基于轮廓跟踪的连通域标记算法优化①

黄金宝, 周赢武, 罗志灶

(闽江学院 物理与电子信息工程系, 福州 350108)

摘　要: 由于需要大量堆栈操作和反复搜索像素邻域, 一次扫描算法往往效率不高. 基于轮廓跟踪的连通域标记算法先跟踪目标的封闭轮廓, 再线扫描轮廓内的像素, 以减少像素邻域搜索及堆栈访问的次数, 提高算法的效率. 本文提出的基于轮廓跟踪的连通域标记算法,屏弃堆栈访问,并采用高效的轮廓跟踪算法,以提高算法的效率. 本算法与其它连通域算法相比,具有效率更高、稳定性好等优点.

关键词: 二值图像; 连通域; 像素扫描; 标记

Optimizing the Algorithm of Labeling Connected Components Based on Contour Tracing

HUANG Jin-Bao, ZHOU Ying-Wu, LUO Zhi-Zao

(Department of Physics & Electronic Information Engineering, Minjiang University, Fuzhou 350108, China)

Abstract: One scan algorithms of labeling connceted components aren't so efficient because of the need of a large number of stack operations and repeated search pixel neighborhood. The algorithm of labeling connceted components based on contour tracing, could decrease times of searching neighbours and accessing stack, improves it's efficiency, through labeling all contour pixels of one object in image before scanning and labelling it's pixels in image. In order to improve the efficiency, the proposed algorithm of labeling connected components based on contour tracing discards accessing stack, and adopts efficient algorithm of contour tracing. The algorithm has the advantages of higher efficiency, better stability than others.

Key words: binary images; connected components; pixels scanning; labelling

通常二值图像将值为 1 的像素点作为目标像素点, 而将值为 0 的像素点作为背景像素点. 图像的连通域标记是将二值图像中符合某种连通规则[1](4 领域或 8 领域)的目标像素用唯一的相同标号标记. 连通域标记算法是将连通域标记是数字图像处理的重要步骤, 是目标识别的必要的途径, 其算法的优劣直接影响数字图像处理系统的性能.

自 Rosenfeld A[2]提出连通域标算法, 各类连通域标记算法相继提出, 有效地提高了图像连通域标记的效率. 文献[3]分析了连通域算法的发展进程及类别,并将现存的连通域标记算法大致归纳为如下三种: 一次扫描连通域标记算法[4-8], 该类算法要求仅扫描图像一次, 完成连通域的标记, 其主要有基于轮廓跟踪[6]、区域增长法[5,8]、递归搜索算法[4]、等价连通域标号传递[7]等方法; 二次扫描连通域标记算法[9], 也称线扫描标记算法. 该类算法主要是通过二次扫描图像达到标记连通域的目标. 基于并查集[10](Union-Find)的连通域标记算法是有效的改进算法; 多次扫描连通域标记算法[11], 该类算法主要是反复自顶向下和自底向上扫描图像, 并传递临时连通域标号, 直到不再出现连通域标号冲突. 文献[11]提出等价标号快速传递法, 并用决策树(Decision tree)分析 8-邻域或 4-邻域内的像素点的遍历秩序, 以减少邻域内的像素点扫描次数. 目前标记算法[12-14]主要是对算法的并行运行的研究.

文献[6][17]提出基于轮廓跟踪的连通域标记算法, 其思想是: 扫描图像时, 若遇到未标记的目标像素点,

① 基金项目:国家创新训练项目(201410395011)

收稿时间:2016-03-03;收到修改稿时间:2016-04-24　[doi:10.15888/j.cnki.csa.005498]

则该目标点必是目标的轮廓, 从该点开始遍历并标记目标的轮廓; 然后以轮廓目标点为起点, 线扫描目标的像素点并标记; 最终完成目标连通域标记. 该算法可有效地减少目标点的邻域搜索次数, 对于轮廓内的目标点, 用线性扫描方式, 仅需搜索其左侧或右侧的邻域即可.

但基于轮廓跟踪的连通域标记算法[6]仍需用堆栈保存大量的轮廓目标点, 且反复访问堆栈, 严重影响算法的效率; 对内外轮廓交织的目标, 缺乏判断内外轮廓的方法, 特别对与内轮廓连通的目标点, 无法识别, 导致算法不能完整准确标记目标. 文献[14]提出首先遍历轮廓然后线扫描轮廓内的像素点, 以减少邻域搜索的目的, 提高效率明显, 但仍不能准确标记目标连通域, 对于复杂图像仍有漏标记现象, 且仍需堆栈协助.

本文针对基于轮廓跟踪标记算法的上述问题, 利用有效的轮廓跟踪算法[6]及根据内轮廓与目标连通的原理, 提出同适用于内外轮廓跟踪的算法, 并根据不同扫描阶段的目标点的标记状态不同: a)某一目标的外轮廓有且仅有一个, 当线扫描图像时, 最先遇到的只能是某目标的外轮廓目标点, 所以跟踪外轮廓时, 遇到的新的外轮廓目标点均未标记过; b)当每遇一个未标记的轮廓目标点时, 立即以该轮廓目标点为起点, 向右线扫描目标点, 直至遇到另一侧轮廓目标点为止; c)在跟踪轮廓目标点, 若遇到已标记的且标号与当前轮廓目标点不一致, 则表明当前跟踪的轮廓是已标记目标的内轮廓, 则重新开始跟踪该内轮廓, 并修改标号, 使之与已标记的目标点的标号相同, 即实现内轮廓的跟踪.

1 算法描述

1.1 算法原理

基于轮廓跟踪连通域标记算法, 仅需对轮廓目标点执行邻域搜索, 8 邻域或 4 邻域, 而对其它目标点则可线扫描, 只需检查目标点的右侧邻域是否是目标点, 因此基于轮廓跟踪连通域算法较之其它一次扫描标记算法如区域增长法[4][5]相比, 其效率较高. 但如上节所述的问题, 严重影响其运行效率.

对于图像的连通域, 其内外轮廓均是封闭的, 因此从某轮廓上的目标点遍历时, 当遍历结束时, 其终点恰好是起点. 本算法的思想是: 1)自顶向下、从左向右扫描图像, 当遇到未标记的目标点时, 表示遇到未标记的轮廓, 此时轮廓既可能是新的连通域的外轮廓, 也可能是某已标记连通域的内轮廓; 2)以该目标点为起点, 遍历轮廓, 根据算法搜索出下一个轮廓目标点, 并对每个轮廓目标点均以该目标点为起点向右标记目标点, 直至遇到另一侧轮廓点为止; 3)在遍历轮廓目标点, 将会遇到三种情况: a、遇到的轮廓目标点未标记, 则用当前的标号标记该目标点, 并以该点为起点向右标记目标点, 直至遇到另一侧目标点为止; b、遇到的轮廓目标点已标记为当前连通域标号, 则不再向右标记目标点, 而是继续遍历它的下一个轮廓目标点; c、遇到的轮廓目标点已用不同于当前连通域标号标记, 则表明遇到的轮廓点是已标记连通域的内轮廓, 重新从起点遍历轮廓目标点, 并用已标记的连通域标号标记. d、当遇到当前轮廓目标点及下一个轮廓目标点与起始轮廓目标点及其下一个轮廓目标点分别相同, 则轮廓遍历完成.

1.2 轮廓跟踪原理

轮廓扫描是通过某种算法,遍历图像的连通域轮廓. 文献[11]提出高效的轮廓跟踪算法, 假设目标点正上方为 0 邻域点,按逆时针方向 8 邻域点分别标为 0 邻域点、1 邻域点、直到 7 邻域点. 如图 1 所示, 则任意互为相邻的两个像素点, 符合这样的规律:当 A 点为 B 点的 X 邻域时,B 点必是 A 点的按八进制计算(X+4)结果的个位数表示的邻域点. 因此, 假设 A、B 两个目标点互为 8 邻域邻居, 若 B 点是 A 点的第 4 个邻域, 则 A 点是 B 点第 0 个邻域; 若 B 点是 A 点第 3 个邻域, 则 A 点是 B 点第 7 个邻域.

连通域轮廓目标点具备如下的特征: a、轮廓目标点的邻域至少有一个背景; b、连通域的轮廓是连续和封闭的, 可从轮廓某目标点出发遍历轮廓, 遍历完所有轮廓目标点后, 又回到出发点; c、对于轮廓上目标点 p, p 的前轮廓点和后轮廓点分别是 p 的 8 邻域目标点, 按 8 邻域连通规则, 逆时针方向从前轮廓点搜索到后轮廓点时, 遇到的第一个目标点是后轮廓点; d、孤立目标点, 其轮廓目标点是其本身, 且没有后续轮廓点.

1	0	7
2	p	6
3	4	5

图 1 目标点的 8 邻域序列

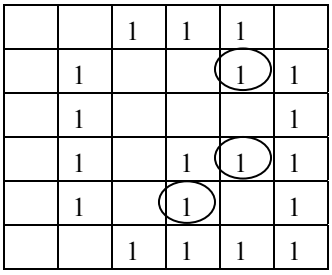


图 2　连通域轮廓示例

因此对于两个紧邻的轮廓目标点 p1、p2, 假设 p2 是 p1 的后续目标点, 从 p1 转入 p2 后, 若要从 p2 处搜索下一个轮廓目标点, 可从 p1 在 p2 的邻域的下一个邻域开始, 可根据式(1)确定搜索邻域的起始位置, 式(1)x 是 p2 在 p1 的邻域位置.

$$d = \mathrm{mod}(x + 4,8) + 1 \tag{1}$$

基于以上特征, 连通域轮廓遍历原理如下:

1) 从上向下、自左向右扫描图像, 当遇到未标记轮廓目标点时, 以该轮廓目标点为起始点, 遍历连通域的轮廓. 从起始目标点的第 2 邻域, 按逆时针方向搜索其邻域, 直到遇到目标点为止, 则遇到的目标点是起始点的后续目标点, 并保存这两个目标点的坐标, 转入后续目标点, 并按式(1)计算后续目标点搜索邻域的起始位置; 若按逆时针搜索其 8 邻域, 未遇到目标点, 则说明该目标点是孤点.

2) 对于轮廓上的任一目标点, 确定下一个目标点时, 则根据式(1)计算搜索邻域的起始位置, 式(1) x 是前轮廓点在当前轮廓点的邻域的位置.

3) 当搜索到的轮廓目标点及它的下一个轮廓目标点与起始目标点及它的下一个目标点坐标位置相同, 则表明轮廓遍历完成.

假设轮廓目标点的前继目标点和后续目标点间的间隔邻域数为 l, 且假设 l 在(1,7)范围内随机出现, 则每个 l 出现的概率 $p_l = 1/7$. 可按式(2)计算出搜索一个轮廓目标点所需平均搜索邻域的次数 s=4 次, 与传统轮廓跟踪算法搜索邻域平均 8 次相比, 效率提高 1 倍.

$$s = \sum_{l=1}^{7} p_l * l \tag{2}$$

当某个轮廓目标点的邻域有多于 2 个轮廓点时, 即连通域的内、外轮廓或多个内轮廓交织情况, 则遍历某轮廓时, 算法无法搜索到另一轮廓点. 如图 2 所示, 当遍历外轮廓时, 与外轮廓紧邻的内轮廓点却无法搜索到, 即圈示的目标点无法搜索到. 因此轮廓跟踪线扫描连通域标记算法会出现少量漏标记或误标记现象, 其主要原因是: 对于多个轮廓交织的情况, 无法完整遍历所有的轮廓点. 当内轮廓遍历时, 内轮廓至少有一个目标点已被标识, 且标号与外轮廓的标号一致, 因此当遇到已标记的目标点时, 则对内轮廓按该标号进行标记, 以解决少量漏标记或误标记现象.

1.3 内外轮廓判定

针对上节所述的轮廓交织会出现少量漏标记或误标记的现象, 本文在连通域内、外轮廓遍历时同时判定该轮廓是否属于内轮廓, 若属于该轮廓是外轮廓, 则用新的标号标记目标点; 若是内轮廓, 则用该轮廓的所属的连通域标号标记目标点.

连通域内外轮廓判定的原则是: 在遍历轮廓目标点时, 若遇到已标记且标号与当前轮廓点的标号不一致时, 则该轮廓必是连通域的内轮廓; 否则是连通域的外轮廓.

依据 1.1 节算法原理, 每搜索到未标记的轮廓点时, 即时以该点为起点向右线扫描并标记目标点, 直至遇到另一侧轮廓点为止.

定理 1.当自顶向下, 从左至右扫描图像时, 最先遇到未标记的目标点时, 此目标点必是左侧轮廓目标点, 即该目标点的第 2 邻域是背景.

证明: 当自顶向下, 从左至右扫描图像时, 如果最先遇到的未标记的目标点不是轮廓目标点时, 则在它左侧必然有目标点存在, 则遇到它之前, 必然会先遇到与它同行的左侧目标点, 如果它未标记, 则它的左侧目标点也未标记. 因此它是左侧的轮廓目标点.

定理 2. 在遍历轮廓目标点时, 若遇到目标点已标记且标号与当前轮廓点的标号不一致的目标点时, 则该轮廓必是连通域的内轮廓; 否则是连通域的外轮廓.

证明: 由定理 1 可知, 当自顶向下, 从左至右扫描图像时, 最先遇到的未标记目标点是轮廓目标点, 若该点是外轮廓点, 则该点所处的连通域还未标记, 不可能有与该点标号不一致的已标记的目标点. 因此若遇到与当前轮廓点的标号不一致的已标记的目标点, 则该轮廓只能是内轮廓.

定理 3. 遍历连通域内轮廓时, 至少会遇到一个已标记且标号与当前轮廓标号不一致.

证明: 根据本文算法, 遍历轮廓的过程中, 当遇

到未标记的轮廓目标点时, 随即向右标记目标点, 直到遇到另一侧轮廓点. 假设某连通域有 n 个内轮廓, 分别用 C1、C2…Cn 表示, 且将它们按它们的最高顶点从上向下排列, 最高顶点相等的轮廓, 按从左到右排列. 对于 C1 内轮廓, 由于在它左侧没有内轮廓, 因此当扫描图像并遇到C1未标记目标点时, 并用新的标号标记 C1 内轮廓时, 外轮廓已完成遍历及向右标记, 且必定会遇到 C1, 因此此时 C1 内轮廓最高顶点已被标记, 且与 C1 轮廓的新标号不一致. 同理, 当自顶向下, 从左至右扫描图像时并遇到某内轮廓 Ci,的未标记的目标点(X, Y)时, 则该点的左侧及上侧的所有目标点均已标记. 根据本文算法, 遍历轮廓的过程中, 当遇到未标记的轮廓目标点时, 用新的标号标记, 因此在遍历内轮廓时, 必定会遇到已标记且标号与当前轮廓标号不一致的目标点. 至此得证.

根据定理 3, 判断内、外轮廓的标准是: 在遍历轮廓时, 是否会遇到已标记且标号与当前轮廓标号不一致的目标点. 若未遇到遇到已标记且标号与当前轮廓标号不一致的目标点, 则继续遍历轮廓, 直到回到起点; 若遇到遇到已标记且标号与当前轮廓标号不一致的目标点, 则从起点重新遍历轮廓, 并用遇到的不一致的标号重新标记轮廓. 在内轮廓遍历时, 不可能遇到多于两个不一致的标号, 因此最多只要重新遍历轮廓一次即可完成内轮廓遍历和标记.

1.4 算法流程

1) 图像预处理, 在图像四周添加一圈背景像素, 将图像包围在内.

2) 自顶向下, 左至右扫描图像, 遇到未标记的目标点 p(x,y)时. 则取新标号 label=label+1;

3) 按 1.1 节算法, 从 p 点开始遍历轮廓目标点, 若遇到未标记的目标点, 则用 label 向右标记目标点,直到遇到背景点为此; 若遇到已标记的且标号与 label 相等的目标点时, 则忽略该目标点继续遍历轮廓目标点; 若遇到已标记且标号不等于 label 的目标点,则从 p 点重新遍历轮廓并用已标记目标点的标号向右标记目标点. 轮廓遍历结束转入 4)

4) 从 p 继续自顶向下, 左至右扫描图像,遇到未标记目标点时,将 p 点指向遇到的目标点, 转入 3).

5) 扫描图像, 不再遇到未标记目标点时, 算法结束.

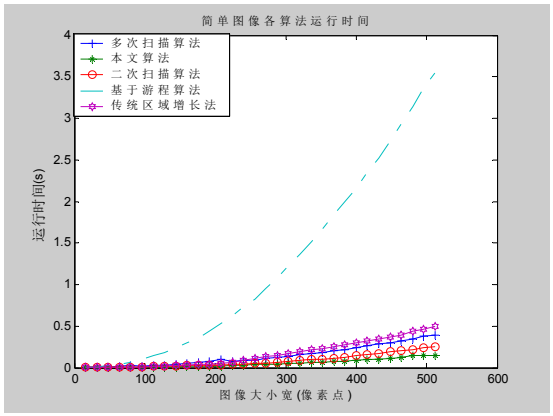
2 实验分析

本文实验图像选自南加州大学图像测试库[18],并将 95 张测试图像按结构复杂程度不同分为简单、中等、复杂等不同的类别, 并将本文算法与常规的一次扫描算法, 如区域增长法[4], 传统轮廓跟踪算法、改进型轮廓跟踪算法[6]、二次扫描算法[9]、多次扫描算法[15]、基于游程算法[16]等进行比较.

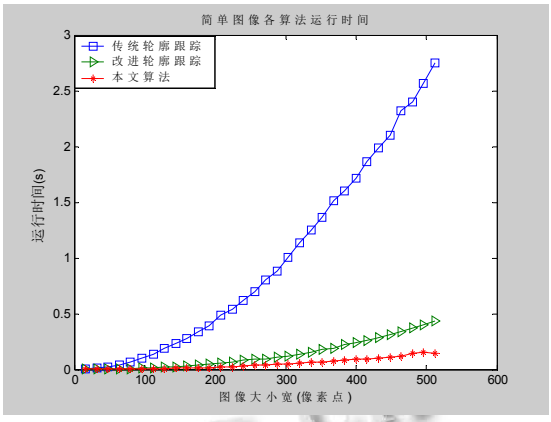
本文对不同复杂程度的图像(512×512), 在 Matlab 环境下, 比较各算法的平均执行时间和执行时间的标准差, 实验结果如表 1 所示. 本文将 512×512 图像, 按 16*n×16*n 划分, 将图像分成 32 幅不同尺寸, 以分析不同尺度图像的各算法的平均执行时间, 结果如图 3、图 4、图 5 所示. 结果表明: 与传统轮廓跟踪算法和改进型轮廓跟踪算法[6]运行时间的比较, 本算法的性能有较大提高; 本算法平均运行时间少于其它类型算法, 且标准差显示本算法具有更好的稳定性; 本文算法与其它算法相比, 随图像尺寸增大, 平均运行时间的增长速度小与其它算法, 这对处理大尺度图像非常有利.

表 1 不同图像特性与各算法性能

分类图像		本文算法	传统轮廓跟踪算法	改进轮廓跟踪算法	游程算法	多次扫描算法	二次扫描算法	传统区域增长法
简单图像	平均时间(s)	0.14871	2.7495	0.43743	3.5515	0.39424	0.25714	0.50052
	方差	0.058802	0.89567	0.1596	0.61385	0.070022	0.16207	0.10931
中等图像	平均时间(s)	0.17179	3.952	0.60733	4.3828	0.41554	0.50205	0.51938
	方差	0.085837	5.505	0.35061	2.18	0.057647	0.99261	0.10003
复杂图像	平均时间(s)	0.19949	4.2406	0.77649	5.0263	0.42917	0.63766	0.4682
	方差	0.074674	2.5514	0.40695	1.8091	0.047578	0.72027	0.1029

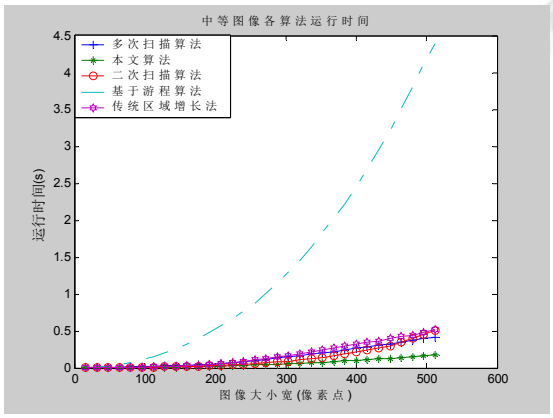


(a)简单图像在不同尺度下五种算法的执行时间(秒)

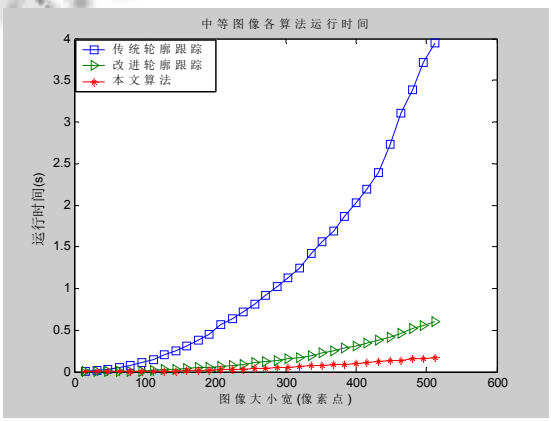


(b)简单图像在不同尺度下三种轮廓算法的执行时间(秒)

图 3　简单图像运行时间

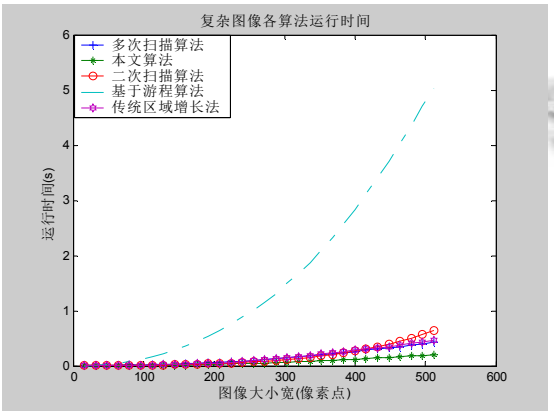


(a)中等图像在不同尺度下五种算法的执行时间(秒)

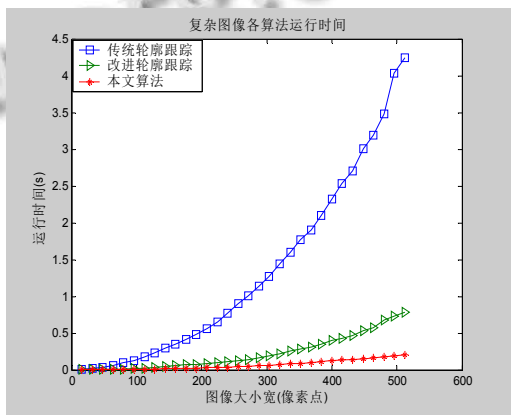


(b)中等图像在不同尺度下三种轮廓算法的执行时间(秒)

图 4　中等图像运行时间



(a)复杂图像在不同尺度下五种算法的执行时间(秒)



(b)复杂图像在不同尺度下三种轮廓算法的执行时间(秒)

图 5　复杂图像运行时间

3　结语

本文分析了现有的连通域标记算法的优劣势，采用高效的轮廓跟踪算法，减少轮廓跟踪的耗费时间；分析本算法的原理，利用不同阶段目标点的标记状态，替代堆栈的访问．实验结果表明本文算法是效率较高，且稳定性较高．

参考文献

1 Gonzalez RC, Woods RE. Digital Image Processing (second edition).北京:电子工业出版社,2006.

2 Rosenfeld A, Pfaltz JL. Sequential operations in digital Picture processing. Journal of the ACM, 1966, 13(4): 471–494.

3 Grana C, Borghesani D, Cucchiara R. Connected component labeling techniques on modern architectures. Image Analysis and Processing, 2009, 5716: 816–824.

4 徐正光,鲍东来,张利欣.基于递归的二值图像连通域像素标记算法.计算机工程,2006,32(24):186–225.

5 AbuBakerl A, Qahwajil R, Ipsonl S, Saleh M. One scan connected component labeling technique. 2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007). 2007. 1283–1286.

6 Chang F, Chen CJ. A component-labeling algorithm using contour tracing technique. 7th International Conference on Document Analysis and Recognition. 2003. 741–745.

7 冯海文,牛连强,刘晓明.高效的一遍扫描式连通区域标记算法计算机工程与应用,2014,50(23):32–35.

8 夏晶,孙继银.基于区域生长的前视红外图像分割方法.激光与红外,2011,41(1):107–111.

9 Samet H, Tamminen M. An improved approach to connected component labeling of images. Proc. of CVPR. 1986. 312–318.

10 Wu KS, Otoo E, Suzuki KJ. Optimizing two-pass connected-component labeling algorithms. Pattern Anal Applic, 2009, 12: 117–135.

11 Suzuki K, Horiba I, Sugie N. Linear-time connected-component labeling based on sequential local operations. Computer Vision and Image Understanding, 2003, 1(89): 1–23.

12 Klaiber M, Rockstroh L, Wang Z, et al. A memory efficient parallel single pass architecture for connected component labeling of streamed images. Field-Programmable Tech Nology, 2012, 10(12): 159–165.

13 赵菲,张路,张志勇,等.基于硬件加速的实时二值图像连通域标记算法.电子与信息学报,2011,33(5):1069–1075.

14 Santiago DJC, Ren TI, Cavalcanti GDC, et al. Fast block-based algorithms for connected components labeling. Proc. of 2013 IEEE International Conference on Acoustics, Speech and Signal. [S.l.]. IEEE. 2013. 2084–2088.

15 崔凤魁,张丰收,等.二值图像目标邻域点法边界跟踪算法.洛阳工学院学报,2001,3(22):28–34.

16 朱云芳,叶秀清,顾伟康.视频序列的全景图拼接技术.中国图象图形学报,2006,11(8):1150–1155.

17 张云哲,赵海,宋纯贺,景巍.一种新的连通区域标记算法.计算机应用研究,2010,11(27):4335–4340.

18 Computer Vision Group.测试图像库.http://decsai.ugr.es/cvg/dbimagenes/g512.php.