第 36 卷 第 4 期 计 算 机 工 程 2010 年 2 月
Vol.36 No.4 Computer Engineering February 2010
·工程应用技术与实现· 文章编号：1000—3428(2010)04—0226—03 文献标识码：A 中图分类号：TP391.9

基于 VTK 的线框仿真组件实现技术

王晓宇[1,2]，陈吉红[2]，唐小琦[2]
(1. 襄樊学院机械与汽车工程学院，襄樊 441022；2. 华中科技大学国家数控系统工程技术研究中心，武汉 430070)

摘 要：对刀位点数据进行实时、全交互的加工仿真是高档数控系统的基本功能需求。针对该需求，提出一种基于可视化工具箱(VTK)的数控加工线框仿真组件实现技术。通过动态插入刀位点构造并维护一个 VTK 标准数据集，利用 VTK 功能全面、性能优异的三维数据可视化引擎进行 VTK 数据管道的显示和交互。采用该技术的 ActiveX 组件已成功应用于基于工业以太网的新一代数控系统中。

关键词：可视化工具箱；数控系统；线框仿真

Component Implementation Technology of Wire-frame Simulation Based on VTK

WANG Xiao-yu[1,2], CHEN Ji-hong[2], TANG Xiao-qi[2]
(1. College of Mechanical and Automotive Engineering, Xiangfan University, Xiangfan 441022;
2. Research Center of National Numeric Control System Engineering Technology, Huazhong University of Science and Technology, Wuhan 430070)

【Abstract】It is a basic requirement for modern numeric control system to visualize the cutter location points immediately and interactively. Aiming at the problem, this paper proposes a implementation technology of numeric control machining wire-frame simulation system based on Visualization ToolKit(VTK). A normal VTK dataset is generated and maintained dynamically by inserting cutter location coordinates. A special VTK data pipeline routine is carried out to display and perform interaction operation, it fully utilizes the excellent 3D visualization engine of VTK. The ActiveX using that technology is successfully applied in a new generation of Ethernet based numeric control system.

【Key words】 Visualization ToolKit(VTK); numeric control system; wire-frame simulation

1 概述

对刀位点数据进行实时仿真，直观地展示当前数控加工状态是高档数控系统的基本功能需求，由于实体仿真需要占用更多的计算资源，因此通常采用线框仿真[1]，该线框仿真可以实现刀位轨迹的三维显示，但不具备交互能力，无法实现轨迹的旋转、缩放等操作。另外，它没有维护刀位数据，当窗口进行切换时，已显示的轨迹信息会全部丢失，因此，实用性较低。面向对象开源可视化工具箱(Visualization ToolKit, VTK)，具有定义规范的数据集、丰富的二次开发接口以及强大的计算机图形、图像处理及可视化算法[2-4]。VTK 通过数据管道实现信息的集成与处理，管道中的对象包括源、过滤器、映射器、演员、渲染器、渲染窗口等类型。VTK 中的所有算法均以过滤器的形式存在，在 VTK 应用中，过滤器的使用及扩展是开发重点，要建立包括全部 6 种对象构成的数据管道[3]。

本文提出一种基于 VTK 的数控加工线框仿真实现技术，该技术采用一种只包括显示与交互子系统(从映射器到渲染窗口)的特殊管道，无过滤器和源对象。直接将由程序动态生成及维护的数据集接入到该管道中，利用 VTK 提供的可视化及交互能力。且由于维护了刀位点轨迹信息，当窗口进行切换时，所有信息均得以保留并可以重新显示，因此具有较强的使用价值。该方法中直接生成轨迹单元格的实现技术在已有的 VTK 相关文献中均未涉及，因此，具有较强的推广应用价值。

2 VTK 数据集

VTK 数据集是表达图形或图像类数据的内部数据结构，由数据集结构和属性数据 2 个部分组成，数据集结构规定了数据集的拓扑和几何信息，其中，几何信息是点及其坐标；单元格是数据集的原子单位，反映了数据集中点的连接关系，因此，单元格反映了数据集的拓扑信息。数据集的属性数据给出了某个点或单元格的属性信息，属性数据类型可以是标量、矢量、张量等。

VTK 可以处理图像数据和图形数据，其中，图像数据点和单元格是隐式表示的；图形数据是显式表示的。VTK 支持的数据集结构包括 6 种：图像，矩形网格，结构网格，散乱点，图形数据和无结构网格。

例如一个气象信息数据集结构即数字地图，这个地图可以是栅格化的地图 vtkImageData(图像数据)表示，也可以是矢量化的地图 vtkPolyDat(图形数据)表示，它们构成了数据集的结构。数据集上的点或单元格可以包含如气压、温度、风力、风向等与天气有关的信息，这些信息可以作为数据属性赋予到每个点或单元格上。这些信息中包括气压、温度、风力等标量信息，也包括风向等矢量信息，这些属性信息和数据集结构一起构造一个完整的 VTK 数据集。

本方法生成的目标数据集是一个如图 1 所示的 vtkPolyData 类型的图形数据集，该数据集包括 4 个相互联系

基金项目：国家“十一五”基金资助重大项目“开放式全数字高档数控装置”(2009ZX04009-011-02)
作者简介：王晓宇(1972－)，男，讲师、博士，主研方向：红外图像处理，数控系统；陈吉红、唐小琦，教授、博士生导师
收稿日期：2009-06-11 **E-mail：**wxyisok@263.net

的动态数组，其中，vtkPoints 数组保存了刀位点轮廓的空间信息，即数据集的几何信息；vtkCellArray 保存了点之间的连接关系即单元格信息；vtkCellTypes 提供了对单元格的随机访问和单元格类型；vtkPolyData 包括 14 种单元格类型(本文类型为 VTK_POLY_LINE)；vtkCellLinks 保存了单元格之间的拓扑关系[4]。

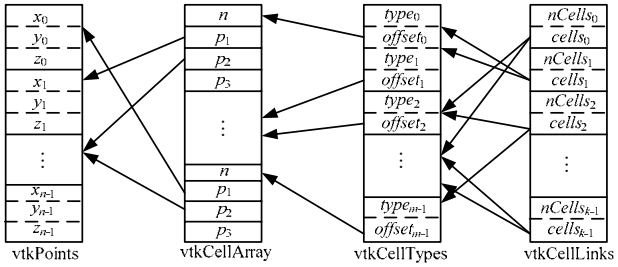

图 1　vtkPolyData 数据集格式

本文提出的技术通过刀位点信息构造这 4 个动态数组来构成 vtkPolyData 类型的数据集，并将该数据集插入到一个只包括显示与交互子系统的特殊管道中，完成刀位轨迹数据的实时渲染和可视化交互。

3　关键技术

本文主要涉及数据管道设置和坐标数据集生成 2 个关键步骤。

3.1　数据管道设置

本文的数据管道包括映射器、演员、渲染器、交互器等对象。映射器、演员、渲染器及交互器等对象以底层 OpenGL 或 DirectX 图形系统为基础，主要完成三角片离散、纹理映射、层次细节处理(Level of Detail, LOD)、拾取与交互、相机与光照设置等与底层图形系统相关的低级图形处理功能。

建立数据管道后可获得 VTK 提供的功能全面、性能优异的空间数据可视化及其交互能力。设置 VTK 数据管道的关键代码片段如下：

```
/*vtkRenderer* ren;//渲染器
vtkRenderWindow *renWin;//渲染窗口
 vtkRenderWindowInteractor *iren;//交互器
vtkInteractorStyle * irstyle;//交互风格*/
…
//设置边界左下角文本
memset(sbuf,0,200);
sprintf(sbuf,"S:%.2f,%.2f,%.2f",Boundary[0].x,Boundary[0].y,Boundary[0].z);
//atext0 是 vtkVectorTextle 类型对象
atext0->SetText(sbuf);
//atext0Mapper 是 vtkPolyDataMapper 类型对象映射器
atext0Mapper->SetInput(atext0->GetOutput());
//textActor0 是 vtkFollower 类型对象一种特殊的演员
textActor0->SetMapper(atext0Mapper);
//设置花朵位置、颜色、指向、缩放
textActor0->SetPosition(Boundary[0].x,Boundary[0].y,Boundary[0].z);
textActor0->GetProperty()->SetColor(TextColor[0],TextColor[1],TextColor[2]);
textActor0->SetCamera(ren->GetActiveCamera());
textActor0->SetScale(TextSize*ren->GetActiveCamera()->GetParallelScale());
//在渲染器中加入文本花朵演员
ren->AddActor(textActor0);
…
//用一可缩放的小球作为刀具
//Tool 是 vtkSphereSource 类型对象
ToolMapper->SetInput(Tool->GetOutput());
//演员指向映射器
ToolActor->SetMapper(ToolMapper);
//设置刀具演员属性
ToolActor->GetProperty()->SetColor(ToolColor[0],ToolColor[1],ToolColor[2]);
//原来刀具在 xoy 平面上以 y 轴为轴心，绕 x 轴 90 度即与 xoy 面
//垂直
ToolActor->SetPosition(0,0,0);
ToolActor->RotateWXYZ(90,1,0,0);
//在渲染器中加入刀具演员
ren->AddActor(ToolActor);
…
//显示走刀刀位轨迹 LineTo
//TravePth 为 vtkPolyData 类型走刀数据集,mapTravePth 为映射器
mapTravePth->SetInput(TravePth);
//演员指向映射器
TravePthActor->SetMapper(mapTravePth);
//设置刀位轨迹演员属性
TravePthActor->GetProperty()
->SetColor(LineToColor[0],LineToColor[1],LineToColor[2]);
ren->AddActor(TravePthActor);
//显示跳刀刀位轨迹 MoveTo
//TravePthMoveTo 为 vtkPolyData 类型跳刀数据集
mapTravePthMoveTo->SetInput(TravePthMoveTo);
TravePthActorMoveTo->SetMapper(mapTravePthMoveTo);
TravePthActorMoveTo->GetProperty()
->SetColor(MoveToColor[0],MoveToColor[1],MoveToColor[2]);
ren->AddActor(TravePthActorMoveTo);
…
//将渲染器加入渲染窗口
renWin->AddRenderer(ren);
//Win32 环境下窗口绑定
renWin->SetParentId((void*)this->GetSafeHwnd());
//设置交互器及其风格
```

上述代码中出现的花朵对象是一种只有提示文本信息的特殊演员对象，在 3D 环境中用户旋转相机视角时，该对象根据当前相机视角自动将提示文本旋转到用户正视方向。代码涉及 2 个数据集：走刀数据集和跳刀数据集，它们都是图 1 所示的 vtkPolyData 类型数据集，由程序动态维护，走刀和跳刀分别对应数控加工 G 代码中的 G01 和 G00 指令执行结果，用下划线给予标记。

3.2　数据集维护

从图 1 可以看到，vtkPolyData 的几何和拓扑信息均为显式表示而且结构复杂，在线框仿真中，前 2 个部分的结构是 vtkPoints 和 vtkCellArray，分别表示几何信息和拓扑信息。构造 vtkPolyData 的实现细节如下：

```
//修改拓扑信息，参数 newid 是插入到 vtkPoints 数组后得到的 ID
vtkIdType ReplaceIdt(vtkIdType newid)
{
 //获得 vtkCellArray
 vtkCellArray *ca=TravePth->GetLines();
 long ncells=ca->GetNumberOfCells();
 vtkIdType* ids=ca->GetPointer();
 vtkIdType size=ca->GetSize();
long celli=0,length,num;
```

```
long i,j,idx,lastnodeidx=0;
for (i=0,idx=0;i<ncells;i++)
{
   //celli 记录第 i 个单元格的点数
   celli=ids[idx];
   //lastnodeidx 记录下最后一个单元格点数所在的位置
   lastnodeidx=idx;
   //idx 为数组 vtkCellArray 的索引
   idx+=celli+1;}
 //获得 AllLines 的下一个 ID，如果==-1 则表示 AllLines 为一空链表
 long x=AllLines->GetMaxId();
 //如果链表为空
 //则插入一个长度为 1 的 cell 并将 newid 作为第 1 个点插入
 if (x==-1)
 {
      AllLines->InsertNextValue(1);
      AllLines->InsertNextValue(newid);
      ncells=1;}
 //否则插入 newid 到最后一个 cell，并修改该 cell 的点数+1
 //最后一个 cell 的点数所在位置为 lastnodeidx
 else
 {
      AllLines->InsertNextValue(newid);
      num=AllLines->GetValue(lastnodeidx);
      num+=1;
      AllLines->SetValue(lastnodeidx,num);
   }
 return ncells;//返回单元格 Id
}
//构造数据集
int Travesed()
{
 long ncells;
 vtkIdType* idt;
 vtkPoints* points=TravePth->GetPoints();
 ToolActor->SetPosition(px,py,pz);
 CurToolPos.x=px;CurToolPos.y=py;CurToolPos.z=pz;
 PointsNum=points->GetNumberOfPoints();
 vtkIdType newid=points->InsertNextPoint
(CurToolPos.x,CurToolPos.y,CurToolPos.z);
//插入几何信息，并得到点 ID
 TravePth->SetPoints(points);
 ncells=ReplaceIdt(newid);//走刀
 //构造单元格数组
 vtkCellArray* l=vtkCellArray::New();
 l->SetCells(ncells,AllLines);
 //删除旧单元格
 TravePth->GetLines()->Delete();
 //生成新单元格
 TravePth->SetLines(l);
 //缩放刀具和花朵提示文本大小
 CameraScale=ren->GetActiveCamera()->GetParallelScale();
 Tool->SetRadius(m_ToolRatio*CameraScale);
 textActor0->SetScale(TextSize*CameraScale);
 return 1;
}
```

上述代码的全局变量“ AllLines ”是 vtkIdTypeArray 类型的指针，它本身是一个 ID 列表，代码维护其内容并生成图 1 所示的 vtkCellArray 类型数组。

在上述代码中，函数“ Travesed ”为主函数负责刀位点轮廓数据集的维护；函数“ ReplaceIdt ”负责维护数据集的拓扑信息，负责单元格的维护；对当前空间点“ CurToolPos ”首先插入到 vtkPoints 数组中，得到一个点的 ID 值“ newid ”，然后调用函数“ ReplaceIdt ”更新单元格；在函数“ ReplaceIdt ”中首先获得目标 vtkPolyData 数据的 vtkCellArray 数组，vtkCellArray 是一个形如：($n1$,$ID1$,$ID2$,…,$IDn1$,$n2$,$ID1$,$ID2$,…,$IDn2$,…)的数组，其含义为：第 1 个 Cell 的点数 $n1$，第 1 个 Cell 第 1 个点的 $ID1$，第 1 个 Cell 第 2 个点 $ID2$，…，第 $n1$ 个点 $IDn1$，第 2 个 Cell 点数 $n2$，第 2 个 Cell 第 1 个点 $ID1$…第 $n2$ 个点的 $IDn2$,…)；首先搜索最后一个 Cell 点数在 vtkCellArray 中的位置“ lastnodeidx ”，如果当前 ID 列表为空，则插入一个新的单元格，单元格点数为 1，且第 1 个点的 ID 为当前空间点对应的“ newid ”；如果当前 ID 列表不为空，则在最后一个 Cell 上插入 ID 为“ newid ”的点，并修改点数为原值+1；根据上述步骤维护的 ID 列表“ AllLines ”可以通过 vtkCellArray 成员函数“ SetCells ”生成 vtkCellArray 数组，并通过 vtkPolyData 类成员函数“ SetLines ”构造数据集拓扑信息，生成数据集。上述代码没有给出跳刀的处理，函数“ ReplaceIdt ”应在 ID 列表“ AllLines ”中插入一个值为 0 的空白 Cell，然后再次以当前空间点 ID 以非跳刀标记调用函数“ ReplaceIdt ”即可。

vtkIdTypeArray 类型的 ID 列表及构造 vtkPolyData 的拓扑信息涉及了较多 VTK 数据接口的调用。VTK 开发文档[5]中与本文技术有关的接口函数文档难懂且没有实例，笔者在 VTK 新闻组[3]中也未找到相关接口的解释，本文完整地表示了从空间点到列表 ID 的维护及 vtkPolyData 生成过程，在 VTK4.0+VC6.0 环境下已验证通过。

4 应用实例

本文提出的基于 VTK 的数控加工线框仿真 ActiveX 组件已在华中数控基于工业以太网的新一代数控系统 HNC-32 系统中得到应用，图 2 是四通道同时显示的界面，其中，通道 1 和通道 4 为本文线框仿真组件的 2 个实例(通道 2 和通道 3 分别为 G 代码显示和实体仿真显示)。

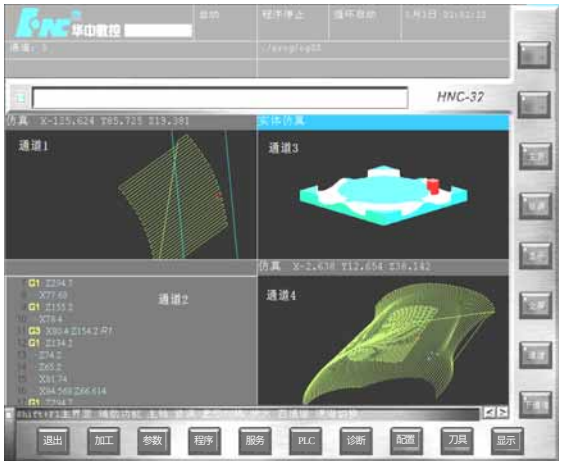

图 2 应用实例

由于刀位点轨迹数据集由程序维护并记录，因此仿真窗口可任意关闭或切换，轨迹信息得到保留不受影响。由于完整地采用了 VTK 的显示与交互子系统，因此获得了缩放、平移、相机固定视角设置等全交互操作能力，可以以任意视角显示所有的加工细节。 (下转第 232 页)