基于链码的分水岭变换算法

孙 涵 任明武
(南京理工大学计算机科学与技术系图像处理研究室,南京 210094)

摘 要 为了快速准确地进行图像分割,通过对现有分水岭变换算法的分析,并借鉴图像处理中常用的链码思想,提出了基于链码的分水岭变换算法,并首先扩展了传统链码的定义,将其分为指出链码和指入链码;然后提出并阐述了利用链码实现分水岭变换的两个性质;最后给出了基于链码的分水岭变换算法的具体描述,并详细分析了新算法的时间和空间复杂度。实验结果表明,新算法具有较低的时间和空间复杂度,且变换结果更有利于后续的图像理解。

关键词 计算机图像处理 分水岭变换 链码 连通分支 图像分割
中图法分类号: TP391.41 **文献标识码**: A **文章编号**: 1006-8961(2004)09-1025-07

A Watershed Transformation Algorithm Based on Chain Code

SUN Han, REN Ming-wu
(*Department of Computer Science, Nanjing University of Sci. and Tech., Nanjing* 210094)

Abstract For implementing image segmentation fast and accurately, and by analyzing the existing watershed transformation algorithms as well as considering the idea of chain code this paper presents a chain code based watershed transformation algorithm. In this paper, the traditional concept of chain code is first expanded into point-out chain code and point-in chain code. And then two characters, which depict how to make watershed transformation based on chain code are proposed and discussed. In the end, the detailed description of the chain code based watershed transformation algorithm is presented, and its complexity is analyzed in detail. The new algorithm is different from the traditional watershed algorithms, and it has two main steps: the first step is to generate point-out and point-in chain codes of every pixel by simulating raining, and the second step is to mark the labels of regions by simulating flooding. Experiments show that the new algorithm's time and space complexity is very low. Further more, such transformation result is more helpful for the following image understanding.

Keywords watershed transformation, chain code, connected component, image segmentation

1 引 言

图像分割中的分水岭变换通常是通过模拟浸水过程来实现的[1~3]。在分水岭变换过程中,整幅图像被看作是一片起伏跌宕的山地模型,其中每个像素的灰度值代表该点的海拔。在这样的地形中,既有地势很低的山谷(局部极小区域),也有高高耸立的山脊(分水岭),还有山脊和谷底之间的或陡或缓的山坡(贮水盆地)。模拟浸水就是在各个谷底刺穿一个小孔,然后把整个模型慢慢浸入水中,随着水位的上升,水面慢慢顺着山坡向上扩展,当到达山脊时就会溢出,这时就在此处建筑堤坝,如此直到整个模型浸入水中,所有堤坝就成为分开各个贮水盆地的分水岭。图 1 演示了一维模拟浸水过程。

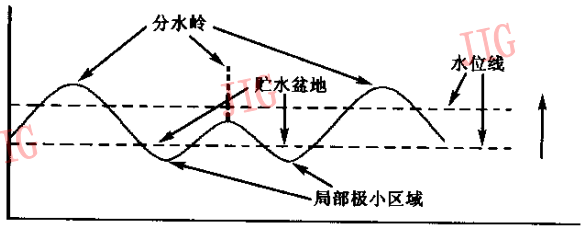


图 1 一维模拟浸水过程示意图

Luc Vincent 和 Pierre Soille 提出的算法(VS

收稿日期:2004-02-20;改回日期:2004-04-30

算法)是采用模拟浸水过程来实现分水岭变换的典型[1],其包括像素排序和模拟浸水两个过程。分水岭变换时,首先,按照像素灰度值递增顺序对图像中所有像素排序,以便直接访问同一灰度值的像素;然后,从最小灰度值开始进行模拟浸水,假设小于和等于 h 灰度级的像素所属的贮水盆地已经标识出来了,则在处理 $h+1$ 灰度级的像素时,就将这一灰度级中与已标记的贮水盆地相邻的像素送入一个先进先出(FIFO)队列,再由这些像素开始,根据测地距离,将已经标注的贮水盆地扩展至 $h+1$ 灰度级,若 $h+1$ 灰度级还有未被标记的像素,则作为新出现的局部极小区域,赋予新的区域标号;最后,在分水岭变换结果中,相同标号的像素属于同一贮水盆地,而将离不同贮水盆地距离相等的像素标记为分水岭点。采用模拟浸水过程的方法来实现分水岭变换虽非常贴近分水岭本身的含义,易于理解,但其算法复杂度高,而且没有完全实现区域分割,即还存在不属于任何贮水盆地的分水岭点。

Bieniek 和 Moga 采用了基于连通分支的方法(BM 算法)来实现分水岭变换[4]。该方法与模拟浸水过程相反,是一个模拟降雨过程,即当雨滴落到山地模型上时,必将沿着山坡流入谷底,雨滴所经过的路线就是一个连通分支,也是雨滴落点到谷底的一条最陡峭路径,而通往同一谷底的所有连通分支就形成了一个贮水盆地。该方法具体操作时,仅需扫描图像 4 遍,即可实现所有贮水盆地的标注。与模拟浸水过程的方法相比,由于该方法无需像素排序,并且不要进行贮水盆地扩展时的出入队列操作,因此其算法复杂度大大降低,而且,能够实现图像区域的完全分割。

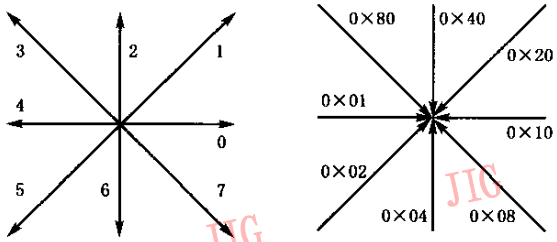
本文受到基于连通分支方法的启发,并借鉴图像处理中常用的链码思想,提出了基于链码的分水岭变换方法。

2 基本思想

本文将传统的链码扩展为指出链码(point-out chain code)和指入链码(point-in chain code),如图 2 所示。

定义 1 指出链码是当前像素指向邻居像素的方向码。

指出链码的取值范围为{0,1,2,3,4,5,6,7,8},其中方向码 8 代表当前像素不指向任何邻居。本文限定每个像素最多只能指向一个邻居像素。



(a) 指出链码 (b) 指入链码

图 2 指出链码和指入链码示意图

定义 2 指入链码是邻居像素指向当前像素的方向码。

由于存在多个邻居同时指向当前像素的情况,所以采用单个字节中的每个比特位分别表示一种指入方向码。这样,指入链码的取值为{0×00,0×01,0×02,0×04,0×08,0×10,0×20,0×40,0×80}中一个或多个 OR 运算的结果,其中 0×00 表示没有任何邻居指向当前像素。

由指出链码和指入链码的定义可知,图像中任意两个像素 p_i 和 p_j 间的连通关系可由链码串表示,即从 p_i 开始,可以由一串指出链码到达 p_j;同时也可从 p_j 开始,通过指入链码回溯到 p_i。在分水岭变换中,由于各个贮水盆地内部的像素是连通的,因此可以用链码来标识和描述贮水盆地。

下面将提出并阐述利用链码实现分水岭变换的两个性质。

性质 1 图像中任意一个像素到对应的局部极小值点的最陡峭路径都可由该像素开始的一串指出链码表示。

说明 当前像素 p_l 到对应的局部极小值像素 p_0 的最陡峭路径 L 是由{$p_l p_{l-1}\cdots p_1 p_0$}这样一串像素组成,其中各像素的灰度值满足如下关系:$f(p_l)\geqslant\cdots\geqslant f(p_i)\geqslant f(p_{i-1})\geqslant\cdots\geqslant f(p_0)(i=l,\cdots,2,1)$,并且像素 p_{i-1} 是像素 p_i 邻居中灰度值最小的一个。使用指出链码来描述上述的最陡峭路径如下:从当前像素 p_l 开始,令各像素 $p_i(i=l,\cdots,2,1)$ 的指出链码指向灰度值小于或等于自己,且是最小的一个邻居,则从像素 p_l 开始的与这串指出链码连接的所有像素就是像素 p_l 到对应的局部极小值点 p_0 的最陡峭路径。由此可见,只要图像中所有像素都有符合要求的指出链码,则任意一个像素都能根据指出链码沿最陡峭路径到达对应的局部极小值点。下面给出满足上述要求的指出链码的生成方法。

由链码定义可知,指出链码的取值由当前像素

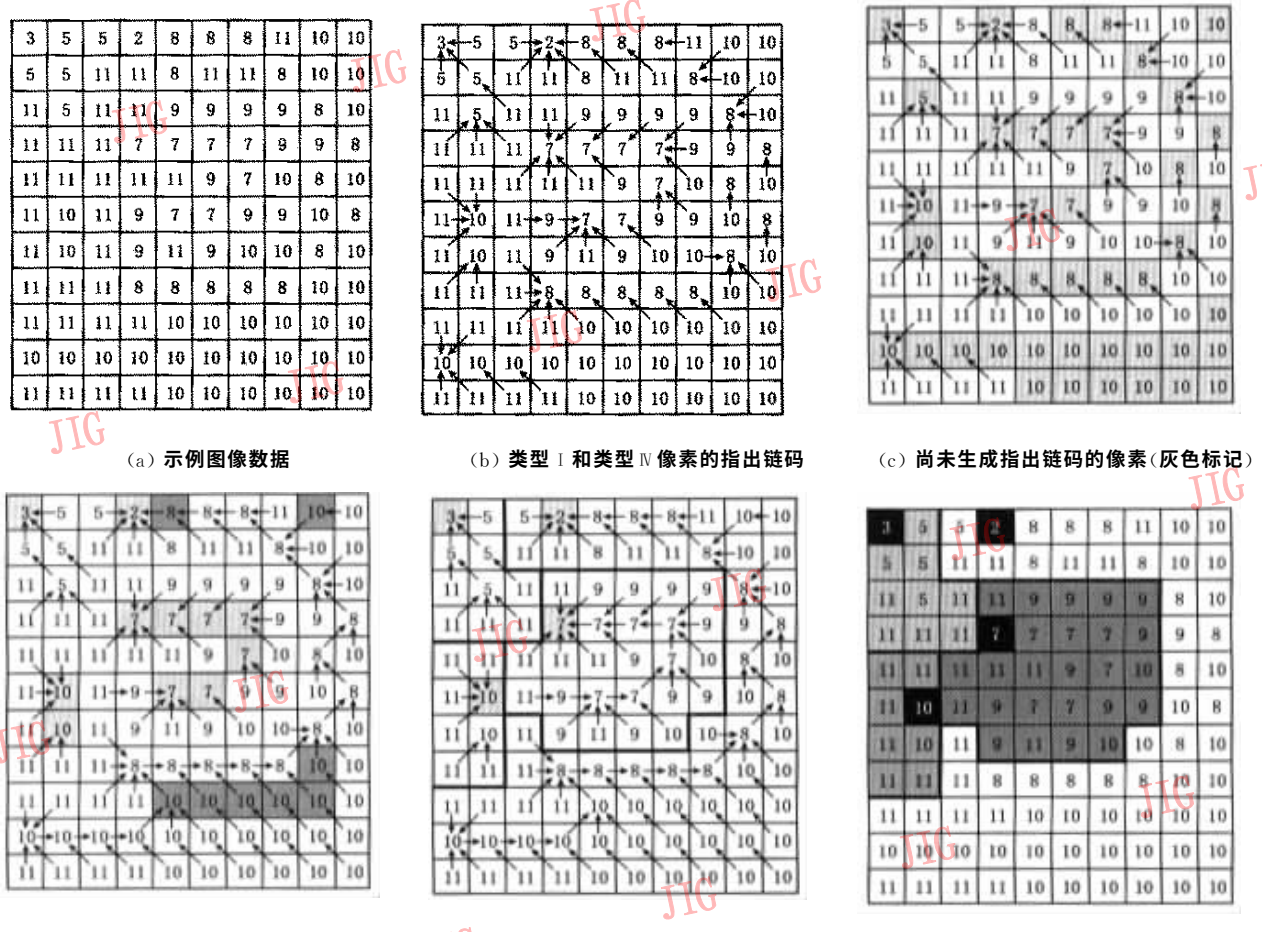
和其邻居像素的关系决定。设当前像素 p 的灰度值为 $f(p)$,邻居像素 N_p 中灰度最大值为 $\{f(N_p)\}_{\max}$,最小值为 $\{f(N_p)\}_{\min}$,则当前像素与其邻居的灰度值大小关系共有 4 种(见表 1)。

表 1　当前像素与其邻居像素灰度值之间的关系

关系类型	当前像素与其邻居像素灰度值之间的关系描述
类型 Ⅰ	$\{f(N_p)\}_{\min}<f(p)<\{f(N_p)\}_{\max}$
类型 Ⅱ	$\{f(N_p)\}_{\min}=f(p)=\{f(N_p)\}_{\max}$
类型 Ⅲ	$\{f(N_p)\}_{\min}=f(p)<\{f(N_p)\}_{\max}$
类型 Ⅳ	$\{f(N_p)\}_{\min}<f(p)=\{f(N_p)\}_{\max}$

显然,指出链码的取值完全由上述的 4 种关系类型决定。下面针对这 4 种关系类型来论述图像中各像素指出链码的生成过程。本文使用文献[4]中的演示数据(图 3 (a))为例。该图像大小为 11×10,各像素的位置为 (i,j),其中 $0\leqslant i<11$,$0\leqslant j<10$。这幅图像中就包含了上述 4 种类型的像素,例如,(0,2),(0,4)等像素属于类型 Ⅰ;(9,5),(9,6)等像素属于类型 Ⅱ;(0,5),(3,3)等像素属于类型 Ⅲ;(1,2),(4,2)等像素属于类型 Ⅳ。

生成各像素的指出链码时,首先,生成属于类型 Ⅰ 和类型 Ⅳ 的像素的指出链码,由于这两种类型的像素有一个共同点,即存在灰度值比自己小的邻居,因此,可令这两类像素的指出链码指向灰度值最小的邻居,如果满足条件的最小邻居不止一个,则指向第 1 个最小邻居,例如,对图 3(a)中这两种类型的像素进行上述操作,结果见图 3(b),共有 67 个满足条件的像素生成了相应的指出链码,而剩余未生成指出链码的 37 个像素在图 3(c)中则用灰色标记;



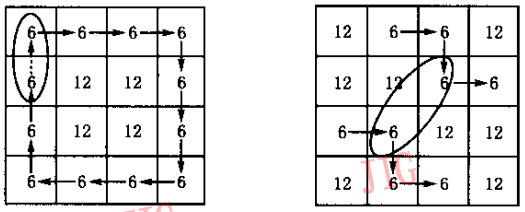
(a) 示例图像数据　(b) 类型 Ⅰ 和类型 Ⅳ 像素的指出链码　(c) 尚未生成指出链码的像素(灰色标记)

(d) 贮水盆地坡面平台区域和宽分水岭区域内像素的指出链码(深色箭头标记)　(e) 所有像素的指出链码　(f) 分水岭变换结果

图 3　基于链码的分水岭变换示意图

然后,生成属于类型 Ⅱ 和类型 Ⅲ 的像素的指出链码,由于这两种类型的像素灰度值与其邻居的最小灰度值相等,因此不能像上述的那样简单决定其指出链码,否则有可能出现链码环和链码断裂现象,例如,图 4(a)中像素(1,0)的指出链码,若指向邻居像素(0,0),则形成一个链码环。在本文的算法中,由

于一旦出现链码环，后续的依据指入链码进行的回溯过程就无法终止，因此，必须避免形成这样的链码环。同时，也要避免同一区域内的链码发生断裂，否则会将一个区域分割成多个，如(如图 4(b)所示)。



(a) 链码环 (b) 链码断裂

图 4 两种可能出现的问题

观察发现，图像中这两种类型的像素会在局部极小区域、贮水盆地坡面上的平台区域和宽分水岭的内部区域 3 种情况下出现。再从上一步的结果来看，又可以分成以下两类：一类是灰度值与自己相同的邻居已标记指出链码，这类像素出现在贮水盆地坡面平台区域和宽分水岭区域的边界，如图 3(c)中的像素(0,5)，(0,9)等；另一类则都没有标记指出链码，这类像素出现在局部极小区域、贮水盆地坡面平台区域内部和宽分水岭区域内部，如图 3(c)中的像素(3,3)，(1,7)和(10,5)等。

由以上分析可知，标记属于类型Ⅱ和类型Ⅲ的像素的指出链码需要分以下两步完成：首先，标记贮水盆地坡面平台区域和宽分水岭区域内像素的指出链码，若当前像素未标记指出链码，但存在灰度值相同，且已标记指出链码的邻居，则将这个邻居像素作为“种子”放入一个先进先出队列；然后依次从队列中取出像素，同时令像素值与之相同，且还未标记指出链码的邻居指向当前像素，并将这些邻居作为新的“种子”放入队列，重复上述操作直至队列为空。该过程是从区域边界的“种子”像素出发，采用扩展的方法来实现贮水盆地坡面平台区域和宽分水岭区域中像素的指出链码的标记。现以图 3(c)为例来说明该标记过程。首先扫描图像，将满足上述条件的像素放入队列，首批进入队列的 9 个像素在图 3(d)中已用深色标记，最终结果见图 3(d)。

现在仅剩局部极小区域中的像素未标记指出链码。本文采用另一种方法进行标记，即在局部极小区域中，若当前像素未标识指出链码，但存在灰度值相同，未标识指出链码的邻居，则先令所有满足此条件的邻居都指向当前像素，同时并把这些邻居压入堆栈；然后依次从堆栈中弹出像素，再观察弹出的像素 p 是否也存在灰度值相同的未标识指出链码和指入链码的邻居，若存在这样的邻居，则令其指出链码指向像素 p，并将这些邻居压入堆栈，重复上述操作，直至堆栈为空。这样，每个贮水盆地内仅剩一个像素没有标记指出链码(如图 3(e)中的 4 个灰色像素)。本文令其指出链码取值为 8，即不指向任何邻居，或者可以认为指向本身。它们就是各个贮水盆地对应的局部极小值点。显然，上述标记方法有效避免了链码环和链码断裂现象的出现。

到目前为止，本文已经给出了图像中所有像素指出链码的生成方法。例如，图 3(e)就标出了图 3(a)中各个像素的指出链码。也可以用实例来说明，例如，图 3(a)中像素(0,9)到对应的局部极小值点(0,3)的最陡峭路径就是在图 3(e)中这样的一条指出链码串：(0,9)→(0,8)→(1,7)→(0,6)→(0,5)→(0,4)→(0,3)→(0,2)。

实际上，任意像素通过指出链码串到达对应的局部极小值点的过程就是一个模拟降雨过程。

性质 2 贮水盆地可从对应的局部极小值点开始的指入链码回溯得到。

说明：由性质 1 可知，属于同一贮水盆地的所有像素均可通过一串指出链码到达该贮水盆地对应的局部极小值点，并且在指出链码的形成过程中，就已生成了相应的指入链码。由于局部极小值点没有灰度值比自己更小的邻居，它的指出链码值就为 8(不指向任何邻居)，但存在有效的指入链码，因此可从这点开始，根据指入链码回溯，凡是能回溯到的像素，必属于该贮水盆地。因为指出链码串中不存在环，所以回溯过程到达没有指入链码的像素就终止。上述过程说明，贮水盆地可从对应的局部极小值点开始的指入链码回溯得到，例如图 3(f)。

由性质 2 可知，从局部极小值点开始回溯标注贮水盆地的过程相当于一个模拟浸水过程。

3 算法描述与复杂度分析

3.1 算法描述

根据上述分析，本文提出的算法具体步骤如下：

(0)初始化。

令图像中各像素的指出链码初始值为 8，各像素指入链码初始值为 0。

(1)标记属于类型Ⅰ和类型Ⅳ的像素的指出链码和指入链码。

扫描图像:

设当前像素 p 的像素值为 $f(p)$,邻居像素 N_p 中灰度最小值为 $\{f(N_p)\}_{\min}$。

若 $f(p)>\{f(N_p)\}_{\min}$,则像素 p 的指出链码指向灰度值最小的邻居 n_p,n_p 的指入链码更新值为原有值与像素 p 指入的方向码进行 OR 运算的结果。

(2)标记贮水盆地坡面平台区域和宽分水岭区域内像素的指出链码和指入链码。

①扫描图像:

若当前像素虽未标记指出链码,但存在值相同,且已标记指出链码的邻居,则将这个邻居像素添加到队列的尾部。

②while(队列非空)

{ 从队列头取出一个像素 p;

若存在还未标识指出链码的邻居 n_p,且 p 和 n_p 的像素值相同,则令邻居 n_p 的指出链码指向像素 p,同时更新像素 p 的指入链码,并将 n_p 放入队列。

}

(3)标记局部极小区域内像素的指出链码和指入链码。

扫描图像:

若当前像素未标识指出链码,且存在值相同,也未标识指出链码的邻居,则

{ 令满足上述条件邻居指向当前像素,并把这些邻居压入堆栈。

while(堆栈非空)

{ 从堆栈中弹出一个像素 p;

若存在还未标识指出链码的邻居 n_p,且 p 和 n_p 的像素值相同,则令邻居 n_p 的指出链码指向像素 p,同时更新像素 p 的指入链码,并将 n_p 压入堆栈。

}

}

(4)根据指入链码回溯标注各个贮水盆地。

①当前区域标号 $current\_label$ 初始值为 1。

②扫描图像:

若当前像素的指出链码值为 8(局部极小点),则

{ 从该像素的指入链码开始回溯,凡能回溯到的像素,其所属区域标号均为 $current\_label$;

$current\_label=current\_label+1$;

}

3.2　算法复杂度分析

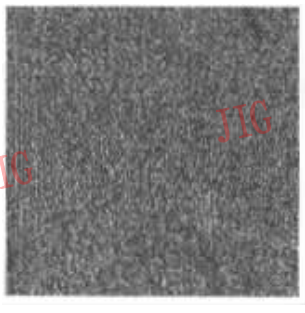
假设待变换的图像总像素数为 n,属于类型 Ⅰ 和类型 Ⅳ 的像素数为 n_1,贮水盆地坡面平台区域和宽分水岭区域内的像素数为 n_2,局部极小区域内的像素数为 n_3,显然,$n=n_1+n_2+n_3$。上述算法中,第 1 步在处理类型 Ⅰ 和类型 Ⅳ 的像素时,需遍历图像一遍,时间复杂度是 $O(n)$;第 2 步在处理贮水盆地坡面平台区域和宽分水岭区域内的像素时,将所有符合要求的像素加入队列的操作也需遍历图像一遍,时间复杂度为 $O(n)$,而将这类像素从队列中取出,并标记的时间复杂度为 $O(n_2)$;第 3 步处理局部极小区域内的像素时,需遍历图像一遍,时间复杂度为 $O(n)$;第 4 步标注贮水盆地时还需遍历图像一遍,时间复杂度也为 $O(n)$。综上所述,整个算法的时间复杂度为 $O(4n+n_2)$。

下面分析算法的空间复杂度。该算法需要的存储空间包括:原始图像存储空间、与原始图像同样大小的指出链码和指入链码存储空间,以及一个用于队列和堆栈操作的辅助空间。由于队列和堆栈操作是串行的,因此可以共用同一存储空间。

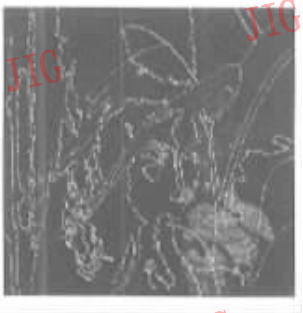
4　实验分析与结论

为验证本文算法效果,对一组经典的原始图像(图 5(a),图 6(a),图 7(a))进行了分割对比实验。实验中,首先求取它们的形态学梯度,然后在梯度图像上分别采用 VS 算法、BM 算法和本文的算法进行分水岭变换,实验对比数据见表 2。图 5(c),图 6(c)

(a) Lena 原始图像(512×512)



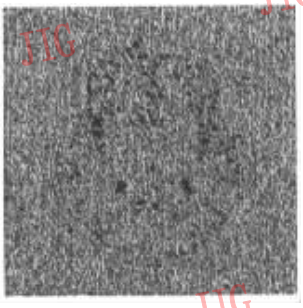
(b) 分水岭变换结果(13 257 个区域)



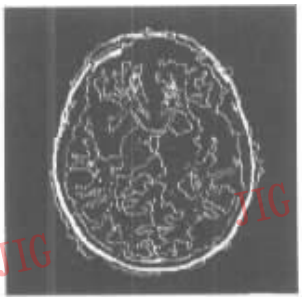
(c) 区域合并结果(32 个区域)

图 5　Lena 图像分水岭变换分割结果

(a) 颅骨切片原始图像(388×395)

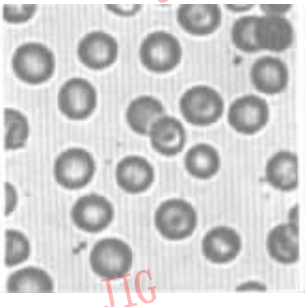


(b) 分水岭变换结果(8 579 个区域)

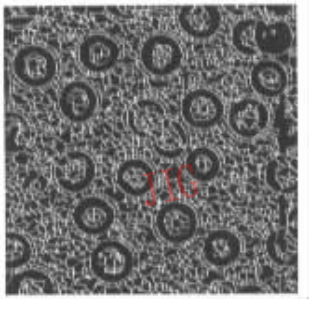


(c) 区域合并结果(11 个区域)

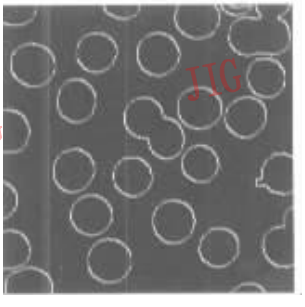
图 6 颅骨切片图像分水岭变换分割结果



(a) 细胞原始图像(272×265)



(b) 分水岭变换结果(1 963 个区域)



(c) 区域合并结果(28 个区域)

图 7 细胞图像分水岭变换分割结果

和图 7(c)是针对分水岭变换中通常出现的过度分割进行区域合并的结果,区域合并采用的是文献[5]中的快速区域合并算法。

从表 2 中的数据可以看出:(1)3 种算法的最终分割区域个数相同,这说明这 3 种算法在功能上是等价的;(2)由于 VS 算法需要对像素进行排序操作,因此 VS 算法的时间复杂度远大于 BM 算法和本文提出的算法,而且本文算法的时间复杂度还略低于 BM 算法,因为本文的时间算法复杂度是 $O(4n+n_2)$,而 BM 算法的时间复杂度是 $O(4n+n_2+n_3\log n_3)$;(3)VS 算法没有实现完全分割,如宽分水岭上的像素未归并到任何区域中。这种现象虽会影响分割区域的大小,但并不影响分割区域的个数,因为分割区域个数取决于分水岭变换中的贮水盆地个数,所以这 3 种算法在确定贮水盆地的原理上是等价的。

另外,本文提出的分水岭变换算法得到的结果更利于后续处理,因为首先本文算法可以从任意一个局部极小点开始,无需遍历图像,仅根据指入链码就可以得到相应的贮水盆地,而 BM 算法若不遍历图像,则无法实现;其次,本文算法在根据指入链码进行回溯操作的同时,可以得到各个区域的相关信息,例如区域大小、区域灰度均值、方差等,而 BM 算法在相应的步骤中则无法实现。

表 2 3 种分水岭变换算法实验对比数据

测试图像	比较参数	VS 算法	BM 算法	本文算法
Lena 图像 512×512 (图 5(a))	分割区域个数	13 527	13 527	13 527
	执行时间(ms)*	352	90	86
	分水岭上的像素数	49 928	0	0
颅骨切片图像 388×395 (图 6(a))	分割区域个数	8 579	8 579	8 579
	执行时间(ms)*	205	58	53
	分水岭上的像素数	29 682	0	0
细胞图像 272×265 (图 7(a))	分割区域个数	1 963	1 963	1 963
	执行时间(ms)*	53	26	25
	分水岭上的像素数	8 455	0	0

* 注:在 P4 1.8GHz 的微型计算机上的执行时间

本文在分析已有的分水岭变换方法的基础上,借鉴图像处理中的链码思想,提出了基于链码的分水岭变换方法,从一个新的角度解决分水岭变换这个经典问题。实验结果表明,基于链码的分水岭变换算法不仅时间和空间复杂度都很低,且变换结果更有利于后续的图像理解。

参 考 文 献

1 Luc Vincent, Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations[J]. IEEE

Transactions on Pattern Analysis and Machine Intelligence, 1991, **13**(6):583～598.

2 Hagyard D, Razaz M, Atkin P, *et al*. Analysis of watershed algorithms for grayscale images[A]. In: Proceedings of IEEE International Conference on Image Processing[C], Lausanne Switzerland, 1996, **3**:41～44.

3 **杜啸晓,杨新,施鹏飞. 一种新的基于区域和边界的图像分割方法[J]. 中国图象图形学报**, 2001, **6A**(8):755～759.

4 Bieniek A, Moga A. An efficient watershed algorithm based on connected components[J]. Pattern Recognition, 2000, **33**(3):907～916.

5 Haris Kostas, Efstratiadis Serafim N, Maglaveras N, *et al*. Hybrid image segmentation using watersheds and fast region merging[J]. IEEE Transactions on Image Processing, 1998, **7**(12):1684～1699.

孙　涵　1978 年生,2000 年获南京理工大学计算机系学士学位,现为南京理工大学计算机科学与技术系博士研究生。感兴趣的研究领域主要为数字图像处理、模式识别与计算机视觉。

E-mail:henrysun2000@vip.sina.com

任明武　1969 年生,2001 年获南京理工大学计算机系博士学位,现为南京理工大学计算机科学与技术系副教授。主要研究方向为数字图像处理、计算机视觉、图像压缩与编码。

《中国图象图形学报》

关于公式及正文中符号表示的一些约定

根据有关规定,对外文符号作如下约定:

1、凡是量符号均用斜体表示,其中标量用白斜体,矢量(向量)、矩阵、集合用黑斜体(整数、有理数、自然数集用正黑体)。

2、凡代表某一名称(含意)的符号用白正体,如:最大(max)、最小(min)、左(left)、右(right)、边缘(edge)有时用一个字母 e 表示,若用斜体则成了一个量符号,所以正斜体一定要区分。一般名称符号用于限定量符号,放量符号的上下角,如:$X_{\max}$,A_{left},A_{edge}。

3、一篇文章中,同一个量用同一符号,不同量用不同符号(包括希文字母、英文字母、大小写、黑体、白体),如,一个字母的黑、白体即为两个符号,因此不要一个量有时用黑体,有时用白体,或有时用大写,有时用小写(易误解为两个量)。同一个量正文中与公式中应一致。

4、一个符号表示一个量,最好不用多个字母代表一个量,主量符号居中,其余限定(说明)符号放上、下角。如 A 代表面积,海洋的面积则为 A_{sea},第 i 块海洋面积则为 A_i^{sea}。一般限定符号均放下角,但有时有几个限定符号,既有正体,又有斜体,不好放一起,正体的符号可放上角(因不易与量的幂相混)。若量符号放上角,又不代表量的幂,则应加括号。如 $A_i^{(n)}$,指迭代 n 次形成的第 i 块输出。

5、单位符号(如 s、m、dB),微分符号 d,三角函数符号 sin、cos,自然对数底 e,圆周率 π 等用正体。