1997 年 11 月 重庆大学学报 (自然科学版) Vol. 20, №. 6
第 20 卷第 6 期 Journal of Chongqing University (Natural Science Edition) Nov. 1997

# 一种改进的并行细化算法*

## An Improved Parallel Thinning Algorithm

何 宇 张太怡 陈廷槐
He Yu Zhang Taiyi Chen Tinghuai
(重庆大学电子信息工程学院,重庆,400044;第一作者 26 岁,男,博士生)

**摘 要** 分析了现有的一种并行细化算法中的象素删除条件,并指出在该条件下,不仅孤立四象素阵列会完全消失,而且会引起细化结果中的"蚕食"反应和"毛刺"现象。然后本文提出了改进的算法,并证明了算法结果为严格八连通。并以实例说明了该算法的优点。

**关键词** 图象;细化;并行;连通 / 蚕食;毛刺
**中国图书资料分类法分类号** TP 391.41

**ABSTRACT** One recent parallel thinning algorithm is analyzed on detail in this paper. The analytical emphasis is put on its pixel-deletion conditions. It is shown that owing to the inherent faults in these conditions, not only all four elements of an isolated 2x2 square are removed but also there exist "worm-eaten" response and "thin-hair" phenomina in its result skeleton. Subsequently, an improved one is proposed which is also strictly 8-connected. Experiments indicate that the modification effectively overcomes the original's shortages and has a little better time efficiency.

**KEYWORDS** image; thinning; parallel; connectivity / worm-eaten; thin-hair

## 0 引 言

细化算法由于在图象预处理中的重要性,一直受到普遍关注。所谓细化,就是在保持原图像拓扑结构的情况下,尽可能快地抽出一个单象素宽的骨架。在大多数情况下,要求细化结果能达到八连通。

20 年来,人们已提出了很多种细化算法,Suen 等人[1]将细化算法分为两类,一类是以非象素为基础的方法[2],也就是利用图象的轮廓信息及综和特征来进行细化。

另一类方法则是以象素迭代删除为基础的方法[3~6],即用一个 3×3,4×4,5×5 或 $n\times n$ 窗口放在图象上,然后使用一个查找表来确定究竟是删除,还是保留中心的 1 象素,这个查找表是预先就根据判定规则生成好了的。用窗口运算符来执行细化的优点在于,操作简单且可作为一种并行算法来执行,从而大大加快图象预处理速度。

但目前完全并行的算法(每次迭代只扫描图象一次)还存在一些问题,如在[5]中,Holt 提出一种 4×4 窗口基于 Zhang[3]的全并行算法,但随即 Hall[6]证明,该算法的并行速度其实是和 Zhang 的串并行方法(每次迭代须扫描图象两次)一样的。

* 收文日期 1997-01-13

因此，比较完善的还是串并行算法，周新伦于 1994 年提出了一种类似算法(以下简称 Zhou 算法)[4]，该算法的窗口运算符含有 11 个象素，如图 1 所示。

$$\begin{array}{ccccc} & & q_0 & & \\ & n_4 & n_5 & n_2 & \\ q_1 & n_6 & p & n_1 & \\ & n_6 & n_7 & n_8 & \end{array}$$

图 1　Zhou 算法算子

其中，$p$ 为 1 象素，Zhou 算法先通过一些条件保证图象的四连通性，然后对称地删除 8-可删点(8-deletable，即删除后不影响图象 8 连通性的点)，来实现最终骨架的严格八连通。Zhou 算法还有速度较快，易于用硬件实现的优点。

## 1　算法分析

Zhou 算法中有下列条件，使 1 象素 $p$ 如同时满足，则被删除：

1) $X(p) = 2$；

2) $N(p) > 2$；

3) $n1 = 0$ or $n3 = 0$ or $n5 = 0$ or $n7 = 0$.

其中 $X(p)$ 为 $p$ 的交叉数(cross number)，即 $p$ 的 8 邻域象素中 0-1(或 1-0) 依次翻转数，$N(p)$ 为 8 邻域中的 1 象素个数。

由于对于孤立的四象素阵列，其模式满足 1)2)3) 条件，且在 Zhou 算法中又无相应的保留模式与之匹配，故在迭代中将被完全删除。这就使细化结果未完全保持拓扑不变。

下面讨论条件 3). 设 1 象素 $p$ 满足条件 1)2) 而不满足条件 3)，则易知 $p$ 的 $3 \times 3$ 邻域内只有一个 0 象素，而其余的七个 1 象素两两 4- 连通，其邻域模式有下面四种可能：

$$\begin{array}{ccc|ccc|ccc|ccc} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & p & 1 & 1 & p & 1 & 1 & p & 1 & 1 & p & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}$$

图 2　防止侵蚀的领域模式

这时 $p$ 被保留，防止了这四种模式下的侵蚀，但这样有一些危险，即当 $p$ 位于下面任何一种模式时：

$$\begin{array}{ccc|ccc|ccc|ccc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & p & 1 & 1 & p & 0 & 1 & p & 1 & 0 & p & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ \multicolumn{3}{c|}{(a)} & \multicolumn{3}{c|}{(b)} & \multicolumn{3}{c|}{(c)} & \multicolumn{3}{c}{(d)} \end{array}$$

图 3　未防止侵蚀的模式

也就是当 p 的八邻域中仅有一个 4- 邻象素值为 0 时(对应于图中 $a,b,c,d$)，因 p 满足 1)2)3) 条件，于是就被删除了，这就使得 p 邻域中的某些 1 象素也有被删的可能，并进而引起一连串的删除，形成较严重的侵蚀。

$$\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & r & p & 0 \\ 1 & 1 & 1 & 1 \end{array}$$

图 4　"蚕食"反应之一

如以图 3($b$) 为例，设 p 左邻域象素 r 为内部象素(即 8 邻域内全为 1 象素)，见图 4，则据 *Zhou* 算法条件 3)，当 p 被删后，在下一迭代中，r 也将会被删掉。

类似地，若 r 的左邻域也为内部象素，则也将在下一迭代中被删掉，如此递推，如笔道足够粗，就会被侵蚀得越来越厉害。形象而言，就象一棵树，正在被一条蛀虫往左蛀一个越来越深的洞。我们可把这种现象称为“蚕食”反应。

若图 3 中的模式结合起来，侵蚀就会向多个方向进行，“蚕食”反应更明显。比如将 c) 和 d) 相结合，如图 5.

从上面的分析可知，$u$ 和 $r$ 也是下次被删的对象。此时，侵蚀是沿右和上两个方向进行的。

总之，由于 Zhou 算法未顾及到可能引起侵蚀的其他四种模式，使得在细化过程中存在着一种危险：当笔道边缘模式与它们相匹配时，就会引起“蚕食”反应，且笔道越粗，“蚕食”反应越明显。“蚕食”反应的后果是造成最后骨架的局部极不规则。

```
1  1  1
1  u  1
1  p1 1  1  1
1  0  p2 r  1
   1  1  1  1
```

图 5 “蚕食”反应之二

另外，对条件 2)，由于算法中忽略了 $N(p)$ 为 2 的情况，使得细化结果中有可能出现“毛刺”现象。下面以汉字笔画中常见的勾为例。

```
        *  *  *  *  *                          1  2  *  2  1
*  *  *  *  *  *  *  *        *  *  1  1  2  *  2  1
   *  *  *  *  *  *  *        1  *  *  *  4  2  1
      *  *  *  *  *  *        1  2  *B 3  2  1
      *  *  *  *  *           1  2  *A 2  1
         *  *  *  *           1  2  1  1
         *  *                 1  1
           (a)                    (b)
```

图 6 “毛刺”现象

其中，a) 为一典型勾笔画的象素模式，b) 为 Zhou 算法的细化结果，b) 中标 1,2,3,4, 的象素分别表示在第一，二，三，四次迭代中被删除的象素，‘*’为细化结果中被保留下来的象素，显然 $A,B$ 为“毛刺”。这是因为在第三次迭代中，由于 $N(A)=2$，所以 $A$ 被保留下来，并进而禁止了 $B$ 在后续迭代中被删除。易知，如在条件 2) 中令 $N(p)>=2$，则 $A,B$ 将先后被删，从而消除“毛刺”。

## 2 新算法描述

针对 Zhou 算法的缺点，我们在修改的基础上给出另外一种算法，叙述如下：

迭代的第一遍(pass) 中，当 1 象素 p 邻域不满足下面模式：

a)
```
      0
   1  1  0
0  1  p  0
   0  0  0
```

且满足下列条件 b)c) 中任一个时，$p$ 被删除：

b) ①$X(p)=2$；

② $N(p)>=2$ and $N(p)<7$;

③ $n1=0$ or $n7=0$ or ($n3=0$ and $n5=0$).

c) 邻域模式与下面任一个匹配时：

```
  1  0        0  0
0 p  1   或   0 p  1
0  0           1  0
```

迭代的第二遍只须将 b)③ 改为 $n3=0$ or $n5=0$ or ($n1=0$ and $n7=0$),将 c) 改为：

```
  0  0        0  1
1 p  0   或   1 p  0
0  1           0  0
```

说明：

1) 条件 a) 用于防止孤立四象素阵列的消失。由于条件 b) 的 ③ 已可保证双象素宽横竖笔道不断裂，故不再需要对应的匹配模式。

2) 条件 b) 的 ② 保证不删去所有 $N(p)=7$ 的象素，使"蚕食"反应失去了产生的前提，从而有效阻止了破坏性的侵蚀，该条件还考虑了 $N(p)=2$ 的情形，这就有效地剔去了"毛刺"。

3) 容易看出，本算法也易于用硬件实现。

4) 易证本算法所得骨架是严格八连通的，下面简要给出：

设 1 象素 $p$ 不为此骨架的特征点，即 $p$ 不是骨架中的孤立点，端点，也不是交叉点，则易知 $N(p)=2$，且 $p$ 与这两个邻域象素八连接，以下证明这两个邻域象素间非八连接。用反证法，设这两个象素(记为 $A$,$B$) 八连接，则此时 $p$ 的邻域模式只有下列两种可能：

```
0 0 0          0 0 0
0 p x          y p y
0 x A          x A x
  a)             b)
```

图 7　连通性证明

其中，$x$,$y$ 为 $B$ 可能的位置，易知其它模式与 a),b) 同构。

1) 当为 a) 模式时，则满足本文算法的条件 b)，故 $p$ 应被删去，矛盾；

2) 当为 b) 模式时，如 $B$ 位于 $x$ 处，则与 1) 类似可得出矛盾；

当 $B$ 位于 $y$ 处，则满足 c) 条件，故 $p$ 也应被删去，矛盾；

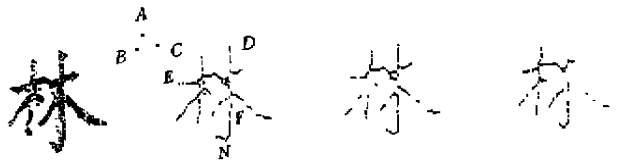
所以每一非特征点象素均只与相邻两象素八连接，故此骨架为严格八连通。

## 3　实验结果

我们将两种算法性能用实验作了比较，实验结果如下：

其中，a) 为中文字符"林"，b) 为 Zhou 算法的结果，c) 为 Zhou 算法中令 $N(p)>=2$ 后所得的结果，d) 为本文算法的结果。将它们加以比较，可以看出：

1) 由于在(a)中的 A,B,C 处存在图 3 中的模式(其中 A 对应于图 3(b)，B 对应于组图 3




(a) (b) (c) (d)

图 8 实验结果比较

(c),C 对应于图 3(c)和 3(d)的结合),结果就在(b)中对应的 A,B,C 处形成了非常明显的"蚕食"反应,而在本文算法下,(d)中无此反应。

2) 下表中列出了 Zhou 算法与本算法的速度比较,可见本算法速度较快。

附 表 Zhou 算法与本算法的速度比较

| 算法 | 迭代次数 | 时间 (t) |
|---|---|---|
| Zhou | 6 | 0.65 |
| 本文 | 6 | 0.62 |

3) 比较(b),(c),(d) ,可以发现(b)中 N 处的毛刺在(c),(d)图中消失了,这是因为后面两个算法考虑了 $N(p)=2$ 的情形。

4) 通过比较还发现 Zhou 算法对轮廓噪声的敏感性要大于本算法,如(b)中的 D,E,F 处,这说明本算法对轮廓噪声有较大的免疫力。

## 4 结 语

本文提出了一种并行算法(Zhou 算法)的改进方案,不仅保持了 Zhou 算法的优点(较高速度和利于用硬件实现),还严格避免了"蚕食"反应的出现,进一步保持了拓扑结构的不变性。另由于本算法能消除"毛刺",且对轮廓噪声有较大的免疫力,所以本算法有更好的适应性。

## 参 考 文 献

1 Iam L,lee S W,Suen C Y. Thinning Methodologies-A Comprehensive Survey. IEEE Trans. PAMI, 1992,14(9): 869～883

2 Kwok P C K. A Thinning Algorithm by Contour Generation,Comm ACM 1988,31(11): 1314～1324

3 Zhang T Y,Suen C Y. A Fast Parallel Algorithm for Thinning Digital patterns. Comm ACM 1984, 27(3): 236～239

4 周新伦. 一种并行细化算法及其硬件实现方案. 模式识别与人工智能, 1994,7(1): 1～6

5 Holt C M,Stewart A,Clint M et al. An Improved Parallel Thinning Algorithm,Comm ACM 1987,30(2): 156～160

6 Hall R W. Fast Parallel Thinning Algorithms:Parallel Speed and Connectivity Preservation. Comm ACM 1989,32(1): 124～131