快速 Blob 分析算法在气门摇臂位置检测中的应用①

孟宪臣, 郭立侠, 潘 丰

(江南大学 轻工过程先进控制教育部重点实验室, 无锡 214122)

摘 要: 针对基于机器视觉的气门摇臂安装位置检测中的高实时性图像处理要求, 提出了一种基于游程的快速 Blob 分析算法. 通过使用游程链表和动态数组, 每个游程仅需扫描一次, 不必与相邻所有游程进行比较. 在算法中增加追溯扫描过程, 消除标记冲突引起的标记误差, 简化了判断过程, 提高了操作效率. 实验结果表明算法具有鲁棒、高效的特性, 已在气门摇臂位置检测系统得到应用.

关键词: 图像处理; Blob 分析; 游程; 位置检测

Efficient Blob Analysis of Binary Image for Defects Inspection in Valve Rocker Arm Installation

MENG Xian-Chen, GUO Li-Xia, PAN Feng

(Key Lab of Advanced Process Control for Light Industry(Ministry of Education), Jiangnan University, Wuxi 214122, China)

Abstract: One of the key issues for a vision system used for inspection of defects in Valve rocker arm installation is the real-time performance request for image process. A new Blob analysis algorithm based on Run-Length Encoding(RLE) for binary image was proposed. This algorithm requires only a single pass over the image by using the method of run-lists and dynamic array, without any need to be compared with all the RLEs at the upper line. Moreover, add the rescan algorithm in order to eliminate the labeling errors aroused by label conficts, which simplifys the process and improves the entire efficiency. The experiment results show that the proposed approach is a viable alternative for effectively labeling connected component. The algorithm has been successfully applied to a vision system fort the inspection of defects in valve rocker arm installation.

Key words: image processing; Blob analysis; run-length encoding; position detection

1 引言

在基于机器视觉的气门摇臂位置检测过程中, 为了正确获得气门摇臂的安装位置信息, 必须对通过激光照射的气门摇臂上的光斑进行 Blob(斑块)分析, 通过目标像素的连通性判断找出每个光斑所对应的像素点集. 由于像素的连通性分析是一个全局性的运算, 当摇臂经过激光照射后所产生的光斑形态复杂、数目未知且变动范围较广时, 算法的稳定性和执行效率成为影响系统性能的关键. 现有的大量 Blob 分析算法仍存在着搜索空间冗余度高、连通性判断分支复杂和标记冲突过程处理低的缺点, 难以满足气门摇臂安装位置检测的应用需要. 本文在现有研究基础上, 通过改善游程编码及 Blob 数据结构, 采用步进式动态搜算方式, 简化了游程连通性判断分支, 达到提高算法整体效率的目的. 在计算机视觉和图像处理中, Blob 的定义是指具有相似图像特征(如颜色、纹理等), 而且在空间上是连通的像素组成的块[1,2]. Blob 分析的核心就是连通区域检测算法[3], 目前文献中出现的标记算法大致可以分为四类: (1)轮廓扫描法[4]. 即先对连通区域进行边界跟踪, 找到区域边界并标记, 然后搜索并标记连通区域内部点. 此法对目标点距离近且形状不规则的情况处理不理想, 准确率低. (2)区域增长法[5]. 扫描二值图像中每一个像素点, 如该点未标记则将其压入堆栈并从该点反复标记其邻域, 直到堆栈为空. 该

① 基金项目:国家自然科学基金(61273131);江苏高校优势学科建设工程资助项目

收稿时间:2013-07-11;收到修改稿时间:2013-08-09

算法重复扫描次数较多，对连通区域形状适应性不强. (3)基于游程编码的标记算法[6,7]. 游程是一种逐行存储目标特征信息的结构，该方法首先对图像进行游程编码，通过判断上下连续两行中游程之间的连通性，编写标号冲突表，扫描完图像后整理冲突表，按照某种搜索算法得到标号等价表来记录标记冗余. 由于标记值冲突现象在图像处理过程中非常普遍，实际的标号冲突处理算法比较复杂，在一定程度上限制了该方法的应用.

本文在分析以上算法的基础上，提出了一种适合工业应用的快速 Blob 分析算法，主要思想是将动态链表引入到连通区域标记算法中，存储相邻两行的游程信息，同时将标号域和行(列)坐标引入到游程结构中以便计算质心面积等特征. 该算法每次只处理相邻两行图像数据,只需一次扫描图像就能完成所有连通区域的标记，由于抛弃了等价表,回避了通过建立等价表来解决标记冲突的麻烦. 然后再标记算法中增加追溯扫描算法对标记冲突进行处理，确保可以准确标记各种形状的连通区域. 实验测试表明，本文算法能较好的适应气门摇臂位置在线检测的实时要求.

2 算法描述

2.1 游程及斑块信息数据结构的定义

假设二值图像(W×H)中 0 为背景像素值，1 为目标像素值. 游程是图像中一行或一列中灰度值连续为 1 的连通段表示方法. 在传统游程结构(起始坐标、终止坐标、长度)的基础上定义了新的游程数据结构并加以描述:

Struct Info(int label, start, end, tempner[]; char cpixel; info*lp;Blob**lpB;)

其中: *label* 代表游程标号，*start* 为游程所在行的起始坐标，*end* 为游程所在行的终止坐标，*tempner*[] 数组用来暂时存放两个冲突的标记: 其中第一列为当前点的标记，第二列为邻居的标记，*cpixel* 为中间变量，表示当前点的像素值. 因为每个游程数据单元必包含且仅包含于某个唯一的 Blob 对象，将标号相同的所有游程放在同一线性链表中，用指针来 *lp* 指向缓存 DIB 图像，用来把原图复制出来，防止在斑块分析中图像被破坏. 数据域 *lpB* 为间接指向 Blob 的二级指针，通过它可以快速检测到链表的头部和尾部，对相邻 Blob 进行概念合并，要明确的是要将大号的往标号小的合并，因为标号较小的斑块的起点肯定是合并后斑块的起点. Blob 对象定义为:

Struct Blob (Info*pb; Info*pt)

其中，*pb* 指向 Blob 的最下一行(第一行)，*pt* 指向 Blob 的最后一行(不一定是最上面一行). DIB 的字节数组是从图象的最下面一行开始的逐行向上存储的，也即等于把图像倒过来然后在逐行扫描的. 可见一个 Blob 对象实际上描述了一个 *info* 链表，通过它可以知道同属于一个标号的所有 *info* 对象. 算法结束后，可以标记出所有的连通区域.

2.2 连通性判据

所谓连通区域是指图像的一个最大连通子集，这个集合的元素都是目标像素[8,9]. 连通性定义如下: 在一个连通集中任意两个像素之间都存在一条完全由这个集合中元素构成的路径. 用公式表示就是: 两个前景像素 P，Q 是连通的当且仅当存在一条路径 $P_1-P_2-P_3-\cdots-P_n$，使得 $P_1=P, P_n=Q, \forall 1\le i\le n-1$ 有 P_i 与 P_{i+1} 相邻. 其中 $P_1, P_2, \cdots, P_n$ 都是前景像素. 连通性的判断只需进行相邻两行 *level*(> 0) 上的游程与上一行 $level-1$ 的游程之间的连通性判断. 本文运用八连通法则进行连通性判断. 当两个像素点 $P_1(x_1, y_1)$ 和 $P_2(x_2, y_2)$ 满足:

$$1\le (x_1-x_2)^2+(y_1-y_2)^2\le 2 \tag{1}$$

就说 P_1 和 P_2 是八连通的. 即

$$Info[i].start\le Info[i-1].end+1 \tag{2}$$

$$Info[i].end+1\ge Info[i-1].start \tag{3}$$

如图 1 所示.

3 算法流程

算法约定: 输入图像高度为 *Height*，图像宽度为 *Width*，*Sequence* 表示扫描图像连通区域的像素数目，*pixel* 为指针数组，指向当前扫描像素的灰度值. 算法按照从上到下，从左到右的顺序扫描改行的游程数组. 算法通过当前像素的像素值和临近像素值的比较来确定当前行的游程数，并对扫描到的游程通过式(2)(3)与上一行游程进行比较，判断连通性. 通过追溯扫描算法将最小标号回传到目标结构中，保证在同一连通体内的目标结构有相同的根标号. 算法完整步骤流程如图 2 所示.

第 1 步: 初始化斑块内的像素数目、连通区域标号 *label* 级行号 *level* 为 0. 在当前行中寻找未标记过的像素，遇到即认为一个新的斑块的开始，记录该像素

的起点 X, Y 坐标, 并标记该像素为 $Label+1$, 扫描邻居点, 如果邻居点为白点且未标记, 则标记设为当前点的标记并归结为一个游程, 进行下一步, 否则转入第 3 步.

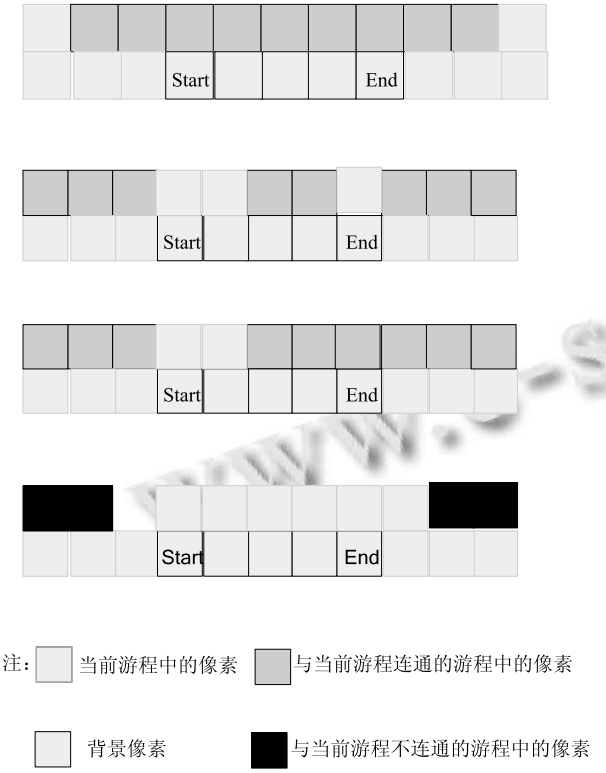


图 1 基于八连通规则的游程连通性表示

第 2 步: 如果 $Level>0$, 在上一行即 $Level-1$ 行寻找与当前游程连通的所有未标记过的游程, 如果邻居未标记, 则标记为当前点的标记, 如果邻居标记过, 且不论标记号大于或者小于当前点的标记, 都把两个标记存入临时数组 *tempner*, 为了防止"隐式邻接"关系被漏掉, 需要读取 *tempner* 中记录的数对, 对于每个数对, 先将 *info* 中对应的较大标号所属象素数合并到较小标号的象素数中, 然后在 *tempner* 中将当前数对之后的数对中所有相应数对作转换.

第 3 步: 如果当前行的第 k 个游程数据 $info(k)$ 大于当前行的游程个数 $size(Level)$, 即当前行的游程已经分析完毕, 则将行号加 1, 转第 1 步; 否则转第 4 步.

第 4 步: 如果上一行的第 k 个游程 $info(k')$ 大于 $size(Level-1)$, 说明上一行的游程已经比较完毕, 则执行第 5 步; 否则转第 7 步.

第 5 步: 当前游程的 *lpB* 不为空, 转第 6 步; 否则, 向链表 Blob 中添加新的目标索引 $\mathrm{Re}f \leftarrow \&Blob$, 并设置 *pb*, *pt* 指针同时指向当前游程: $Blob.pb, Blob.pt \leftarrow \&Info(k)$; 继续执行第 6 步.

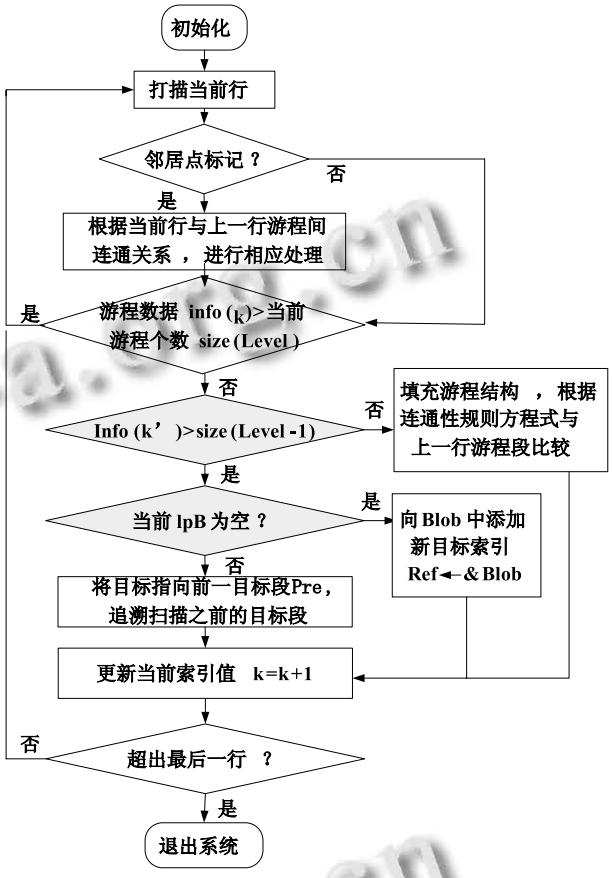


图 2 Blob 分析算法流程图

第 6 步: 将当前索引值 k 加 1, 转第 3 步.

第 7 步: 当扫描到一个游程 A 后, 填充该游程结构, 并根据连通性规则式(2)(3)与邻近上一行游程段进行比较, 这时可能出现三种情况:

(1) 找不到匹配的游程段

认为 A 为新增加的游程段, 上下两个游程不连通, 即 $Info[i].start>Info[i-1].end+1$ 或 $Info[i].end+1<Info[i-1].start$. 申请新的目标结构空间, 将其地址赋值给目标结构指针数据 *Sequence*, *Label* 数累加 1, 转入第 5 步.

(2) 与唯一游程段匹配

说明该游码段 A 为某一目标的一部分, 设匹配游码结构为 B, 这时只需将 A 中相关信息添加到该目标信息中去即可. 判断游程 *lpB* 是否为空, 若为空应更新目标的尾部, 使其指向当前游程: $(*Info(k').lpB).pt$

← &*Info*(*k*) 同时更新当前游程所包含于的 *lpB* 指针，使之与参考游程相同.

若 *lpB* 不为空，此时比较二者的 *lpB* 域是否相同，如一致，无需进行任意操作；否则转入第 2 步.

(3) 与两个或更多的游程段匹配

当A找到两个或更多的匹配游码结构 $C_1, C_2, \cdots C_N$ 时，图像出现向上的开叉. 修改当前游程标号为所匹配的游程中最小的那个，即：*Info*(*k*).*label* ← min(*Info*(*k*').*label*)

第 8 步: 遍历 Blob 动态数组，将所有指向当前游程合并前所属 Blob 标号值修改为参考游程所属的 Blob. 得到 Blob 区域后，就要计算其面积、周长等特征参数. Blob 区域面积是所有属于这个 Blob 游程所代表的像素数目之和.

由于标记冲突非常普遍，即使算法在精确还是会有一些标记冲突出现，如图 3，图 4 所示. 为了确保查找的 Blob 数正确，增加追溯扫描算法，消除标记冲突给后续处理带来的麻烦.

0	0	0	0	1	1
0	0	1	1	1	1
1	1	1	1	1	1
0	0	1	1	1	1
0	0	0	0	1	1

图 3　原二值图像 A

0	0	0	0	1	1
0	0	1	1	1	1
1	1	1	1	1	1
0	0	3	1	1	1
0	0	0	0	2	2

图 4　图像 A 的标记冲突

要直接从根源上消除标记冲突，比较困难. 间接解决较容易，当一个像素点为前景像素并且周围标记值不同时，即标记冲突出现时，增加追溯扫描算法，把错误的标记值修改为同一连通区域的最初标记值，可实现连通区域的准确标记. 具体流程为在第五步对当前目标段的上下领域扫描完成后，将目标指向前一目标段 Pre，追溯扫描之前的目标段.

4　实验结果与分析

本文算法采用 VisualC++2008 编程实现，对不同结构和形状的二值图连通区域进行了测试. 实验结果如图 5，图 6，图 7 所示，其中属于同一像素的连通区域被赋予了同一颜色值.

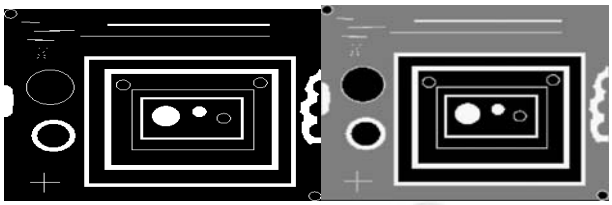


图 5　Blobs

图 6　Lena

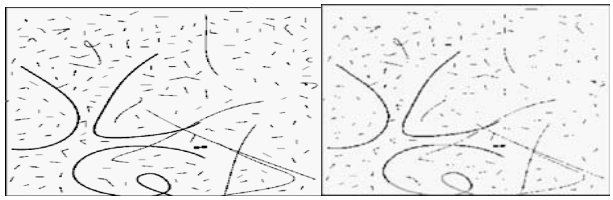


图 7　不规则形状

为了与同类算法作比较，对文献[10,11]的算法做了同等测试. 对每幅图像，算法各运行 500 次，取平均时间，结果如表 1 所示.

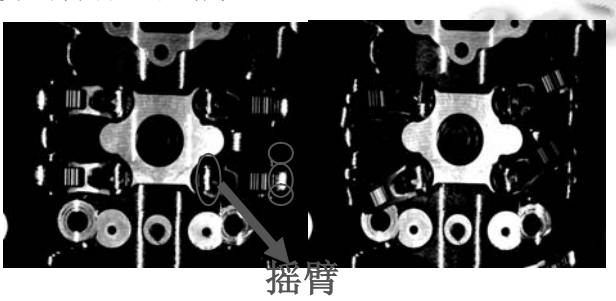
表 1　算法性能比较

图像	像素尺寸	Blob个数	耗时/ms		
			文献[10]算法	文献[11]算法	本文算法
Blobs	329×272	39	18.42	0.085	0.116
Lena	256×256	102	27.25	0.263	0.175
不规则	240×275	243	112.4	0.875	0.329

比较结果表明本文算法的实现效率十分明显，不受所标记图形形状的影响，在目标个数较多时这一优势更加明显. 文献[10]的区域增长法耗时多的原因是不但对目标像素需要扫描两次，目标像素点的邻域比较次数也较多，而且需要较多的堆栈操作时间. 文献[11]在游程数目较少的情况稍具优势，因为其操作比较简单，但随着游程和 Blob 个数的增多，其运算速度明显劣于本文算法，其主要原因是每行的游程都必须与上

一行的游程作比较, 造成游程的冗余, 扩大了搜素空间. 由于本系统中激光照射到气门摇臂后所产生的光斑数大约 100 多个, 所以文献[10]和文献[11]的方法都不适合运用到本系统中.

将本文算法应用到气门摇臂位置检测系统中, 通过比较图 8 中所画的四个区域的连通区域个数和面积, 可以准确查找到气门摇臂安装位置是否正确, 只要四个区域中的一个区域查找到的连通区域在所限定的界限之外, 即判断为摇臂安装错误. 如限定正确安装位置查找的像素个数应为 3000, 图 7(a)中摇臂头部查找到的为 3400, 而(b)中为 1800, 所以判断图 7(b)为安装位置错误的摇臂. 其中图 9, 图 10 为查找 Blob 时的灰度直方图和斑点结果.



(a) (b)

图 8 安装位置正确和错误的气门摇臂

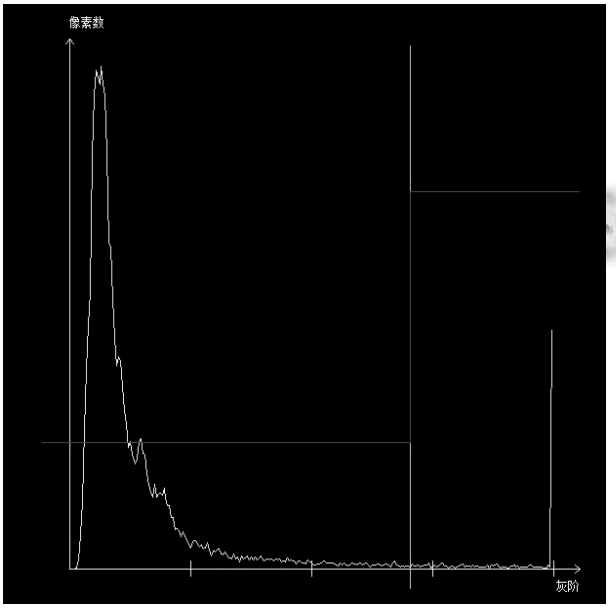


图 9 灰度直方图

在实验中利用 Blob 面积这一特征可以滤除像素数目小于特征值的连通区域, 从而实验噪声的滤除.

N	ID	面积	CenterMassX	CenterMassY	ConnectivityLabel
0	5	1813.97	-377.4	-309.705	1: 斑点
1	8	961.983	-370.916	-211.195	1: 斑点
2	1	325.994	-367.723	-409.159	1: 斑点
3	4	225.996	-377.42	-377.247	1: 斑点
4	11	60.999	-382.615	-188.802	1: 斑点
5	2	42.9993	-383.15	-405.353	1: 斑点
6	12	25.9996	-368.093	-183.487	1: 斑点
7	9	19.9997	-371.627	-237.609	1: 斑点
8	6	15.9997	-370.7	-332.784	0: 孔
9	10	15.9997	-384.473	-208.099	1: 斑点
10	0	15.9997	-369.866	-421.518	1: 斑点
11	7	11.9998	-368.434	-294.62	1: 斑点
12	3	9.99983	-353.563	-401.097	1: 斑点

图 10 查找 Blob 结果

5 算法复杂性、鲁棒性分析

由于文献[10]和文献[11]的传统算法在工业环境复杂时无法准确标记连通区域多且形状复杂的连通区域, 而且对噪声非常敏感, 综合看来本文算法更具优势. 表 2 为两种传统算法和本文算法的时间复杂度和空间复杂度分析, 图像大小为 N×N.

表 2 3 种连通区域标记算法复杂度比较

算法名称	时间复杂度	空间复杂度
文献[10]算法	O(n3)	O(n2)
文献[11]算法	O(n2log)	O(n2)
本文算法	O(n2)	动态分配

6 总结

本文算法运行速度稳定、高效, 适于实时性要求高的场合, 在气门摇臂安装位置检测系统中得到验证. 本文算法充分利用了游程之间的空间信息, 通过特殊的数据结构设计, 无需遍历链表就可以游程节点访问到所属链表的头部和尾部, 进一步压缩了搜索空间, 在算法标记冲突出现时, 立即进行追溯扫描, 对引起误差的标记值及时修改, 减少了算法的重复扫描次数, 具有很好的鲁棒性, 不受标记图形形状的影响.

参考文献

1 Messom CH, Demidenko S, Subramaniam K. Size poison identification in realtime image processing using run length encoding. Instrumentation and Measurement Technology Conference (IMTC)2002. Anchorage, Reading. 2002. 1055–1060.

2 Gupta GS, Win TA, Messom C, Demidenko S, et al. Defect analysis of gritblasted or spraypainted surface using vision sensing techniques. Image and Vision Computing, New

(下转第 95 页)

系统基于 Websphere Portal 实现了用户访问门户. 系统实现分系统服务类、Portlet 类和 jsp 界面三个层次实现.

系统服务类是系统实现的核心类, 实现了集成环境中的资源信息查询和监控. 其中最重要的是 Discover 类和 Monitor 类. Discover 类生成整个社区资源索引列表, 根据用户请求查询资源信息; Monitor 类根据用户请求获取目标资源历史数据.

所有 Portlet 类继承 ActionPortlet 类, 用于与系统服务类交互和管理 Web 页面显示, 是沟通系统服务类和 jsp 页面的桥梁. 其中 UserPortlet 读取并显示用户信息和角色权限、管理用户, 并管理 userinfo 和 help 页面的显示. ResMon 类根据用户请求管理资源信息列表, 绘制并更新资源历史信息图表.

jsp 界面为用户提供查询监控访问界面.

4 结论

本文工作是地震勘探应用网格平台建设的重要组成部分. 系统在网格基础设施 Globus Toolkit 信息服务组件的基础上扩展了信息采集和信息查询接口, 并为地震勘探应用环境下高性能计算资源的信息查询与监控提供了资源实体的统一视图和可视化管理 Portal. 系统实现了位于不同管理域、分布异构的高性能计算资源的信息查询与监控. 系统通过测试, 运行稳定、可扩展性强. 不足之处在于对元调度系统任务调度的决策支持有待进一步扩展. 下一步工作拟在监控资源历史信息的基础上, 对历史数据进行分析以支持作业调度策略的执行. 例如通过分析某个资源节点的 cpu 历史负载信息, 指导元调度系统在调度作业时避免将作业分配到过载节点执行.

参考文献

1 赵连功,刘洪.地震勘探数据资料处理软件集成化研究现状和发展趋势.地球物理学进展,2003,18(4):598–601.

2 Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications, 2001, 15(3): 200–222.

3 Foster I. Globus toolkit version 4: Software for service-oriented systems. Journal of Computer Science and Technology, 2006, 21(4): 513–520.

4 Schopf JM, Pearlman L, Miller N, Kesselman C, Foster I, D'Arcy M, Chervenak A. Monitoring the grid with the Globus Toolkit MDS4. Journal of Physics, 2006, 46(1): 521–525.

(上接第 159 页)

Zealand. 2003. 18–23.

3 Dillencourt MB, Samet H, Tamminen M. A general approach to connected component labeling for arbitrary image representations. Journal of the Association for Computing Machinery(ACM), 1992, 39(2): 253–280.

4 Chang F, Chen CJ, Lu C J. A lineartime componentlabeling algorithm using contour tracing. Technique Computer Vision and Image Understanding, 2004, 93(2): 206–220.

5 张桂林.基于跑长码的连通区域标记算法.华中理工大学学报,1994,22(5):11–14.

6 蔡世界,于强.基于游程编码的连通区域标记算法优化及应用.计算机应用,2008,28(12):3150–3153.

7 徐利华,陈早生.二值图像中的游程编码区域标记.光电工程,2004,31(6):63–65.

8 高红波,王卫星.一种二值图像连通区域标记的新算法.计算机应用,2007,27(11):2776–2778.

9 He L, Chao YY, Suzukik. A runbased twoscan labeling algorithm. IEEE Trans. on Image Processing, 2008, 17(5): 749–756.

10 聂欢欢,伊磊,刘任平.基于区域生长法提取二值图像中的连通区域.计算机时代,2012,6:23–24.

11 胡涛,郭宝平,郭轩等.一种串行/并行两用的区域标记算法.计算机工程,2010,36(9):17–22.