第 34 卷　第 21 期　　计　算　机　工　程　　2008 年 11 月
Vol.34　No.21　　Computer Engineering　　November 2008
·人工智能及识别技术·　文章编号：1000—3428(2008)21—0199—03　文献标识码：A　中图分类号：TP18

基于多线程的集控式足球机器人上位机系统

景征骏[1,2]，周　军[2]
(1. 江苏技术师范学院计算机科学与工程系，常州 213001；2. 河海大学机电工程学院，常州 213022)

摘　要：为满足系统的稳定性和处理数据的实时性，在分析集控式足球机器人系统运作原理的基础上，对系统进行任务分解并构造了基于多线程技术的其上位机子系统的体系结构。在实现该系统时，采用内核同步机制和 Windows 消息映射机制相结合的方式完成线程之间的同步，通过以循环链队列方式设计的多级缓冲区实现数据信息的传递。

关键词：足球机器人；多线程；同步机制；循环链队列

Host Computer System Based on Multithreading in Centralized Control Soccer Robot

JING Zheng-jun[1,2], ZHOU Jun[2]
(1. Department of Computer Science and Engineering, Jiangsu Teachers University of Technology, Changzhou 213001;
2. College of Mechanical and Electrical Engineering, Hohai University, Changzhou 213022)

【Abstract】To meet the stability and real-time to process data of centralized control robot soccer system, the architecture of its host computer system based on multithreading technology is introduced and constructed while system task is discomposed by analysing its working principles. In the realization process of system, this paper adapts the way of combining windows kernel synchronization mechanisms and message mapping mechanism to accomplish the synchronization between threads, and uses multi-level buffers designed in loop linking queue to transfer data.

【Key words】soccer robot; multithreading; synchronization mechanism; loop linking queue

当前，多智能体的研究已经成为人工智能领域研究的热点之一，而足球机器人系统是一个典型的多智能体系统[1]。足球机器人系统是对动态的环境进行研究，对获取的信息做实时的处理，从而让每个机器人及时地做出最合理的动作。在Windows操作系统平台上，多线程技术是实现足球机器人系统实时性和较高执行效率的最有效的方法。

1　集控式足球机器人系统的运作机理及系统结构

1.1　任务的分解与上位机系统分析

本文介绍的集控式机器人足球比赛系统的工作流程如图 1 所示。

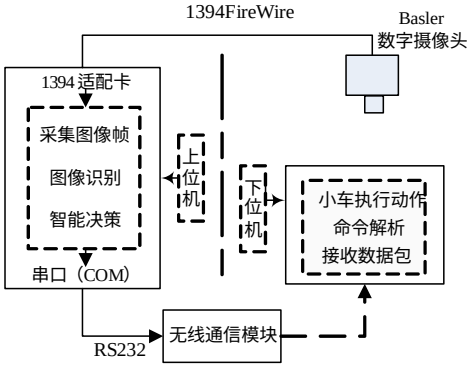

图 1　集控式足球机器人系统结构示意图

首先系统通过挂在比赛场地的数字摄像头采集场地的信息图像，通过 1394 数据线传送到上位机上，上位机上的应用软件应对获取的图像信息进行目标识别(识别场地上的机器人小车和球)并对识别结果进行分析并及时地做出决策，该决策通过RS232 连接的通信器发送到场地上的每一个机器人小车使其做出动作。因此，从工作的流程看，可以从功能上将整个系统分为 5 个子系统：视觉子系统，智能决策子系统，通信子系统，机器人控制系统以及足球机器人车体[2-3]，其中上位机(PC机)负责视觉和决策部分。视觉子系统作为场上机器人的眼睛，它必须实时地捕捉场上的图像信息并从中识别出重要的数据——场上机器人和球的位置。而决策子系统则相当于机器人的大脑，它必须要根据获得图像处理后识别出的信息快速地做出正确的决策，以使机器人小车能相互配合做出合理动作来完成进攻或者防守的任务。因此上位机系统的设计是实现整个足球机器人系统实时性、可靠性的关键。

1.2　上位机系统的需求与结构设计

(1)系统响应。集控式足球机器人系统作为一个半自主多智能体系统，在系统运行中，操作员要根据场上的情况和裁判的口令来调整系统的运行模式，如在开球、争球、罚点球、任意球等情况下，随着裁判的一声哨响，操作员选择某个模式进行点击后，系统必须快速地作出响应。否则，就会错失控球权或者破门得分机会，甚至本方球门会被对手攻破。

(2)实时性。集控式足球机器人系统是一个实时性要求很强的系统。本文介绍的系统中采用的 Basler 数字摄像头的图像采集周期是 0.02 s，即每秒钟 50 帧，这样的采集速度完全能达到实时需求。但是，要让下位机(智能体小车)在场上及

作者简介：景征骏(1978－)，男，硕士，主研方向：智能信息系统；周　军，教授、博士

收稿日期：2007-12-27　**E-mail：** jzj@jstu.edu.cn

时快速地做出合理的动作，系统上位机在单位时间内发出的命令次数必须要满足实时要求。也就是说，上位机系统要能够实时地处理采集到的图像信息并进行智能决策，这就要求除了要有一个高效的图像识别算法外，上位机软件系统的结构也要进行精心设计。

根据以上建立的集控式机器人足球系统框架和上位机子任务需求分析，一个并行的、多线程系统实现方案是必须的，系统方案如图 2 所示。该方案在上位机系统中有 4 个线程：GUI(界面)线程，图像采集线程，图像处理线程和决策线程。其中 GUI 线程为主线程，它负责界面处理、系统数据的初始化以及产生子线程等工作，而图像采集线程、图像处理线程和决策线程是后台的工作线程并通过系统的调度来并发执行的。这一方案的关键就是要建立一个稳定而高效的同步机制，以实现对应用程序内的多个线程的触发与调度以及相互间的通信。

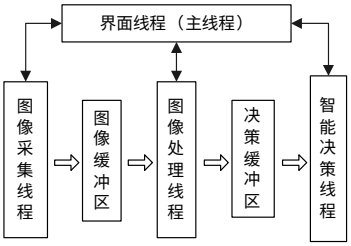

图 2　上位机系统结构示意图

2　多线程技术在系统中的应用

2.1　线程的创建

在VC环境下创建线程主要有以下几种方式：(1)直接使用Win32 API函数CreateThread()；(2)通过MFC的CWinThread类；(3)使用C runtime library所提供的_beginthreadex()来创建[4]。为了在程序的实际运行中能更好地掌控其运行状态以及为了方便代码的调试，本文结合了前两种方式的特点，利用面向对象的技术自行设计了一个CRobotThread类。该类封装了Win32 API函数中一些与线程操作相关的函数，并向外提供了操作线程的接口函数。

```
Class CRobotThread
{
private:
    HANDLE m_hThread; //线程局柄
    DWORD m_ID; //线程号
    bool m_bSuspended; //标志线程是否被挂起
CThread()   //构造函数
BOOL Create(//! 创建线程
LPTHREAD_START_ROUTINE process,//线程处理函数
                LPVOID pParam=NULL,
                int Priority=THREAD_PRIORITY_NORMAL,
//线程优先级
DWORD CreationFlags = 0);
  BOOL SetPriority (int Priority);   //! 设置线程优先级
  BOOL Suspend ()//! 将线程挂起
BOOL Release(DWORD timeout = INFINITE)//释放线程
      …}
```

那么，系统框架中的线程(GUI 线程、图像采集线程、图像处理线程、决策线程)就可以用 CrobotThread 类来创建了。比如创建 GUI 线程：先定义类变量 CRobotThread threadGUI，再通过该类的 Create()函数建立线程，即 threadGUI.Create(…)。

2.2　线程同步机制

同步机制是否合理是多线程应用程序运行是否稳定的关键。多线程应用程序设计步骤中的主要工作，就是要决定在何处存在可能潜在地引起数据毁坏的多线程数据访问冲突，以及如何使用线程的同步来避免这种冲突。在Windows平台上，有多种内核同步机制可以使用互斥器Mutexes、信Semaphores、事件Events以及关键区域Critical Sections[5]等。为了使线程的触发更为灵活，线程调度的可控性更强，本文将操作系统的内核同步机制与Windows消息机制以及全局变量相结合，从而形成了一种全新的线程同步应用模型。图 2 所描述的系统方案中存在着 2 个“供应者——消费者”模式：图像采集线程——图像处理线程、图像处理线程——智能决策线程。在系统实现时，由于两个“供消”模式采用了相同的同步模型机制，因此下面叙述图像采集线程与图像处理线程之间的同步过程。

(1)为了保持各个线程的同步，防止系统资源的泄漏，GUI 主线程中主要工作有以下几点：1)初始化数字摄像头的驱动代码，为图像采集做好准备。本系统使用 Basler 公司提供的驱动程序开发包中的 BCAM 类。2)创建 3 个动态子线程，并分配资源。3)定义系统消息 WM_GRAB_STOPPED 以及相应的消息处理函数，该消息有采集线程发出，通知终止采集，从而使系统释放各个线程资源，保证系统安全结束。4)定义全局标志量 g_bOCROk，该标志量主要是为了线程在对数据缓冲区进行存、取数据时，保持处理数据的连续性。

(2)图像采集线程是动态子线程，是获取数据的起始线程，其他子线程都是由它来激活的。该采集线程负责接收 GDI 主线程的命令(如系统开始、结束等)。在系统开始后，采集线程等待摄像头数据驱动的触发事件 GrabImageEvent 来激活并获取采集端口的一帧图像数据，然后将数据存入图像信息缓冲区，此时设置触发 NewDataEvent 事件来激活图像处理线程。该线程的同步流程如图 3 所示。

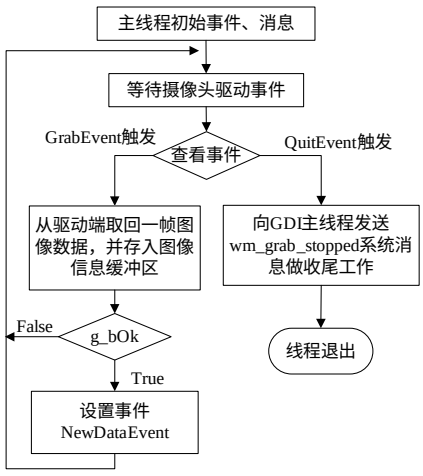

图 3　图像采集线程流程

(3)在图像数据处理线程等待的 NewBuffer Available Event 事件被采集线程触发后，它将进入数据处理模块。首先，它将从图像数据缓冲区中获得新一帧的图像数据，如果缓冲区不为空，则取出数据进行信息识别，并将识别的结果存入识别决策数据缓冲区以待决策。如果缓冲区为空，则再次等待有新一帧数据的触发事件。在此过程中，通过全局标志量 g_bOk 来保持存取数据的一致性和处理数据的连续性。该线程的同步流程如图 4 所示。

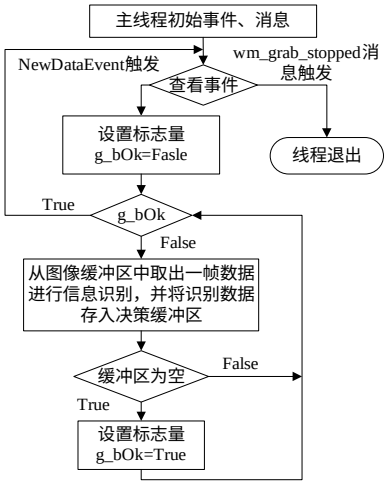

图 4　图像处理线程流程

(4)采集线程作为系统的数据源头，它激活了所有其他的子线程，与之相对应，其他子线程应随着采集线程结束而结束，也就是说，其他子线程退出的事件将由采集线程来触发。在采集线程收到 QuitEvent 事件后，它在结束前向主线程发出安全退出消息 WM_GRAB_STOPPED，而主线程再通过 windows 系统消息映射机制来触发其他静态子线程的退出事件，并释放各个系统资源。采用这种“事件触发，消息回调”的方式结束各个子线程，最大限度地保证了前台的处理用户界面的 GUI 线程与后台的工作线程之间的独立性，从而使系统能快速地响应用户的操作指令，而且也提高了系统运行时的稳定性。

3　数据缓冲区

由于集控式足球机器人系统是一个实时系统，而且各个线程处理的数据类型和处理数据的时间是不一样的，为了能让数据在各个线程之间进行平滑流动，并保持高度的连续性和时效性，在实现该系统时采用了多级缓冲的方式来完成线程之间的匹配。系统的缓冲区是采用了循环链表和动态内存相结合的数据结构，并且利用队列的先进先出(FIFO)思想进行设计的。

3.1　缓冲区数据结构

循环链队列在系统运行的操作过程中的内存示意如图 5 所示。

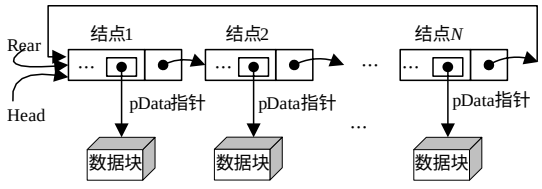

图 5　循环链队列的内存示意图

在系统中有 2 个缓冲区：图像数据缓冲区和决策数据缓冲区。虽然这 2 个缓冲区中所存放的数据的类型不一样(前者存放实时采集的图像信息，后者主要存放经过图像处理后识别出来的机器人和球的位置信息)，但都采用了循环链队列数据结构来实现。根据循环链队列的特性并结合系统的要求，设计封装了 CDataQueue 类。该循环链中可以存放并维护的结点数据类型 NewDataStruct 有 2 种：ImageDataNode 结构或 DecisionDataNode 结构，其中，在创建图像数据缓冲区循环链队列时，ImageDataNode 结构体是循环链队列结点的数据结构，该结点的 PBYTE 类型指针变量 pData 将指向存储某一帧图像数据的内存空间。而在创建决策数据缓存区循环链队列时，DecisionDataNode 结构体为该队列链的结点数据结构，该结点的 PBYTE 类型指针变量 pData 将指向存储图像处理后识别出的某一帧数据信息，主要包括机器人和球的位置信息。

3.2　数据管理

CDataQueue 类主要有 4 个成员变量：m_nDataSum(队列中总的数据个数)，m_nLength(队列的结点总数)，m_pFront 队列头，m_pRear 尾指针。向外提供的操作接口主要是队列的建立、数据进队列和出对列。在系统初始化过程中，由 GUI 主线程定义了 2 个全局的 CQueue 类变量(图像数据缓冲区循环链队列、决策数据缓冲区循环链队列)，并调用类构造函数分配内存资源建立循环链队列。在管理链表队列中的数据时，采用 FIFO 思想与标志量相结合的方式。也就是，当标志量 m_nDataSum 的值大于零，即队列中有数据可读，则通过头指针 Head 来读取数据。

```
PBYTE CMyQueue::OutQueue()
{
  if(m_nDataSum == 0) //表示队列里无存储数据
      return NULL;
  m_nDataSum--; //队列中数据个数减 1
  NewDataStruct * s = head->next;        //获取但前头指针的位置
  head->next = s->next;//头指针后移
  return s->pData;
}
```

而向队列中存数据时则通过尾指针 Rear 来执行，但如果尾指针和头指针相等并且标志量 m_nDataSum 的值大于 m_Length 的话，就表示队列已满，此时就要做丢帧处理。

```
bool CQueue::InQueue(PBYTE pInfo)
{
if(rear->pNext==front->pNext    && m_iDataSum>0 )
  return FALSE; //队列满
memcpy(rear->pNext    ->robotInfo,    pInfo,    DATASIZE);
//通过内存拷贝将数据存入
rear->pNext =rear->pNext ->pNext ;//尾指针后移
          m_iDataSum++; //队列中的数据个数增加 1
   return TRUE;
}
```

对图像数据循环链队列的写操作由图像采集线程执行，读操作由图像处理线程执行。而对决策数据循环链队列的写操作由图像处理线程执行，读操作由智能决策线程来执行。其中，为了防止互斥状况的发生，即 2 个线程同时操作同一个内存空间的事件，在读取队列中的数据时作了加锁操作：

```
EnterCriticalSection
m_pMyQueue->OutQueue();
LeaveCriticalSection
```

4　结束语

集控式足球机器人系统是一个复杂的智能体系统，上位机子系统作为整个系统的首脑，必须要能够实时地捕捉机器人在场地上的图像信息，在进行分析和处理之后，再快速地发出指令以命令场上的智能体小车做出下一个动作。为了能完成这样的任务，除了要有高效的图像识别算法和高度智能的决策模型外，设计一个结构合理的上位机系统体系结构是整个工作的核心。目前，以上设计的方案已经成功地应用到河海大学足球机器人 Mirosot5:5 和 11:11 系统上。

（下转第 204 页）