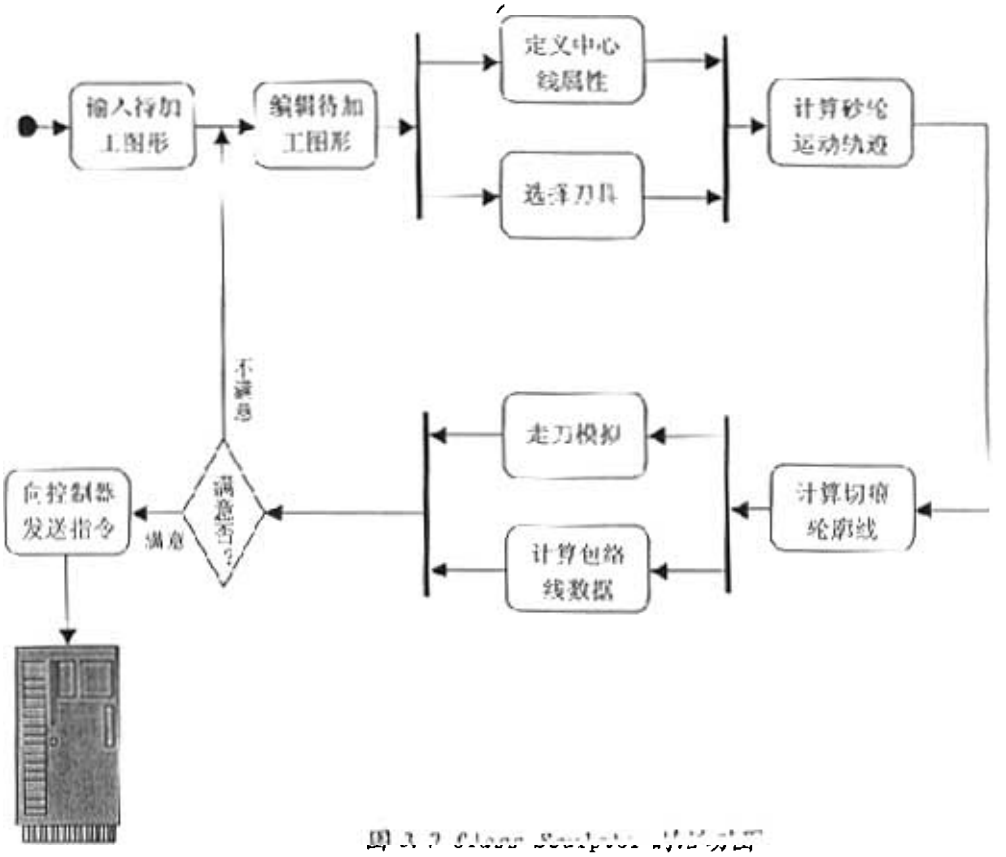


图3.7 Glass Sculptor的活动图

Figure 3.7 The Activity Graph of Glass Sculptor

开始由用户输入中心线，然后用户定义中心线的属性和砂轮的几何特征，根据中心线的属性和砂轮的几何特征，得到砂轮的运动轨迹；再由切痕轮廓线离散化算法得到砂轮每一个位置的切痕轮廓线数据组；根据一系列的切痕轮廓线数据组，可以进行加工模拟，同时也可通过产生包络线算法的求得整个加工过程的所形成的包络线，也就是在玻璃表面加工形成的图形；用户如果对将要加工的图形感到满意，即可将运动轨迹数据组将指令下载给控制器，进行实际加工；如果对图形并不满意，可重复以上过程，直至满意为止。

本节从整体上研究探讨了 Glass Sculptor 的模块结构，至于产生砂轮运动轨迹、切痕轮廓线离散化和生成包络线等算法的具体实现细节将在第五章中加以介绍。

第四章 数据的获取、处理及输出

数据的处理是本课题的重点和难点所在，Glass Sculptor 的最主要的任务就是根据输入的待加工图形计算符合要求的雕刻指令，可以说，本课题的研究都是围绕这个目的进行展开的。本文前两章在整体上为数据的求取所设计的数据结构和模块组成进行了探讨，本章将细化到数据处理算法的内部，进一步详细介绍。

Glass Sculptor 中数据流程如图 4.1 所示，数据从输入到输出，一共经历了 A、B、C、D、E 和 F 六次处理（如图中箭头所示），其中，过程 A 为待加工图形数据的输入，过程 B、C、D 为对原始数据的处理，过程 E 和 F 为数据的输出，本章将按着这个划分对这六个过程逐一进行介绍。

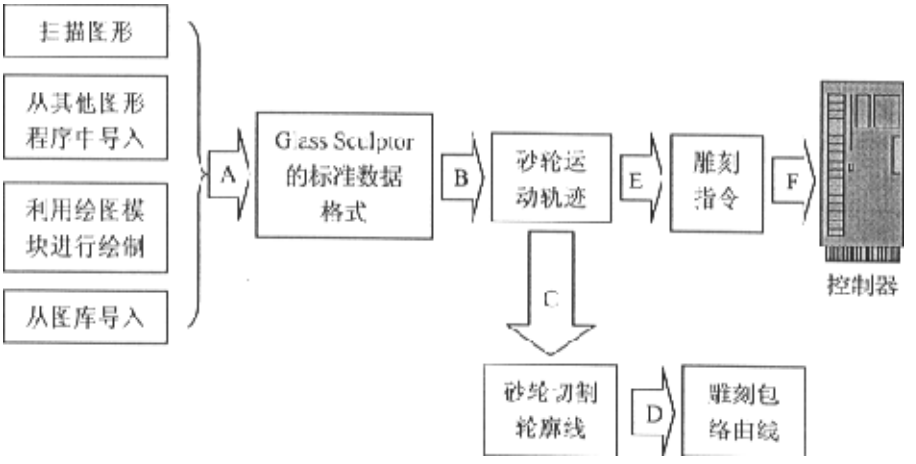


图 4.1 Glass Sculptor 的数据流程

Figure 4.1 Data Flow Chart of Glass Sculptor

4.1 待加工图形的输入

玻璃雕刻机应对多种待加工图形的输入方式加以支持，支持的输入方式越多，对用户而言，玻璃雕刻机的使用就更方便。如图 4.1 所示，常见的获取加工图形的方式有：

- 从其他图形编辑程序中导入；
- 扫描获得的光栅数据；
- 利用绘图模块进行绘制；
- 从图库中导入。

其中，后三种方式在本文第二章介绍数据结构时曾讨论过，这几种方式获得的数据可以比较方便地转换为 Glass Sculptor 的标准数据格式。但从其他图形程序中导入数据并加以转换需要一些专门的处理，可以采取两种方式：1.对这些图形应用程序进行二次开发，如使用 ObjectARX 接口进行 AutoCAD 二次开发，这种方法使用简便，但无形中增加了玻璃雕刻机的成本，而且，对导入其他方式输入的图形而言并不方便；2.将这些应用程序导出保存为通用的文件格式，然后从 Glass Sculptor 中导入这些文件。目前，本课题采用的正是后者。

目前使用最广的图形应用程序为 AutoCAD，它支持以文本文件的形式导出编辑图形，

其中最常见的格式有 DXF 和 HPGL 文件，本节将对这两种文件进行介绍。

4.1.1 从DXF文件导入图形

DXF(Autodesk Drawing eXchange Format)文件是 AutoCAD 中的矢量文件格式，它以 ASCII 码方式存储文件，在表现图形的大小方面十分精确。DXF 文件中以“主要点”的方式表示图形，如对直线记录两个端点坐标，对圆记录圆心和半径，因此 DXF 文件表示的图形可以不失真的还原和放大，是真正意义上的矢量图形格式。DXF 文件可以被许多软件调用或输出，其扩展名为.dxf。

DXF 文件使用标记数据来表示图形，标记数据的意思是指在每个数据元素前都带一个称为组码的整数。组码的值表明了其后数据条目的类型，也指出了数据元素对于给定对象(或记录）类型的含意。通过使用这些配对的组码和数据条目，DXF 文件被组织成区域，这些区域由记录组成，而记录又由依次排列的组码和数据条目组成。在 DXF 文件中，每个组码和每个数据条目都各占一行。

每个区域都是以一个其后跟随着字符串 SECTION 的组码 0 开始，接着是组码 2 和表示区域名称的字符串（例如 HEADER)。每个区域都是由定义它的元素的组码和组值组成。其后跟着字符串 ENDSEC 的组码 0 表示该区域结束。

从小图形中生成 DXF 文件时组码非常有用，用户可以将以后读到关于组码的信息析取出来，从而复原图形。常用的组码及紧跟其后的数据条目的含义如下表：

组码	数据条目
0	表示图元类型的文字字符串（固定）
2	名称（属性标记、块名称等）
6	线型名（固定）
8	图层名（固定）
9	变量名标识符（仅用于 DXF 文件的 HEADER 区域）。
10、20、30	主要点的 X 、Y 和 Z 坐标的值 （主要点：直线或文字图元的起点，圆的圆心等等）
62	颜色代码（固定）
1011、1021、1031	三维世界空间位置的 X 、Y 和 Z 坐标的值

DXF 文件完整的组织结构说明如下：

- HEADER 区域。此区域包含图形的基本信息，它由一个 AutoCAD 数据库版本号和许多系统变量组成。每个参数包括一个变量名及其组值。
- CLASSES 区域。此区域包含有关应用程序定义类的信息，这些类的实例包含在 BLOCKS 区域、ENTITIES 区域和 OBJECTS 区域的数据库中。类定义在类的层次结构中是固定不变的。
- TABLES 区域。此区域包含如下符号表的定义：
  - APPID（应用程序标识表）
  - BLOCK_RECORD（块引用表）
  - DIMSTYLE（标注样式表）
  - LAYER（图层表）
  - LTYPE（线型表）

- STYLE（文本样式表）
- UCS（用户坐标系表）
- VIEW（视图表）
- VPORT（视口配置表）

- BLOCKS 区域。此区域包括块定义和组成图形中每个块引用的图形图元。
- ENTITIES 区域。此区域包含图形中的图形对象（图元），包括块引用（插入图元）。
- OBJECTS 区域。此区域包含图形中的非图形对象，所有那些非符号表记录的、符号表的和非图元的对象都储存在此区域中。在 OBJECTS 区域中的条目样例是包含多线和组的字典。
- THUMBNAILIMAGE 区域。此区域包含图形中的预览图像。该区域为可选。 如果用户使用了 SAVE 和 SAVEAS 命令选择对象选项，输出的 DXF 文件将只包含的 ENTITIES 区域和 EOF 标记，且在 ENTITIES 区域中只包括用户选择输出的对象。如果选择一个插入图元，则在输出文件中不包括对应的块定义。

在了解了 DXF 文件的格式后，即可实现对 DXF 文件的读取。

4.1.2 从HPGL文件导入图形

HPGL 文件格式是 HP 绘图仪支持的格式，能直接驱动绘图仪工作，并完全支持三维线框模型、尺寸、图形块等输出。此外，多种绘图应用程序，如 AutoCAD、UG、ProE 等，都支持将编辑图形转存为 HPGL 格式，HPGL 文件的扩展名为.plt。

与 DXF 文件相比，HPGL 文件更简单， HPGL 不采用“主要点”的方法描述图形，而是将矢量图形离散为连贯的一组折线，所以，HPGL 文件中保存的是折点的信息。以下以记录一条直线图形的 HPGL 文件的全部内容为例，对 HPGL 文件的格式进行分析。

```
IN;SC;PU;PU;SP1;PU;VS6;PU;PA4964,6743;PD;PA4954,3461;PU;
PA0,0;SP;
```

- 开头部分：即“IN;SC;PU;PU;SP1;PU;VS6;PU;”，为文件头部分，记录 HPGL 文件的信息；
- 数据部分：即上文件中的“PA4964,6743;PD;PA4954,3461;PU;”，这是我们所关心的部分，是图形信息保存的地方。在上文件中，记录了两个折点和两个操作，其中：“PA4964,6743”为第一个折点的 x、y 坐标；“PD”为“Pen Down”的缩写，本意是开始绘图，表示折线的开始；“PA4964,3461” 为第二个折点的 x、y 坐标；“PU”是“Pen Up”的缩写，表示折线的结束；
- 结束部分：即文件中的“PA0,0;SP;”

对于直线，HPGL 文件的数据部分比较简单，因为只需两个折点即可，相比而言曲线图形需要的折点非常多，但总的格式是一样的。我们通过分析 HPGL 文件的数据部分，即可得到需要的图形。

由上面对 DXF 和 HPGL 文件的分析可知，HPGL 文件的格式和 Glass Sculptor 的标准数据结构很类似，只需简单的处理就可转换为 Glass Sculptor 可以识别并进行加工的图形数据格式，可以作为数据导入的主要文件格式。同时，从功能扩展的角度考虑，对 DXF 文件也应加以支持。

4.2 数据的计算处理

在第一章的 1.2.4 节对本课题进行了分析，如何从只包含 x，y 两轴坐标的平面待加工图形中心线计算得出包含四轴信息的砂轮运动轨迹，是本课题的主要难点所在。显然，从两维数据到四维数据，必须设计合理的算法来获取额外的信息，并且，为保证生成的砂轮运动轨迹能满足玻璃雕刻的要求，还应该对算法进行优化。

4.2.1 主要算法介绍

本节分别对图 1.1 中所示的过程 B、C、D 的算法进行介绍。其中，过程 D 实现砂轮运动轨迹，过程 C 实现砂轮切割轮廓线，过程 D 实现雕刻效果预览，即包络曲线。

4.2.1.1 实现砂轮运动轨迹的算法

第二章中详细介绍了 Glass Sculptor 的数据结构，其中用来描述砂轮空间姿态的数据结构如下：

```
typedef  struct _POSEF
{
    float x;
    float y;
    float z;
    float Theta;
}POSEF;
```

其中 x、y、z 分别为砂轮 X 坐标、Y 坐标和吃刀深度，Theta 为砂轮与 X 轴的夹角，0<=Theta<π。确定了以上四个参数，即可确定砂轮的空间位置。

描述砂轮运动轨迹的数据结构为：

```
ypecef  struct _PATHF
{
    unsigned short Total;
    POSEF Pose[1];
}PATEF;
```

其中，Total 是沿着一条中心线运动的空间姿态总数；Pose 是一个空间姿态数组，其下标在此仅是做占位之用，数组的实际大小是根据 Total 的值而定的，其大小的计算方法应为 sizeof(PATHF)+(Total-1)*sizeof(POSEF)。

本算法的主要目的是要确定砂轮运动到中心线上任何一点时的空间姿态，根据每一个（x、y）找到一个与之对应的（x、y、z、θ），即得到 POSITIONF，从而组成与之对应的 PATHF，从而实现对砂轮运动轨迹的求取。

本算法的实现可分为以下几个步骤进行。

1. 待加工图形中心线的插点算法

第二章中，我们比较了几种数据结构，最后选取了用折线的形式来描述输入中心线，并在雕刻前，对于由绘图模块绘制或由 DXF 文件导入等方式获得的“纯矢量”形式的中心线数据进行离散化，转换成统一的折线形式的数据结构。然而，折线形式的数据必须统一折点间的距离，否则间距很大的两个折点而言，其中心线上砂轮空间姿态 Z 和 θ 的变化很难精确描述，更重要的是，间距不同将导致砂轮在某两点间雕刻的时间比别的点长，玻璃磨损量更

多，从而雕刻出来的图形深浅宽窄不一致，这对于玻璃雕刻来说从视觉上很容易感觉到，在中心线折点间距过大时进行均匀插点是绝对有必要的。

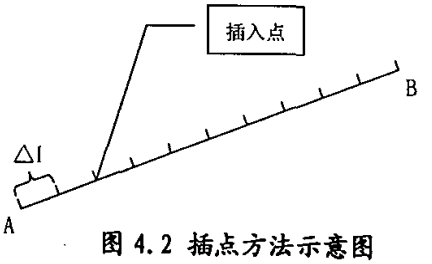


图 4.2 插点方法示意图

Figure4.2 Method of Points Inserting

均匀插点的算法为：如图 4.2 所示，将待加工图形的中心线拓扑拉直，抽象为一条直线 AB，其端点坐标分别为 A（x1，y1），B（x2，y2），设 AB 之间的距离为 L，且中心线上任意两点之间的距离不能超过△l，如图 4.2 所示。

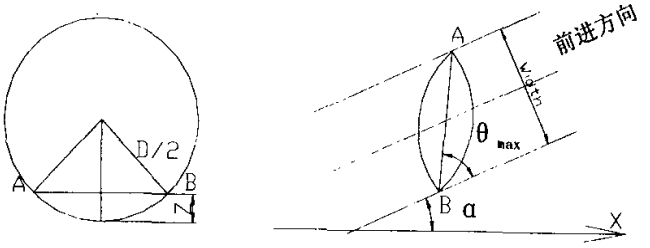
令 $N=\left[\frac{L}{\Delta l}\right]$，N 即为应插入的点的个数。

插入的任一点的坐标计算公式为：

$$x_i = x1 + i\cdot\Delta l\cdot(x2-x1)/L \qquad \text{（式 4—1）}$$

$$y_i = y1 + i\cdot\Delta l\cdot(y2-y1)/L \qquad \text{（式 4—2）}$$

注意，中心线的两个端点一定要包括进去。



图a　　图b

图 a：砂轮的正面视图；图 b：砂轮在 AB 处的截面图

图 4.3 求砂轮最大转角的示意图

Figure4.3 Calculating For The Max Angle of Grinding Wheel

2．砂轮最大转角的确定

如图 4.3（a）所示，当吃刀深度为 z（z<=0）时，

砂轮的前进方向与截面的切点并不一定就是 A、B 两点，但可近似地看作是点 A、B，故

$$Width = \sin\theta_{max}\cdot AB \qquad \text{（式 4—4）}$$

$$AB = 2\sqrt{(D/2)^2-(D/2+z)^2} = 2\sqrt{-Dz-z^2} \qquad \text{（式 4—3）}$$

有：

将（式 4—3）代入，有

$$\theta_{\max} = \arcsin(\frac{width}{2\sqrt{-Dz - z^2}}) \qquad (式\ 4—5)$$

其中，Width 为最大宽度，由用户给定。当宽度达到最大时，砂轮的转角最大，吃刀深度也最大，即此时

$$z = -Depth \qquad (式\ 4—6)$$

由上图可以看出，砂轮与 X 轴之间的夹角应该是 $\theta + \alpha$

3．砂轮 z 轴坐标的计算

用同样的方法，将待加工图形的中心线拓扑拉成一条直线，以每一点与起始点之间的距离为参数来计算。如图 4.4 所示。

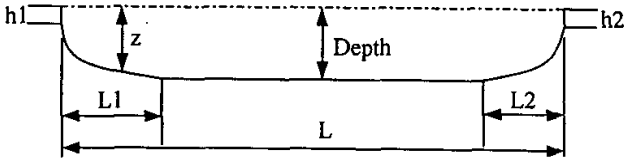


图 4.4　加工深度与拓扑拉直后的中心线示意图

Figure4.4 Calculating For Grinding Wheel's Depth

图中，L 为拉直后的中心线总长度，L1、L2 为开始过渡段和结尾过渡段的长度（在程序中 L1、L2 为过渡段占总长度的百分数），在中间段时深度达到最大值 Depth，h1、 h2 为起始深度和结尾深度。在起始过渡段，深度由 h1 逐渐过渡到 Depth；在结尾过渡段，深度由 Depth 逐渐减小到 h2。为了使加工后的图形达到比较好的效果，过渡段必须平滑，因此不能采用直线，而在各种曲线中，抛物线方程简单，易于计算，而且比较平滑，无疑是最好的选择。因此，我在此对于深度的计算采用的是抛物线。

由图 4.4 可知，h1、h2 是起始深度和结尾深度，L1、L2 的长度由用户给定，L 由计算得出，因此，所求的 $z = f(x, y)$ 必须满足以下约束条件：

- 当 $l = 0$ 时，$z = h_1$；
- 当 $L1 < l < L - L2$ 时，$z = -Depth$；
- 当 $l = L$ 时，$z = h_2$；
- 在 $l = L1$,或 $l = L - L2$ 处，$z = f(x, y)$ 左右两边的切线斜率必须一致。

对于中心线上的任一点，设其到起始点的距离（拉直以后）为 l，结合抛物线的方程和图 4.4，可知：

当 $0 \le l \le L1$ 时，即点处于起始过渡段；

$$z = \frac{Depth + h1}{L1^2}(L1 - l)^2 - Depth \qquad (式\ 4—7)$$

当$L1 < l < L - L2$时，点处于中间段；

$$z = -Depth \qquad (式\ 4—8)$$

当$L - L2 \le l \le L$时，点处于结尾过渡段。

$$z = \frac{Depth + h2}{L2^2}(l - L + L2)^2 - Depth \qquad (式\ 4—9)$$

4. 砂轮转角θ的求取

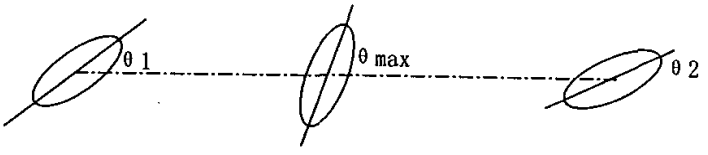


图 4.5　砂轮转角随距离改变而变化的示意图

Figure4.4 Calculating For Grinding Wheel's Angle

由上图可知，θ 1、θ 2 是起始和终止角度，由用户给定，θ max 根据最大宽度和深度计算得出。因此，所求的 $\theta = f(x, y)$ 应满足以下三个约束条件：

- 当$l = 0$时，$\theta = \theta_1$；
- 当$L1 < l < L - L2$时，$\theta = \theta_{max}$；
- 当$l = L$时，$\theta = \theta_2$；
- 在$l = L1, 或 l = L - L2$处，$\theta = f(x, y)$左右两边的切线斜率必须一致。

这里，θ 随距离变化的曲线选择正弦曲线，具体的计算公式如下：

当$0 \le l \le L1$时，即点处于起始过渡段，有：

$$\theta = \theta_1 + (\theta_{max} - \theta_1)\sin(\frac{l}{L1} \cdot \frac{\pi}{2}) \qquad (式\ 4—10)$$

当$L1 < l < L - L2$时，点处于中间段，有：

$$\theta = \theta_{max} \qquad (式\ 4—11)$$

当$L - L2 \le l \le L$时，点处于结尾过渡段，有：

$$\theta = \theta_2 + (\theta_{max} - \theta_2)\sin(\frac{L-l}{L2}\cdot\frac{\pi}{2}) \qquad (式\ 4—12)$$

注意，以上所求的 θ 均为砂轮与中心线之间的夹角，必须为锐角，在计算完以后均需转变成在世界坐标系中的角度。

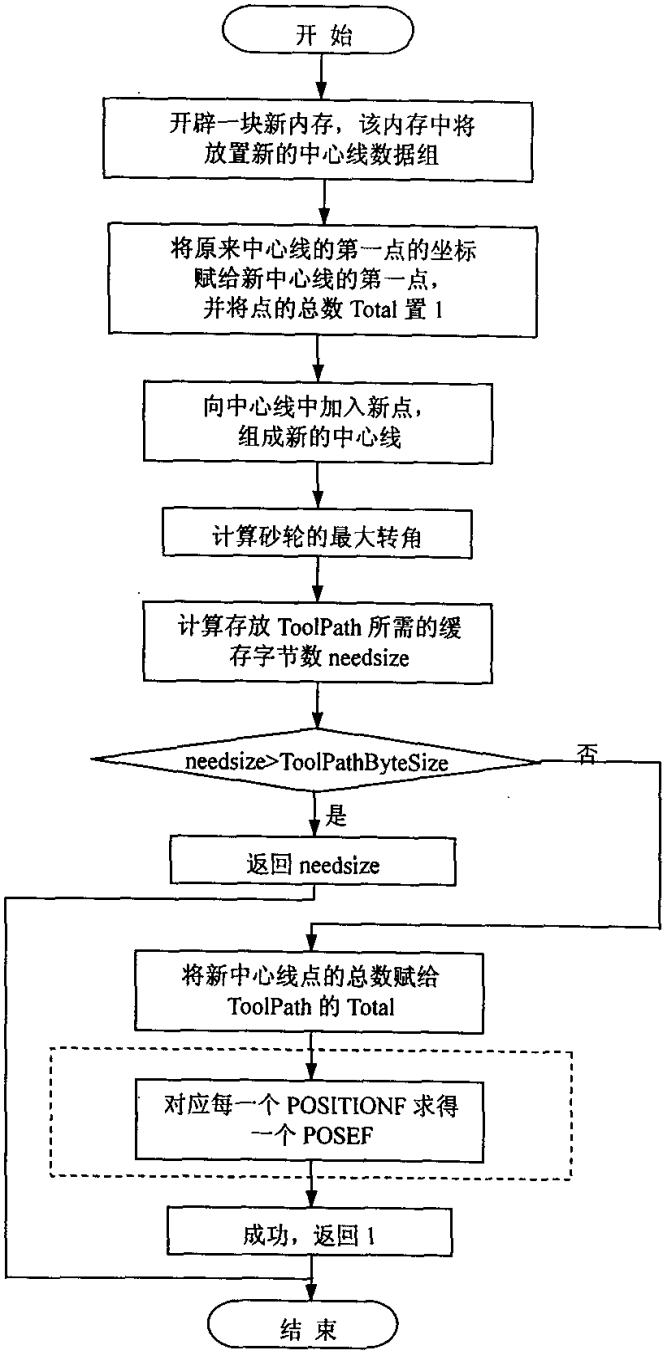


图 4.6　GenerateToolPath 函数的整体流程图

Figure4.6　Flow Chart of GenerateToolPath

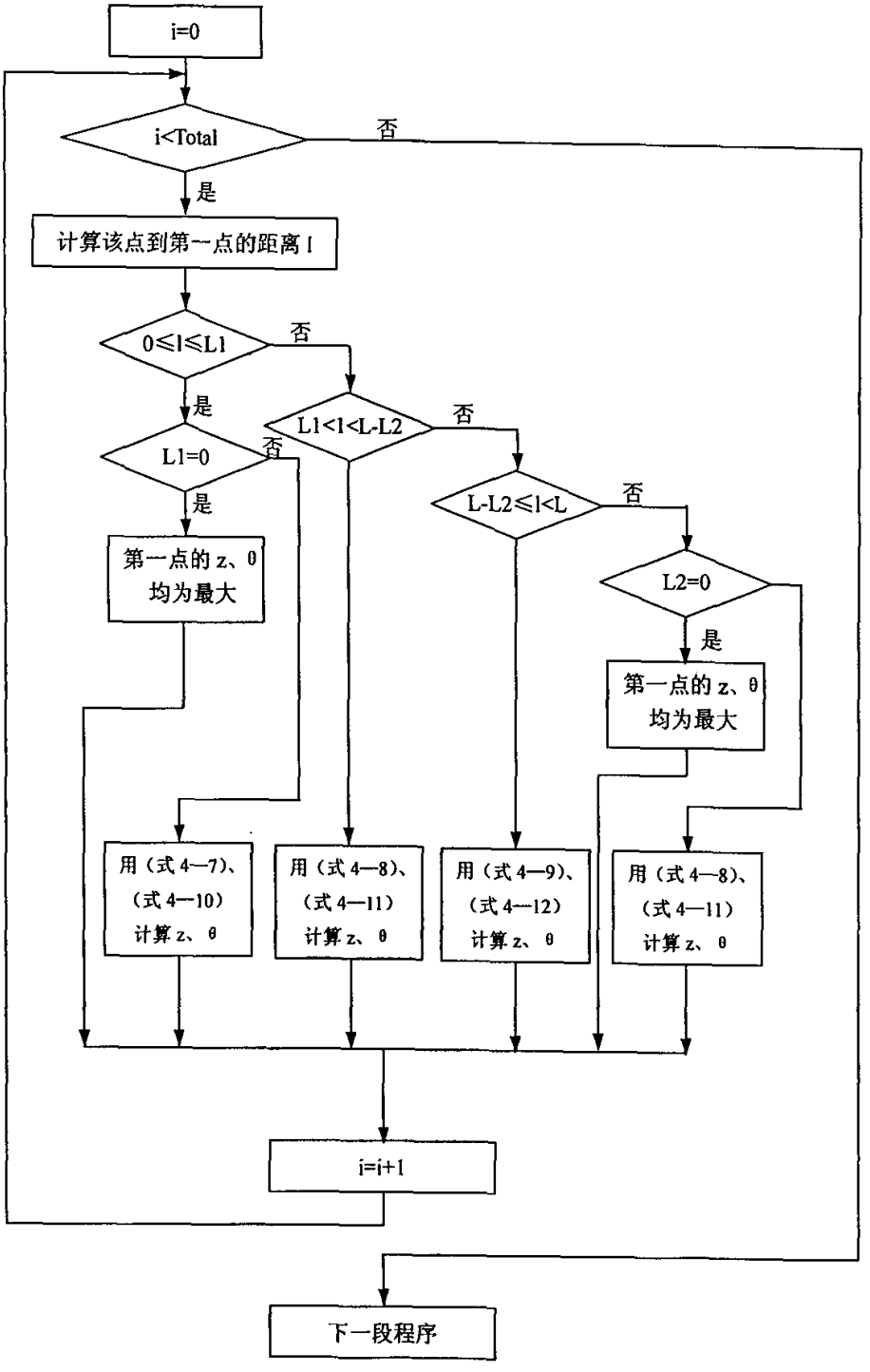


图 4.7 计算 z、θ 的流程图

Figure4.7 Flow Chart For Calculating Z and θ

整个生成砂轮加工路径数据的算法由函数 GenerateToolPath 实现(第三章介绍 DLL 的代码中有对此函数的调用)，该函数的定义如下：

```
unsigned long GenerateToolPath(
                    const Tool *Tool,
                    const CENTERLINEATTR *CenterLineAttr,
                    const LINESF *CenterLine,
                    unsigned long CenterLineByteSize,
                    PATHF *ToolPath,
                    unsigned long ToolPathByteSize)
```

其它的变量在前面均已介绍过，CenterLineByteSize 是中心线数据组缓存的字节尺寸，ToolPathByteSize 是 ToolPath 指向的缓存字节尺寸。

如果函数成功，返回值为 1，如果 ToolPathByteSize 太小，返回值为 ToolPath 指向缓存所需的字节数，如果失败，返回值为 0。

GenerateToolPath 函数的流程图如图 4.6 所示，其中的虚线部分的具体流程见图 4.7。

4.2.1.2　实现砂轮切痕轮廓线的算法

在第二章已经推导了砂轮的几何模型，(式 2—4) 和 (式 2—7) 给出了两种典型砂轮的几何模型的表达式，该表达式所表示的实际上就是砂轮工作时的切割轮廓线。该轮廓线是一个代数方程，为了便于计算，应将它离散处理用折线来逼近，如果间距足够小，就可以很好地描述出切痕轮廓线。表示出了该轮廓线以后，才可以完成走刀模拟和预览功能。

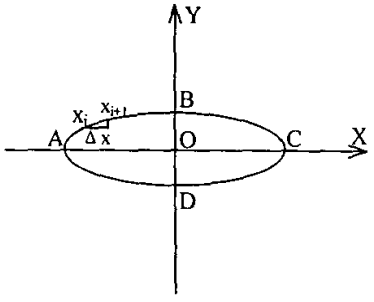


图 4.8 刀具坐标系中的切痕轮廓线

Figure4.8 Grinding Wheel's Profile

由于切割轮廓线是轴对称图形，因此利用这一特性，可以极大地简化计算量。只需计算出一个象限中的点，然后依次将沿 X、Y 对称即可。如图 4.8 所示，从 A 点开始，以后每一点的 X 坐标上都加上 Δx，即 xi+1=xi+Δx，y 由 (式 4—6) 和 (式 4—9) 来计算，一直到 B 点。然后将所得到的点由 Y 轴对称过去，再将所得到的点由 X 轴对称，即得到所需要的离散点。值得注意的是，以上所计算的点均是在刀具坐标系中的坐标值，为了方便以后的计算，这里并不将它们转化成世界坐标。

以上所描述的功能由函数 GenerateToolProfile 来实现，该函数的定义如下：

```
unsigned long GenerateToolProfile(
                    const Tool *Tool,
                    const POSEF *Pose,
                    LINESF *ToolProfile,
```

unsigned long ToolProfileByteSize)

其中，Pose 指针指向一个 POSEF 结构，即图 4.8 中的坐标原点；ToolProfile 指针指向用来保存函数生成的砂轮切割轮廓线数据的缓存，ToolProfileByteSize 是 ToolProfile 指向的缓存字节尺寸。如果函数成功，则返回值为 1，如果 ToolProfileByteSize 太小，返回值为 ToolProfile 指向缓存所需的字节数，如果失败，则返回值为 0。

其具体的流程图如图 4.9 所示。

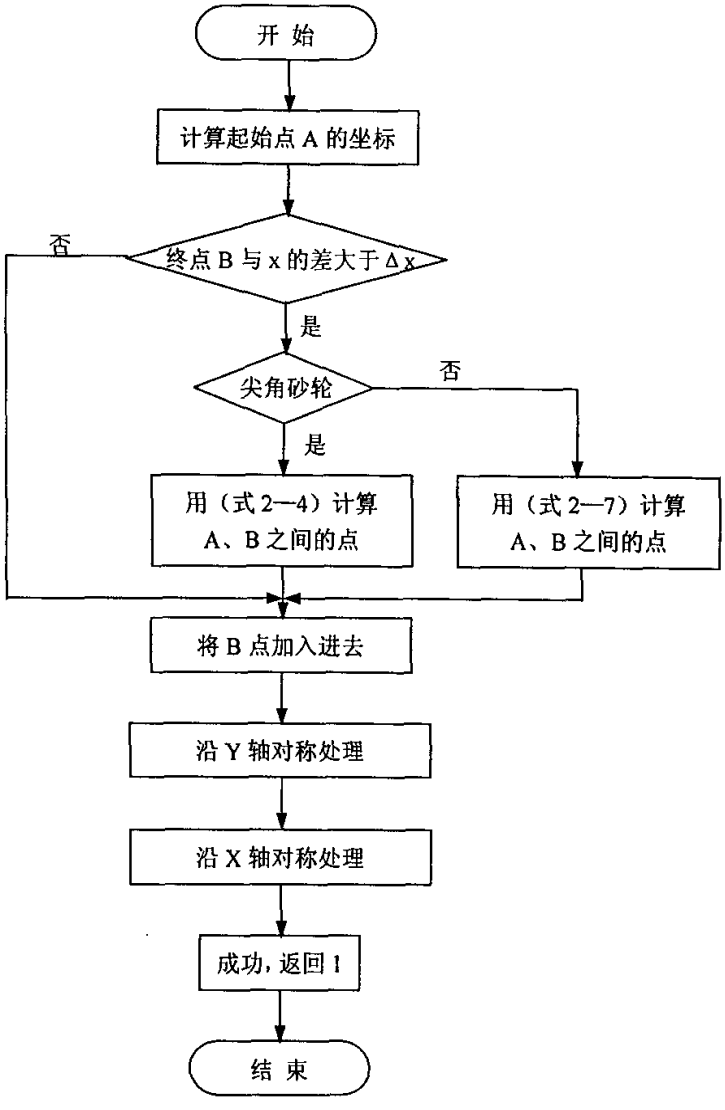


图 4.9 函数 GenerateToolProfile 的流程图

Figure4.9 Flow Chart of GenerateToolProfile

4.2.1.3 实现雕刻预览（包络曲线）的算法

预览是让用户与计算机进行交互的一个非常重要的功能。通过预览，用户可以及时地调整输入图形和雕刻参数，使整个版面的搭配更加和谐，从而提高成品率，节省加工成本。要

达到预览的功能，就必须求出砂轮在沿着中心线运动，且转角和深度在不停地变化时在玻璃表面产生的切痕轮廓线的包络线。

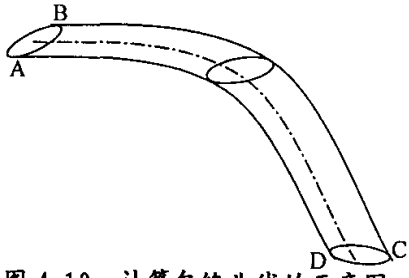


图 4.10　计算包络曲线的示意图

Figure 4.10 Calculating Of Outline

如图 4.10，包络线由四段曲线组成，AB 段（砂轮在起始点形成的切割轮廓线上顺时针由最低点到最高点的一段弧），BC 段（一系列切割轮廓线上的最高点的连线），CD 段（砂轮在终止点形成的切割轮廓线上顺时针由最高点到最低点的一段弧），DA 段（一系列切割轮廓线上的最低点的连线）。在第二章中已经讨论过 BC 段和 DA 段的求法，由于产生的包络线仅仅是给用户观察效果，并不需要很高的精度，因此在本软件的编制中采用离散法来求解 BC 和 DA。

应该注意，所求得的一系列的包络线的点坐标都是在运动坐标系中得到的，因此，还必须将这些运动坐标系中的点坐标转化为世界坐标系中的点坐标，参照 2.3.1 中转换矩阵：

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & x_0 \\ \sin\alpha & \cos\alpha & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \qquad \text{（式 4—13）}$$

其中，α 为前进方向与 X 轴的夹角，x0、y0 为砂轮在该点的坐标。在这里一共牵涉到三种坐标系，刀具坐标系、运动坐标系和世界坐标系，前两种均为相对坐标，世界坐标系为绝对坐标。

在确定包络线的起止部分，即 AB、CD 段时，应注意具体取轮廓线上的哪一段：

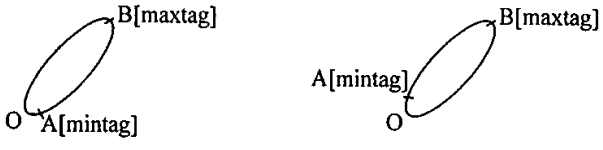


图 4.11　砂轮在起始点处 AB 段弧的确定

Figure4.11 Calculating For the Arc Between A and B

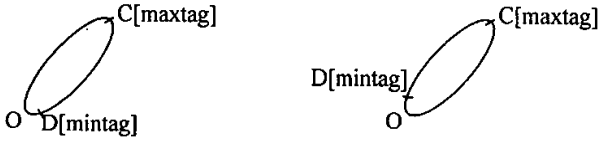


图 4.12　砂轮在终止点处 CD 段弧的确定

Figure4.12 Calculating For the Arc Between C and D

在离散化砂轮切痕轮廓线时，都是在刀具坐标系中以轮廓线与 X 轴的负半轴的交点为起点，顺时针方向进行离散。如图 4.11 所示，0 点为起点，数组下标为 0。在运动坐标系中求得最高点和最低点分别为 A、B，下标为 mintag 和 maxtag。如果 mintag >maxtag，则起始弧应该是（a）中的 AOB 段；如果 mintag<maxtag，则起始弧应该是（b）中的 AB 段。

砂轮在终止点的终止弧 CD 与求取起始弧 AB 段类似，如图 4.12 所示。如果 mintag >maxtag，则终止弧应该为图（a）中的 CD 段；如果 mintag<maxtag，则终止弧应该是图（b）中的 COD 段。

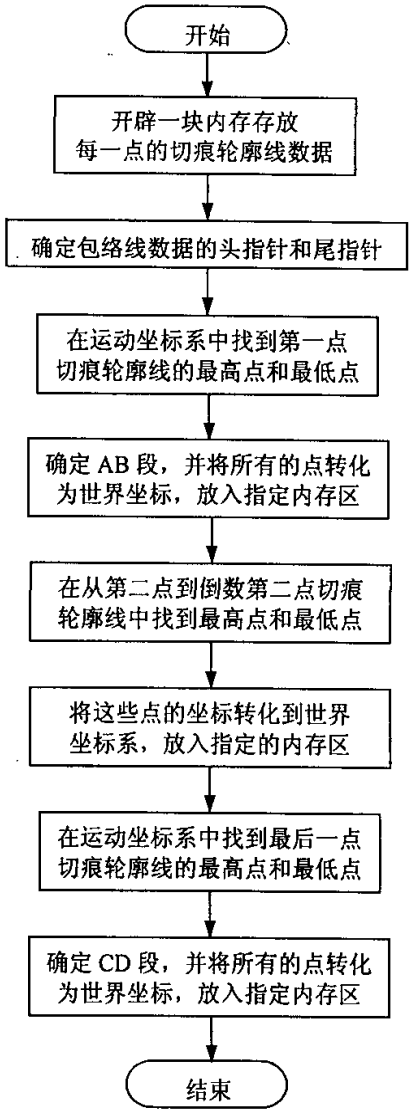


图 4.13 函数 GenerateCutLine 的流程图

Figure4.3 Flow Chart of GenerateCutLine

整个算法由函数 GenerateCutOutline 来实现，其定义如下：

```
unsigned long GenerateCutOutline(
                                const Tool *Tool,
                                const PATH *ToolPath,
```

```
unsigned long ToolPathByteSize,
LINESF *CutOutline,
unsigned long CutOutlineByteSize)
```

ToolPath 指向砂轮加工路径数据组缓存，由一个 PATHF 和随后的一个或多个 POSEF 组成；ToolPathByteSize 为砂轮加工路径数据组缓存的字节尺寸；CutOutline 指向用来保存函数生成的砂轮切痕包络线数据组，由一个 LINESF 和随后的一个或多个 POSITIONF 组成；CutOutlineByteSize 为 CutOutline 指向的缓存字节数。如果函数成功，返回值为 1；如果 CutOutlineByteSize 太小，返回值为 CutOutline 指向的缓存所需的字节数，如果失败，则返回值为 0。

该函数的流程图见图 4.13。

4.2.2 重要优化

通过上一节所介绍的算法可以实现图 4.1 中的过程 B、C、D，但是，为了使雕刻效果更精美，雕刻效率更高，还应采取一些其他的优化措施，本节将对最重要的两个算法进行详细介绍。

4.2.2.1 圆弧拟和优化

在 Glass Sculptor 中，待加工图形的中心线以一组连贯的折线的方式保存，如图 4.14 所示，折线 ABE 就是一条加工中心线的一部分。有些时候，尤其是当待加工图形是扫描获得的光栅图形时，折线间的夹角（如图中，φ 为从 DB 到 BE 之间的夹角，有正负之分）偏大，若不加优化而直接操纵砂轮进行雕刻，砂轮将在 B 点突然转动 φ 度角，必然会雕刻出不连贯的图形，严重时甚至会使玻璃破损。

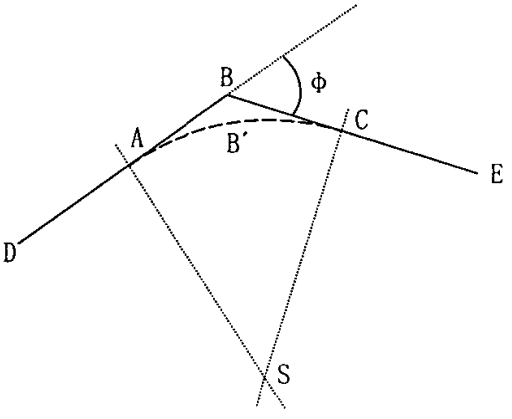


图 4.14 圆弧拟和优化示意图

Figure 4.14 Optimized By Inserting Arc

如果对上述加工图形进行拟和优化，使砂轮沿着直线 DA->圆弧 AB′C->直线 CB 进行雕刻，将使砂轮沿着圆弧 AB′C 逐步均匀平稳地转过 φ 角，雕刻出一条光滑的加工图形。

计算过渡圆弧 AB′C 可以有很多种方法，但必须保证一个前提，即最终计算结果中圆弧 AB′C 上每一点的坐标必须具有相同等级的计算误差，如果圆弧上的点从 A 到 C 误差逐渐放大，将使得过渡圆弧在 A 点与直线 DA 相切，却无法在 C 点平稳过渡到直线 CE 上。因此，不能基于圆弧上前一点的坐标来计算下一点的坐标。

Glass Sculptor 中圆弧拟和优化算法思路如下：

1. 判断是否进行圆弧优化。判断的依据为 φ 角的大小，分三种情况：
   - $|\phi| \leq 1.5°$ 时，无需优化；
   - $1.5° < |\phi| \leq 10°$，进行圆弧拟和优化；
   - $10° < |\phi|$，转角过大，超出圆弧优化的限度，应提刀处理。
2. 选取圆弧插入点 A、C，判断标准为从（不难看出，AB=BC）：

- max{DB, BE}≤4mm，则 AB=BC=Min{DB, BE}/2，即取较短边的中点，有（假设 BD 为较短边）：

$$\begin{cases} x_A = \dfrac{x_B - x_D}{2} \\ y_A = \dfrac{y_B - y_D}{2} \\ x_C = x_B - \dfrac{(x_B - x_E)\times\sqrt{(y_B - y_D)^2 + (x_B - x_D)^2}}{2\times\sqrt{(y_B - y_E)^2 + (x_B - x_E)^2}} \\ y_A = y_B - \dfrac{(y_B - y_E)\times\sqrt{(y_B - y_D)^2 + (x_B - x_D)^2}}{2\times\sqrt{(y_B - y_E)^2 + (x_B - x_E)^2}} \end{cases} \quad \text{式（4—14）}$$

- min{DB, BE}>4mm，则 AB=BC=2mm，即有：

$$\begin{cases} x_A = x_B - \dfrac{2\times(x_B - x_D)}{\sqrt{(y_B - y_D)^2 + (x_B - x_D)^2}} \\ y_A = y_B - \dfrac{2\times(y_B - y_D)}{\sqrt{(y_B - y_D)^2 + (x_B - x_D)^2}} \\ x_C = x_B - \dfrac{2\times(x_B - x_E)}{\sqrt{(y_B - y_E)^2 + (x_B - x_E)^2}} \\ y_A = y_B - \dfrac{2\times(y_B - y_E)}{\sqrt{(y_B - y_E)^2 + (x_B - x_E)^2}} \end{cases} \quad \text{式（4—15）}$$

3．计算过渡圆弧 AB′C 的圆心 S 的坐标：

由于 AS⊥DB，CS⊥BE，故有：

$$(y_B - y_D)\times(y_S - y_A) = -(x_B - x_D)\times(x_S - x_A) \quad \text{式（4—14）}$$

$$(y_E - y_B)\times(y_S - y_C) = -(x_E - x_B)\times(x_S - x_C) \quad \text{式（4—15）}$$

联立上述方程，可得 S 点坐标（x_S，y_S）：

$$\begin{cases} x_S = \dfrac{(y_B - y_D)\times((y_A - y_C)\times(y_E - y_B) + x_C\times(x_E - x_B)) - x_A\times(y_E - y_B)\times(x_B - x_D)}{(y_E - y_B)\times(x_B - x_D) - (y_B - y_D)\times(x_E - x_B)} \\ y_S = \dfrac{(x_B - x_D)\times((x_A - x_C)\times(x_E - x_B) + y_C\times(y_E - y_B)) - y_A\times(x_E - x_B)\times(y_B - y_D)}{(x_E - x_B)\times(y_B - y_D) - (x_B - x_D)\times(y_E - y_B)} \end{cases}$$

式（4—16）

其中，上式分母不会为 0，否则，D、B、E 三点同线，不满足 $1.5°<|\phi|\leq 10°$ 的条件。

因此，可以计算出半径 R_S：

$$R_S = \sqrt{(y_S - y_A)^2 + (x_S - x_A)^2} \quad \text{式（4—17）}$$

4．计算过渡圆弧 AB′C 上点的坐标：

令 N=[L/步长]（取整函数），则将圆弧 AB′C 均匀地分成 N 部分，即取 N-1 个插补点，则

以求得第 i 个点的坐标为（i=1, 2, ……, N-1）：

$$\begin{cases} x_i = x_S + R_S \times \cos(\psi + \dfrac{\phi}{i}) \\ y_i = y_S + R_S \times \sin(\psi + \dfrac{\phi}{i}) \end{cases} \qquad \text{式（4－18）}$$

其中，ψ 为 x 轴正向到有向线段 SA 的夹角，有：

$$\begin{cases} \text{当} x_A < x_S \text{时，有} \psi = \arctan(\dfrac{y_A - y_S}{x_A - x_S}) \\ \text{当} x_A > x_S \text{时，有} \psi = 180 + \arctan(\dfrac{y_A - y_S}{x_A - x_S}) \\ \text{当} x_A = x_S \text{时，有} \psi = 90 \end{cases} \qquad \text{式（4－19）}$$

5. 将 A 点、上述 N-1 个过渡点和 C 点按顺序插入中心线数据中，并将中心线数据中 B 点删除。

图 4.15 中的 DataBrower 为本课题的一个辅助程序，可以用来查看上位程序发送给控制器的雕刻指令，其中 a)图为未经过圆弧拟和优化处理的雕刻指令，b)图为经过优化处理的雕刻指令，输入的待加工图形为同一个圆，很明显，b)图中的数据明显优于 a)图中的数据。

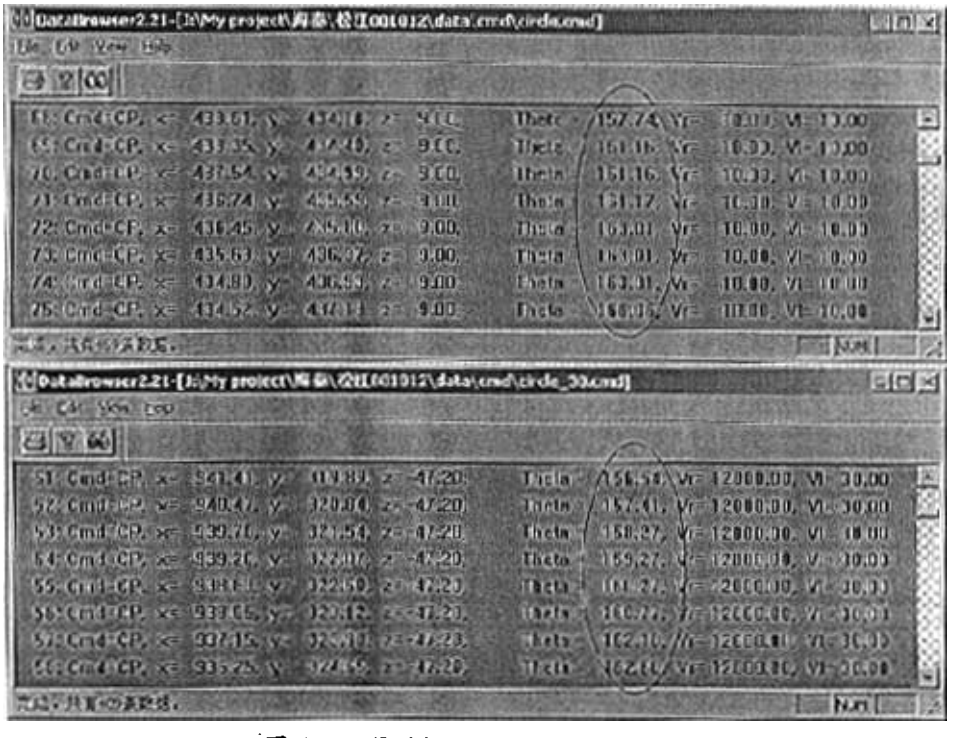


图 4.15 圆弧拟和优化的效果比较

Figure4.15 The Compare of Optimized By Inserting Arc

4.2.2.2 最优加工路径的选取

玻璃雕刻机的雕刻顺序为待加工图形中心线的绘制顺序，但大多数时候这样的顺序很

不合理。试想，如果砂轮先加工完左上角的一条中心线后，为重新调整开始角度而自转 120 度，然后移动到右下角加工另一条中心线，这样的加工顺序显然不理想，因此出于加工效率的考虑，应对加工顺序进行优化选取。

最优加工路径问题与数据结构中图的最小生成树计算有很一定的相似性，可以参考其中的 Prim 算法（详见参考文献 9 第 187 页）来设计本课题的最优加工路径优化算法。所不同的是，最优加工路径求取的是最小遍历链表，而非最小生成树；对于玻璃雕刻机而言，不仅要考虑两节点之间的距离，还要对节点间的转动角

度加以计算，因为在加工一个线条之前，既要使砂轮移动到线条的起始点位置，又要使砂轮转动到该加工线条的起始点的转角。

相邻两条加工图形之间的砂轮进给时间为：

$$T_{ij}=\max\{\frac{dis_{ij}}{v},\frac{ang_{ij}}{w}\}$$

式中，dis_{ij} 表示节点 i 与节点 j 之间距离，ang_{ij} 表示节点 i 与节点 j 之间转角，v 砂轮的进给平移速度，w 为砂轮 θ 角的进给转速。

最优加工路径算法的实现思路为：先选取与砂轮零点之间砂轮进给时间最小的节点 a，再以节点 a 所在中心线的另一节点 b 为参考点，找下一个进给时间最小的节点，周而复始，直至所有的中心线都遍历为止。这样找到节点的顺序就是所求的最优加工路径的节点顺序。

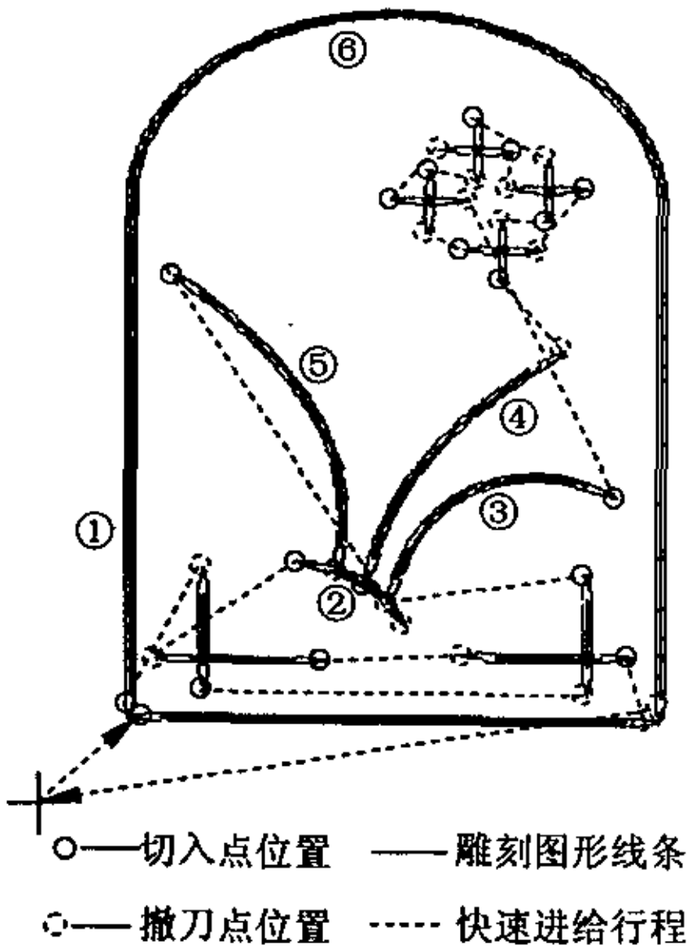


图 4.16 最优加工路径的求取

Fig.4.16 The Best Grooving Sequence

对于图中实线所示的雕刻路径，其优化后的快速进给行程如图中的虚线所示，图中的实线小圆圈表示砂轮切入玻璃时的位置，虚线小圆圈表示砂轮离开玻璃时的位置。

4.3 雕刻指令的发送

在计算求得砂轮运动轨迹后，还必须将计算结果转换为雕刻指令，并通过上位计算机的串行口与控制器通信，达到对玻璃雕刻的最终实现。

4.3.1 雕刻指令的数据格式

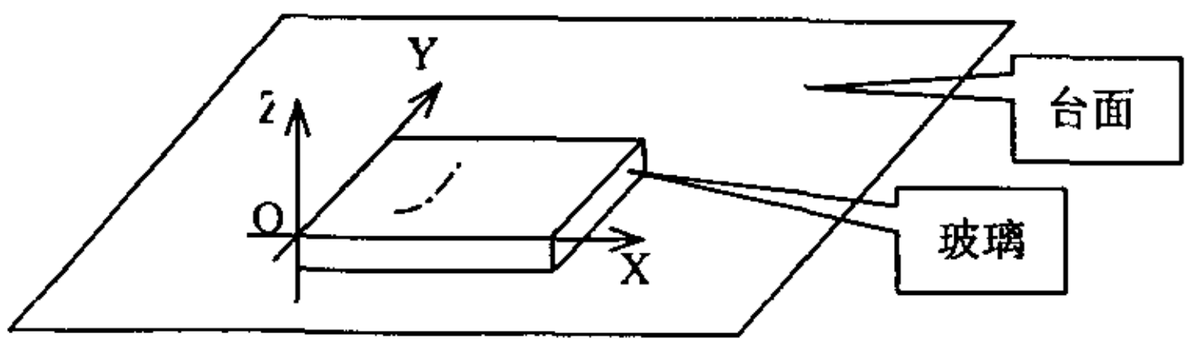


图 4.17　工件坐标系的定义

Figure 4.17 The Definition of Workpiece's Coordinates

在定义通信的数据格式之前，首先定义工件坐标系。如图 4.17 所示，以工件的左下角为坐标原点，建立工件坐标系，坐标系的定义符合右手螺旋法则。

下图是通信数据格式，作为上位程序和控制器共同遵循的标准：

X	Y	Z	θ	R	V_R	V

图 4.18　通信时的数据格式

其中，X、Y、Z、θ 分别为砂轮在工件坐标系中的位置参数，θ 以逆时针方向为正，以角度制表示，且 θ 的范围为 $-180° < \theta \leq 180°$。R 为砂轮的半径，VR 为砂轮自转的速度，V 为砂轮加工的线速度。所有数据都以 float 型变量存储，长度单位均为 mm，时间单位均为 s。

4.3.2 串口通信的方法研究

在工业应用中，串口是常用的计算机与外部串行设备之间的数据传输通道，由于串行通信方便易行，故应用非常广泛。

与以往 DOS 下串行通信程序不同的是，Windows 不提倡应用程序直接控制硬件，而是通过 Windows 操作系统提供的设备驱动程序来进行数据传递。串行口在 Win 32 中是作为文件来进行处理的，而不是直接对端口进行操作，对于串行通信，Win 32 提供了相应的文件 I/O 函数与通信函数，通过了解这些函数的使用，可以编制出符合不同需要的通信程序。与通信设备相关的结构有 COMMCONFIG ，COMMPROP，COMMTIMEOUTS，COMSTAT，DCB，MODEMDEVCAPS，MODEMSETTINGS 共 7 个，与通信有关的 Windows API 函数共有 26 个，详细说明可参考 MSDN 帮助文件。下面结合实例，对实现串行通信最常用的两种方法进行介绍。

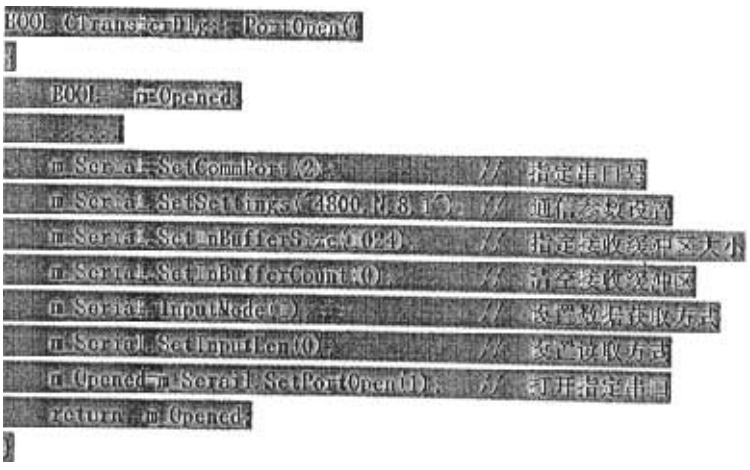
4.3.2.1 使用串行通信控件MSComm

通信控件 MSComm 在对话框中应用，它封装了一些内部操作，因而使用简单直观，只需要关心控件提供的对 Windows 通讯驱动程序的 API 函数的接口。换句话说，只需要设置和监视 MSComm 控件的属性和事件就可以实现串行通信

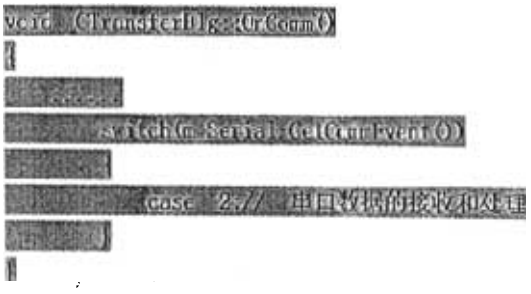
在 ClassWizard 中为新创建的通信控件定义成员对象（CMSComm m_Serial），通过该对象便可以对串口属性进行设置，MSComm 控件共有 27 个属性，这里只介绍其中几个常用属性：

- CommPort 设置并返回通讯端口号，缺省为 COM1。
- Settings 以字符串的形式设置并返回波特率、奇偶校验、数据位、停止位。
- PortOpen 设置并返回通讯端口的状态，也可以打开和关闭端口。
- Input 从接收缓冲区返回和删除字符。
- Output 向发送缓冲区写一个字符串。
- InputLen 设置每次 Input 读入的字符个数，缺省值为 0，表明读取接收缓冲 区中的全部内容。
- InBufferCount 返回接收缓冲区中已接收到的字符数，将其置 0 可以清除接收缓冲区。
- InputMode 定义 Input 属性获取数据的方式（为 0：文本方式；为 1：二进制方式）。
- RThreshold 和 SThreshold 属性，表示在 OnComm 事件发生之前，接收缓冲区或发送缓冲区中可以接收的字符数。

以下是通过设置控件属性对串口进行初始化的实例：



打开所需串口后，需要考虑串口通信的时机。在接收或发送数据过程中，可能需要监视并响应一些事件和错误，所以事件驱动是处理串行端口交互作用的一种非常有效的方法。使用 OnComm 事件和 CommEvent 属性捕捉并检查通讯事件和错误的值。发生通讯事件或错误时，将触发 OnComm 事件，CommEvent 属性的值将被改变，应用程序检查 CommEvent 属性值并作出相应的反应。在程序中用 ClassWizard 为 CMSComm 控件添加 OnComm 消息处理函数：



4.3.2.2 直接使用API函数设计串口通信类

控件简单易用，但由于必须拿到对话框中使用，在一些有特殊需要的应用场合，使用控件就显得捉襟见肘了。可以按不同需要定制灵活的串口通信类。

该通信类 CTransferComm 的基类为 CObject，建立步骤如下：

1. 打开串口，获取串口资源句柄

通信程序从 CreateFile 处指定串口设备及相关的操作属性。再返回一个句柄，该句柄将被用于后续的通信操作，并贯穿整个通信过程。CreateFile()函数中有几个值得注意的参数设置：串口共享方式应设为 0，串口为不可共享设备；创建方式必须为 OPEN_EXISTING，即打开已有的串口。对于 *dwFlagAndAttribute* 参数，对串口有意义的值是 FILE_FLAG_OVERLAPPED，该标志表明串口采用异步通信模式，可进行重叠操作；若值为 NULL，则为同步通信方式，在同步方式下，应用程序将始终控制程序流，直到程序结束，若遭遇通信故障等因素，将导致应用程序的永久等待，所以一般多采用异步通信。

2. 串口设置

串口打开后，其属性被设置为默认值，根据具体需要，通过调用 GetCommState(hComm,&dcb)读取当前串口设备控制块 DCB (Device Control Block) 设置，修改后通过 SetCommState(hComm,&dcb)将其写入。再需注意异步读写的超时控制设置，通

过 COMMTIMEOUTS 结构设置超时，调用 SetCommTimeouts(hComm,&timeouts)将结果写入。以下是串口初始化成员函数：

```
BOOL CTransferComm::Oper( )
{
    DCB dcb;
    m_hIDComDev=CreateFile( "COM2",
            GENERIC_READ | GENERIC_WRITE,
            0,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_
            NORMAL|FILE_FLAG_OVERLAPPED, NULL );// 打开串口，异步操作
    if( m_hIDComDev == NULL )
        return( FALSE );
    dcb.DCBlength = sizeof( DCB );
    GetCommState( m_hIDComDev, &dcb );// 获得端口默认设置
    dcb.BaudRate=CBR_4800;
    dcb.ByteSize=8;
    dcb.Parity= NOPARITY;
    dcb.StopBits=(BYTE) ONESTOPBIT;
    ......
}
```

3. 串口读写操作

主要运用 ReadFile()与 WriteFile()两个 API 函数，若为异步通信方式，两函数中最后一个参数为指向 OVERLAPPED 结构的非空指针，在读写函数返回值为 FALSE 的情况下，调用 GetLastError()函数，返回值为 ERROR_IO_PENDING，表明 I/O 操作悬挂，即操作转入后台继续执行。此时，可以用 WaitForSingleObject()来等待结束信号并设置最长等待时间，代码如下：

```
BOOL bReadStatus;
bReadStatus=ReadFile(m_hIDComDev, buffer, dwBytesRead,
                     &dwBytesRead,&m_OverlappedRead );
if( bReadStatus)
{
    if(GetLastError()==ERROR_IO_PENDING)
    {
        WaitForSingleObject(m_OverlappedRead.hEvent, 1000);
        return ((int)dwBytesRead);
    }
    return(0);
}
return ((int)dwBytesRead);
```

定义全局变量 m_Serial 作为新建通信类 CTransferComm 的对象，通过调用类的成员函数即可实现所需串行通信功能。与方法一相比，方法二赋予串行通信程序设计较大的灵活性，端口的读写可选择较简单的查询式，或通过设置与外设数据发送时间间隔 TimeCycle 相同的定时器：SetTimer(1,TimeCycle,NULL)，进行定时读取或发送。

```
CSampleView:: OnTimer(UINT nIDEvent)
```

```
{
    char InputData[30];
    m_Serial.ReadData(InputData,30);// 数据处理
}
```

若对端口数据的响应时间要求较严格，可采用事件驱动 I/O 读写，Windows 定义了 9 种串口通信事件，较常用的有：

EV_RXCHAR：接收到一个字节，并放入输入缓冲区。

EV_TXEMPTY：输出缓冲区中的最后一个字符发送出去。

EV_RXFLAG：接收到事件字符(DCB 结构中 EvtChar 成员)，放入输入缓冲区。

在用 SetCommMask()指定了有用的事件后，应用程序可调用 WaitCommEvent()来等待事件的发生。其中，SetCommMask(hComm,0)可使 WaitCommEvent()中止。

上述两种方法各具优点，适合于不同的场合，在本课题中，选用的是第一种方法，但方法二可以作为课题进一步完善的参考。

4.3.3 多线程技术在通信模块中的应用

4.3.3.1 线程的概念

为了了解线程的概念，我们必须先讨论一下进程的概念。一个进程通常定义为程序的一个实例。与它们在 MS-DOS 和 16 位 Windows 操作系统中不同，Win32 进程并不执行什么指令，它只是占据着 4GB 的地址空间，此空间中有应用程序 EXE 文件的代码和数据。进程是不活泼的，从来不执行任何东西，它只是线程的容器，每个进程至少需要一个线程。EXE 需要的任意 DLL 也将它们的代码和数据装入到进程的地址空间。除了地址空间，进程还占有某些资源，比如文件、动态内存分配和线程。当进程终止时，在它生命期中创建的各种资源将被清除。

进程只是一个静态的概念，为了让进程完成一些工作，进程必须至少占有一个线程，所以线程是进程内的执行单位，正是线程负责执行包含在进程的地址空间中的代码。实际上，单个进程可以包含几个线程，它们可以同时执行进程的地址空间中的代码。为了做到这一点，每个线程有自己的一组CPU寄存器和堆栈。每个进程至少有一个线程在执行其地址空间中的代码，如果没有线程执行进程地址空间中的代码，进程也就没有继续存在的理由，系统将自动清除进程及其地址空间。为了运行所有这些线程，操作系统为每个独立线程安排一些CPU时间，操作系统以轮转方式向线程提供时间片，这就给人一种假象，好象这些线程都在同时运行。创建一个Win32进程时，它的第一个线程称为主线程，它由系统自动生成，然后可由这个主线程生成额外的线程，这些线程，又可生成更多的线程。

进程是由两个部分构成的，一个是进程内核对象，另一个是地址空间。同样，线程也是由两个部分组成的：

- 一个是线程的内核对象，操作系统用它来对线程实施管理。内核对象也是系统用来存放线程统计信息的地方。
- 另一个是线程堆栈，它用于维护线程在执行代码时需要的所有函数参数和局部变量（第 16 章将进一步介绍系统如何管理线程堆栈）。

4.3.3.2 为什么在 Glass Sculptor 的通信模块中使用多线程技术

一般来说，多线程技术适用于需要并行处理某些事务的场合，对于玻璃雕刻机来说，由于雕刻图形比较复杂，数据量很大，传送一些雕刻指令需要一段时间，如果不使用多线程，

在发送数据给控制器时，用户将不能对上位程序进行操作，只能等待数据传输完毕，这显然是影响工作效率的。

此外，如果数据量很大，发送过程比较耗时，用户极有可能在不知情的情况下可能会误认为机器死机或程序出现问题，从而强行关掉程序甚至重启计算机，很可能会使数据传送发生中断，并产生一些不可预知的问题。

综上，我们使用多线程技术来实现 Glass Sculptor 的通信模块，这样，用户在向下位机传送雕刻指令的同时可以对下一幅雕刻图案进行编辑修改，大大的提高了 Glass Sculptor 的工作效率。

4.3.3.3 多线程的编程实现

1、编写线程函数

所有线程必须从一个指定的函数开始执行，该函数称为线程函数，其代码将作为一个单独的线程存在，它必须具有下列原型：

```
DWORD WINAPI YourThreadFunc(LPVOID lpvThreadParm);
```

该函数输入一个 LPVOID 型的参数，可以是一个 DWORD 型的整数，也可以是一个指向一个缓冲区的指针，返回一个 DWORD 型的值。象 WinMain 函数一样，这个函数并不由操作系统调用，操作系统调用包含在 KERNEL32.DLL 中的非 C 运行时的一个内部函数，如 StartOfThread，然后由 StartOfThread 函数建立起一个异常处理框架后，调用线程函数。

Glass Sculptor 中，负责通信任务的线程函数为一个全局函数，代码如下：

```
UINT CommSending(LPVOID pParam)
{
    CDownloadApp* pWinApp=(CDownloadApp*)AfxGetApp();
    CMainFrame* pFrame=(CMainFrame*)(pWinApp->m_pMainWnd);
    CDownloadView* pView=(CDownloadView*)(pFrame->GetActiveView());
    ……//与通信相关的操作
}
```

2、创建一个线程

一个进程的主线程是由操作系统自动生成，创建额外的线程可以调用 CreateThread 完成：

```
HANDLE CreateThread(LPSECURITY_ATTRIBUTES lpsa,
                    DWORD cbstack,
                    LPTHREAD_START_ROUTINE lpStartAddr,//线程函数
                    LPVOID lpvThreadParm,
                    DWORD fdwCreate,
                    LPDWORD lpIDThread);
```

其中 lpsa 参数为一个指向 SECURITY_ATTRIBUTES 结构的指针。如果想让对象为缺省安全属性的话，可以传一个 NULL，如果想让任一个子进程都可继承一个该线程对象句柄，必须指定一个 SECURITY_ATTRIBUTES 结构，其中 bInheritHandle 成员初始化为 TRUE。参数 cbstack 表示线程为自己所用堆栈分配的地址空间大小，0 表示采用系统缺省值。参数 lpStartAddr 用来表示新线程开始执行时代码所在函数的地址，即为线程函数。lpvThreadParm 为传入线程函数的参数，fdwCreate 参数指定控制线程创建的附加标志，可以取两种值。如果该参数为 0，线程就会立即开始执行，如果该参数为 CREATE_SUSPENDED，则系统产生线程后，初始化 CPU，登记 CONTEXT 结构的成员，准备好执行该线程函数中的第一条指令，但并不马上执

行，而是挂起该线程。最后一个参数 lpIDThread 是一个 DWORD 类型地址，返回赋给该新线程的 ID 值。

Glass Sculptor 中，调用线程函数代码如下：

```
hCommSrcThread = CreateThread(    (LPSECURITY_ATTRIBUTES) NULL,
                                  0,
                                  (LPTHREAD_START_ROUTINE)CommSendProc //线程
                                  (LPVOID)&m_comm,
                                  0, &dwThreadID);
if(hCommSendThread != NULL)
{
    ……//相应处理
    return;
}
```

3、终止线程

如果某线程调用了 ExitThread 函数，就可以终止自己：

这个函数为调用该函数的线程设置了退出码 fuExitCode 后，就终止该线程。调用 TerminateThread 函数亦可终止线程：

```
VOID ExitThread(UINT fuExitCode);
```

该函数用来结束由 hThread 参数指定的线程，并把 dwExitCode 设成该线程的退出码。当某个线程不在响应时，可以用其他线程调用该函数来终止这个不响应的线程。

```
BOOL TerminateThread(HANDLE hThread,DWORD dwExitCode);
```

4、设定线程的相对优先级

当一个线程被首次创建时，它的优先级等同于它所属进程的优先级。在单个进程内可以通过调用 SetThreadPriority 函数改变线程的相对优先级。一个线程的优先级是相对于其所属的进程的优先级而言的：

```
BOOL SetThreadPriority(HANDLE hThread,int nPriority);
```

其中参数 hThread 是指向待修改优先级线程的句柄，nPriority 可以是以下的值：

THREAD_PRIORITY_LOWEST,
THREAD_PRIORITY_BELOW_NORMAL,
THREAD_PRIORITY_NORMAL,
THREAD_PRIORITY_ABOVE_NORMAL,
THREAD_PRIORITY_HIGHEST

5、挂起及恢复线程

前面曾提到，可以直接创建挂起状态的线程（通过传递 CREATE_SUSPENDED 标志给函数 CreateThread 来实现）。当这样做时，系统创建指定线程的核心对象，创建线程的栈，在 CONTEXT 结构中初始化线程 CPU 注册成员。然而，线程对象被分配了一个初始挂起计数值 1，这表明了系统将不再分配 CPU 去执行线程。要开始执行一个线程，另一个线程必须调用 ResumeThread 并传递给它调用 CreateThread 时返回的线程句柄：

一个线程可以被挂起多次。如果一个线程被挂起 3 次，则该线程在它被分配 CPU 之前必须被恢复 3 次。

```
DWORD ResumeThread(HANDLE hThread);
```

此外，除了在创建线程时使用 CREATE_SUSPENDED 标志，还可以用 SuspendThread 函数挂起一个现有的线程：

```
DWORD SuspendThread(HANDLE hThread);
```

对于 Glass Sculptor 而言，可以使用线程的挂起和恢复来实现传送数据过程的暂停和继续。

第五章 Glass Sculptor的实现

本章在前几章理论研究成果的基础上，以自行研制的玻璃雕刻机为例对这些理论分析进行实践应用。本课题中，玻璃雕刻机离线编程系统的程序名称为 Glass Sculptor，下面从程序界面、应用举例和雕刻效果三个角度对 Glass Sculptor 进行介绍。

5.1 Glass Sculptor的程序界面简介

本文第一章规划了课题的设计方案，并对本课题的设计原则——简单易用性，进行了介绍，出于这方面考虑，Glass Sculptor 采用纯中文的标准 Windows 应用程序界面，由通用格式的标题栏、菜单条、工具条、编辑区和状态条组成。可以说，用户可以在短期内掌握玻璃雕刻机的应用。

运行安装目录中的 Glass Sculptor.exe 文件，即可打开如图 5.1 所示的程序界面，不难看出，界面由图中五个主要部分组成。从应用角度说，Glass sculptor 实际上是一个交互式的 CAD 系统，从输入中心线、对加工图形进行编辑修改到产生走刀轨迹、雕刻走刀模拟、效果预览和最后的雕刻指令下载，都是在交互式的流程中完成的，而且这些功能的实现可以简单地通过菜单、工具栏按钮、键盘快捷键配合鼠标拖动来实现。

图 5.1 中编辑区域中的两条待加工图形即第二章图 2.4 中的基本加工对象，由这些类型的线条，我们可以组合得到各种复杂精美的图案。

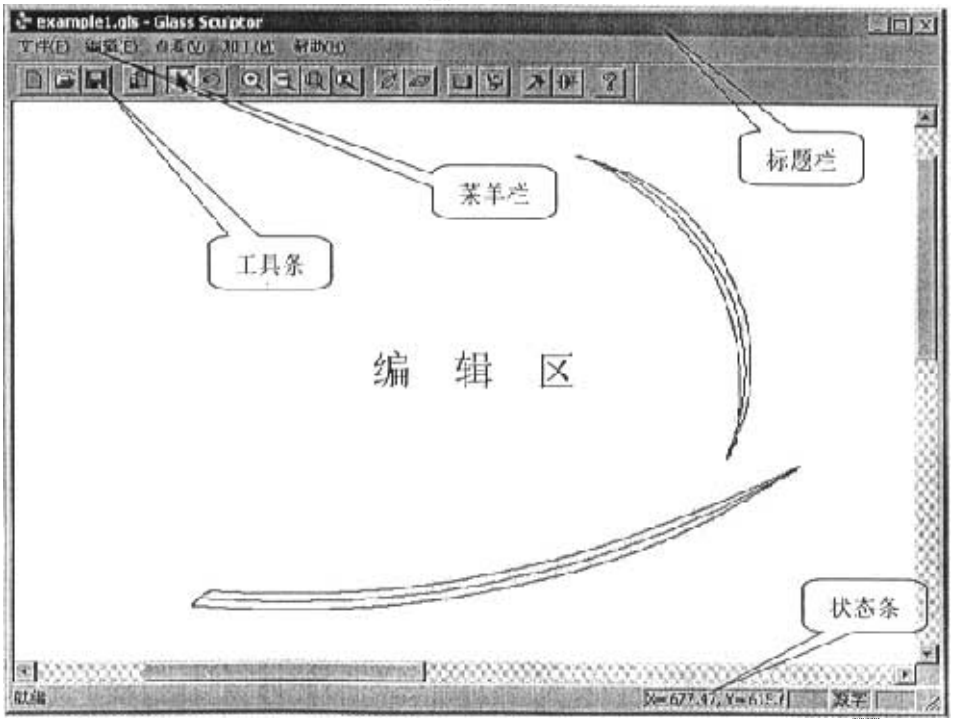


图 5.1 Glass Sculptor 主程序界面

Figure 5.1 The Main Interface of Glass Sculptor

5.2 Glass Sculptor的应用

本节将以第四章中分析的基本数据流程为顺序，通过对一个常见的镜面玻璃图案（第四章曾以此图形为例介绍了最优加工路径的计算算法，参见图 4.16）进行玻璃雕刻来简单介绍玻璃雕刻机的基本功能。

同时，本节也是前几章分别讨论的基本理论的具体实践应用。

1. 导入加工图形：

Glass Sculptor 以中心线的方式导入的待加工图形，如图 5.2 所示。可以采用图 4.1 中的多种待加工图形数据输入方式，图中的待加工图形是以 HPGL 文件的方式由 AutoCAD 绘制而得的。

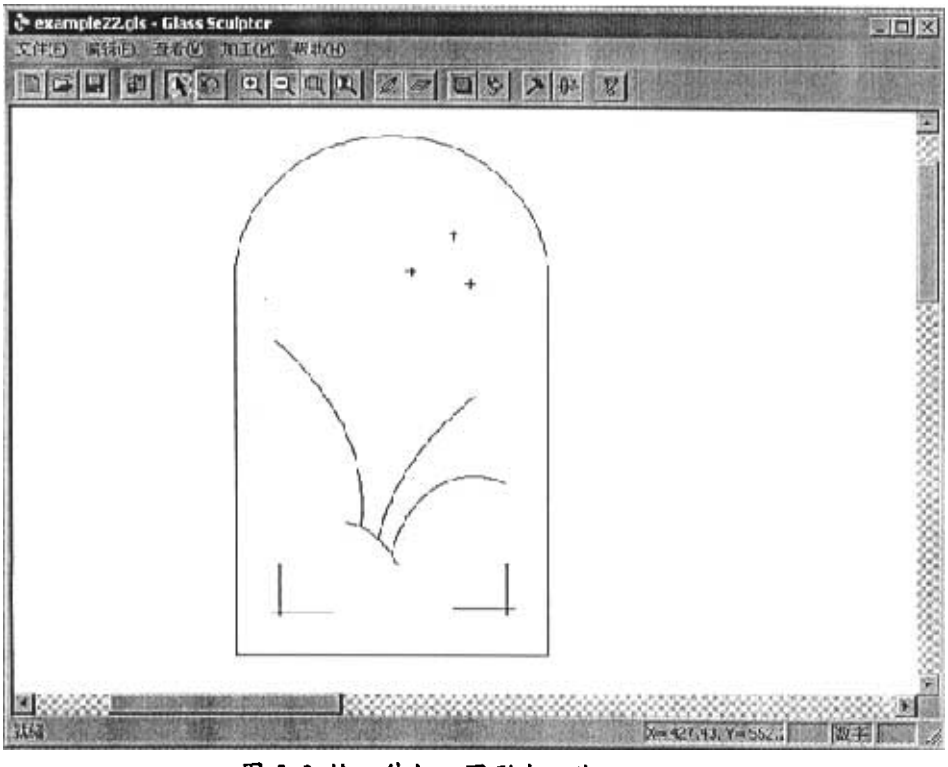


图 5.2 输入待加工图形中心线

Figure 5.2 The Inputing of Graph Data

2. 玻璃雕刻的效果预览和走刀模拟：

根据图 5.2 的加工图形，用户可以对每条中心线的加工属性如线形、线宽、深度、起始终止转角和两个过渡段长度（L1 和 L2）进行设置，根据第四章所介绍的主要算法和优化处理，可以计算得到包含四维信息的的砂轮运动轨迹。在此基础上根据第二章讨论的砂轮界面几何方程，可以计算出砂轮在每一个点处的切割轮廓线，最后求取并绘制这些切割轮廓线的包络曲线，即可在屏幕上得到待加工图形的预览效果图，图 5.3 为图 5.2 中的原始加工图形的雕刻效果预览图。

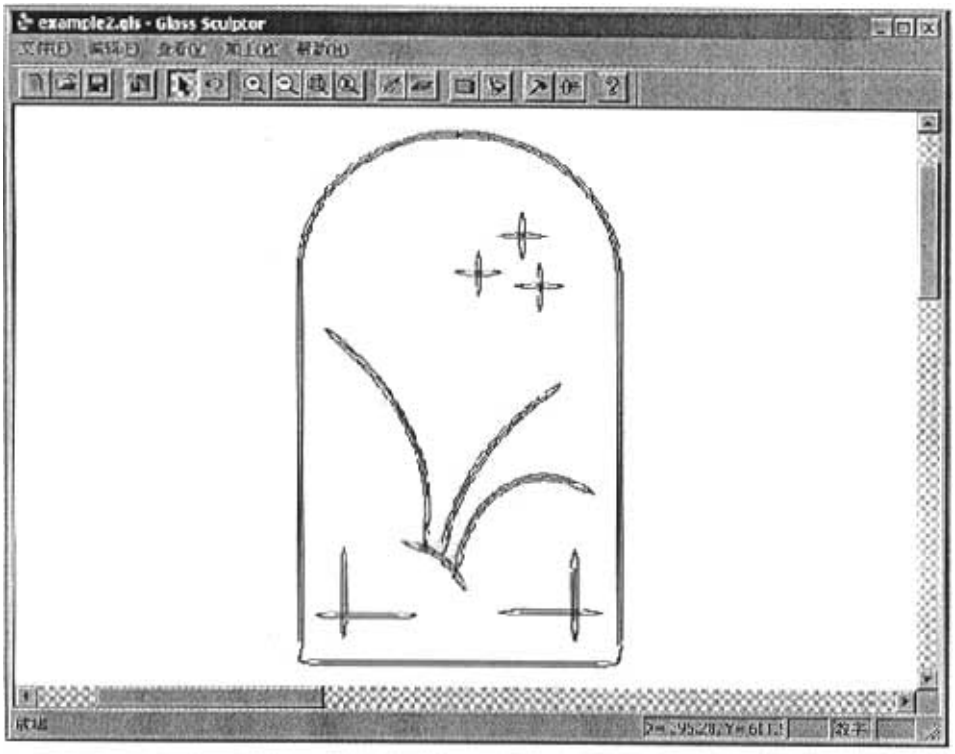


图 5.3 玻璃刻花加工效果图

Figure 5.3 The Preview of Process

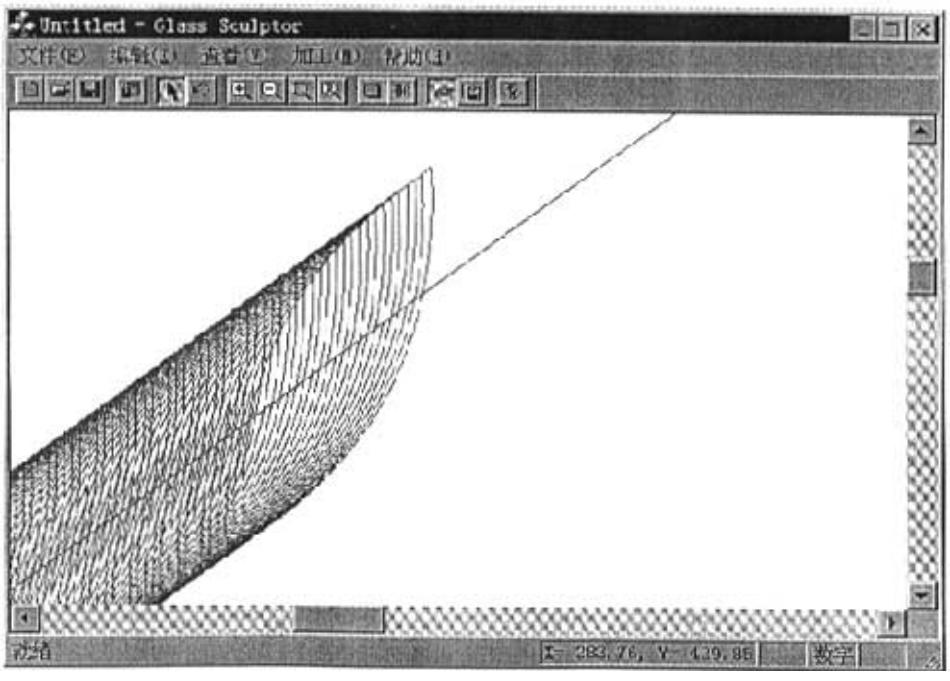


图 5.4 放大以后的走刀模拟图

Figure 5.4 The Simulation of Process

按顺序在屏幕上绘制出中心线上每一点处的切割轮廓线，可以对砂轮在玻璃上雕刻操作进行模拟。图 5.4 为放大后的雕刻走刀模拟图。

通过效果预览和雕刻走刀模拟，可以发现加工图形从图纸角度难以觉察的不美观和不完善的地方，甚至是可能导致产生次品的因素，因此用户可以对比预览效果对待加工图形重新编辑修改，直至满意为止。

3．雕刻指令的发送：

图 5.5 为 Glass Sculptor 与控制器进行串口通讯的程序界面，可以对串口的工作方式、通讯参数进行设置，本模块的具体实现详见本文 4.3 节。

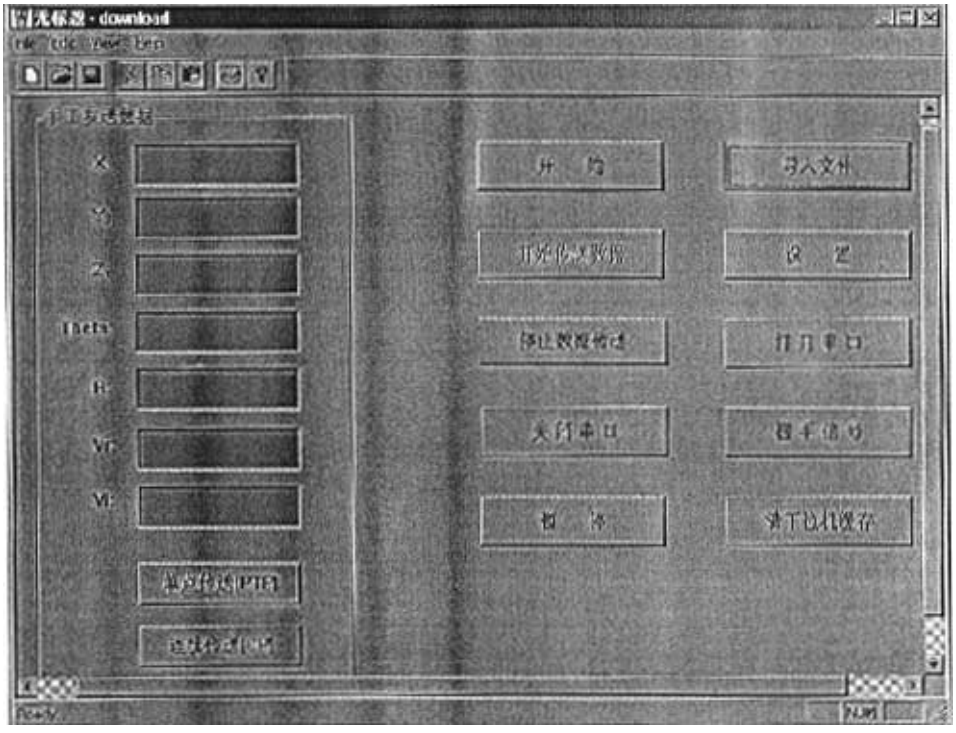


图 5.5　玻璃雕刻机雕刻指令的传输

Figure 5.5 The Transfer of Grooving Instruction

5.3 玻璃雕刻实物

采用本课题开发的玻璃雕刻机，对图 5.2 所设计的图形进行雕刻，雕刻形成的实际图形如附录二所示。雕刻过程中，机床的运行平稳，雕刻以后的线条光滑平整，雕刻速度和加工精度符合设计要求，各轴重复加工精度最大值为 0.005mm，平均移动速度为 9.09m/min，远低于设计指标（具体加工参数见附录三），并通过国家 863 项目验收和上海市经济委员会高新产业办技术成果鉴定，在整体上达到国外进口设备同等水平。

第六章 总结与展望

6.1 课题的总结

本论文是在完成课题研发工作的基础上，对发现问题、研究问题和解决问题过程中所涉及并应用的理论和方法进行介绍，同时也是对所做的实际开发工作的总结，文中所提出的模块划分思想、设计采用的数据结构、数据生成算法和优化处理都是在解决开发过程中出现的实际问题时提出的。

6.1.1 本课题的主要工作

本课题的完成，是在几乎没有任何玻璃雕刻机资料的情况下完成的，经过几个月的工作，作者在设计玻璃雕刻机上层软件时主要完成了如下工作：

- 前期准备阶段，通过研究从图书馆、网络等途径获得的关于国外玻璃雕刻机设备的技术资料，收集国内玻璃生产的市场需求和玻璃雕刻机用户的反馈信息，我们对本课题的开发要求、设计原则进行了总结，并在此基础上制定了课题的设计思路（详见第一章）；
- 通过对比玻璃雕刻机和通用电脑雕刻机的区别，总结分析了本课题的难点和技术创新点，实际上，本课题的展开主要都是围绕着这些难点和技术创新点进行的（详见第一章）；
- 对玻璃雕刻机所涉及的一些重要对象，如加工对象、加工工具、基本加工图形和图形单元进行了抽象分析，设计了易于编程实现，并便于将来升级更新的程序描述方法和基本数据结构（详见第二章）；
- 将课题中面对的问题按功能划分为相对独立的模块，并设计了相应的接口，模块化思想的应用使得玻璃雕刻机的开发更易于实现。为了管理模块化的项目开发，必须借助相应的规划工具，因此又对标准建模语言 UML 进行了研究并应用其建立了玻璃雕刻机的软件模型；（详见第三章）
- 数据的处理是本课题的关键，因此我们对如何输入待加工图形，计算得到包含四维信息的砂轮运动轨迹，并将其发送给控制器，实现最后的玻璃雕刻等核心问题进行了重点研究，将数据的流程细化为六个过程（见图 4.1 中 A、B、C、D、E 和 F 六个过程），在此基础上提出并实现了产生走刀轨迹、离散化切痕轮廓线和产生包络线三个主要程序算法，以及圆弧拟和、最优加工路径两个优化处理方法，这一部分是本课题的主体（详见第四章）；
- 编码并程序实现。整个系统由 Visual C++ 6.0 编制完成，涉及了 MFC、动态链接库、多线程等技术，程序调试成功，界面简单易用，雕刻效果良好（详见第五章和附录）。

6.1.2 本课题研究的收获

本课题属控制软件开发，其开发成果——离线编程系统 Glass Sculptor 将直接应用于我们自行研制、设计、开发的玻璃雕刻机的实际工作当中，并将起着重要的作用。

玻璃雕刻机的研究过程中，从前期资料收集分析、产品规划设计到系统的正式开发实现乃至最后的现场调试，无一不凝聚着本课题开发组数位教授和同学的心血和汗水，可以这么说，玻璃雕刻机的研发成功，不仅仅是课题方面的成功，同时也是团队合作精神的成功。

Glass Sculptor 是一个应用于玻璃雕刻机专用控制软件，然而，本课题研究过程中使用

的方法和思想具有普遍性，不但为今后对玻璃雕刻机软件的进一步研究提供了一定的实践经验，对其他相关课题而言也具有一定的参考价值。在本课题的开发过程中有如下几点经验是本课题的成功之处：

- 前期准备阶段时间比重较大。课题初期，我们花了相当长的时间来收集并分析资料。磨刀不误砍柴工，资料收集工作使得后来的开发少走了很多弯路——研究国外进口设备，使我们了解了玻璃雕刻机的工作方式和基本功能，并以此设计了本课题的开发原型；市场调查让我们了解了客户的需求，从而研究工作更具针对性；通过与电脑雕刻机的横向比较，我们了解了玻璃雕刻机的特殊之处和课题重难点，并以此制定了本课题的研究重心……。可以说，准备充分是任何课题研究都应遵循的原则。
- 将复杂的问题细化为数个简单子问题来逐步实现。玻璃雕刻机涉及的领域较多，开发起来较为复杂，本课题即采用了将复杂问题细化的方法来实现，。如，由简单的线条组合出复杂精美的雕刻图形；将各种风格的线条分类为最简单的三种线型，并以统一的数据结构方式用特征参数进行描述；程序实现时也将界面部分、数据生成算法、优化处理部分、和输入输出部分用不同的模块来分别实现，易于同课题组成员的分工合作。
- 实现方案选取方面以实用为导向。本文前几章介绍玻璃雕刻机的实现机制时，对某些可用方案都进行了比较并介绍了选择的标准，本课题在对系统整体影响不大的方面，更倾向于采用那些实现简单的方案，而不用功能强大却实现复杂的方案，因此大大的缩短了开发时间。如，包络线的求取采用离散法；DLL 文件映象映射到调用进程的地址空间采用隐式链接；用正规 DLL 实现相应模块；用通信控件 MSComm 实现串行通信……。
- 设计模块接口时考虑将来的升级更新。为了便于项目的完善，我们将主要算法和优化处理以独立的模块来实现，升级只需替换相应模块即可；在方案选取时，对未采用的复杂却功能更强大的方案同样也进行了研究；在设计模块时，也为将来可能添加的功能保留了接口。可以说，除非从根本上对本系统进行重新设计，大部分的更新只需进行小规模调整即可。
- 力求在技术和功能上突破创新。人无我有，人有我优是任何产品获得成功的根本之道。本课题的开发借鉴参考了国外玻璃雕刻机的相关技术，但决不是简单的模仿套用，而是力争有技术突破。在总结用户反馈信息的基础上，我们加入了不少新的功能，而且雕刻的方式也有很大不同（详见本文 1.2.4 节），因此加工出来的图案更丰富，可以说，我们研制的玻璃雕刻机与国外进口设备相比也有自己的优势，在技术上也有竞争力。

6.2 前景展望

任何课题，都有一个从粗放到完善的过程。本课题也不例外，到目前为止，我们研发玻璃雕刻机已能应用于玻璃雕刻生产，达到了国外进口设备的水平，但离产品的完善和成功实现产业化还有一定距离。

一方面，本课题的一个设计原则便是将复杂问题细化为子问题，使开发经历了一个由复杂到简单的过程，现在课题的整体框架已成功构建，下一阶段，应由简单返回到复杂，在细节部分进行完善。另一方面，研究开发的实践过程中，有不少思路想法富有创意，但迫于时间有限而未能完成。

玻璃雕刻机离线编程系统的下一步完善可以从以下几个方面来考虑：

- Glass Sculptor 的图形获取方式中，目前绘制修改功能与专业的 CAD 软件相比较为简单，可以考虑采用 OpenGL、DirectX 技术实现绘图模块，也可以考虑使用 ObjectARX

接口对 AutoCAD 进行二次开发；此外，建立通用的图库，并规划方便合理的图库更新方案也能大大方便 Glass Sculptor 的使用；

- 在机械本体加入对刀系统和垂直刀头后，应考虑在程序中进行相应修改；
- 应考虑在防盗版技术进行研究，从技术上保护知识产权。由于玻璃雕刻机与常用的应用软件相比，产量不会很高，因此，防盗版技术实现起来也将会有较大的特殊性；
- Glass Sculptor 目前以图 2.4 中的三种线型为主，可以考虑加入新的基本雕刻单元，如波浪线型等；
- 应对玻璃的刚度进行分析，从动力学角度规划雕刻各种中心线时的最优速度和加速度。

可以说，玻璃雕刻加工将有很广阔的发展前景，本文初步对其中所涉及的一些技术进行了有益的探索和尝试，希望本课题的研究能对推动我国玻璃深加工产业的发展起到抛砖引玉的作用。同时，本文乃至本课题由于时间有限，难免有不足之处，希望各位专家学者不吝赐教，批评指正。

参 考 文 献

[1]冯德坤、马香峰，《包络原理及其在机械方面的应用》，冶金工业出版社，1994 年第一版。

[2]孙家广、许隆文，《计算机图形学，》清华大学出版社，1986。

[3]冯玉琳、黄淘、倪彬，《对象技术导论》，科学出版社，1998 年第一版。

[4]刘超、张莉，《可视化面向对象建模技术——标准建模语言 UML 教程》，北京航空航天大学出版社，1999。

[5]周伯生、张莉、葛科等，《标准建模语言 UML 及其支持环境》，计算机世界，1998。

[6]薛卫，《三维 CNC 雕刻软件研究及开发》，上海交通大学硕士学位论文，1996。

[7]雷霆，《玻璃雕刻机上层软件的研究及开发》，上海交通大学硕士学位论文，2000。

[8]陈金诚，《多轴联动高性能数控加工的运动优化与复杂轨迹实时控制策略研究》，上海交通大学博士学位论文，2001。

[9]赵文静，《数据结构——C++语言描述》，西安交通大学出版社，1999。

[10] 陈国学、钟伟群、刘兵等，《模具和模型 CAD/CAM 集成系统的研究》，机械工程学报，1991，27（6）。

[11] 谭浩强，《C 程序设计》，清华大学出版社，1991。

[12] 李长江、赵志辉、陈杰，《C++使用指南》，电子工业出版社，1995。

[13] 徐军等译，《Visual C++开发人员指南》，机械工业出版社，1996。

[14] Jeffrey Richter，Advanced Windows，Microsoft Press，1998。

[15] Charles Petzold，《Windows 程序设计》，北京大学出版社，1999。

[16] 薛定宇，《系统计算机辅助设计——Matlab 语言及应用》，清华大学出版社，1996。

[17] Sabine Coquillart, Extended Free-Form Deformation: a Sculpturing Tool for 3D Geometric Modeling, Computer Graphics, 1990, 24(4): 187-193。

[18] Ronald J. Norman, Object-Oriented Systems Analysis and Design, Prentice Hall, 1996.（影印本，清华大学出版社，1998）

[19] Edward Yourdon & Carl Argila, Case Study in Object Oriented Analysis & Design, Prentice Hall, 1996.（中译本，殷人昆等译，电子工业出版社，1998）

附录一：国家八六三计划项目验收专家意见表

国家八六三计划项目验收专家意见表

专家姓名	黄振训	电话	021-56331615	传真	021-56331844
工作单位	上海大学 机械工程学院			职务职称	教授
通讯地址	上海延长路149号 上海大学 108人大楼			邮政编码	200072

对项目的整体评价：（包括：项目的目标和完成情况、技术和经济指标、经费实用、研究成果的水平、社会效益和应用前景等）

受国家 863 计划智能机器人主题专家组的委托，验收组于 2000 年 10 月 28 日在上海交大海泰科技发展有限公司对“雕刻机器人产业化”（合同号 863-512-06-04）项目进行验收。验收组听取了课题组的工作报告，技术报告和观看了国家 863 计划机器人产业化基地通用雕刻机器人和玻璃雕刻机的现场操作，专家们经讨论一致认为：

1 雕刻机器人已获得：1999 年由上海市政府新兴产业办进行的产品技术鉴定，取得《三维电脑雕刻机》实用新型专利（专利号 ZL99 2 40381.2），并取得《上海市高新技术成果转化项目证书》A 级。制定了经上海市技术监督局认可的产品企业标准（Q/SGKV01-1999）。

2 D-400 D-600 D-900 三种型号的通用雕刻机器人实现了小批量生产销售，从原年销售量不足十台的情况下发展到 2000 年 7 月份开始月销售量达十台左右，为 2001 年 200 台的销售计划打下基础。

3 玻璃雕刻机的新产品开发达到产品样机的要求，为今后生产打下基础。该机为平板玻璃表面装饰加工提供装备，市场前景良好。

4 在完成本合同过程中扩展开发了计算机应用教学实验系统（教学雕刻机器人），已可为学生配合计算机应用教学提供有效设备，市场前景十分广阔。

验收组认为：项目组工作达到合同要求，技术文档资料齐全，经费使用合理。验收组一致同意通过验收。

验收结论	A、通过验收	B、需要复议	C、未通过验收

专家签名：黄振训

2000 年 10 月 28 日

附录二：雕刻实物图

附录三：上海市经济委员会科学技术成果鉴定

测试报告（节选）

鉴 定 委 员 会 专 家 测 试 报 告

上海交大海泰科技发展有限公司研制的平板玻璃刻花机采用五轴四联动控制方式，四个联动轴定义为：X、Y、Z和C轴，分别控制磨削砂轮的X、Y、Z空间位置以及砂轮偏转角度。另一个轴为主轴，控制砂轮作旋转运动。四个联动轴通过计算机实现同步运动，主轴的转速也在计算机中设定。玻璃刻花机在设计中根据项目计划任务书的要求即：最大线速度9m/min，重复定位精度小于0.2mm。

一、 五轴四联动传动系统的设计值

轴号	电机额定转速	传动方式	传动比	传动分辨率	最大传动速度
X	2500rev/min	滚珠丝杠	丝杠导程：5mm	0.0024mm	12500 mm/min
Y	2500rev/min	减速带 滚珠丝杠	1:1.25 丝杠导程：5mm	0.0020mm	10000 mm/min
Z	2500rev/min	滚珠丝杠	丝杠导程：5mm	0.0024mm	12500 mm/min
C	2500rev/min	齿轮减速器 谐波减速器	1:5.8 1:80	0.0038”	5.38 rev/min
主轴	3000rev/min	同步带	1:1		3000 rev/min

二、 测试项目

1. 重复精度测试　对X轴行程1000mm、Y轴行程500mm、Z轴行程50mm往复重复精度测试，观察千分表读数如下：

测量次数	1	2	3
X轴(1000mm)	0.001mm	0.005mm	0.005mm
Y轴(500mm)	0	0	0
Z轴(50mm)	0	0	0

2. 最大传动速度测试　X轴移动1000mm、Y轴移动500mm距离，测试移动的时间如下：

测量次数	1	2	3
X轴(1000mm)	6.6s	6.5s	6.5s
Y轴(500mm)	3.3s	3.3s	3.2s

三、 测试结果

1. 重复精度测试：经反复3次对X轴的试验结果，重复精度最大值为0.005mm；
经反复3次对Y轴的试验结果，重复精度为0；
经反复3次对Z轴的试验结果，重复精度为0。
则X、Y、Z轴重复精度均远远小于设计指标。

2. 位移速度测试：经反复3次对X轴的试验结果，移动速度为9.09m/min；
经反复3次对Y轴的试验结果，移动速度为9.09m/min。
则X轴、Y轴的最大移动速度均超过设计指标。

3. 结论：测试结果完全达到并超过原设计要求，可提供鉴定委员会予以鉴定通过。

测试组长：[illegible]签字　成员：王治中、陈金成、＿＿＿、＿＿＿

2000 年 11 月 27 日

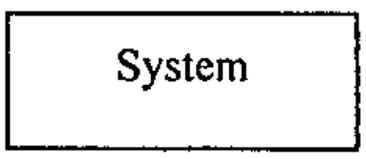
附录四：标准建模语言 UML

用标准建模语言 UML 建立的软件模型都是用 UML 所提供的一套表示符来表示的，为了让读者对本文用 UML 所建立的模型图理解得更清楚，特在附录中给出所用到的 UML 表示符的意义，供参考。

用例图中所用到的表示符：

表示用例，每个用例用于表示所建模系统的一项外部功能需求

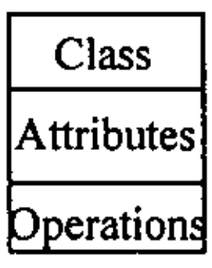
与系统功能有关的外部实体，可以是用户，也可以是外部系统



用于界定系统功能范围

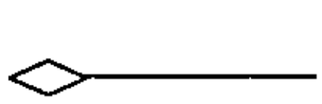
连接执行者与用例，表示执行者与用例所描述的系统需求有关

类图中所用到的表示符：

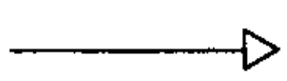


表示类

关联，用来表示类的对象间的关系

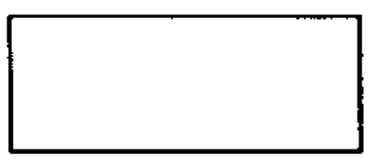


聚集关联，用来表示类的对象之间的关系是整体与部分的关系



泛化关系（也称继承关系），用来表示类的一般元素和特殊元素之间的分类关系

合作图中所用到的表示符：



表示合作图中参与交互的对象

表示对象之间的各种关系，包括组成关系和聚集关系的链接

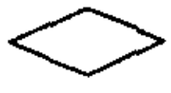
表示对象间从源对象向目的对象发送同步消息

活动图中所用到的表示符：

表示活动图中所有活动的终点

表示活动图中所有活动的起点

表示活动图所描述的过程或算法的某一步



表示活动流程中的判断、决策

同步条，表示活动之间的同步

致 谢

在玻璃雕刻机的研究过程中，从课题选取、开发设计到调试实现，无一不凝聚着本课题开发组每一位成员的心血和智慧。可以这么说，玻璃雕刻机的研发成功，同时也是团队合作精神的成功。

本文是在导师赵维江副教授的悉心指导下完成的，他在论文的选题和研究过程中富有启发性的建议使研究工作得以顺利完成，每一次指导均能使人思路更加开阔。导师学识渊博，为人谦恭，他高尚的品德、敏锐的洞察力、高瞻远瞩的思维方式、踏实严谨的治学作风和诲人不倦的工作态度，将使我终身受益。本文成稿之际，谨向导师致以最崇高的敬意和衷心的感谢！

本文的研究工作得到了上海交大海泰高科技发展有限公司总经理蒋厚宗教授的大力支持，蒋教授思路开阔，经验丰富，虽年事已高却仍然不懈地奋斗在高科技的产业化第一线，令我等后生晚辈肃然起敬。蒋教授对课题方面提供的点拨让我获益匪浅，同时在人生观和个人发展方面的教导也对我启发颇深，在此表示深深的感谢！

感谢上海交大海泰高科技发展有限公司薛卫工程师、李书训博士和上海交通大学的陈金成博士，几位师兄更多的时候是我的老师，在他们的言传身教下，我的理论水平和动手能力获得了很大提高，任何言语都无法概括我对他们的感激之情！

感谢上海交大海泰高科技发展有限公司研究生实验室的其他成员：丁富强博士、徐志明博士、吴嘉盟硕士、詹亚平硕士，大家一起营造的和谐友善的实验室气氛让原本枯燥的研究过程也变得妙趣横生。

感谢上海交通大学机器人研究所所长杨汝清教授，翁新华副教授，顿向明老师，张伟军老师及所有给我提供过帮助的老师、同学和朋友，原谅我无法一一尽述他们的姓名，但他们的帮助和关怀确实是我最大的财富！

感谢 B9906092 班的所有同学，与他们相处的两年半时光让我体会到了挑战与乐趣并存的研究生生活的快乐，他们的每一次激励，每一次关怀甚至每一次调侃打闹都让人永生难忘，两年半时光犹如白驹过隙，一晃毕业在即，祝愿每一位同学都一帆风顺，前程似锦！

深深感谢我的亲人，即使在童年那段艰苦的岁月里也让我一直生活在幸福的亲情之中，是他们一直在鼓励我勇敢地接受人生中的每一次考验，是他们一直在默默地为我祝福，给我信心，也正是他们让我觉得任何艰难的挑战，任何辛苦的工作都是值得的！祝福我的家人永远健康快乐！

吴 炯

2002 年 1 月于上海交通大学

攻读硕士学位期间发表的学术论文

1．四轴联动玻璃刻花加工中心集成控制系统研究，《组合机床与自动化加工技术》，已录用，并列入 2002 年 6～8 期选题计划，论文署名：上海交通大学 机械工程学院：吴炯 赵维江 陈金成 蒋厚宗 薛卫