基于 OpenGL 的 6R 工业机器人动态仿真*

郝矿荣 东华大学信息学院(201103)
孟 飞 朱玉蓉 郑州工程学院 (450052)

Abstract

This paper gives a new kinematic simulation method by using the sets upright kinematics model aim at 6R jointing robot based on kinematicanalysis.Adopting OpenGL and VC++,and empolders a suit of real time simulation system.It presents detail design process, empolering system interface and arithmetic chart.To a certainty,it is universal to dynamic simulation of robot.

keywords:robot,OpenGL,kinematics simulation

摘 要

本文针对 6R 焊接机器人,在进行运动学分析的基础上,建立了正确的 6R 运动学模型。并采用 OpenGL 和 VC++技术开发了一套实时仿真系统。给出了详细的设计、开发步骤,以及系统的界面和算法框图,该系统对机器人的动态仿真具有一定的通用性。对降低机器人的设计制造的成本有着重要的现实意义。

关键词:机器人,OpenGL,运动学仿真

随着对机器人研究的不断深入和机器人领域的不断拓展，机器人仿真系统作为机器人设计和研究的安全可靠,灵活方便的工具,在本领域发挥着重要作用。本文结合 6R(旋转关节)焊接机器人,对 6R 机器人仿真系统的开发进行了探讨。并给出了开发的界面、系统界面和图形的交互过程。实现多自由度,多关节机器人的动态仿真,达到考查、研究多自由度,多关节机器人的目的。

1 开发工具和开发环境

本系统将在 Window9X/NT 平台上开发，采用面向对象(objiect-orien)的编程技术。选用 Vc++6.0 的编程环境。Vc++6.0 与 Windows9X 具有相同接口，OpenGL 被嵌入 Windows 中,不存在接口问题。同时 OpenGL 的强大强大图形处理功能,使它成为我们开发这套图形仿真系统的首选。

OpenGL(open graphics library)是 SGI 公司开发的一套 3D 图形软件接口标准，是独立于操作系统和硬件环境三维图形库，能十分方便地在各平台间移植,已成为开发计算机图形的工业标准。具体功能有:几何建模,坐标变换,颜色模式设置,光照和材质,图象效果,位图和图象,纹理映射,实时动画,交互技术等。

2 6R 机器人运动学分析

2.1 建立数学模型

机器人关节参数如表 1 所示。根据设定的机器人几何模型，建立合理的数学模型。对于任一空间机器人,为了建立运动学方程,首先要在每个臂杆上建立与各臂杆相固连的动坐标系,我们称之为杆件坐标系统。为了求解运动学方程,用 Denavit-Hartenberg 法建立 A 距阵(相临两杆件坐标系的奇次坐标变换距阵)为:

表 1 6R 机器人关节参数

杆号	关节变量	扭角	杆长	转角范围	偏移
1	θ	90°	0	±165	a_1
2	θ	0	a_2	±75	0
3	θ	90°	0	±130	0
4	θ	-90°	0	±180	a_4
5	θ	90°	0	±105	0
6	θ	0	0	±180	a_6

$An=Rot(Z,\theta_n)Trans(0,0,dn)Trans(an,0,0)Rot(X,\alpha_n)$

$$=\begin{bmatrix} c\theta_n & -s\theta_n & 0 & 0 \\ s\theta_n & c\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & an \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dn \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_n & -s\alpha_n & 0 \\ 0 & s\alpha_n & c\alpha_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$=\begin{bmatrix} c\theta_n & -s\theta_n c\alpha_n & s\theta_n s\alpha_n & a_n c\theta_n \\ s\theta_n & c\theta_n c\alpha_n & -c\theta_n s\alpha_n & a_n s\theta_n \\ 0 & s\alpha_n & c\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其中:令 s=sinθ,c=cosθ。d_n:为沿杆件的轴线两个公垂线的距离;θ_n:为垂直于杆件 n 的轴线的平面内两个公垂线的夹角;an：为两个关节轴线沿公垂线的距离;α_n：为在垂直于 an 的平面内的两个关节轴线的

* 国家教育部归国留学基金支持项目

夹角；这一变换可用 OGL 提供的坐标变换函数 glTranslate()和 glRotate()来实现。

(1)运动学正解问题

这一问题是根据给出的机器人 6 个关节变量 θ_i 及结构参数,求出末端操作器位姿各矢量$\vec{n},\vec{o},\vec{\alpha}$和$\vec{p}$。

$$T_n=\prod_{n=1}^{n}A_n=\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} (1\leqslant n\leqslant 6)$$

则末端操作器的位置为(p_x,p_y,p_z),姿态角用横滚角,俯仰角,侧摆角表示。

$\gamma=\text{atan}(n_y,n_x)$, $=\text{atan}(-n_z,n_x\cos+n_y\sin)$,

$\alpha=\text{atan}(a_x\sin-a_y\cos,o_y\cos-o_x\sin)$;

(2)运动学逆解问题

这一问题是已知机器人末端操作器要到达的空间位姿$\vec{n},\vec{o},\vec{\alpha}$和$\vec{p}$的情况下,求出各关节变量 θ_i.即 $(p_x,p_y,p_z,\alpha,\beta,\gamma)\Rightarrow(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6)$。

态角和位姿向量的转化关系式同上，在文献[2]中给出了机器人的运动学逆解的解析解。只要输入手部位置距阵及机器人必要的结构参数。计算机便可以迅速地得出全部解。这样我们可以在开发的系统界面上显示图形的状态的同时输出相关联的数据。

(3)轨迹规划

轨迹规划是指根据作业任务的要求，确定轨迹参数并实时计算和生成运动轨迹。它是工业机器人控制的依据，所有控制的目的都在于精确实现所规划的运动。常用的轨迹规划方法分为:关节空间的轨迹规划和直角坐标空间的轨迹规划。关节空间进行轨迹规划,能保证机器人手爪通过路径点到达目标点。然而实际运动轨迹,很难给出空间曲线的规范表达式,所以只适用于点到点作业的轨迹规划。在本文中由于要实现焊接机器人的直线运动,所以对机器人的路径、姿态两者的瞬时变化规律都有严格的要求，这样我们就必须在直角坐标空间进行轨迹规划。

在直角坐标空间中进行轨迹规划，首要的问题是在起始点和终止点之间如何生成一系列中间点。显然空间的直线运动轨迹比较容易描述和生成因为只要给出轨迹起始点和终止点的位姿信息，就可以利用线性插补，简单地计算出由这两点确定的直线段上一系列点的位姿信息。然后再用求逆解的方法转化到关节空间求出各瞬时一系列的关节角。

2.2 图形的建模

在建立图形模块前,先要极小化 opengl。即：

1）接口设置，也就是在 link 中加入需要的库函数,如 opengl32.lib、glu32.lib 等库函数

2)设备环境的像素格式。它先告诉 OpenGL 的绘制风格,颜色模式,颜色位数等

```
    static PIXELFORMATDESCRIPTOR pfd=
    {
        sizeof(PIXELFORMATDESCRIPTOR), //pfd 告诉窗口我们所希望的对象
        1,                                  // 版本号
        PFD_DRAW_TO_WINDOW                  |// 格式必须支持窗口
PFD_SUPPORT_OPENGL|     // 格式必须支持 OpenGL
PFD_TYPE_RGBA|          // 申请 RGBA 格式
PFD_DOUBLEBUFFER,       // 必须支持双缓冲
        24,             // 选定色彩深度
        0,0,0,0,0,0,    // 忽略的色彩位
        0,              // 无 Alpha 缓存
        0,              // 忽略 Shift Bit
        0,              // 无聚集缓存
        0,0,0,0,        // 忽略聚集位
        32,             // 32 位 Z-缓存 (深度缓存)
        0,              // 无模板缓存
        0,              // 无辅助缓存
        PFD_MAIN_PLANE,     // 主绘图层
        0,                  //保留
        0,0,0               // 忽略层遮罩
    };
```

3)建立“翻译描述表”,并选定环境设备当前的翻译描述表。

```
void CRobotView::Initial()
{.....
n=::GetPixelFormat(m_pDC->GetSafeHdc());
    ::DescribePixelFormat(m_pDC->GetSafeHdc(),n,sizeof(pfd),&pfd);
//这里调用前面的设备环境格式
    hrc=wglCreateContext(m_pDC->GetSafeHdc());    //适合程序的设备上下文
    wglMakeCurrent(m_pDC->GetSafeHdc(),hrc);
GetClientRect(&m_oldRect);
    glClearDepth(1.0f);                 // 设置深度缓存
  glEnable(GL_DEPTH_TEST);              // 启用深度测试
.....
}
```

4)舞台的布置。也就是把景物放置在三维空间的适当位置,设置三维透视视觉体(OnSize)。

```
void CRobotView::OnSize(UINT nType, int cx, int cy)
{
CView::OnSize(nType, cx, cy);
    if(cy>0)
    {
if((m_oldRect.right>cx)||(m_oldRect.bottom>cy))
            RedrawWindow();
m_oldRect.right=cx;
        m_oldRect.bottom=cy;
if(cy==0) cy=1;
glViewport(0,0,cx,cy);                  //设置窗口大小
glMatrixMode(GL_PROJECTION);    // 选择投影矩阵
```

```
        glLoadIdentity();              // 重置投影矩阵
        gluPerspective(60,cx/cy,5.0,40.0);    // 计算窗口的外观比例
        glMatrixMode          (GL_MODELVIEW);
// 选择模型观察矩阵
        glLoadIdentity();              // 重置模型观察矩阵
    }
}
```

建立零件模型的基本构架是体、面、边、点。通过他们之间的循环构成整个物体的拓扑结构，同时赋予面的颜色和法线等重要信息。这样在光照的条件下，才能真实的展现三维物体，达到虚拟现实的目的。可以说颜色、法线、光照三者对真实再现物体模型来说缺一不可。现在可以建立机器人各零部件模型，确定出各零部件与相连接的关节点，便于组装机器人时进行定位。

```
void Cube (GLdouble length,GLdouble width,GLdouble heigth) //参数化设计方体
{
    GLdouble p[8][3],x,y,z;        //设置方形的八个顶点
    x=length/2;
    y=width/2;
    z=heigth/2;
.....
glPushMatrix();
        Polygon(p[0],p[1],p[2],p[3]);      //由点->线->面
        Polygon(p[3],p[2],p[4],p[5]);
    Polygon(p[5],p[4],p[6],p[7]);
        Polygon(p[7],p[6],p[1],p[0]);
    Polygon(p[0],p[3],p[5],p[7]);
    Polygon(p[6],p[4],p[2],p[1]);      //由六个面组成立方体
    glPopMatrix();
}
```

2.3 系统界面的设计

本仿真目的是为了实时地控制，观察机器人的运动过程，因此必须有图象显示和控制两部分。为此我们采用 MDI 结构，用 CsplitterWnd 类对象把窗口分为两部分，左边用于控制和输出，右边用于显示图象。由于有正反解和轨迹规划，这个相当于我们在一个程序中要做三个实验，互不干涉，为此我们在左视图采用了切换视图的功能，便于在各实验间转换，实现自动控制。同时又在工具栏添加手动控制按钮，样既能实现自动控制也能实现手动控制。图形界面如图 1 所示。

2.4 实时动画和交互技术

OpenGL 用双缓存技术实现动画，在双缓存模式下，位平面被分为前台位平面和后台位平面。只有前台位平面被显示，因为绘制函数绘制的场景首先被写向后台位平面。当完整的画面在后台视频缓存中画出以后，就调用 SwapBuffers()函数，成为可见视频缓存。

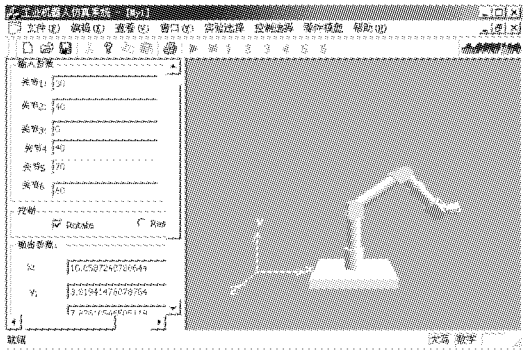

图 1 仿真系统界面

这样循环往复，隐藏了画图的过程，屏幕上显示的总是已经画好的图形，于是看起来所有的画面都是连续的。同时，建立时间和消息驱动的动画机制，很好的实现了动态仿真的目的。流程如表 2 所示。

表 2 动态仿真演示的结构化流程

开始(选择要进行的试验)
在视图左边输入参数
在 OnRotate ()成员函数中响应“进行动态仿真”命令，设置视窗类的成员函数和成员变量：速度快慢选择函数
暂停函数，运动停止标志
创建系统定时器，每事 60ms 发送定时消息 WM_TIMER，引发 OnTimer()成员函数
在 OnTimer()成员函数中处理 WM_TIMER 消息，计算该时该末端操作手的位置，进行动态模拟演示
结束

3 结束语

下一步工作是，在以上的基础上对机器人动力学进行仿真，并对运动精度、误差进行分析。

参考文献

1 徐元昌遍著.工业机器人.中国轻工业出版社，1997
2 刘成良，张为功，等.RV12L 6R 机器人运动学及计算机仿真系统研究.机器人，1998
3 向世明编著.精通 OpenGL 图象编程.电子出版社，1999(9)
4 侯俊杰编著.MFC 深入浅出.华中科技大学出版社，2001(2)
5 潘爱民，王国印译.Visual C++技术内幕.清华大学出版社，1999(1)
6 陈哲，吉熙章编著.机器人技术基础.机械工业出版社，1997(10)

[收稿日期：2002.12.10]

德州仪器推出经济的 DSP 开发工具包

日前，德州仪器公司(TI)推出两款用于 TMS320F2812 DSP 控制器的新型开发工具捆绑包和一款用于 TMS320LF2407 DSP 控制器的升级评估模块(EVM)，使嵌入式控制开发人员能够迅速开始 TMS320C24x 与 C28x DSP 控制器系列的开发工作。两款捆绑包预计将于六月初开始供货。