一种新的基于链码描述的轮廓填充方法

任明武 杨静宇 孙 涵
(南京理工大学计算机科学与工程系图象处理研究室,南京 210094)

摘 要 基于链码描述的轮廓填充是图形图象处理的基础算法,已广泛应用于图象处理、目标分析、图象压缩和计算机图形学中,但存在需要较大的辅助空间和速度较慢的问题,为此,在分析现有算法的基础上,提出了一类基于将整条码链的填充分解成子链填充的算法,这样每条子链都是简单封闭轮廓.与现有算法相比,该算法最大仅需要与码链等大的辅助空间,而且在非二值图象或码链允许更改时,可不需要任何辅助空间;另外,该算法既不需要排序操作,也不需要用人工交互的方式给出种子,即可通过在子链中根据相邻链码的值来自动给出种子.理论和实验表明,该方法能正确填充任意复杂形状的轮廓,并具有实现方便、速度快、算法简单、易于理解等特点.此快速简单算法具有很大的应用价值.

关键词 轮廓填充 链码 填充算法
中图法分类号:TP391.4 **文献标识码**:A **文章编号**:1006-8961(2001)04-0348-05

A New Contour Filling Algorithm Based on Chain Codes Description

REN Ming-wu, YANG Jing-yu, SUN Han
(*Department of Computer Science image processing Lab, Nanjing University of Science and Technology, Nanjing* 210094)

Abstract Contour filling is one of the most common problems in image and graphics processing and plays a very important role in many in many applications, such as image analysis, pattern recognition and computer graphics. The paper proposed a new concept of sub-chain and a new sub-chain based filling algorithm. The new algorithm is based on separating the whole contour into many simple sub contours and then does filling in each sub contour. Compared with the conventional algorithms, the new algorithm has five advantages. (1) Its speed is faster and its implementation is easier. (2) It needs less storage-just need size-fixed temporary memory or none in many cases. (3) It finds seeds for filling automatically and easily. (4) It needs neither sorting operation nor discrete Green's theorem to find the seeds for filling. (5) It can be implemented with flexible forms according to the different background of applications. Theory and experiments prove it can fill any complex regions with more convenient, faster speed, more simple, more applicable. It can be used widely in applications of graphics and image processing.
Keywords Contour filling, Chain Code, Filling Algorithm

0 引 言

基于链码描述的轮廓填充是图形图象处理的基础算法[1],鉴于其在图形图象处理领域中占有重要地位,且已广泛应用于图象处理、目标分析、图象压缩和计算机图形学中,因此研究完善和高效的填充算法,在今天仍然具有很大的应用和研究价值.

传统的填充算法大致可以分为奇偶性检测[2]和连通性填充[3]两大类别.前者是基于"一条直线与任何一条封闭曲线(一个区域的轮廓)相交偶数次"的原理,首先对扫描线与轮廓的交点进行计数,同时对处于两交点之间的线段进行编号,并定义在编号为奇数的线段上的所有象素是区域内部的点;后者(又称种子填充)则需要给定一个区域内部的点作种子,并定义由种子出发能达到轮廓,而又不穿过轮廓的所有象素是区域内部的点,而且前者在因采样精度不够而把原始轮廓上的多个点采样成一个点时,或轮廓中存在着与其相交的线条时,则会产生错误的线段编号,而导致错误填充,而后者在很多应用中只能使用人机交互的方式给出种子,而且对于存在多个不连通区域内部的轮廓,则必须给出多个种子.由于轮廓形状复杂多样,因此很多奇偶性和连通性填充的改良算法[1,4]最终也只是部分地克服了某些问题.

收稿日期:2000-03-21 改回日期:2000-08-30

随着链码[5]技术的广泛应用，很多文献[6~9]提出了当轮廓是以链码形式给出时的填充算法，且它们都能取得比传统算法正确而快速的填充结果．虽文献[6]～[8]仍使用奇偶性填充，但它可以分析被填充区域轮廓的 8 链码编码属性，并将轮廓点分成孤立点、标志点和忽略点、不存在点，从而很好地解决了删除孤立点或将一个孤立点当作两个点的问题，这样就能避免产生错误的线段编号，从而取得正确的填充结果．根据文献[9]介绍的基于以 Freeman 编码[5]表示的轮廓和离散 Green 定理[10]，从而可把链码当作符合 Green 定理要求的闭合单向曲线，然后通过计算从某一点出发的对整条轮廓线的积分，即可判断该点是否是区域内部的点(即种子)．

本文提出了一类新的基于子链的快速轮廓填充算法，与现有算法[6~9]相比，该算法最大仅需要与码链等大的辅助空间或不需要任何辅助空间，且速度更快，不仅实现简单，而且运用灵活．本文中轮廓点的颜色设为 $boundary\_color$，可是区域内部现有象素的颜色均不为这一颜色；其轮廓点描述为 $(x_0, y_0)\, d_0 d_1 \cdots d_{n-1}$，方向为逆时针，$(x_0, y_0)$为跟踪起点，$d_i$ 为链码；填充色为 $fill\_color$．

1　现有填充方法的缺陷

现有基于链码的填充算法大致可以分为以下 2 类：(1)局部分析法或称为轮廓点分类法[6~8]．该方法即通过分析每一个轮廓点的进入链码和出发链码，以判定该轮廓点的类型，看其是否能作为填充起点或该点是否既是起点又是终点．采用奇偶性填充算法进行分类，虽分类速度很快，但需要较繁的数据结构；(2)全局分析法或称为种子寻找法[9]．该方法是基于离散格伦(Green)定理，并通过计算从某一点出发的整条轮廓线的积分，来判断该点是否是区域内部的点，其采用的是种子填充算法，其需要很大的计算量，但不需要复杂的数据结构．

CAI 提出了对轮廓点按链码属性进行分类，来解决奇偶性填充算法存在的问题[6]．其基本思想是临时移去在 X 方向孤立的轮廓点(孤立点)，而使保留下来的轮廓点(填充标志点)在 X 方向完全符合常规的奇偶性填充算法要求，最后在 X 方向进行奇偶性填充，填充完毕后，再加入被临时移去的孤立点，最终得到正确完整的填充结果．由于一个轮廓点可能被经过多次，即可能被多次分类，因此该算法采用了一个与图象 A 等大的辅助内存 B，以便分别标记标志点和孤立点，然后 A、B 相互参照即得到赋值．由于 CAI 算法中将填充标志点记录在图象 A 中，因此它在填充时，需要扫描图象 A 中的每一个象素来寻找填充标志点，当图象的宽度和高度值相等，均为 N 时，则进行比较次数为 N^2．

CHANG 和 Frank 提出了一种快速、低内存消耗的算法[7,8]．它扩展了 CAI 对轮廓点的分类，即首先将孤立点(0)细分为孤立点(0)和忽略点(3)；然后使用一个辅助表，为保证此表符合奇偶性填充原则，只将孤立点和标志点置入表中，孤立点(0)作为 2 个相同的点进入表中，而标志点(1)则直接进表；最后对表中的点按坐标值排序，再顺序从此有序表中取标号为奇数的点对进行填充，便完成了最终填充．然而排序需花费大量时间，而且难以事先确定表的大小，故应用不便．若以每点 4byte 计算时，则辅助内存至少为 $4 \times n$byte，且为此至少需付出 $o(n\log_2 n)$次的比较和大量的移动操作．

Tang 先后提出了离散格伦定理[10]和基于此定理的填充算法[9]．该填充算法是在 X 方向采用种子填充方法，即在每一水平扫描行上寻找待填充的各水平线段的起点，若此起点是区域内部点，则从此点填充到该线段终点即可．因为给每个水平线段寻找种子，所以该算法不受多连通子区域的影响，且其最大优点是不需要辅助内存，然而确定一个点是否是内部点的积分，都必然要用到所有的轮廓点，且判断一个点至少需要 $2n$ 次比较、$2n$ 次减法和 n 次加法．当一条轮廓跨度 m 行时，其填充一般至少需要进行 m 次内点判决．一般来说，该算法为寻找填充起点所付出的最少时间代价为 $2mn$ 次比较和 $3mn$ 次加减，因此当链码个数较多或轮廓较复杂时，将花费大量时间．

2　子链填充法

实质上，基于链码的填充算法，其最大难题是因为轮廓不能保证连续光滑，即会存在一些点被跟踪多次的问题．对此问题的不同考虑，就导致了上述算法的差异，例如先消除孤立点或孤立点算作 2 次，以及孤立点被多次经过时的重新分类等均会导致上述算法的差异．虽然轮廓点分类法的优点是可以对轮廓点进行快速分类，然而需要额外的内存空间来记录填充标志点，并对它们进行排序，而基于全局分析的种子寻找法则不需要额外的内存空间来记录种子，也不需要排序，然而寻找种子要花费大量的时

间.本文算法的基本思想是:(1)研究如何对码链进行分解,即如何将码链分解为多条各自封闭的简单轮廓(本文中称为子链);(2)在简单轮廓中,分析研究利用轮廓点分类法对轮廓点进行快速分类的优点和种子寻找法不需要排序的优点,以便能直接寻找种子.

2.1 码链分解方法

本文定义的子链就是封闭的简单轮廓,在子链中每个轮廓点仅被跟踪 1 次.本文中的链码 d_i 指的就是 Freeman 链码[5],其定义如图 1 所示.

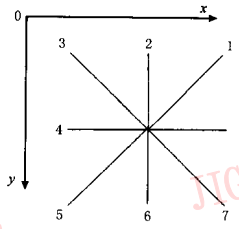

图 1 Freeman Code 链码

在码链中,除跟踪起点(x_0,y_0)是先出发,而最后到达外,其他的轮廓点肯定是先到达后出发,且进入和出发肯定是成对出现.这样被跟踪多次的轮廓点肯定被多次到达和出发.如果一个轮廓点 $p_{x,y}$ 再次由链码 d_k 到达时,且其已有出发链码 d_i,则存在的闭合子链即为($p_{x,y}$) $d_i \cdots d_k$,而 $p_{x,y}$ 则称为割点.码链的分解就是从整个码链中删除 $d_i \cdots d_k$,继续检测删除,如此反复,直到整个链变空为止,从而得到全部的子链.由于割点 $p_{x,y}$ 是公共的轮廓点,因此它存在于不只一条子链中,例如,在图 2 中,□是起点,○、△是割点,原始链为□71771104434535,方向为逆时针,将其分解可得到 3 条子链,其中子链 Ⅰ 为△04,子链 Ⅱ 为○77114345,子链 Ⅲ 为□7135,方向为逆时针.

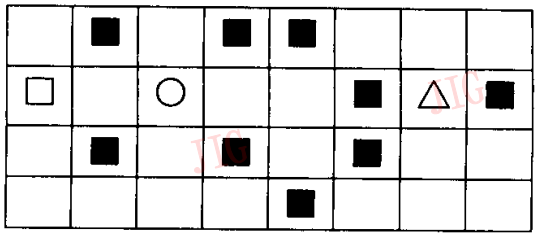

图 2 子链分割示意图

2.2 子链中的种子寻找

本文采用行内种子填充法来对每条子链进行填充,即在子链中,本文定义的填充起点是其右边区域内部点的轮廓点,而填充过程则是从填充起点开始从左到右填充,直到碰到另一个轮廓点(文中称为填充终点)为止.子链中轮廓点的类别是通过分析到达该轮廓点的码 d_i 和从该轮廓点出发的码 d_{i+1} 而得到的,如图 3(a)至(h)所示,图上 B 点为当前点,d_i 表示 $A \rightarrow B$ 的路径.在 8 连通时,当 d_{i+1} 为 $B \rightarrow 1$ 时,则 B 是填充起点;当 d_{i+1} 为 $B \rightarrow 0$ 时,则 B 不是填充起点,而 $B \rightarrow 2$ 或 $B \rightarrow$① 是不可能出现的,因为此点肯定是区域外部点(背景点),另外,由于 8 连通包含了 4 连通,因此在 8 连通时不可能出现的码组合,却可能出现在 4 连通中,$B \rightarrow$① 表示在 4 连通时,是填充起点.

图 3(a)至(h)的正确性可用反证法证明.设连通性为 8 连通,B 的右边点为 $p_{x,y}$,当 $d_i = 1$ 时,若后继路径为 $B \rightarrow 1$,则 $p_{x,y}$ 肯定是内部点,否则根据连通性,即说明所有的 $p_{x+j,y}$ 都是外部点,且轮廓若要闭合,则 B 必定被经过多次,这违反简单轮廓的约定.当 $d_i = 2$ 时,则 $p_{x,y}$ 肯定不是区域内部的点,否则 $A \rightarrow B$ 就违反逆时针定则(应该 $A \rightarrow p_{x,y}$).对图 3(a)至(h)进行综合即得到表 1.

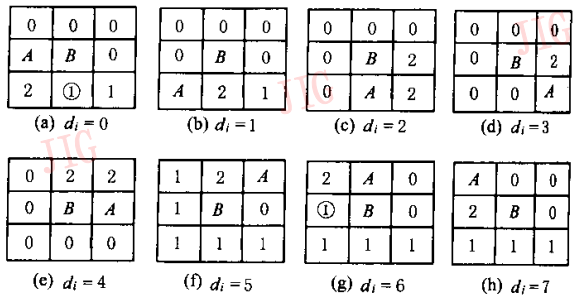

图 3

表 1 逆时针走向时的填充起点判定表 RenLut

d_i	d_{i+1} 0	1	2	3	4	5	6	7
0	0	0	0	0	0	2	①	1
1	0	0	0	0	0	0	2	1
2	2	0	0	0	0	0	0	2
3	2	0	0	0	0	0	0	0
4	0	2	2	0	0	0	0	0
5	0	0	2	1	1	1	1	1
6	0	0	0	2	①	1	1	1
7	0	0	0	0	2	1	1	1

图 3 及表 1 中 0 代表不是填充起点,1 代表是填

充起点，2 代表在 8 连通时不可能出现的码组合，①虽也代表 8 连通时不可能出现的码组合，但它是 4 连通时的填充起点.

证明 1　在简单轮廓中，填充起点和填充终点成对出现.

证明：设 $p_{y,x}$ 为任意填充起点，则根据填充起点的定义可知，最长线段 $p_{y,x+1}p_{y,x+2}\cdots p_{y,x+j}(j\geq 1)$ 肯定是由区域内部的点组成. 根据连通性准则可知，$p_{y,x+j+1}$ 轮廓点即为所求的填充终点.

算法 0　水平线段的种子填充算法

FillOneLineSegment(x, y, incode, outcode, boundary_color, fill_color)

step1 如果 RenLut[incode][outcode]等于 1，那么
{x = x + 1；
while ((x,y)点的颜色值不等于
boundary_color)
将(x,y)点的颜色值改为 fill_color；x = x + 1；}

step2 结束

2.3　算法实现

根据码链分解原理，即可以从原始链中依次移走各个子链，直到原始链变空. 这样，在算法实现时，就可将已经访问过的轮廓点颜色值改为 *boundary_color*，若当前访问轮廓点的颜色值已经为 *boundary_color*，则表示已发现了以当前访问的轮廓点为割点的一条子链；发现子链后通过逆行即得到其链码，且边逆行，边判断和填充；当子链填充完毕后，即可将其从原始链中删除，因为是边逆行边填充，所以不需要额外的内存空间来存储种子，而且在原始链不允许改动时，可事先备份之. 下面给出一个用 C 语言编写的伪程序，以简单说明该类方法的算法实现过程. 假定 $d_0d_1\cdots d_{n-1}$ 存放在数组 code 中.

算法 1　REN.1

step1　$x_1 = x_0$；$y_1 = y_0$；codenum = n；i = 0；
pixel(x_0,y_0) = boundary_color；

step2　$x_1 = x_1$ + offsetx[code[i]]；$y_1 = y_1$ + offsety[code[i]]
/* 计算当前轮廓点的坐标 */

step3　if pixel(x_1,y_1)不等于 boundary_color 那么
pixel(x_1,y_1) = boundary_color；i = i + 1；goto step2；
否则执行 step4 到 step6 完成子链填充
/* 找到了割点为(x_1,y_1)的一条子链 */

step4　$x_2 = x_1$；$y_2 = y_1$；j = i /* 初始化子链的信息 */

step5　$x_2 = x_2$ + offsetx[(code[j] + 4) mod8]；
$y_2 = y_2$ + offsety[code[j] + 4) mod8]
/* 逆行计算子链上轮廓点的坐标 */

step6　if x_2 等于 x_1，并且 y_2 等于 y_1，那么
FillOneLineSegment(x_2, y_2, code[i], code[j], boundary_color, fill_color)，子链填充完毕，执行 step7 到 step8，完成链的调整；否则，FillOneLineSegment(x_2, y_2, code[j - 1], code[j], boundary_color, fill_color)；j = j - 1；goto step5
/* 执行填充并判断该子链是否填充完毕 */

step7　for (k = j；k < codenum；k = k + 1)
code[k] = code[i + 1 + (k - j)]；
codenum = codenum - (i - j + 1) /* 删除子链 */

step8　if code 不等于 0，那么 i = j，goto step2，继续检测子链；否则结束./* 检测子链，直到链空 */

其中，mod 为取模操作，*offsetx* 和 *offsety* 分别为下一个轮廓点相对于当前轮廓点在 X 和 Y 方向的坐标偏移量，由图 1 可知，在链码分别为 {0,1,2,3,4,5,6,7} 时，*offsetx* 则分别为 {1,1,0,-1,-1,-1,0,1}，*offsety* 分别为 {0,-1,-1,-1,0,1,1,1}.

虽然在简单轮廓子链中进行填充的算法复杂度极小，但关键的问题是如何实现子链的删除或跳过已被填充的子链. 根据具体的应用环境，至少有下述 5 种方法：

(1)在原始链不需要保存时，算法可以直接从原始链中删去已填充的子链；

(2)在 *boundary_color* 不等于 *fill_color* 时，因为已填充子链的颜色已经被赋值为 *fill_color*，而未填充子链的颜色为 *boundary_color*，所以算法只需要通过颜色来跳过已经被填充的子链即可；

(3)很多实际应用中，考虑到方便处理等原因，链码采用 8bit 存储，但考虑到字节对齐，有时也采用 4bit 存储. 由于这样就存在冗余比特，因此算法可使用该比特来标记已经被填充子链的链码；

(4)在象素用 1 个 byte 以上的空间存储时，由于子链中的填充起点肯定是填充区域的填充起点，因此整个区域的填充起点即是各子链中填充起点的并集. 可用此象素的 3 个比特(b6b5b4)来记录最新到达链码的值，用 4 个比特(b3b2b1b0)来记录最新出发链码的值加 1，而剩余比特(b7)则用来标记是否曾是填充起点. 每当该轮廓点再次到达时，即根据其记录的出发链码(b3b2b1b0)和当前到达链码来进行填充起点的判定，若是填充起点，则将剩余比特(b7)置 1；同时，重新修正用于记录最新出发链码的 4 个比特(b3b2b1b0)的值. 而且只要一个轮廓点曾经是

一条子链的填充起点,即使它也是另条子轮廓的非填充起点,那该点仍是整个区域的填充起点.这样即可首先得到所有的填充起点,尔后再进行填充;

(5)使用大小为链码个数的辅助内存,来记录一个链码的前一个链码的编号,即逆行后退时,应使用的链码编号,再通过动态修正其编号,则逆行时即可跳过已经被填充的子链,此方法可减少算法 REN.1 中 step7 所花费的时间,方法(1)、(2)、(3)、(4)不需要额外的辅助内存.由于具体的算法实现很简单,故本文不再讨论.

图 4 Lena 图象的尺寸为 512×512,8bit 存储;轮廓为 176 个,8 连通,目标颜色为白色;链码总个数为9 420;最长链为4 913.

图 4 Lena 测试图象

使用本文方法在 Pentium Ⅱ 400 微机上对图 4 Lena 图象的测试结果如表 2 所示.

表 2 算法效率比较表

	辅助内存(byte)	100 次填充用的时钟数
Chang	1.33n8n 变化	4 680
Tang	0	422 350
REN.1	0.375n,定长	2 850
REN.2	0	1 270
REN.3	0	1 260
REN.4	0	980
REN.5	4n,定长	1 370

注:n 为链码个数

因为算法 Chang 和算法 Cai 均属于一种类型,且性能略优,所以本文只与算法 Chang 进行比较.其中,REN.1、REN.2、REN.3、REN.4、REN.5 分别是采用方法(1)、(2)、(3)、(4)、(5)运算的结果.其辅助内存指标使用原始链中链码个数 n 的倍数字节表示.通过比较可知,本文方法速度较 Chang 算法提高了 1~5 倍,且不需要辅助内存,或内存将减少 2 到 20 倍,且定长.

3 结 论

本文给出了逆时针链码的快速填充算法,根据同样原理,可以很容易地得到顺时针链码的填充算法.因为该算法是 8 链码的填充,所以也适用于 4 链码.

本文提出了将整条链码分割为多条各自封闭的简单子链,进而对子链进行填充的方法.由于该算法的最大优点是最大仅需要与码链等大的辅助空间,且在非二值图象或码链允许更改时,可不需要任何辅助空间,因而应用更方便;而且由于不需要逻辑或运算及不需要为维持表的有序性付出排序时间,因而速度更快;决定任意点是内部点的运算代价小于离散格伦定理方式.基于子链的链码填充算法在实际应用中,具有大的灵活性,可根据具体的应用环境,设计不同的实现方法;该算法具有 较广泛的应用性和实用性.

参 考 文 献

1 Pavlids T. Algorithm for graphics and image processing, Washington: Comput. Sci., 1982.
2 Ackland B D, Weste N. The edge flag algorithm—A fill method for raster scab display. IEEE Trans. Comput. 1981, 30: 41-47.
3 Sahni U. Filling regions in binary restore images. In: SIGGRAPH'80, 1980: 321-327.
4 Merrill R D. Represntation of contours and regions for efficient computer search, Commun, ACM, 1973, 16(2): 69-82.
5 Freeman H. Computer processing of line drawing images, Comput. Surveys 1974, 6: 57-97.
6 Cai Zuguang. Restoration of binary images using contour direction chain codes description. CVGIP, 1988, 41: 101-106.
7 Chang Long-Wen, Leu Kuen-Long. A fast algorithm for the restoration of images based on chain codes description and its application. CVGIP, 1990, 50: 296-307.
8 Frank Y. Shih, Wai-Tak Wong. An improved fast algorithm for the restoration of images based on chain codes description, CVGIP, 1994, 56: 348-351.
9 Tang G Y. Region filling with the use of the discrete Green theorem, CVGIP, 1988, 42: 297-305.
10 Tang G Y. A discrete version of green's theorem. IEEE Trans. Pattern Anal. Mach. Intell. 1982, PAMI-4(3): 242-249.

任明武 1969 年生,博士研究生,南京理工大学计算机系讲师.主要研究方向为数字图象处理、计算机视觉、图象编码与压缩.

杨静宇 1941 年生,教授,博士生导师,现任南京理工大学信息学院院长.主要研究方向为人工智能、机器人等.

孙 涵 1978 年生,硕士研究生.研究方向为数字图象处理、计算机视觉等.