多线程下的视频微动目标检测与提取算法研究

瞿 中[1,2] 马庆伟[1] 张庆庆[1]

(重庆邮电大学计算机科学与技术学院 重庆 400065)[1] (重庆邮电大学移通学院 重庆 401520)[2]

摘 要 在多线程并发情况下提取实时视频图像中的微动人物目标,基于多线程的思想,在多路并发请求的情况下,运用肤色检测和聚类思想对图像中的人物进行提取分割,并对处理过程中的多路访问资源的问题进行访问冲突避免。首先采用边缘检测的思想对运动目标边缘进行范围定位,然后采用聚类算法和肤色检测等方法进行目标模板的提取和完整性补充,根据得到的目标模板结合原始图片对应像素点坐标,将原始像素点颜色着色到新的背景模型的对应位置。仿真结果表明,在保证处理质量的情况下,提出的基于多线程思想的算法能满足实际工程应用实时处理的要求。

关键词 多线程,微动目标,聚类算法,肤色检测,目标模板

中图法分类号 TP391.41 **文献标识码** A

Research on the Algorithm of Micro-movement Target Tracking and Extraction in Video by Multi-thread Technology

QU Zhong[1,2] MA Qing-wei[1] ZHANG Qing-qing[1]

(College of Computer Science & Technology, Chongqing University of Posts & Telecommunications, Chongqing 400065, China)[1]

(College of Mobile Telecommunications, Chongqing University of Posts & Telecommunications, Chongqing 401520, China)[2]

Abstract We extracted micro-movement target of real-time video images under the multi-thread conditions. Based on the multi-thread ideas, we used skin colour detection and clustering methods to process and solve the access violation condition. The ideology of edge detection was used locate the range of the moving object, and then clustering algorithm and skin color detection and some other methods were used extract the object template and complement the integrity of the object template, and according to the object template and corresponding pixel coordinates of the original image, the original pixel color onto the corresponding position of new background model. The simulation results show that the proposed method ensure the quality requirements of real-time processing and has a certain robustness, so this method satisfies the needs of the project.

Keywords Multi-thread, Micro-movement target, Clustering algorithm, Skin colour detection, Object template

1 引言

视频中的运动目标跟踪与提取是计算机视觉、视频目标跟踪等研究领域的重要内容,并已经得到广泛的应用。运动目标提取是指从图像序列中将运动的区域分割出来,然后根据提取的目标进行特征提取或目标再处理等操作[1]。

准确性是运动目标提取的关键因素,常见的算法有帧差法、背景减除法以及光流法等。帧差法算法简单,适用于实时性要求高的场合,但该算法存在提取的目标边缘模糊和孔洞现象;背景减除法指当前帧和存留的背景进行相减运算,同样算法简单,适合实时处理的要求,但该方法对背景和当前帧的图像范围要求高,即摄像头捕获的场景范围不能发生位移,否则减运算后得到的结果完全错误;光流法是根据目标随时间变化得到的光流特征来检测运动目标,该算法运算时间复杂度高、实时性差,另外对微动目标的检测效果不明显。

为了对视频序列中微动目标在多个用户响应事件中进行实时提取,本文采用肤色检测、形态学处理和聚类算法相结合的方法进行目标模板的获取和补充。

2 多线程图像处理技术

多线程思想是指每一个进程至少有一个主执行线程[2],它无需用户去主动创建,是由系统自动创建的。用户根据需要在应用程序中创建其它线程,多个线程并发地运行于同一个进程中。一个进程中的所有线程都在该进程的虚拟地址空间中,共同使用这些虚拟地址空间、全局变量和系统资源,线程间的通讯非常方便。多线程可以实现并行处理,避免了某项任务长时间占用CPU资源。

2.1 多线程

当多线程执行时,对应的进程包含并发执行的多个线程。利用多线程可以执行实时性、交互性很强的运算和操作,从而提高CPU的利用率。

线程是动态存在的,从创建到终止具有一定的生命周期。

到稿日期:2011-07-09 返修日期:2011-10-12 本文受重庆市科委自然科学基金计划项目(CQCSTC2010BB2399),重庆市教委科学技术研究项目(KJ110528)资助。

瞿 中(1972—),男,博士,主要研究方向为数字图像处理、普适计算等,E-mail:quzhong@cqupt.edu.cn;马庆伟(1986—),男,硕士生,主要研究方向为数字图像处理;张庆庆(1983—),男,硕士生,主要研究方向为数字图像处理。

创建线程时需指定任务例程(函数),线程创建后开始执行该例程,执行返回后线程处于终止状态,但线程仍然存在,需要其它线程来删除。线程的终止可由其它线程完成。

2.2 多线程与图像处理

利用多线程技术来提高图像处理的速度和实现多并发性:

(1)在支持多任务处理的操作系统中,CPU 时间片以进程为基本单位进行分配,进程进一步划分为线程。任何应用程序都不可能像在单任务操作系统中那样独享所有系统资源。既然 CPU 时间片是根据线程来分配的,在同一优先级下,单线程程序所分配到的 CPU 时间必然要比多线程程序少,而采用多线程程序能得到更多 CPU 时间[3]。

(2)本文中图像的处理主要包括前景图像的运算处理和背景图像加载两个部分。由于一次只能对一帧背景图片数据进行磁盘载入,因此磁盘访问次数比较频繁,磁盘设备的访问速度相对较慢。如果采用顺序处理方法,只能在处理完一帧前景数据后再进行磁盘访问载入背景图片,在等待的过程中,并未充分利用 CPU 资源。而采用多线程技术,则有可能使处理数据与磁盘操作两部分同时进行,可提高对 CPU 的利用率,从而提高数据处理速度。

(3)本文中目标图像的处理存在多对象并发性,即在同一时刻存在不同的目标提取服务请求。在这种情况下,单线程显然无法满足实际需求,因此该情况下,采用多线程技术,则可以使不同并发的处理请求得到互不干扰的响应,从而满足实际多路并发情况。

3 基于多线程的视频微动目标提取技术

多线程技术可使编程任务简化和模块化[4],可以提高处理器的效率,减少任务的切换时间,提高吞吐量和并发度。特别是后台处理时,可执行很重要但时间上要求不高的任务或等待某一事件的发生而控制某一事务的处理等。因此,对图像处理可分为两部分:一部分用于人机交互,即对预处理后的图像进行处理;另一部分后台可不断对分割的子块进行预处理,同时进行校正。采用多线程技术后,可以在预处理后让用户进行局部操作,如矢量化操作等。后台的线程专门用于预处理的实现。

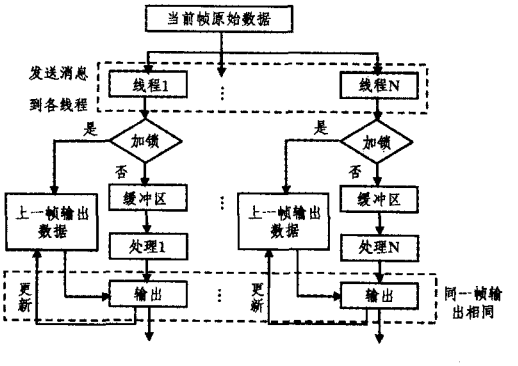

图 1 多线程下视频目标检测与提取流程

本文采用的多线程思想,运用于多用户下响应图像处理的总体流程如图 1 所示。图中,上一个虚线框包含部分是在主进程中创建多个线程,该阶段在线程创建完成后,进程发送消息给各个线程,启动各个线程,消息包含各个线程处理视频资源的路径等;下一个虚线框包含部分是指每个线程在处理同一个视频的同一帧的情况下,所输出的内容应该相同,不同的是由于每个线程执行时间有长短,对应的每个线程输出相同帧图像的先后顺序不同。

3.1 多线程与视频微动目标提取

本文提出的视频微动目标的提取,主要用于多个用户向系统请求对采集到的视频进行处理,为了响应多个客户对服务器上同一个功能接口进行调用,采用动态创建多线程的方式对每个客户的请求进行响应。

视频是一个连续图像序列,当某一帧正在处理时,下一帧可能到达,但当前帧还没处理完。由于一些原始图像数据在处理过程中还存在访问或回取的现象,此时,不能对缓冲区中的当前帧的原始数据进行下一帧覆盖操作,否则会出现错误的结果,因此还需要对线程的缓冲区资源进行加锁保护来实现互斥访问。

3.2 线程间的通信

当开始进行图像处理时,程序的主线程开始工作;当有用户请求视频处理时,主线程与用户处理子线程之间进行信息交换,如子线程启动、子线程数据获取和返回等,这些操作涉及到线程之间的同步问题,通过互斥锁控制对图像数据缓冲区实现同步访问。

主线程和子线程都涉及对图像数据的读取和更新问题,当主线程重新获取数据后,就要向子线程发送消息,激活子线程处理消息响应;当子线程执行完后,向主线程发送处理完成信息。主线程接收到信息后释放子线程。

通过主线程 PostThreadMessage()函数向子线程发送消息来完成线程间的通信,在子线程里面根据对应的线程 ID 号,通过 GetMessage()函数来获取主线程发送的消息。

上述两个函数原型:

(1) 函数 BOOL PostThreadMessage (DWORD idThread, UINT Msg, WPARAM wParam, LPARAM Iparam),该函数的 4 个参数的含义:

idThread:其消息将被寄送的线程的线程标识符;

Msg:指定将被寄送的消息的类型;

wParam:指定附加的消息特定信息;

IParam:指定附加的消息特定信息。

返回值:若函数调用成功,则返回非零值,否则返回零值。

(2) 函数 BOOL GetMessage (LPMSG lpMsg, HWND hWnd, UINT wMsgFilterMin, UINT wMsgFilterMax),该函数 4 个参数的含义:

lpMsg:指向 MSG 结构的指针;

hWnd:取得其消息的窗口的句柄;

wMsgFilterMin:指定被检索的最小消息值的整数;

wMsgFilterMax:指定被检索的最大消息值的整数。

返回值:如果函数取得 WM_QUIT 之外的其他消息,返回非零值;如果函数取得 WM_QUIT 消息,返回值是零。

3.3 共享缓冲区加锁机制

共享缓冲区是实时视频数据存放和处理过程数据提取的公共缓冲区,对于不同的处理线程,存在各自的缓冲区。为了防止出现访问冲突,实现线程同步问题,本文采用加互斥锁的方法避免访问冲突。

有时线程的某些步骤需要以原子的方式来进行操作,尤其在线程对内存、资源访问的时候。用 InterlockedExChange ()来实现循环锁的功能,即在向缓冲区存放数据的线程中,首先对加锁标志进行判断,若缓冲区未加锁,则对标志变量进行 InterlockedExChange()加锁,此时才能向缓冲区存储数据,存储结束后,以同样的方式释放加锁标志;在读操作中,首先也

是判断加锁标志位是否已被加锁，若未加锁，则对标志变量进行 InterlockedExChange()加锁操作，然后读取数据，完成后释放锁标志，一直循环对锁标志进行操作直到处理结束。

加原子锁函数原型：

LONG InterlockedExchange(LONG volatile * Target, LONG Value)，其中函数的两个参数的含义为：

Target：指向待设置原子操作对象变量的指针；

Value：变量需要设定的值。

4 视频微动目标检测与提取算法

本文提出的运动目标检测是从视频序列图像中检测出运动目标[5]，并将目标提取出来。通常，实时视频捕获环境中摄像头位置在某些情况下需要调节，镜头方向会发生角度变化[6]，因此，对于目标运动幅度不大的视频序列，目标的提取必须采取单帧分开的方式处理。本文对视频帧进行处理的流程如图 2 所示。

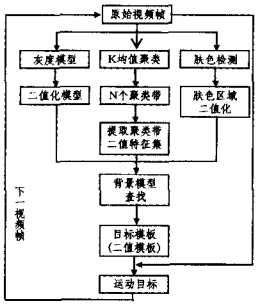

图 2 运动目标检测与提取

本文图像处理算法的处理过程为：

(1)对原始视频单帧图像进行灰度处理，并根据设定的阈值得到二值化模型。

(2)对原始视频单帧图像进行均值聚类操作，并根据聚类后的分类结果进行特征集提取，区分出背景前景聚类带，并对前景背景二值化区分处理。

在 K 均值聚类算法运行前必须先指定聚类数目 N(在最小化误差函数的基础上将数据划分为预定的类数 K)和迭代次数或收敛条件，并指定 N 个初始聚类中心。根据一定的相似性度量准则，将每个像素点分配到颜色相近的聚类中心，然后形成类，以每一类的平均矢量作为这一类的聚类中心重新分配，反复迭代直到类收敛或达到最大的迭代次数。

(3)对原始视频单帧图像进行固定值静态肤色检测，并对肤色区域进行二值化处理。

(4)上述 3 个二值化模板结合原始图像，把属于前景人物的范围按坐标点分割出来，完成目标的分割提取。

该方法具如下优点：

(1)不依赖于运动目标的形状特征和数量。

(2)以往各种方法的运动目标的分割中出现的孔洞误分割现象得到了很好的解决。

该处理过程涉及到的处理方法有：

```
# ifndef _HANDLE_PICTURE_H_
# define _HANDLE_PICTURE_H_
// 初始化函数，参数为待处理图片大小
INT32 Initialize(UINT32 nWidth,UINT32 nHeight);
// 设置转换参数；参数为待处理图片格式和背景图片源等信息
INT32 SetOperationParam(const InputParam * ptInputParam,const
OperationParam * ptOperaParam,UINT32 nParamCount);
// 阈值设置函数，在预处理阶段使用
VOID setThreshold(double ThresholdValue);
// 完整性约束函数，对目标人物的完整性提取控制
VOID setIntegrity(int Integrity);
// 待处理图片背景类型选择函数
VOID setBgStyle(int Value);
// 本算法关键函数，图片的目标检测和分离提取
INT32 HandlePicture(const UINT8 * pSrcPic,UINT8 * pDstPic);
// 释放资源函数
INT32 Uninitialize();
# endif
```

对于核心图像的目标检测与分割提取函数 HandlePicture()，其主要处理流程为：

(1)当收到处理命令后，申请并开始创建多个线程，根据命令初始化指定数目的线程数，并分配满足各自线程需要的内存空间等资源。

(2)对于单一线程来说，对源初始数据缓冲区进行互斥机制处理，以防在处理过程中重复地读取源数据，此时发生重写操作时，引起数据不一致，从而造成错误的发生。

(3)处理源缓冲区中单帧图片数据，流程按照图 2 所示流程进行处理。

(4)把结果放到设定的目的缓冲区中，输出显示过程是直接从缓冲区读取数据，并组合成视频图像序列播放显示。

(5)传递释放源缓冲区的消息给该线程，并读入下一帧图片，重复第(3)(4)步。当时间超过定时器设定的长度而源缓冲区仍然没有被释放时，则跳过第(3)步，直接进入保持状态，即输出目的缓冲区中上一帧的处理结果。

(6)当该线程视频处理完成之后，发送销毁该线程函数，并释放线程占用的内存空间。

5 实验结果

本文所采用的硬件环境：CPU 主频为 2.83GHz×8、内存为 4G 的计算机。软件平台使用 Microsoft Visual C++ 6.0 和由 Intel 开发的开源计算机视觉库 OpenCV 2.0 实现。测试的视频序列像素尺寸大小为 176×144。

实验过程是首先对原始视频的每帧图像进行前景目标的检测定位，接着对目标进行分割提取，并形成二值目标模板，然后对比原始图像，对模板着色后提取出目标。其中，整个系统的测试操作处理平台界面如图 3 所示。

图 3 多线程视频目标提取

实时视频处理的一个重要性能指标就是要达到处理时延低于人眼观察图像序列流为连贯视频的最大帧速的要求，即人眼的视觉暂留特性，人眼能够感觉视频播放流畅的频率是每秒 25 帧，即每张图片处理时间要小于 40ms，这样才能使得视频看起来很流畅。根据视频的帧率，要达到显示流畅的效果，处理时间不能高于视频帧与帧的间隔时长，即满足公式条件[8]：处理单帧时间≤1/视频帧频。

仿真实验结果的实时处理时间性能和内存使用情况的统计如表 1 所列。

表1 时间和内存使用性能统计

并行线程数	帧频 (fps)	平均处理时间 (ms)	内存使用情况 (Mb)	理论帧间隔 (fps)
2	10	23.0	18.4	100
	15	23.2	18.9	66
	30	22.7	19.7	33
10	10	27.0	32.1	100
	15	28.2	32.8	66
	30	29.7	33.1	33
20	10	30.3	44.7	100
	15	31.8	45.8	66
	30	33.5	41.3	33
40	10	38.2	72.4	100
	15	34.7	73.1	66
	30	40.1	73.3	33
80	10	46.7	118	100
	15	41.3	119.3	66
	30	44.2	119.9	33

仿真实验结果表明，在并发处理2路和10线程时，单帧处理时间远小于理论计算得到的帧间隔时间，即达到实时处理的效果；在并发处理20路、40路或80路线程时，由于程序运行在单一服务器上，处理时间需要考虑操作系统基本的线程轮询的时间消耗，在视频的帧率为30fps时，处理时间比理论帧间隔时间长，例如，在并发线程数为40时，30fps实际平均处理时间为40.1ms，而理论值为33ms，实际值比理论值大，但是实际图像处理效果仍然良好，视频播放同样流畅，能达到实际工程应用要求。

结束语 本文提出的在并发多线程环境下对微动视频目标进行检测和提取的算法，结合了K均值聚类算法和肤色检测算法。根据K均值聚类算法的性质，它具有严密性和运算简单的特征；根据固定肤色检测算法的性质，它具有定位肤色区域快捷高效的特征。在实时处理要求很高的情况下，运用多线程思想对实际应用中的多路访问情况进行解决并实行，从时间性能和准确性能上均具有很好的效果。

参考文献

[1] Jasper S, Babak T, Yulia E, et al. Automatic segmentation of video to aid the study of faucet usability for older adults[C]// IEEE Computer Society Conference, Computer Vision and Pattern Recognition Workshops. 2010:63-70

[2] 李实，刘乃琦，郭建东. 多核架构下的多线程负载平衡[J]. 计算机应用，2008，28(12)：139-140

[3] 刘权胜，杨洪斌，吴悦. 同时多线程技术[J]. 计算机工程与设计，2008，29(4)：963-967

[4] Sun Zhi-hai, Zhu Shan-an, Zhang Da-wei. Real-time and Automatic Segmentation Technique for Multiple Moving Objects in Video Sequence[C]// IEEE International Conference, Control and Automation. 2007:825-829

[5] Qu Zhong, Ma Qing-wei. An Algorithm of Slight Moving Object Extraction in Real-time Video[J]. Applied Mechanics and Materials, 2011, 63-64:603-606

[6] Zhan Chao-hui, Duan Xiao-hui, Xu Shuo-yu, et al. An Improved Moving Object Detection Algorithm Based on Frame Difference and Edge Detection[C]// Fourth International Conference on Image and Graphics. 2007:519-523

[7] Qu Zhi-guo, Wang Ping, Gao Ying-hui, et al. Contour detection based on contextual influences[C]// IEEE International Conference on Information and Automation. 2010:1694-1699

[8] Guan Yu-dong, Wang Ying, Zhang Hong. Video object segmentation algorithm integrating temporal and spatial information[C]// 9th International Conference, Electronic Measurement & Instruments. 2009, 4:136-140

[9] 徐保勇，王媛丽，王平，等. 基于高斯混合模型机载下视运动目标检测方法[J]. 重庆理工大学学报：自然科学版，2011，25(11)：56-62

(上接第235页)

结束语 本文提出了一种求解加法群 Z_p^+ 上离散对数问题的DNA计算算法及相关子算法。解空间的生成算法借鉴传统计算机中经典的3表算法思想，其搜索空间小，减少了实验过程中的错误率，进一步提高了算法的实验可行性。从文献[5-11]及本文算法可以发现，DNA计算为密码分析学提供了一种更加有效的可行方案。当未来DNA计算的生物技术进一步成熟后，现存的密码系统的安全性将面临很大的挑战。

参考文献

[1] Harif Ullah A M M. A DNA-based computing method for solving control chart pattern recognition problems[J]. CIRP Journal of Manufacturing Science and Technology, 2011, 133:1-11

[2] Chen R C S, Yang S J H. Applying DNA computation to intractable problems in social network analysis[J]. BioSystems, 2010, 101:222-232

[3] Shin S Y, Lee I H, Kim D M, et al. Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing[J]. IEEE Transactions on Evolutionary Computation, 2005, 9(2): 143-158

[4] 李人厚，余文. 关于DNA计算的基本原理与探讨[J]. 计算机学报，2001，24(9)：973-978

[5] Boneh D, Dunworth C, Lipton R J. Breaking DES Using a Molecular Computer[C]// Proceedings of DIMACS Workshops on DNA Computing. Princeton: American Mathematical Society, 1995:37-65

[6] 王静，李肯立，许进. Pollard $p-1$ 因子分解的DNA计算机改进算法[J]. 系统仿真学报，2008，20(18)：4835-4839

[7] 陈智华. 基于DNA计算自组装模型的Diffie-Hellman算法破译[J]. 计算机学报，2008，31(12)：2116-2122

[8] 张勋才，牛莹，崔光照，等. 基于自组装DNA计算的NTRU密码系统破译方案[J]. 计算机学报，2008，31(12)：2129-2137

[9] Chang W L, Guo Min-yi, Ho Shan-hui, et al. Fast Parallel Molecular Algorithm for DNA-based Computation: Factoring Integers[J]. IEEE Transaction on Nanobioscience, 2005, 4(2): 149-163

[10] Li Ken-li, Zou Shu-ting, Xv Jin. Fast Parallel Molecular Algorithms for DNA-based Computation: Solving the Elliptic Curve Discrete Logarithm Problem over GF(2^n)[J]. Journal of Biomedicine and Biotechnology, 2008(5): 749-752

[11] 章照止. 现代密码学基础[M]. 北京：北京邮电大学出版社，2004，4：152-171

[12] 周康，同小军，许进. 背包问题的闭环DNA算法[J]. 系统仿真学报，2008，20(17)：4605-4608

[13] 周旭，李肯立，乐光学，等. 一种基于分治的三维匹配问题DNA计算算法[J]. 电子学报，2010，38(8)：1831-1836

[14] 卢圣栋. 现代分子生物学实验技术[M]. 北京：中国协和医科大学出版社，1999：23-59