第27卷 第3期
2001年5月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 27, No. 3
May, 2001

短文

# CMRC——多 DSP 并行结构机器人控制器的研究[1)]

胡 健 谭 民

(中国科学院自动化研究所 北京 100080)

**摘 要** 用并行结构实现多机器人协调系统的控制任务是多机器人协调系统研究的重要内容之一. 文中提出了一种新型多 DSP 并行结构机器人控制器的思想,阐述了系统的硬、软件结构,并对系统的主要性能进行了分析,最后给出了在码头多机器人协调工作中的例子分析.

**关键词** 多机器人协调,机器人控制器,数字信号处理器,并行结构.

# CMRC——RESEARCH ON ROBOT CONTROLLER WITH MULTI-DSP PARALLEL ARCHITECTURE

HU Jian TAN Min

(*Institute of Automation, Chinese Academy of Sciences, Beijing* 100080)

**Abstract** The paper presents a new idea for a robot controller of multi-DSP parallel architecture. We first illustrate the architecture, then analyze the main capabilities, and finally provide a detail explanation of how the controller works when used in a multi-robot coordination system on the dock.

***Key words*** *Multi-robot coordination, robot controller, digital signal processor, parallel architecture.*

## 1 引言

目前,多机器人群体协调成为机器人研究的新领域,它需要智能机器人、自动控制理论、通讯技术、计算机科学以及传感器技术和传感信息融合等多方面理论技术的支持,对机器人控制系统的任务显著增加. 1982年 J. Y. S. Luh[1]首次提出机器人动力学并行处理问题,解决了关节机器人动力学方程为一组非线性强耦合的二阶微分方程的复杂计算问题,从而推动了控制器结构并行化的发展. 到20世纪90年代中期,并行系统用于开发机器人控制器一直得到了普遍的重视[2~6].

本文正是立足于这种思想,提出 CMRC(The Controller for Multi-Robot Coordination)系统,它着眼于多机器人协调系统控制中许多任务有较强的并行性,希望通过 CM-

1) "八六三"高技术自动化领域智能机器人主题资助项目.

收稿日期 1999-04-14 收修改稿日期 1999-09-14

RC 并行结构的特点达到较好的控制性能，并针对码头上多机器人协调运送货物的具体例子给出了在 CMRC 上协调算法的实现.

## 2 CMRC 体系结构

多机器人协调系统的任务量大、操作复杂. 这样就对控制器的实时性、可靠性等指标提出了更高的要求. 如图1所示，CMRC 采用双总线(PCI)结构，主机(Pentium-Ⅱ)、共享内存、多 DSP(TMS320C40)、电机驱动单元、外界环境传感器和通讯端口分别与相对的总线相连.

因为任务种类繁多，所以对于 CMRC 系统存在着任务队列优化的问题. 下面介绍任务块链表和重要度两个概念. 如图2所示，其中的 $J_i$ 值表示第 $i$ 项任务块在整个系统中的重要程度，称为第 $i$ 项任务块重要度. 任务块链表是根据各任务块重要度进行排序和动态刷新的.

$$J_i = f(x_i \quad y_i \quad t_i), \tag{1}$$

式中 $x_i$ 表示第 $i$ 项任务块的属性，$y_i$ 是第 $i$ 项任务块中所涉及的机器人的重要性. 本文认为在多机器人协调系统中因各机器人承担的工作不一样，其相应的重要性也有差异. $t_i$ 是任务块列入链表中到目前还未被执行时已等待的时间，有防止重要度低的任务长期停放在链表中和其它一些特殊的用途. $t_i$ 取任务块链表的刷新次数，也就是每次刷新后在各任务块 $t_i$ 域加1. 关于方程(1)的建立，本文在针对码头多机器人协调运送货物系统的具体例子中进行了讨论.

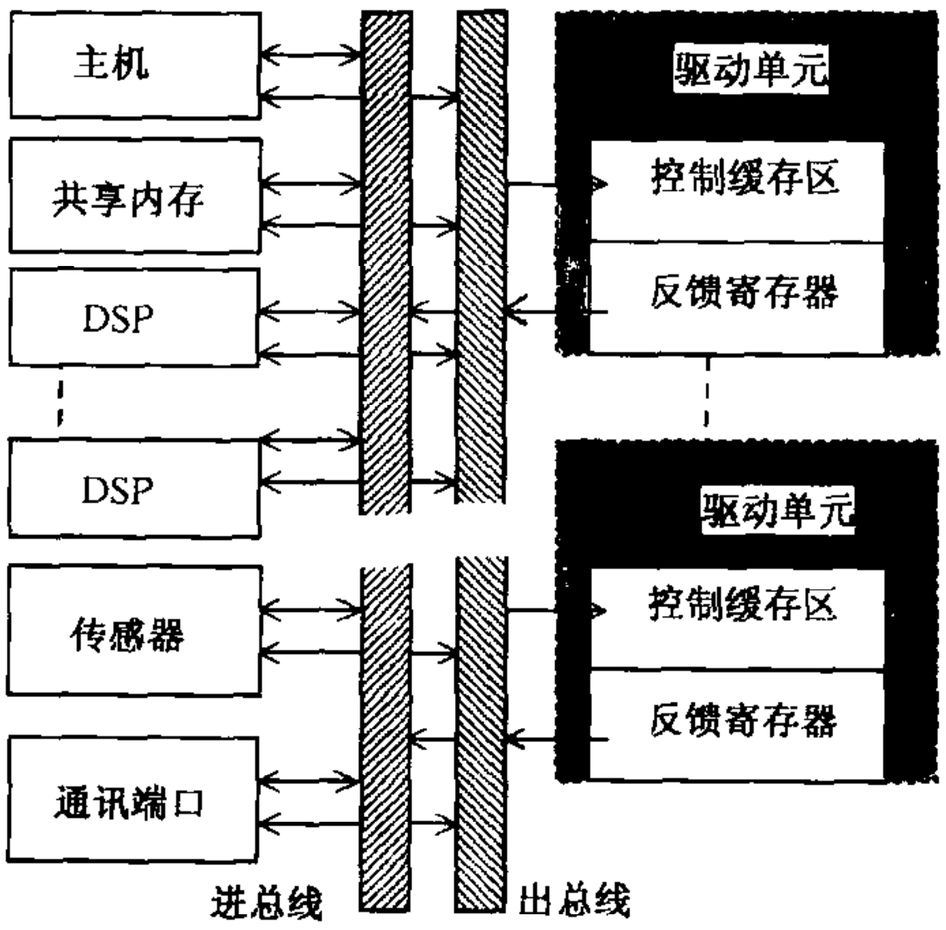



图1 CMRC 系统结构

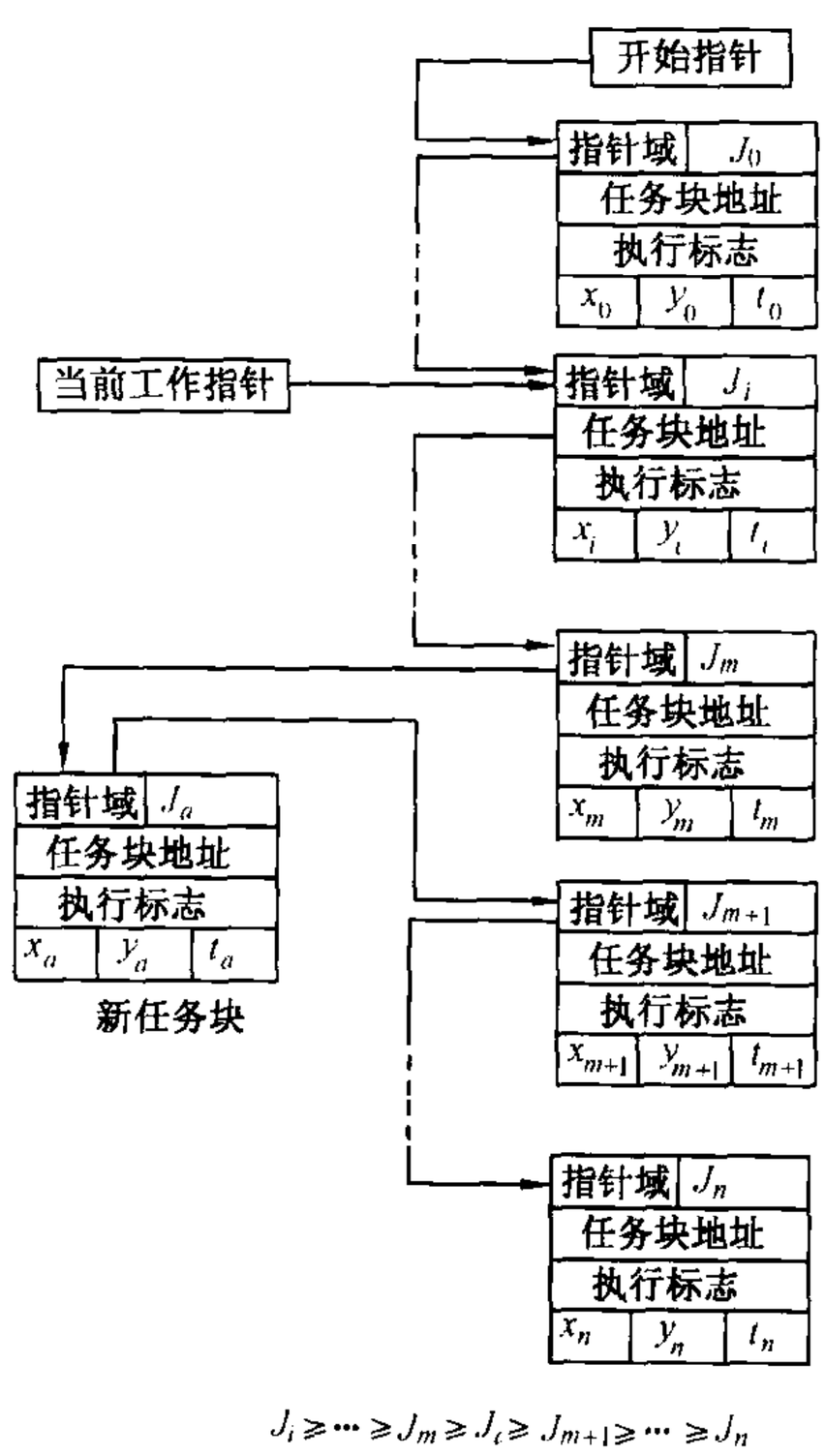



图2 任务块链表

## 3 系统的性能

1)开放性. CMRC 操作系统的软件结构可以归结为硬件、内核、应用层和人机交互界

面,其中操作内核支持系统的通讯和 DSP 的运算,系统作业层提供主机对系统的管理和多机器人协调的各种作业模块. 主机提供人机交互界面,为应用者提供修改影响系统动态特性的控制参数的入口,另一方面又用函数库支持用户开发自己的应用程序,在系统作业层完全向用户开放.

2)实时性. 在 CMRC 中 DSP 作为 CPU 的主动性被强调,即它可以在完成任务后自己查询下一个任务,脱离主机的控制. 所以 DSP 可以灵活地在线插接于系统中. 基于此特性,可以根据 CMRC 系统需要完成的具体任务和工作量通过简单地增加 DSP 的数量,很好地达到实时的要求.

3)可靠性. DSP 取出当前任务块,并把图2中此任务块的执行标志从初始值"0"置为"0Fh",$t_i$ 置为"0",表示此任务块正在执行,但没有结束. 任务块的最后工作是置执行标志为"FFh". 主机在刷新任务块链表时发现执行标志为"FFh"的任务块,立即从链表中清除,但是发现在当前工作指针之前的链表任务块 $t_i$ 的数值已大于一定的数值,就认为执行此任务块的 DSP 损坏或系统进入死循环. 系统将对此任务块重新计算它的重要度,然后再次放回当前的工作指针之后的任务块链表中,等待处理. 有以上的特点,DSP 就可以在任务执行中在线插拔,所以系统结构相对于 DSP 是冗余的,可靠性和可维修性有非常大的提高.

## 4　举例分析——码头运货多机器人协调工作

考虑码头上运送货物的工作情况,当集装箱货船停靠于码头后,由吊车把货物吊装到运输车上,再运送到仓库存放. 本文中考虑用一个吊装机器人和自动小车的协调系统来模拟码头的工作情况. 图3中表示了整个系统的协调工作.

**定义.**

进程1. 识别判断(判断工作是否完成. 如果已完成,发出工作结束信息,通知主机同意装入进程7和进程8;如果没有完成,获取货物终端位置,通知主机必须装入进程2).

进程2. 抓取货物任务规划(优化空间路径,得出系列取道点,通知主机同意装入进程3).

进程3. 抓取货物操作(子进程1;子进程2;子进程3;子进程4;通知主机必须装入进程1和同意装入进程4).

进程4. 装载(子进程3;子进程4;通知主机同意装入进程7,同意装入进程3和必须装入进程5).

进程5. 送回仓库(子进程1;子进程2;子进程5;通知主机同意装入进程8和必须装入进程6).

进程6. 开向码头(子进程1;子进程2;子进程5;通知主机同意装入进

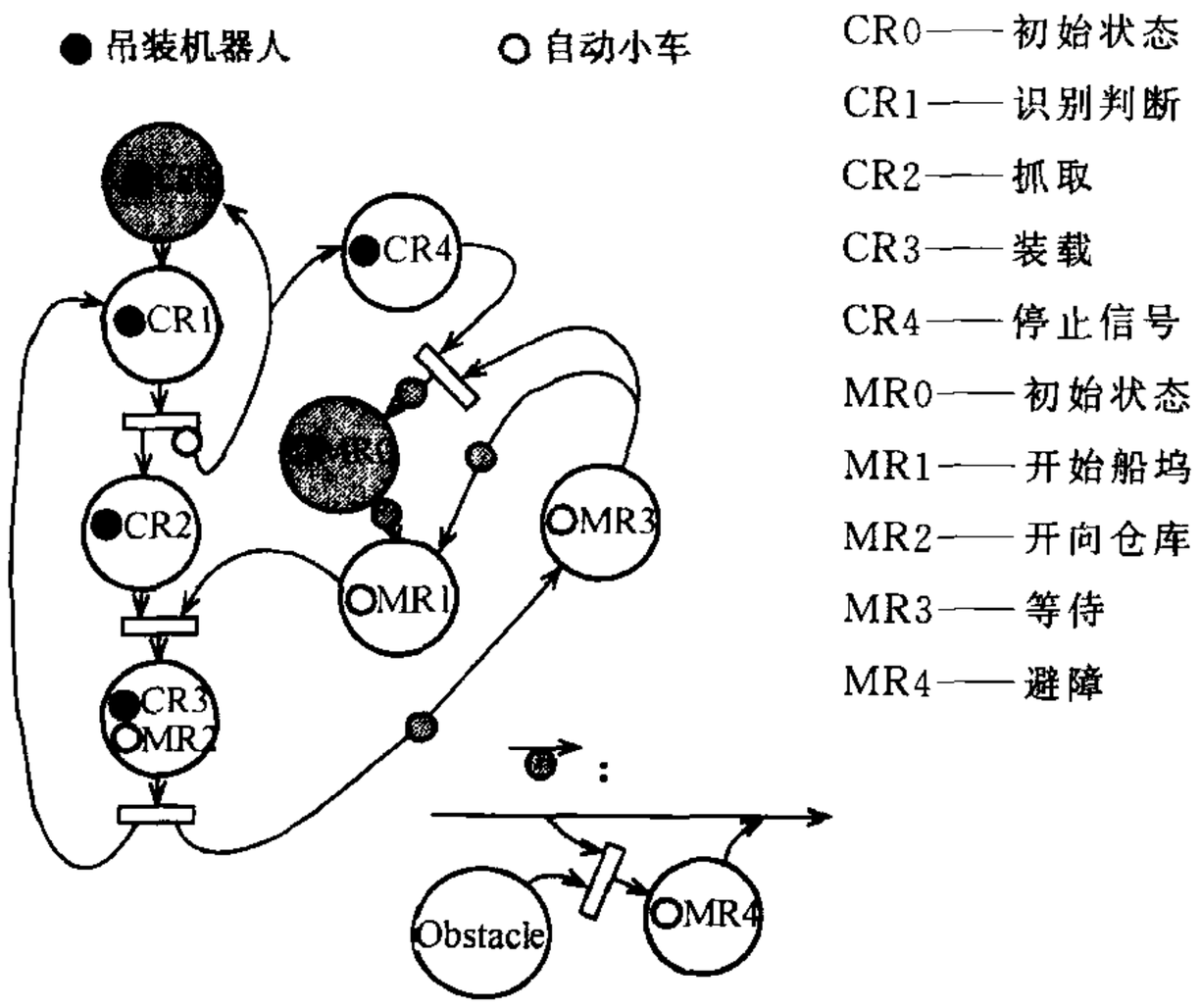



图3　船坞多机器人协调模型

程4).

进程7. 吊装机器人恢复初始状态.

进程8. 自动小车恢复初始状态.

子进程1. 轨迹规划(通知主机必须装入子进程1和2).

子进程2. 逆运动学求解.

子进程3. 力矩补偿.

子进程4. PID 控制.

子进程5. 避障.

在执行顺序上进程1,2,4之间和进程3,5,6之间存在并行实施关系. 根据 CMRC 的并行结构,所以 CMRC 在装载和运行进程时有图4(a)的结构. 码头多机器人协调系统的优化指标是以最少的时间完成货物搬运的工作. 当两种机器人满负荷工作(物理上最大允许的工作条件)时,完成一次循环分别所需要的时间为

1)吊装机器人

$$T_g = t_3 + t_4 + t_v, \tag{2}$$

2)自动小车

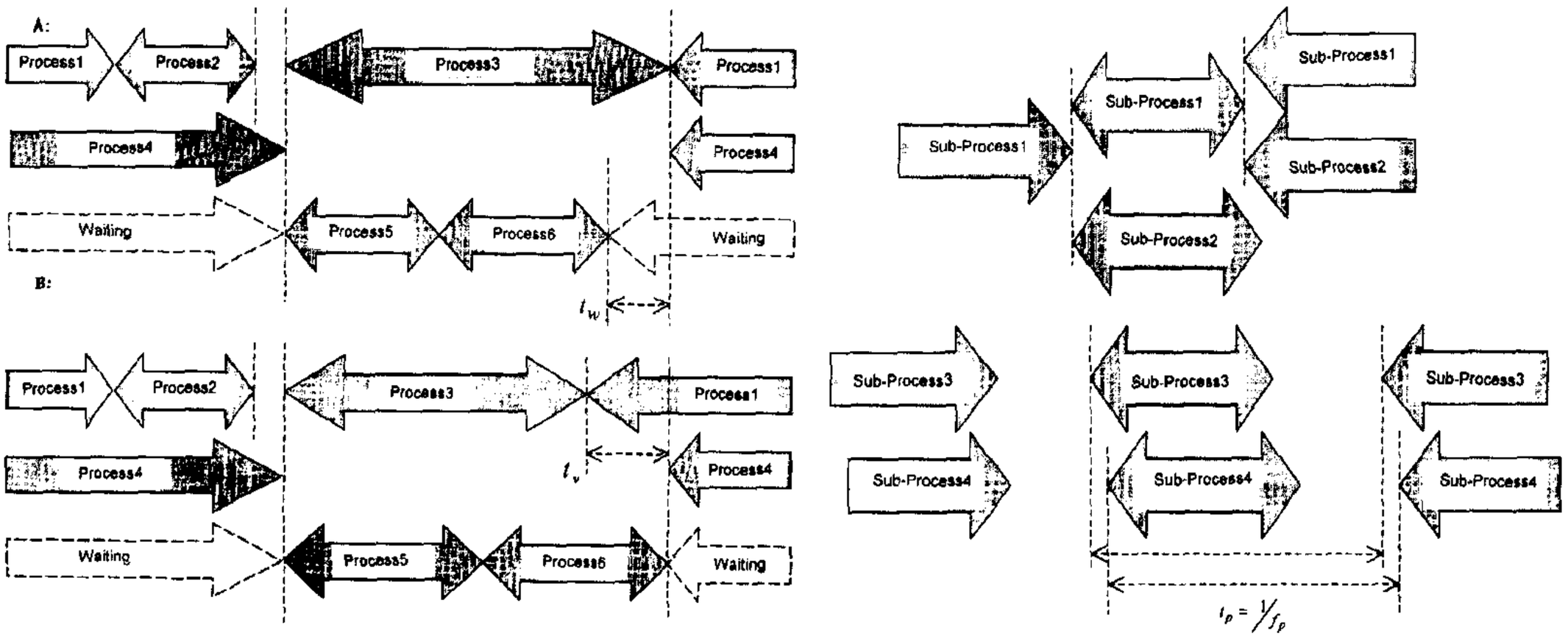
$$T_a = t_5 + t_6 + t_{\text{wait}}, \tag{3}$$

$t_i(i=3,4,5,6)$是第 $i$ 个进程所花费的时间. 方程(2),(3)有如下的关系

$$t_{\text{wait}} = t_4 + t_w, \quad t_w \geqslant 0, \quad t_v \geqslant 0. \tag{4}$$

A)$t_w>0$时,说明吊装机器人的工作较慢,可以采取的策略是降低自动小车的行驶速度,CMRC 用更多的时间处理吊装机器人的任务;

B)$t_v>0$时,说明自动小车的工作较慢,可以采取的策略是放慢吊装机器人在进程3中的速度.




(a)进程调度 (b)子进程调度

图4

进程3,4,5和6都分别需要调用子进程. 各子进程在 CMRC 中执行时也同样有并行的结构,如图4(b)所示. 从机器人控制学可知,子进程3和4必须按照一定的频率被执行;子进程1和2必须在一定的时间范围内被执行,也就是说它们必须及时地为子进程3和4提供下一个插值点. 降低机器人运行速度直接表现为在进程5,6或进程3中所包含的子进程1和

2的处理时间间隔可以延长.设系统对任务块链表的刷新频率为 $f_s$,当吊装机器人满负荷工作时,子进程3和4所要求的工作频率为 $f_p$,子进程1和2的工作频率为 $f_t$,自动小车以最快速度行进时,它的子进程1和2的工作频率为 $f_a$.取

$$N_p = \min[f_s/f_p], \quad N_t = \min[f_s/f_t], \quad N_a = \min[f_s/f_a]. \tag{5}$$

考虑到若出现 $B$ 类情况,通过增加自动小车的数量可以很容易地解决问题.所以这里着重讨论 $A$ 类情况,就是吊装机器人工作慢.如前所述,可以减缓自动小车的行进速度.假设方程(4)中 $t_w=0$时自动小车子进程1和2的工作频率为 $\hat{f}_a$,则

$$\hat{N}_a = \min[f_s/\hat{f}_a]. \tag{6}$$

从以上的讨论以及各进程和子进程之间的执行顺序,可以得出方程(1)的具体表达式如下:

$$J_i = \begin{cases} x_i = 1 & \text{FFFEh}, \\ x_i = 2 & \text{FFFDh}, \\ x_i = 5\begin{cases} y_i = 1\begin{cases} t_i \geqslant N_t - \alpha & \text{FFDFh}, \\ t_i < N_t - \alpha & p, \end{cases} \\ y_i = 2\begin{cases} t_i \geqslant \hat{N}_a - \alpha & \text{FF7Fh}, \\ t_i < \hat{N}_a - \alpha & q + at_i, \end{cases} \end{cases} \\ x_i = 6\begin{cases} y_i = 1\begin{cases} t_i \geqslant N_t - \beta & \text{FFBFh}, \\ t_i < N_t - \beta & p + b \end{cases} \\ y_i = 2\begin{cases} t_i \geqslant \hat{N}_a - \beta & \text{FEFFh}, \\ t_i < \hat{N}_a - \beta & q + b + at_i, \end{cases} \end{cases} \\ x_i = 7\begin{cases} t_i \geqslant N_p - \gamma & \text{FFF7h}, \\ t_i < N_p - \gamma & 0, \end{cases} \\ x_i = 8\begin{cases} t_i \geqslant N_p - \gamma & \text{FFEFh}, \\ t_i < N_p - \gamma & 0, \end{cases} \\ x_i = 11 & \text{FDFFh}, \\ x_i = 12 & \text{FBFFh}, \\ x_i = 13 & \text{F7FFh}, \\ x_i = 21 & \text{FFFBh}, \\ x_i = 22 & \text{EFFFh}. \end{cases} \tag{7}$$

表1列出了 $x_i$ 和 $y_i$ 的取值关系.下面对方程(7)进行一些讨论.

1)错误处理任务块,重新计算的任务块和避障任务块是系统中随机发生事件,必须得到及时处理,其相应的 $J$ 取最大值,是对系统安全性和系统性能的考虑.

2)$\alpha$,$\beta$ 和 $\gamma$ 是可调节值,通过 $\alpha$,$\beta$,$\gamma$ 调节,保证需要执行的任务在规定的时间范围内得到操作.

3)通过学习程序可以调节 $\hat{N}_a$:$t_w>0$时,增加 $\hat{N}_a$;$t_v>0$时,可以减小 $\hat{N}_a$.

4)对于 $p$,$q$ 和 $b$ 的取值范围是

$$b>0, \quad 0<q<p, \quad p+b<7\text{FFFh}. \tag{8}$$

$p$ 和 $q$ 的作用是使轨迹规划和逆运动学求解任务块利用力矩补偿和 PID 控制任务块的执行间歇被实施，根据各子进程的并行性加快系统的计算速度.

5)从图4(a)和(b)中可以得知，进程5和6中包含的子进程1和2因为重要度低于进程3中包含的子进程1和2，所以会被不合理地过分延迟，通过 $a(a>0)$ 的选择可以对其进行有效的调节.

6)系统作业层是利用模块化进行设计的，$x_i$ 在非连续空间取值，是为了新模块很方便地加入.

表1 $x_i$ 和 $y_i$ 对应表

| | | |
|---|---|---|
| $x_i=$ | 1 | 错误处理任务块 |
| | 2 | 重新计算的任务块 |
| | 5 | 轨迹规划任务块 |
| | 6 | 逆运动学求解任务块 |
| | 7 | 力矩补偿任务块 |
| | 8 | PID 控制任务块 |
| | 11 | 识别判断任务块 |
| | 12 | 抓取过程路径优化任务块 |
| | 13 | 吊装机器人恢复初始状态 |
| | 21 | 避障任务块 |
| | 22 | 自动小车恢复初始状态 |
| $y_i=$ | 1 | 吊装机器人 |
| | 2 | 自动小车 |

## 5 结论

本文提出了一种新型的多 DSP 并行结构机器人控制系统的思想，利用并行结构的特点实现多机器人协调系统中可以并行实施的复杂运动控制任务，并通过任务块在链表中的优化排队，加强了系统控制的实时性. 系统发挥了 DSP 的主动性，在进一步提高实时特点的同时，使 CMRC 适用于众多的工作场合. 提供灵活的人机接口，在系统作业层完全向用户开放. 系统相对于 DSP 是冗余的，可靠性和可维修性得到大幅度的提高. 最后对在码头上多机器人协调运送货物的具体例子加以讨论，并通过它给出了 CMRC 中任务块在链表中的优化过程.

## 参 考 文 献


1 Luh J Y S. Scheduling of parallel computation for a computer controlled mechanical manipulator. *IEEE Trans. Syst., Man & Cyber*, 1982, **SMC-12**(2)

2 Nigam Ravi, Lee C S G. A multiprocessor-based controller for the control of mechanical manipulator. In: IEEE International Conference on Robotics and Automation, 1985: 815～821

3 Kasahara Hironori, Narita Seinosuke. Parallel processing of robot-arm control computation on a multimicroprocessor system. *IEEE Journal of Robotics and Automation*, 1985, **RA-1**(2)

4 Chen J B. NYMPH: A multiprocessor for manipulation applications. In: IEEE International Conference on Robotics and Automation, 1986: 1733～1736

5 Kazanzides Peter. A multiprocessor system for read-time robotic control: design and applications. In: IEEE International Conference on Robotics and Automation, 1987: 1903～1908

6 Shalom J I, Kazanzides Peter. SPARTA: multiple signal processors for high-performance robotic control. In: IEEE International Conference on Robotics and Automation, 1988: 284～290



**胡 健** 1972年生，1999年获中国科学院自动化研究所硕士学位，感兴趣的领域有机器人控制、系统可靠性理论及其应用等.

**谭 民** 简介见本刊第26卷第1期.