# 改进的脱机手写体汉字细化算法①

魏 玮，刘亚宁

(河北工业大学 计算机科学与软件学院，天津 300401)

**摘 要**：在手写体汉字识别中，细化是关键的步骤。数学形态学细化方法是近年来汉字细化采用的主要方法。细化效果的好坏取决于结构元素的选取。在对细化算法进行大量分析的基础上，针对手写体汉字的特点对原有算法的结构元素进行了改进，细化的对象由原图像改进为归一化后图像，文中算法使用了 23 组汉字，每组 100 个汉字进行实验。实验表明，改进后的算法解决了原有算法的端点信息丢失问题，降低了算法的运行时间。

**关键词**：汉字；归一化；细化；数学形态学

## Improved Off-Line Chinese Character Thinning Algorithm

WEI Wei, LIU Ya-Ning

(College of Compute Science and Engineering, Heibei University of Technology, Tianjin 300401, China)

**Abstract:** In the recognition of off-line Chinese char, thinning is an important step. Mathematical morphology algorithm is the most useful thinning method in recent years. The results are heavily influenced by the structuring element we used. With the large amount of the analysis of the thinning algorithm and the character of the Chinese char, we improved the previous structuring element. The objects are altered to unitary char. 23 group of Chinese char have been used to experiment and each group have 100 Chinese char. The results show that this algorithm solved the problem of losing the ending information and reduce the running time.

**Keywords:** Chinese character; unitary; thinning; mathematical morphology

## 1 引言

细化，也被称之为骨架化，是广泛应用于图像处理与模式识别的一个重要的图像预处理过程。图像的细化是特指在保持原图像拓扑结构的情况下尽可能快地抽出一个单像素宽的骨架的过程，极大地消除图像中的冗余信息。因此图像的细化算法研究一直受到图像处理领域的普遍关注。在汉字识别领域，细化后的汉字不仅能突出汉字的形状特点和减少汉字的冗余信息，而且更有利于提取汉字的骨架信息，进而为汉字的特征提取和识别作准备。一个好的细化算法需要满足以下几条要求[1,2]：(1)骨架图像的连通性必须与图像保持一致；(2)骨架图像的线条宽度尽可能为单像素；(3)骨架图像中的线条应尽量是原图像的中心线；(4)骨架细化算法应尽可能快。

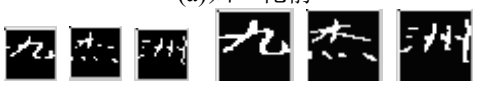
细化算法按照处理每个像素的方式的不同，可以分为两大类，串行算法和并行算法[3]。串行算法在每一次对图像进行循环处理时，采用固定的次序来处理每一个像素(如依次搜索图像象素的左、上，右、下边经轮廓像素)，对像素属性的判断，是依据当前像素的实时处理结果 ，即利用前一次循环处理的结果以及本次循环中那些已被处理过的像素的结果；并行算法在本次扫描处理每个像素时，只利用了上一次扫描处理的结果，而与本次扫描过程中其余像素的处理情况无关，这样，每个像素的处理过程都具有独立性，适宜于并行处理器进行处理。一种有效的快速细化算法[3]是基于 Han 提出的 weight-value 的概念，进而提出了改进的细化算法。一种快速的手写体汉字细化算法[4]利用 FPA 细化算法的思想提出了利用消除模板删除多

① 基金项目:河北省自然科学基金(08M006）
收稿时间:2010-09-28;收到修改稿时间:2010-11-05

余像素的同时可以保证交叉点不变形。索引表细化算法是基于查表的方法进行字符的细化，具体方法是根据某点的八邻域点情况制作出一张表，共 256 个元素，然后根据当前处理点的情形查表决定是否删除。以上算法虽然在一定程度上可以满足细化的要求，但处理时间比归一化后字符处理时间高。如果处理的字符量庞大则处理时间将有较大的区别。

在手写体字符的识别过程中，字符的大小不一也是影响识别率的重要因素之一，因此在字符特征提取之前需要对待识字符进行归一化操作，归一化后图像可以在细化过程中减少字符扫描的行数和列数且不会产生细化后再对字符进行归一化的变形问题。经过实验，汉字字符归一化为 25×25 个点阵象素会导致汉字的笔划断裂，又因手写汉字均小于 64×64 个象素，归一化为 64×64 象素反而会增加处理时间，因此文中算法选择将汉字归一化为 36×36 象素。如下图 1 所示：

(a)归一化前



(b)归一化为 25×25 (c)归一化为 36×36

图 1 汉字归一化

## 2 数学形态学与细化算法结构元素

数学形态学[5]是一种采用模板匹配运算对图像进行各种操作，在细化运算中要用到腐蚀、膨胀、细化等形态运算。腐蚀运算可以平滑图像的边缘，膨胀运算可以填平图像中缺失的部分。细化运算用来删除图像中满足击中运算的部分，从而使图像线划变细。

汉字图像进行二值化，细化部分即汉字字符部分设为前景点值为 1，背景点为 0。每一个待处理像素设为点，其八邻域如图 2(a)所示；细化算法模板[1-8]如图 2(b)~(c)所示：

模板（b）至模板（i）为细化模板用于删除边界元素，模板（j）与（k）为一对，(l）与（m）为一对用于判别(k)与(m)中的变形点并删除，模板(p)与(q)说明当前点为左折点可删除，模板()与()说明当前点为右折点可删除。

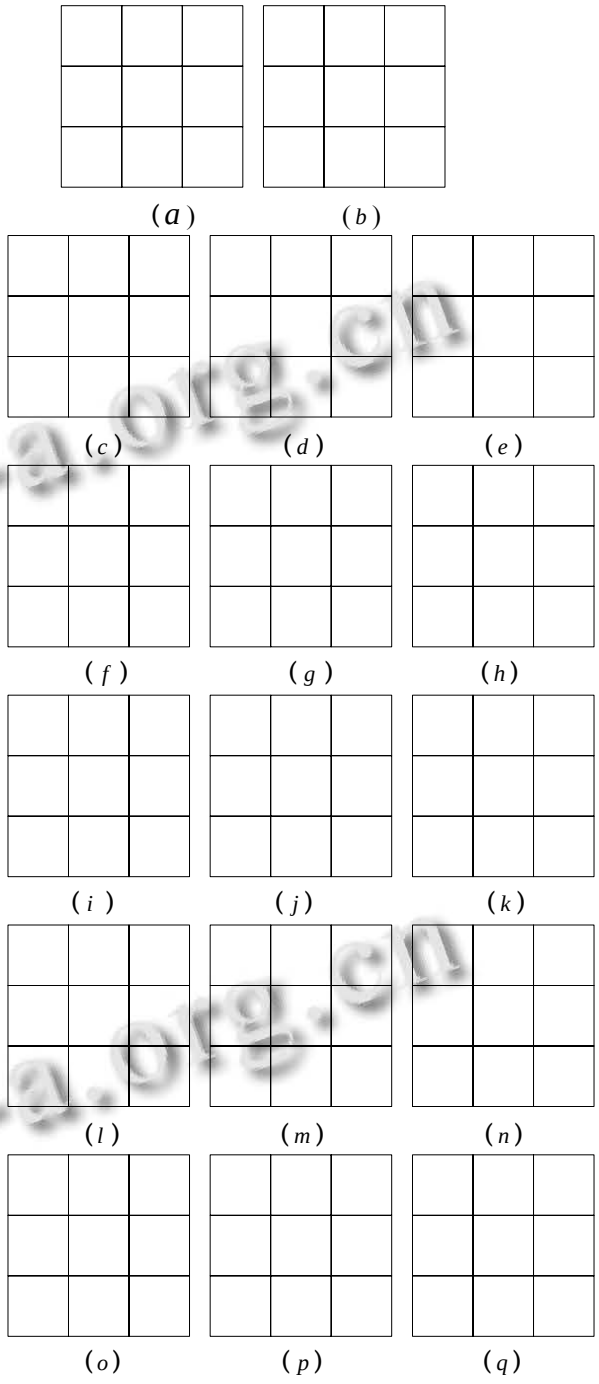



图 2 八邻域模板及细化模板

例如，当我们输入一个待细化汉字时，程序在第一次细化扫描中循环考查汉字图像中的每一个前景点，根据当前像素的八邻域情况与细化模板进行匹配，若模板（a）满足以下条件之一：（一）与模板（b）~（i）中任一匹配；（二）同时满足（1）和（2）或（3）

和（4）；（三）同时满足（5）和（7）或（6）和（8），可将当前像素赋值为 0。然后进行第二次扫描直至无满足条件的象素为止。

（1） $P_1 * P_2 * P_6 * P_7 * P_8 == 1$

（2） $P_3 + P_4 + P_5 == 0$

（3） $P_2 * P_3 == 1$

（4） $P_4 + P_5 + P_6 + P_7 + P_8 == 0$

（5） $P_1 + P_2 + P_3 + P_6 == 0$

（6） $P_2 + P_4 + P_7 + P_8 == 0$

（7） $P_4 * P_7 == 1$

（8） $P_3 * P_5 == 1$

下图为该组模板的细化效果图：

图 3　细化效果图

由图 3 可知该细化算法在对归一化后的汉字进行细化时不能保证汉字的完整性且毛刺较多[9]，因此需要对算法的结构元素进行进一步的改进。

## 3　结构元素改进后细化算法

由以上细化效果可知细化算法在竖笔划和横笔划处不能很好的细化，因此为了克服该算法的一些不足，文中新增加了两个保护模板来防止字符在细化过程中被过度删除，新增两个模板如下图 3 所示，循环条件为，凡是满足以下两个条件的像素点不会被删除。保护模板相应的表达式为：

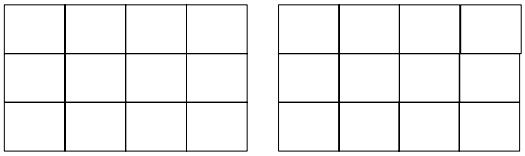
（1） $P == 1;$

（2） $(P_2 + P_3 + P_4 + P_6 + P_7 + P_8 + P_9 + P_{11} == 0 \,\&\, P_1 == 1) |$
$(P_1 + P_2 + P_4 + P_5 + P_6 + P_8 + P_9 + P_{11} == 0 \,\&\, P_7 == 1)$

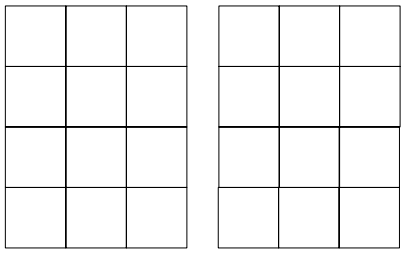
算法运行过程中每次扫描图像中首选调用这两个模板，如果当前像素满足这两个模板任何一个，将跳过该像素继续扫描下一个像素。如此循环直至所有像素点不再满足删除要求为止。

算法具体步骤如下：

(1) 扫描整幅图像并记录符合其八邻域像素点个数满足在>=1&<=7 范围内的像素个数，并记录下该像素所在的行数 *CoorR* ( *sum* )和列数 *CoorC* ( *sum* )；



（r）　　（s）



（t）　　（u）

图 4　新增模板

(2) 扫描中符合要求的像素点，并统计出其八邻域内值为 1 的像素点个数并存到数组中；01 对出现的个数存至中；

(3) 满足 $P_1 + P_2 + P_3 + P_6 == 0$ 的像素存至 *Csum*1 中，满足 $P_4 * P_7 == 1$ 的像素存至 *Csum*2 中，满足 $P_2 + P_4 + P_7 + P_8 == 0$ 的像素存至 *Csum*3 中，满足 $P_3 * P_5 == 1$ 的像素存至数组 *Csum*4 中；

(4) 满足 $P_1 * P_2 * P_6 * P_7 * P_8 == 1$ 的像素存至数组 *Dsum*1 中；满足 $P_3 + P_4 + P_5 == 0$ 的像素存至数组 *Dsum*2

(5) 满足 $P_2 * P_3 == 1$ 的像素存至数组 *Esum*1 中；满足 $P_4 + P_5 + P_6 + P_7 + P_8 == 0$ 的像素存至数组 *Esum*2 中。

(6) 满足 $P_2+P_3+P_4+P_6+P_7+P_8+P_9+P_{11}==0$ 的像素存至数组 *Fsum*1 中；满足 $P1 == 1$ 的像素存至数组 *Fsum*2 中；满足 $P_1 + P_2 + P_4 + P_5 + P_6 + P_8 + P_9 + P_{11} == 0$ 的像素存至数组 *Fsum*3 中；满足 $P_7 == 1$ 的像素存至数组 *Fsum*4 中。

(7) 判断条件(1)为：

$P == 1\&!$
$(P_2 + P_3 + P_4 + P_6 + P_7 + P_8 + P_9 + P_{11} == 0 \,\&\, P_1 == 1)$
$| P_1 + P_2 + P_4 + P_5 +) P_6 + P_8 + P_9 + P_{11} == 0 \,\&\, P_7 == 1)\&!$
$(P_1 * P_2 * P_6 * P_7 * P_8 == 1 \,\&\, P_3 + P_4 + P_5 == 0)\&!$
$(P_2 * P_3 == 1 \,\&\, P_4 + P_5 + P_6 + P_7 + P_8 == 0)\,\&$
$(Asum(t) > 1 \,\&\, Asum(t) < 7) \,\&\, Bsum(t) == 1$

判断条件(2)为：

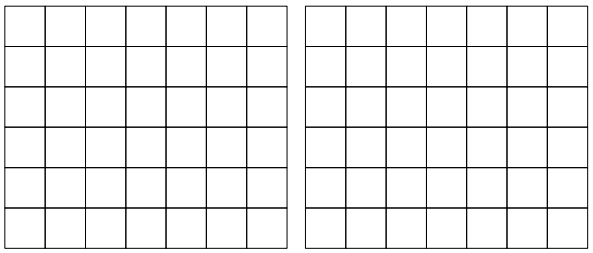
$P == 1\&!(P_2 + P_3 + P_4 + P_6 + P_7 + P_8 + P_9 + P_{11} == 0 \,\&\, P_1 == 1) |$
$(P_1 + P_2 + P_4 + P_5 + P_6 + P_8 + P_9 + P_{11} == 0 \,\&\, P_7 == 1)\&!$
$(P_1 * P_2 * P_6 * P_7 * P_8 == 1 \,\&\, P_3 + P_4 + P_5 == 0)\&!$
$(P_2 * P_3 == 1 \,\&\, P_4 + P_5 + P_6 + P_7 + P_8 == 0)\,\&$
$(Asum(t) > 1 \,\&\, Asum(t) < 7)\,\&$
$(P_1 + P_2 + P_3 + P_6 == 0 \,\&\, P_4 * P_7 == 1)$

判断条件(3)为：

$P == 1\&!(P_2 + P_3 + P_4 + P_6 + P_7 + P_8 + P_9 + P_{11} == 0 \,\&\, P_1 == 1) |$
$P_1 + P_2 + P_4 + P_5 + P_6 + P_8 + P_9 + P_{11} == 0 \,\&\, P_7 == 1)\&!$
$(P_1 * P_2 * P_6 * P_7 * P_8 == 1 \,\&\, P_3 + P_4 + P_5 == 0)\&!$
$(P_2 * P_3 == 1 \,\&\, P_4 + P_5 + P_6 + P_7 + P_8 == 0)\,\&$
$(Asum(t) > 1 \,\&\, Asum(t) < 7)\,\&$
$(P_2 + P_4 + P_7 + P_8 == 0 \,\&\, P_3 * P_5 == 1)$

(8) 重复以上步骤直到没有满足删除要求的像素为止。

汉字字符九在算法改进前后的竖笔划细化效果如下图 5 所示：

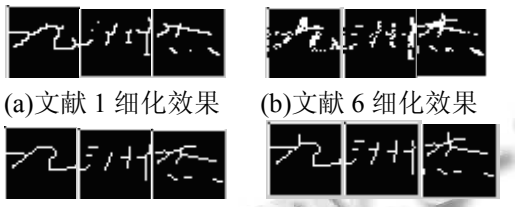


(*a*) 细化前 (*b*) 细化后

图 5 算法改进前后细化效果图

## 4 实验结果与分析

为验证文中所使用算法的优劣，在 6.0 进行实验，所使用的图像均是经过归一化的汉字图像，第一种细化算法未能实现细化后汉字字符的完整性，出现了笔划丢失问题。第二种细化算法中细化后的未能完成汉字骨架的连通性，笔划出现了断笔。文中所使用算法不但能保证字符细化的完整性，连通性且毛刺少，因此可以达到较好的细化效果，同时在处量大量图片的时间可以提高运算速度，经计算归一化后的汉字在运行时间上比未归一化处理的汉字字符的运算时间平均少 0.1522 秒，如表 1 所示。



(a)文献 1 细化效果 (b)文献 6 细化效果

(c)改进前细化效果 (d)改进后细化效果

图 6 细化算法效果对比图

表 1 算法运行时间

| 图像名 | 未归一化 | 归一化 |
|---|---|---|
| 图像 1 | 0.343(s) | 0.250(s) |
| 图像 2 | 0.422(s) | 0.266(s) |
| 图像 3 | 0.453(s) | 0.297(s) |
| 图像 4 | 0.469(s) | 0.328(s) |
| 图像 5 | 0.593(s) | 0.328(s) |
| 平均运行时间 | 0.456(s) | 0.2938(s) |

## 5 结论

好的细化算法应能保留原图骨架并体现原符号的拓扑结构，且细化结果是符号的中心线，通过对细化结果的观察，文中算法能达到这些要求。文中介绍了当前细化算法的概况，针对手写体汉字的特点将汉字归一化后再细化，并将原有算法的结构元素进行改进。进行实验验证并分析了当前细化算法的优缺点，结果表明：算法在保证细化后字符结构完整性与连通性的同时，降低算法运行时间。

## 参考文献


1 李四明.工程图纸输入与自动识别的改进细化算法.计算机工程,2003,29(16):37–39.

2 Xie JH, Cai N, Jing YW. Improvement of two-step parallel thinning algorithm for military map contours. Proc. of 2009 Chinese Control and Decision Conference. China. 2009:1483–1486.

3 杨兴炜,刘文予,白翔.一种有效的快速细化算法.小型微型计算机系统,2006,27(7):1343–1346.

4 张学东,张仁秋,关云虎,王亭.一种快速的手写体汉字细化算法.计算机应用与软件,2009,26(1):17–19.

5 李迎,段汕.对于二值图像形态细化方法中结构元素的研究.第十二届全国图像图形学学术会议论文集.中国图象图形学学会.武汉,2005:304–307.

6 Zhang YY. Redundancy of parallel thinning. Pattern Recognitiion Letters, 1997: 27–35.

7 刘利娜,吴锡生.提高手写体数字细化效果的改进算法.计算机工程与应用,2010,46(3):137–139.

8 安然,张少军,陈华,喻振华.字符识别中毛刺的去除方法.计算机技术与发展,2007,17(9):136–138.

9 Melhi M, Ipson SS, Booth W. A novel triangulation procedure for thinning hand-written text. Pattern Recognition Letters, 2000: 1059–1071.

10 Chiu HP, Tseng DC. A feature-preserved thinning algorithm for handwritten Chinese characters. Signal Processing, 1997: 203–214.