文章编号: 1002-0446(2001)05-0404-03

一种通用的机器人三维图形仿真的实现*

刘振宇　徐　方　陈英林

(沈阳新松机器人自动化股份有限公司, 中国科学院沈阳自动化研究所　沈阳　110015)

摘　要: 机器人三维图形仿真是离线编程的一个重要内容, 本文介绍了一种应用 **ActiveX** 编制的机器人三维图形仿真控件, 包括机器人运动学模型的建立方法和三维图形仿真的实现. 这种控件具有很强的通用性, 可以与多种编程环境兼容.

关键词: 离线编程; 三维图形仿真; 运动学模型; **ActiveX**.

中图分类号:　**TP**24　　文献标识码:　**B**

IMPLEMENTATION OF 3D GRAPHICAL SIMULATION FOR GENERAL ROBOT

LIU Zhen-yu　XU Fang　CHEN Ying-lin

(*Shenyang Institute of Automation, Chinese Academy of Sciences, Xinsong Robot & Automation Co. Ltd., Shenyang*　110015)

Abstract: 3D graphical simulation for robot is an important content of off-line programming. This paper introduces a 3D graphical simulation controller for robot based on the ActiveX, including kinematic modelling and the implemetation of the simulation. This controller has strong compatibility, and it can be compatible with many programming languages.

Keywords: off-line programming, 3D graphical simulation, kinematic model, ActiveX

1　引言(Introduction)

进入 21 世纪, 机器人已成为现代工业不可缺的重要工具. 它标志着工业的现代化程度. 通常, 机器人的编程方式可分为示教再现编程和离线编程两类[4]. 目前, 生产使用的机器人仍是采用示教再现编程, 随着工业生产对自动化水平要求的增高, 机器人的功能也要不断的增强. 所以在示教编程之外, 增加离线编程已经是机器人研究领域中的重要课题. 而三维图形仿真是实用化离线编程技术的重要内容.

中国科学院沈阳自动化研究所多年从事工业机器人的研究开发工作, 近几年, 具有自主产权的焊接机器人已进入生产线使用. 使其具有图形仿真的离线编程功能是我们进行的一项工作. 本文介绍了其中的图形仿真模块, 其特点是:

- 应用 Active 技术实现机器人三维造型, 使此模块具有很强的通用性;
- 便于机器人模型的修改, 适合不同类型机器人的仿真;
- 图形效果逼真, 动画质量好, 软件效率高.

2　三维图形仿真的具体实现 (Implemetation of 3D graphical simulation)

2.1 机器人运动学模型的建立

机器人运动学模型是机器人三维图形仿真的基础, 下面我们以沈阳新松机器人自动化股份有限公司自主研制的 RH-06A 型机器人为例进行建模. RH-06A 型机器人(如图 1 所示). 是一种 6 自由度关节型机器人, 我们把各轴的原点依次称为 $O1$, $O2$, $O3$, $O4$, $O5$, $O6$, 根据坐标变换原理[5]可得机器人末端执行器中心 p(没有工具时被认为是 6 轴的中心点) 相对于 1 轴的坐标变换. 2 轴的坐标系是 1 轴坐标系绕 z 轴旋转 $\theta 1$ 后又沿 y 轴平移 $D1$ 得到的, 因此得到 2 轴坐标系相对于 1 轴坐标系的变换矩阵为

$$^{0}_{1}T = Rot(z, \theta_1)\, Trans(y, D1) \qquad (1)$$

* 收稿日期: 2001-01-15

同理可以得到其余的变换矩阵, ${}^1_2T$, ${}^2_3T$, ${}^3_4T$, ${}^4_5T$, ${}^5_6T$,

最后, 由坐标变换原理可得

$$ {}^0_6T = {}^0_1T\ {}^1_2T\ {}^2_3T\ {}^3_4T\ {}^4_5T\ {}^5_6T = \begin{vmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (2) $$

根据这个变换矩阵我们就可以利用一种 3D API—— OpenGL 来进行机器人的图形仿真了.

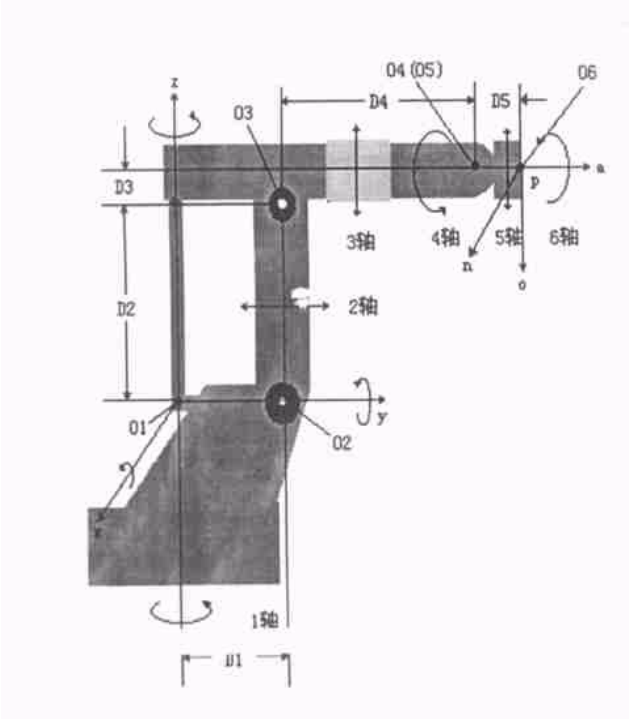

图 1　RH-06A 型机器人简图

Fig. 1　Sketch of robot RH-06A

2.2　基于 OpenGL 的三维图形仿真

由于人脑拥有处理视觉信息所需的高度能力, 而图像又是沟通思维最自然的手段, 因此仿真信息的可视化处理成为仿真技术的一个重要内容[1]. 机器人图形仿真是离线编程系统的重要组成部分[2], 它将机器人仿真的结果以图形的形式显示出来, 从而直观地显示出机器人的运动情况, 可以得到从数据曲线或数据本身难以分析出来的许多重要信息. 以往由于机器人图形和动画仿真的计算量很大, 在微机上实现效果不佳或根本无法实现, 因此大多数的仿真都是基于图形工作站的[6], 由于造价昂贵, 从而限制了这项技术的发展. 随着计算机硬件以及 CAD, CAM 技术的飞速发展, 加上性能卓越的开放式三维图形标准 OpenGL 的出现, 使得在 PC 机上实现高品质的机器人三维图形仿真成为现实[3].

新松公司研制的 RH-06A 型机器人是一个 6 自由度系统, 它有底座、底盘、大臂、小臂、手腕和末端关节等几个部分. 从几何造型上来看, 底座和末端关节是圆柱体, 其它部分都是多面体. 因此, 机器人的造型过程是按照结构化立体造型法的方式来进行的, 整台机器人分解为底座、底盘、大臂、小臂、手腕和末端关节六个部件, 每个部件又可以分为几个基本几何体素(长方体、圆柱体)的组合. 体素经过累加结合组成一个个部件, 部件的连接是按照式(2)的变换矩阵进行的. 机器人具体的几何造型是一个自底向上的过程, 如图 2 所示.

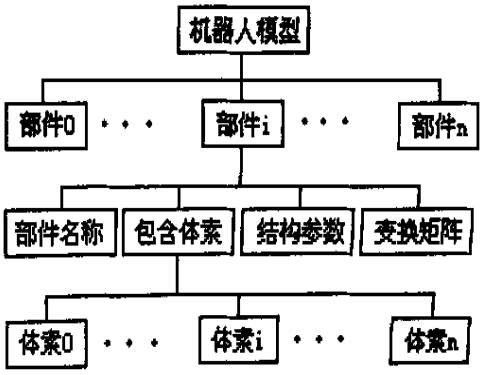

图 2　机器人模型层次结构图

Fig. 2　Hierarchy diagram of robot

(1) 体素级

体素是构成形体的最基本单元, 其数据结构的安排状况直接影响到整个模型处理的复杂度和速度. 本图形系统采用了两种体素结构, 即长方体与圆柱体.

• 长方体

长方体采用贴面的方式构成. 首先按如下数组方式给出长方体 8 个顶点在某个坐标系下的三维坐标:

```
static float
p1 = {p1x, p1y, p1z},
p2 = {p2x, p2y, p2z},
... p8 = {p8x, p8y, p8z}
```

然后, 用每相临的 4 个顶点构成一个有向面, 从而产生 6 个有向面:

```
glVertex3fv(p1, p2, p3, p4);
glVertex3fv(p2, p3, p6, p5);
... glVertex3fv(p5, p6, p7, p8)
```

这六个面就"贴"成了一个长方体. 这里点的选取是

有方向的, 都为顺时针或都为逆时针, 以有助于以后的消隐等工作.

这样, 长方体体素的几何信息和拓扑信息分别存在了 6 个二维数组中, 体素经比例、平移、旋转等变换, 可以拼合成复杂物体.

• 圆柱体

OpenGL 提供的基本体中就含有圆柱体. 按如下方式定义:

首先定义一个 NURBS 曲线指针: GLU quadricObj * objectname;

然后在视图的实现文件中得到圆柱体的具体定义:

gluCylinder(objectname, baseradius, topradius, height, slices, stacks);

其中, objectname: NURBS 曲线指针;

baseradius: z= 0 处圆柱的半径;

topradius: z= height 处圆柱的半径;

height: 圆柱的高度;

slices: 绕 z 轴的离散曲面数目;

stacks: 沿 z 轴的离散曲面数目.

实际上, gluCylinder 函数根据 baseradius 和 topradius 的不同取值可以实现圆锥、圆柱和圆台. 当 baseradius 等于 topradius 时, 就得到圆柱. 同样, 这个圆柱也可以经过坐标变换, 构成新的物体.

(2) 部件级

RH-06A 型机器人包括了六大部件, 除了底座和末端关节外, 都是由多个体素构成的. 根据 OpenGL 的编程规则, 为了提高程序的效率, 把这六个部件分别作成六个列表, 以便于在机器人模型级中调用. 列表是事先存储的用于稍后执行的一组命令序列, 一旦完成列表的建立, 就将其处理成适合于图形硬件的格式, 而且可以避免在绘图过程中因主机计算量太大而影响图形输出的速度, 从而提高了效率. 下例即为底座列表:

```
glNewList(baseN, GL_ COMPILE);
glPushMatrix();
gluCylinder(base, m_ bR, m_ bR, m_ bh, slices, stacks);
glPopMatrix();
glEndList();
```

其中, 夹在 glNewList 与 glEndList 函数之间的部分为新的列表, glPushMatrix() 和 glPopMatrix() 函数可以保证坐标变换的正确.

(3) 机器人模型级

根据变换矩阵(2)以及各个部件列表可以得到机器人仿真模型, 以 RH-06A 型机器人为例:

```
glCallList(baseN);
    glCallList(OBJECT1);
glTranslatef(0. 0, 0. 0, D1);
glCallList(OBJECT2);
glTranslatef(0. 0, 0. 0, D2+ D3);
glCallList(OBJECT3);
glRotatef(- 90, 1. 0, 0. 0, 0. 0);
glTranslatef(0. 0, D4);
glCallList(OBJECT4);
glTranslatef(0. 0, D5);
glCallList(OBJECT5);
```

3 通用控件的实现 (Implemetation of the general controller)

ActiveX(又称 Active 技术)是一种开放式技术, 程序开发人员可以利用它开发出通用的应用程序或模块[8]. ActiveX 的好处在于, 不管应用程序用哪一种计算机语言, 都可以插入 ActiveX 控件, 容器与控件总能融洽的交互. ActiveX 控件提供了三种与容器应用程序集成的基本机制: 属性、事件、方法. 属性就是控件的属性, 可在容器中读取和修改; 方法就是控件提供的函数, 可供容器调用; 事件就是控件发生的情况, 用于通知容器(通过调用容器中特定事件的处理函数).

我们利用 ActiveX 构造了机器人仿真通用控件 Robotctl, 具体实现如下:

(1) 利用 MFC Control Wizard 创建一个新的工程文件;

(2) 在控件实现文件中按照前面介绍的机器人建模方式输入机器人仿真模型的源代码;

(3) 在控件属性页的 DoDataExchange 函数中实现具体的属性, 把机器人各种参数当作 Robotctl 控件的不同属性, 在调用 Robotctl 控件的时候, 只需要改变这些属性值就可以得到不同型号的机器人.

(4) 运行控件, 进行注册, 以供调用.

图 3 就是在 VC++ 6.0 下利用 Robotctl 控件生成的 RH-06A 型机器人三维仿真图例.

这是在有光照的条件下, 并经过反走样处理的图像, 由于采用了双缓存技术, 动画的刷新频率可达到 12.5Hz, 如果仿真图形采用线框模型, 刷新频率还可以进一步提高.

(下转第 449 页)