云南大学学报 (自然科学版), 2006, 28 (5): 398~403 CN 53-1045/N ISSN 0258-7971
Journal of Yunnan University

基于 OpenGL 的关节型机器人实时控制与仿真系统的研究*

钱 谦, 裴以建, 余 江, 宗 容
(云南大学 信息学院, 云南 昆明 650091)

摘要: 以在 Visual C++ 环境下利用 OpenGL 开发的机械手实时控制三维可视化仿真平台为实例, 详细介绍了建立机械手三维模型的数学模型和实现的算法, 该平台的软件设计使用了模拟人手的人机交互界面, 而且在软件功能上实现了其他类似平台所没有的实时抓取物体的功能, 具有较好的可扩展性和交互性.

关键词: 关节型机器人; 计算机仿真; OpenGL; 三维坐标变换; 实时控制
中图分类号: TP 391 文献标识码: A 文章编号: 0258-7971(2006)05-0398-06

机器人是最具代表性的光电一体化技术系统. 现代机器人技术经过数十年的发展, 在科学技术, 以至整个社会领域内正发挥着越来越重要的作用, 21 世纪无疑将是一个更为广泛和深入地开发和应用机器人技术的时代. 在所有类型的机器人中, 关节型机器人模拟动物和人类肢体, 甚至是整体而最具有仿生性, 也就具有了优于其他类型机器人的动作能力, 受到最大程度的关注. 现在在各种领域, 如工业生产和特种作业等, 关节型机器人都发挥着巨大作用.

利用仿真技术来实现机械手的建模, 可以直观地显示机械手的模型和实际操纵过程, 为实物的制作及安装环境提供一切必要的准备, 减少投资和风险. 本文就是研究类似人的手臂的关节型机器人(即机械手) 的实时控制三维仿真. 以前的机械手仿真研究大多停留在建立模型和关节的变换上[1,2], 本文不但介绍了模型的建立和关节的变换, 借助 VC 和 OpenGL 相结合的编程环境对模型的人机交互进行实时控制, 使机械手能在遥感控制装置的实时控制下抓取物体(如小球), 真实地反映其在实际应用中的表现.

1 机器人动态模型的建立及位姿分析

1.1 坐标变换的数学基础 在空间中, 要确定一个物体的几何状态需要确定其 3 个位移坐标(或称位置自由度) 和 3 个旋转坐标(或称姿态自由度), 即 x, y, z 轴上的相对移动和相对旋转. 在机器人学术语中, 将一个空间物体的上述 6 个自由度状态称为该物体的位姿[3].

从运动几何学的角度, 可以将多关节非移动型机器人或机械手看成是其一端与基础固接的一系列具有空间运动能力的刚体的连接组合. 这种组合通常要在每一个关节上建立运动坐标系, 随同关节运动, 这些运动坐标系是相对坐标系. 关节的转动轴线通常也是运动坐标系的一个坐标轴, 坐标的原点一般选在关节转动轴线上的一个便于识别的特征点上, 以方便计算.

机器人手部的空间位置和姿态(简称位姿) 可以借建立一个固接在手部上的坐标系(通常简称为手坐标系) 来描述. 手坐标系是相对坐标系, 随手部运动移动和旋转. 此外, 为操控机器人, 还需要将其任意时刻的位姿状态变换到建立在基础上的固定

* 收稿日期: 2006-01-12
基金项目: 云南省自然科学基金资助项目(2004F0010M); 云南大学重点资助项目(2003Z009B).
作者简介: 钱 谦(1981-), 男, 硕士生, 主要从事实时机器人仿真方面的研究.
通讯作者: 裴以建(1959-), 男, 教授, 主要从事机器人、智能科学方面的研究.

坐标系(通常简称为基坐标系)来描述. 基坐标系是绝对坐标系, 而且对机械手手部运动的控制也要以基坐标系为参照.

由于手部和基座之间存在着空间距离和方位差别, 因此就需要运用一种数学方法, 可以方便地将手坐标系所描述的手部位姿转换为由基坐标系进行描述, 或是相反. 而且, 实际上由于机器人上的每一个关节都位于手部和基座之间, 其工作状态也必然存在着类似的在不同坐标系之间进行转换的问题.

要从一个坐标系变换到另一个坐标系, 需经一个位移变换和三个旋转变换, 这些变换共同组成坐标系之间的变换矩阵. 首先, 将基坐标系原点位移到新坐标系原点位置, 变换矩阵为

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix}, \tag{1}$$

其中(T_x, T_y, T_z)为新坐标系的原点在基坐标系中的位置.

然后以 x 轴为中线旋转 θ_x 角, 变换矩阵为

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2}$$

再以 y 轴为中线旋转 θ_y 角, 变换矩阵为

$$\begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3}$$

最后以 z 轴为中线旋转 θ_z 角, 变换矩阵为

$$\begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4}$$

其中 θ_x, θ_y, θ_z 为不相关的 3 个角度.

从上文中可知空间直角坐标系中任一点 P 的位姿可用行矢量$P(x, y, z)$表示, 在本文中用齐次坐标表示法时, 表示为 $P(x, y, z, 1)$, 其中的 1 为比例因子, 也可写成 $P(k \times x, k \times y, k \times z, k)$.

本文中要求基坐标系中一点 $P(x, y, z, 1)$ 在新坐标系中的位姿, 应用了行矩阵$(x\ y\ z\ 1)$依次矩阵乘上上文的变换矩阵, 即可得新坐标系下的位姿. 相反, 已知新坐标系下 P 的位姿求基坐标系下的位姿, 只需乘变换矩阵的逆矩阵.

1.2　关节型机器人的位姿分析　本文中描述一个关节和下一个关节间相对关系的齐次矩阵通常记为 A. A 矩阵描述了关节坐系间相对平移和旋转的齐次变换, 其值等于上文各个齐次变换矩阵的乘积. 如果 A_1 表示第 1 个关节对于基坐标系的位姿, A_2 表示第 2 个关节相对于第 1 个关节的位姿, 那么第 2 个关节在基系中的位姿可由下列矩阵的乘积给出

$$T_2 = A_1 \times A_2 \tag{5}$$

同理, 若 $A3$ 表示第 3 个关节相对第 2 个关节的位姿, 则有

$$T_3 = A_1 \times A_2 \times A_3. \tag{6}$$

于是, 一个以基坐标系为参照的 6 关节(6 自由度)机器人从手部到基系的总齐次变换矩阵 T 为

$$T_6 = A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6, \tag{7}$$

n 关节机器人从手部到基系的总齐次变换矩阵 T 为

$$T_n = A_1 \times A_2 \times A_3 \cdots \times A_n. \tag{8}$$

因此不论有几个关节, 要求基坐标系中一点 $P(x, y, z, 1)$ 在手部坐标系中的位姿, 只需用行矩阵$(x\ y\ z\ 1)$矩阵乘上 T. 相反, 已知手部坐标系下 P 的位姿求基坐标系下的位姿, 只需矩阵乘 T 的逆矩阵.

实际关节型机器人一般一个关节上只有一个自由度, 但为简便起见, 在仿真系统中我们设计为一个关节上可有多个自由度, 比如上臂部关节就有左右方向和上下方向上共 2 个自由度. 这种设计也更符合本系统模拟人手臂的初衷, 更易于遥感的实时控制.

1.3　研究过程中的坐标变换实例

1.3.1　*模型的建立与位姿的变换*　机器手的位姿由 4×4 矩阵来表示, 建立手部模型首先要从机器手的肩部关节到手指末端, 逐一对机器手各个关节建立自己的坐标系, 每个坐标系与其对应的关节固连在一起, 随关节一起动作. 然后从基础坐标系沿着各个关节依次进行坐标变换, 得到机器手各个关节相对于基础坐标系的位置关系[4].

由于 OpenGL 本身对复杂物体建模支持并不好, 本文用其他辅助软件制作了三维图元函数, 并在此基础上进行简单建模.

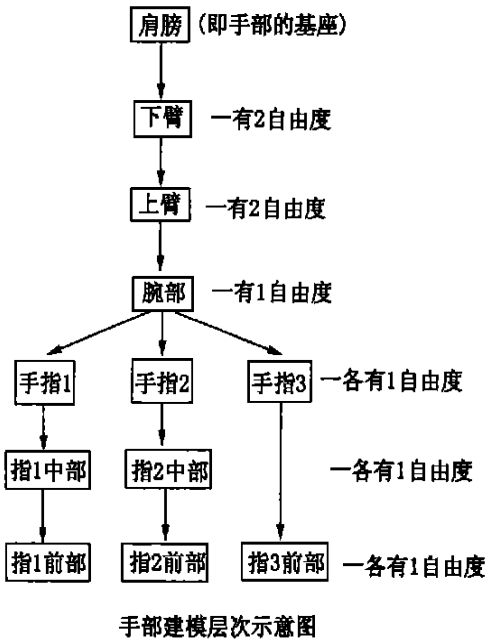

图 **1** 手部的结构

Fig. 1 The structure of hand model

1.3.2 碰撞检测和被抓取物体(小球)的变换 作为本文的关键技术之一,采用包围指尖的 3 个圆球与小球的距离来检测手部是否碰到小球,以及是否抓住了小球. 小球被抓后可随手部各关节的运动而运动,并始终保持在被 3 个手指抓住的相对位姿,还可以由一只手换到另一只手.

为了检测指尖包围球与小球的距离,首先必须使指尖包围球和小球的球心坐标在同一坐标系中,这里本文统一用最终视图的坐标系来计算,这就要先将指尖球心相对于指部坐标系的位姿变换为视图坐标系的位姿,并且这个工作必须随着手部参数的变化实时完成. 为此本文在 Hand 类中定义了 tip[3]和 joint[3]来保存手部各关节的相对于手部基坐标系的位姿数据,在 Draw() 函数中调用 UpdateTipAndJoint() 方法来进行转换,然后在 View 类中的 UpdateBallPos() 方法中完成到最终视图坐标系的转换.

表 **1** 手部各关节参数

Tab. 1 The parameters of joints

关节	关节角度变量及初始值	变量范围	相对上一关节的位移	自由度
肩膀				
下臂	shoulder$x$(0), shoulder$y$(0)	-90~90, -75~75	从肩膀的 0 位置开始 同上	2,分别沿基坐标系的 y 和 z 轴
上臂	elbow$y$(0) elbow$R$(0)	-90~90, 0~360	HandPos_elbow 同上	2,分别沿上一坐标系的 x 和 y 轴
腕部	wrist$x$(-45)	-90~15	HandBos_wrist	1,沿上一坐标系的 z 轴
手指	finger1(45), finger2(45), finger3(45)	0~60 0~60 0~60	HandPos_finger 同上 同上	1,沿上一坐标系的 z 轴
指中部	subfinger1(-30), subfinger2(-30),	-10~-60 -10~-60	HandBos_subfinger 同上	1,沿手指坐标系的 z 轴
指前部	-subfinger3(-15) subfinger11(-15) subfinger22(-15)	-10~-60 -10~-30 -10~-30	Handpos_subfinger HandBos_subfinger2 由上	1,沿指中部坐标系的 z 轴,不过大姆指(即第 3 指)绕 z 轴转了 45°

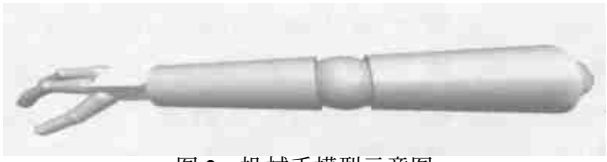

图 **2**　机械手模型示意图

Fig. 2 The image of hand model

得到了各个关节和小球的实际位姿, 就可以进行碰撞检测, 实际代码在 UpdateBallPos( ) 方法中, 以下是算法的描述:

第 1 步: 分别计算小球球心坐标和 3 个指尖包围球坐标的距离 d, 如果 d 大于小球半径与包围球半径之和, 则设置 hand 对象相应状态变量 collflag[i] (表明第 i 个手指是否碰到球) 为真, 其中 i 为 0, 1 或 2.

第 2 步: 判断如果某只手的 collflag[0] , collflag [1] , collflag[2] 都为真, 即 3 根手指都碰到小球时, 计算指部大拇指与第 3 指的距离, 如果距离大于设定值(手指距离过小时不能抓取小球, 因为抓不稳) , 则说明此时小球确实被抓住了, 设置 hand 对象相应状态变量 fastenflag(手是否抓到球) 为真.

第 3 步: 判断如果左手的 fastenflag 和右手的 fastenflag 都为真, 说明两手都抓住小球, 则此时必须先保存小球此时位姿, 再将左右手的可移动标志变量 moveflag(手是否可移动) 设为假, 否则设置 moveflag 为真. 这里使小球同时被两只手抓住时保持不动, 实现了小球由一只手交换到另一只手的功能.

第 4 步: 判断如果 hand 对象的 fastenflag 为真, 则首先保存小球此时位姿, 然后在 Hand 对象中记录球被抓后手部各关节的转角 RotIncrement [i] (球被抓后各个关节的增量) , 对于每一个转角利用前述位姿变化方法计算小球的新位姿, 其中 i 为 0, 1, 2, 3 和 4, 共 5 个关节.

第 5 步: 判断如果左手的 fastenflag 和右手的 fastenflag 都为假, 说明小球未被抓, 此时如果小球不位于地面, 需改变小球位姿使小球做自由落体运动, 直至落到地面.

2　实时控制与仿真应用实例

2.1　程序结构　本程序采用 VC+ + 环境下的 OpenGL 图形库来实现, 利用 MFC 单文档程序框架来构建程序外观和框架, 采用面向对象的编程技术[6].

2.2　控制　实例利用了 Windows 下的外设控制函数来控制, 控制设备采用了 USB 接口的遥感手柄 2 台, 力求做到真实的反映手部运动, 使操纵者就像使用自己的手一样方便.

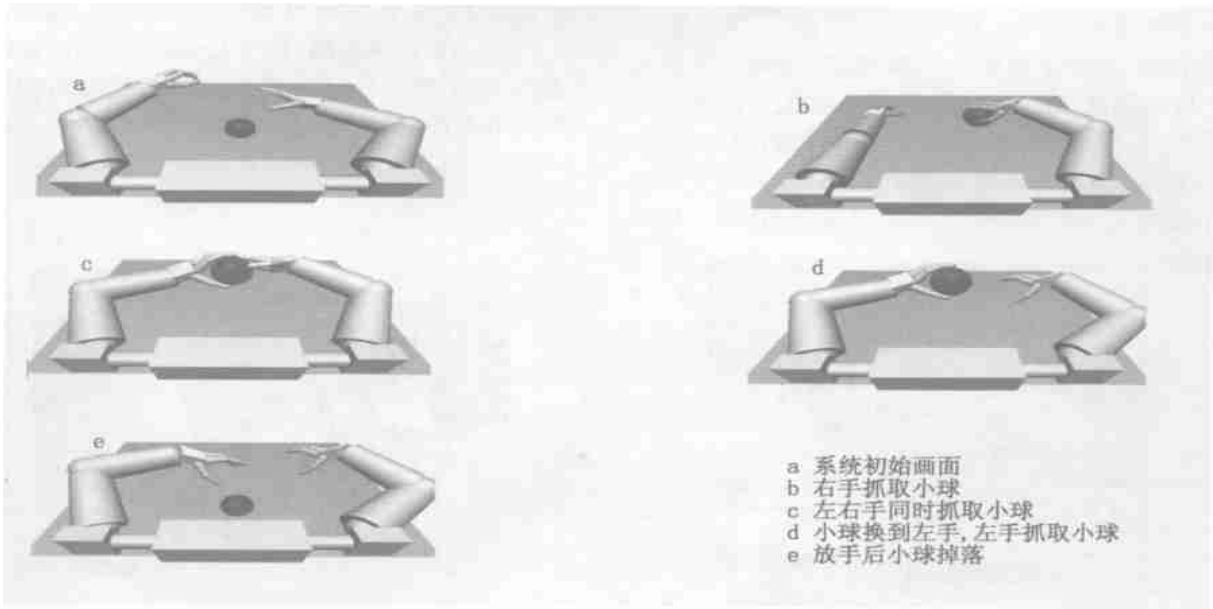

图 **3**　程序运行图

Fig. 3 The image of running system

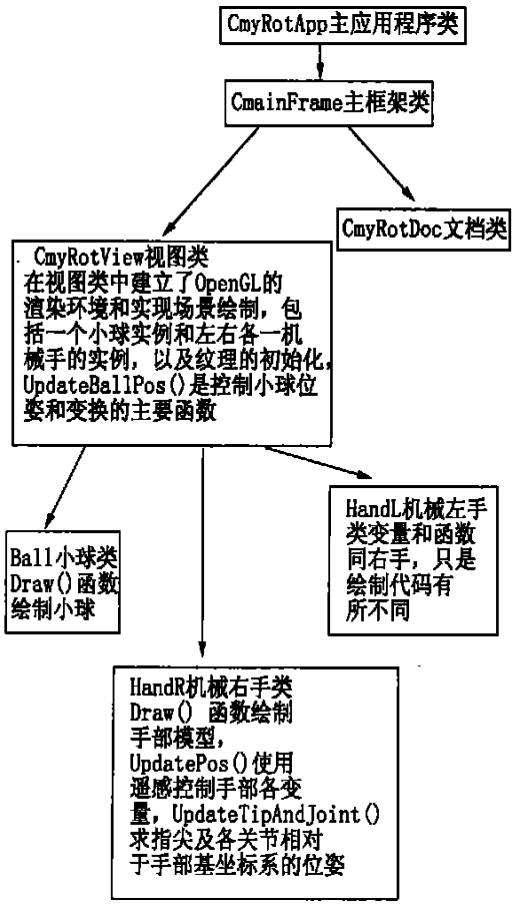

图 **4** 程序的类及主要函数的关系

Fig. 4 The combination of classes and it's functions

在本文的研究实例中使用了 joyGetPos 平台 SDK 函数, 主十字键控制肩部变量, 微调十字键控制上臂和腕部, 2, 3 键控制手部抓与放, 4 键控制上臂旋转. 为使手部抓住小球时、手部碰到地面时、以及两手都抓住小球时限制手部各关节的动作, 在其中还使用了一些标志性布尔变量来协助控制. 另外还将手部位姿变换数值保存到一些变量中用于小球位姿的计算. 在显示每一帧时, 即每次调用视图类 RenderScene( ) 函数时, 更新各个变量的值, 计算小球位置并更新视图区, 利用 OpenGL 双缓存形成动画.

实际控制代码在 hand 类的 UpdatePos( ) 中, 以下为控制部分的算法描述:

第 1 步: 根据手柄状态参数的数值设置 hand 对象的控制步长 m _ speed , 即每次调用时关节变量的改变大小, 可控制机械手的运动速度.

第 2 步: 判断如果 moveflag(是否可移动) 为真且 fastenflag(手是否抓住球) 为真, 或 moveflag 为真且 fastenflag 为假且 collflag [0]、collflag [1]、collflag[2] (手指是否碰到球) 都为假时, 根据手柄状态变量数值改变机械手下臂 shoulder$X$ 下方向和 shoulder Y、上臂 elbow$R$ 和 elbow Y、腕部 wrist$X$ 变量的值, 其中 shoulder$X$ 下方向和 wrist$X$ 的改变还必须判断 coll _ wall(手是否碰到地面) 标志, 使机械手不会穿过地面. 各个关节变量改变时如果 fastenflag 为真, 还必须将变化量加到 RotIncrement[0~ 4] (球被抓后各个关节的增量) 中, 以便于小球被抓时计算小球的位姿.

第 3 步: 根据手柄状态变量改变机械手下臂 shoulderX 上方向和手指的变化数值, 其中同一根手指上的各个关节变量, 如 finger1, subfinger1, subfinger11 是联动的, 而 3 根手指的变化又可一同进行, 故手指部分只需开合 2 个按键即可控制, 此外按下合按键时还必须判断各个手指的 collflag[0 ~ 2] 值, 为真时表明已碰到小球, 关节不再变化.

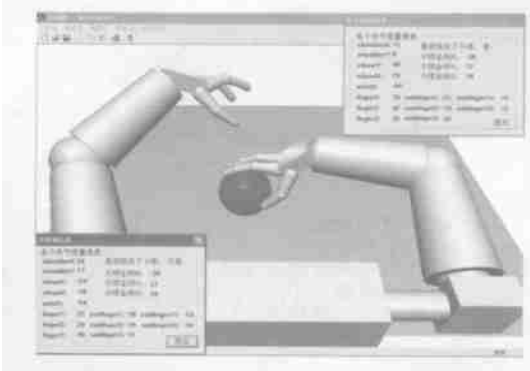

图 **5** 小球随左手移动和左右手的控制信息显示

Fig. 5 The ball moved with left hand and the controlled information of hands

图 **6** 实际操作的照片

Fig. 6 The image of real manipulation

3　结束语

本文介绍了在 Visual C + + 环境下利用 OpenGL 开发的关节型机器人三维可视化控制与仿真平台, 该平台的软件设计使用了模拟人手的人机交互界面, 而且在软件功能上实现了其他类似平台所没有的实时抓取物体的功能, 具有良好的可扩展性和交互性. 在此基础上, 可以进行关节型机器人运动学、动力学、控制和规划等方面的可视化控制与仿真研究. 由于 OpenGL 功能所限, 机械手的模型略显简陋, 下一步将使用 3Dmax 等建模工具进一步建立更精细的模型, 并尝试实现除小球外其他物体的抓取功能.

参考文献:

[1] 张秋豪, 孙汉旭, 李旭. 应用 OpenGL 建立机器人可视化仿真系统[J]. 机械与电子, 2004(1): 56-58.

[2] 翟雪琴, 郝矿荣, 曹自洋. 基于 OpenGL 的工业机器人动力学仿真的研究[J]. 机床与液压, 2004, 7: 14-16.

[3] 殷际英, 何广平. 关节型机器人[M]. 北京: 化学工业出版社, 2003.

[4] Edward Angel. OpenGL 程序设计指南[M]. 第 2 版. 李桂琼, 张文祥, 译. 北京: 清华大学出版社, 2005.

[5] 李晓燕, 张翔, 陈立伟. 基于 vc 6.0 和 opengl 机械手三维仿真演示系统[J]. 计算机工程与设计, 2004, 06: 982-984.

[6] 马志强, 陈一民, 汪地. 面向对象的机器人仿真与监控系统[J]. 计算机工程与设计, 2005(1): 75-78.

Real-time control and simulation system based on the OpenGL joint robot

QIAN Qian, PEI Yi-jian, YU Jiang, ZONG Rong

(College of Information, Yunnan University, Kunming 650091, China)

Abstract: We consider the instance of the 3-dimensional visual simulation platform controlled by the real-time hand model, which is developed by visual C+ + and OponGL. To construct the 3-dimersional model of hands, the mathematical method and the corresponding algorithm are introduced in datail. The software design of this platform makes use of the interface simulating real human hands. As well , the function of real-time scraching has been implemented, compared to other similar systems. This platform is some extensible and friendly.

Key words: joint based robot; computer imitation; OpenGL; translation of 3D coordinates; real time control