一种适于串行机实现的图像并行细化算法

王家隆 郭成安
(大连理工大学电子与信息工程学院,大连 116023)

摘 要 为解决现有的图像并行细化算法在串行机上的高效实现问题,首先提出了一种 4×4 邻域二值图像的双字节图像编码方案,由于在该方案中将每个 4×4 邻域的像素用一个双字节的整数来表示,从而将基于整个邻域 16 个像素的细化处理转化为一个双字节整数的读、写和比较运算的问题;然后在此基础上提出了一种可在串行机上实现的并行细化算法。实验证明,该算法适用于当前通用的各种基于模板匹配的并行细化算法,其不仅可以取得完全相同的细化结果,而且可以大幅度提高图像细化过程在串行机上的执行速度;最后简要讨论了该算法利用 PC 机中的 MMX 技术来进一步提高并行粒度和运算效率方面所具有的潜力。

关键词 图像处理 细化 并行算法 模板匹配法
中图法分类号: TP391.41 TN911.73 **文献标识码**: A **文章编号**: 1006-8961(2004)01-0112-06

An Image Parallel Thinning Algorithm for Serial Computers

WANG Jia-long, GUO Cheng-an
(*School of Electronic and Information Engineering, Dalian University of Technology, Dalian* 116023)

Abstract In order to improve the computation efficiency of image parallel thinning algorithms implemented on serial computers, this paper proposes a coding scheme for each 44 binary pixels, representing the 44 pixels with a double-byte integer, based on which an effective parallel thinning algorithm for serial computers is presented that transforms the 16-pixel based thinning processing into reading, writing and comparison operations of a double-byte integer. By use of this algorithm, the 16 pixels can be processed simultaneously in the thinning operations with serial computers. Computation complexity analysis of the new algorithm is made in the paper that shows about 70% saving in computations compared with the existing parallel template thinning algorithms (e.g., the OPTA algorithm). The algorithm is suitable for all kinds of parallel template thinning algorithms operated on commonly used personal computers, and can significantly improve the computation efficiency of the algorithm with the same thinning results. The potential advantage of the algorithm is also discussed in the paper for using the MMX (or SSE) technology integrated in PCs to further increase the parallel granularity and speed up the thinning processing.

Keywords Image processing, Thinning, Parallel algorithm, Template matching

1 引 言

细化又称为骨架化,即在不影响原图像拓扑连接关系的条件下,尽可能用最少的迭代次数,快速而准确地将宽度大于一个像素的图形线条转变为一个像素宽线条的处理过程,也就是抽取图像的骨架。细化不但能够很好地展现原图像的拓扑结构形状,而且可以大大减少存储图像所需的内存空间。由于它只需存储图像中必需的结构信息,且在图像处理中简化了数据结构,也是图像分析、信息压缩、特征抽取及模式识别中常用的技术,因此在图像处理中占有重要地位,例如在文字识别和指纹识别中,通过细化可以得到其单像素宽度的文字笔划或指纹脊线,这将有利于提取文字和指纹的特征。另外,在工程图识别中,通过细化得到骨架图像,也有利于进行各种

基金项目:辽宁省科学技术基金(20022139)
收稿日期:2002-10-30,收回日期:2003-08-18

图形及线条的提取和各种符号的识别等。这种细化处理可为后续的图像分析处理提供一种紧凑而有效的表示形式,也起到减少后续处理步骤中所需的计算时间和空间的作用。

前人已提出了多种细化算法,根据这些算法的迭代方式不同,通常有串行算法和并行算法之分。在串行细化算法中,每次迭代的结果不仅取决于前一次的迭代结果,而且与当前处理情况有关;而在并行细化算法中,当前迭代则仅仅由上次的迭代情况决定。由于串行细化算法的处理结果依赖于对像素处理的先后顺序,因而某像素点是消除,还是保留与处理顺序有关,事前不可预测;而并行细化算法对图像进行细化时,则可利用相同的条件来同时检测所有像素点,由于其不但可同时对所有像素点进行细化处理,且其结果具有各向同性,因此从算法原理上讲,并行方法优于串行方法。如今已提出的并行算法有 OPTA(One-Pass Thinning Algorithm)细化算法[1]、Hall 细化算法[2]、Rosenfeld 细化算法[3]、Zhang 和 Suen 细化算法[4]以及 ZR 细化算法[5]等。

由于在实际应用中,进行图像处理的工作一般是在串行的 PC 机上实现,并非用专用的并行处理机,所以对现有的并行图像细化算法,只是在 PC 机上"串行"地来实现,但由于这样并不能充分发挥并行算法本身具有的优越性,从而大大降低了运算效率。为了充分发挥并行细化算法本身的优越性,本文对如何在串行机上快速实现并行细化算法进行了探讨,即通过对模板细化算法的研究,采用 C++ 语言作为开发工具,提出了一种适合在串行机上快速实现图像细化的算法。该算法可在保留原算法细化结果的同时,较大幅度地提高图像的细化速度。

2　算法描述

2.1　OPTA 算法

本文的细化算法是基于已有的并行模板细化算法提出的。OPTA 算法[1]是经典的图像并行模板细化算法之一,下面以该算法为例给出本文的问题和解决问题的思路。

OPTA 算法是从图像的左上角像素点开始,按照从左到右,从上到下的顺序对图像进行扫描,同时对每个像素点均抽取出如图 1 所示的 10 个相邻像素(其中 P,Q 都为像素点,P 为当前像素点),然后将这些像素与事先规定的模板进行比较:第 1 步与

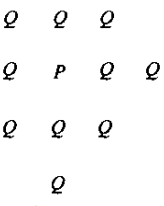

图 1　OPTA 算法抽取的邻域

图 2 所示的 8 个消除模板比较,如果和 8 个消除模板中的任意一个匹配(即邻域中所有像素点的值与模板中相应位置所有非"×"值的点都相等),则第 2 步再和图 3 所示的两个保留模板比较,如果在第 2 步中与其中的任意一个保留模板相匹配,则保留 P 点,否则删除 P 点;如果在第 1 步比对中,和所有的消除模板都不匹配,则保留该点。

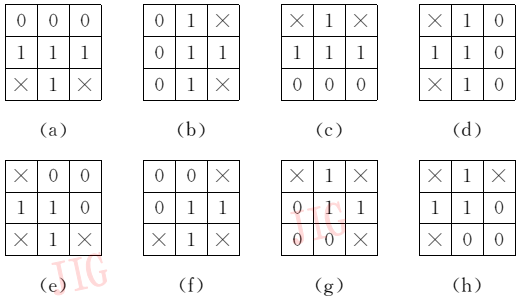

图 2　OPTA 消除模板

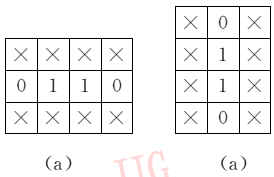

图 3　OPTA 保留模板

下面以图 4 为例来具体说明 OPTA 算法的实现过程:当扫描到第 4 行,第 4 列的像素点时(该点坐标记为(4,4)),则依照图 1 所示的邻域对所需像素点进行抽取,并首先与 8 个消除模板依次进行比对,如结果与消除模板图 2(f)相匹配,这时再与两个保留模板进行比对,如结果与两个保留模板都不匹配,这样便判定该像素点属于可删除像素点,并将该点值改置为 0,以示删除了该点;然后继续对图像进行扫描到像素点(4,5),同时抽取所需邻域像素点,并与各消除模板进行比对,若结果不和任一消除模板相匹配,则判定保留此点(即不改变该点的值)。依照此方法对图像进行细化,反复迭代扫描,直到无像素点可删除为止,则细化结束。

在该算法中的每一次迭代扫描中,对每一个当前

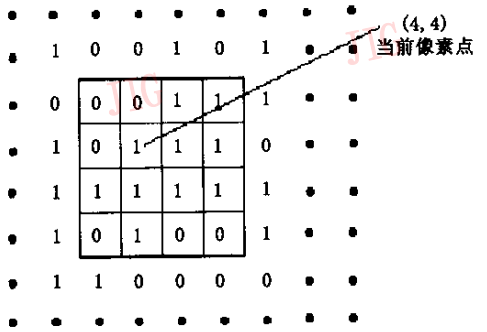

图 4 细化中图像扫描过程示例

像素点均需对 4×4 邻域中每个像素点进行逐点读取，并与消除模板和保留模板中的对应点进行逐点比较。显然，该操作在串行机上的直接实现是一种串行运算，其对每个 4×4 邻域的像素共约需 70 次读操作、70 次比较操作，可见操作次数较多。本文算法由于是对每个 4×4 邻域的 16 个像素点同时进行操作，而不是逐点操作，从而可大幅度提高运算效率。

2.2 在串行机上实现图像细化的快速并行算法

如上所述，由于 OPTA 算法在串行机上的直接实现是一种串行运算，故需要较多的计算量。为能有效地降低这一计算量，本文研究了该算法的并行实现方案。该方案的核心是将每个 4×4 邻域的所有像素用一个双字节整数来编码，并将每个删除模板和保留模板也用一个双字节整数来编码，然后通过对这些双字节数进行运算来实现 OPTA 算法，这样就可实现 4×4 邻域的 16 个像素的并行操作。下面首先讨论该并行算法的编码方案。

2.2.1 用于实现并行模板细化算法的编码方案

首先将需要细化的二值图像的全部像素用 0 或 1 来表示(0 表示背景，1 表示前景)。这样每个像素点只需用 1 bit 的数据(0 或 1)即可表示。

由图 1、图 2 和图 3 可知，在 OPTA 算法的所有消除和保留模板中，由于所涉及到的所有像素点均在 4×4 邻域内，因此用一个标准的 4×4 模板可以包含其中所有像素点(如图 5 所示)。而因串行的 PC 机中一个字节为 8bits，故用一个 16bits 整数(即 2Bytes)就可以表示 OPTA 算法中的任意一个模板或任意一个 4×4 邻域。本文的编码方案如图 5 所示。按此方案，可将每个 4×4 邻域的 16 个像素点用一个双字节整数来表示(记为 $I_{16}(x,y)$)，并可将 8 个消除模板和 2 个保留模板也各用一个双字节整数来表示(记为 $M_i(j)$)。

注意到由图 2 和图 3 所示的模板中，有些像素

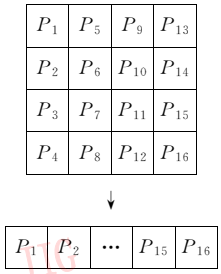

图 5 4×4 模板转化为双字节整数的编码方案示意图

点表示为“×”，这表示在作模板匹配时，该点不需进行比对，也就是在对这些模板进行编码时，要将这些像素点所对应的比特位屏蔽掉(即置为 0)，以便于后面的操作，而在后面进行模板匹配处理时(即将每个 $I_{16}(x,y)$ 分别与各个 M_i 进行比较)，也需要将 $I_{16}(x,y)$ 中不需要进行比对的相应比特位屏蔽掉。这可通过将 $I_{16}(x,y)$ 和一个屏蔽数相“与”来实现。因为对不同的模板所要屏蔽的比特位不同，故其所对应的屏蔽数也不同。下面采用如图 5 所示的编码方案，将 8 个消除模板及其屏蔽数分别表示成双字节的 16 进制整数，其结果如下：

(1)模板数 01×× 011× 01×× ××××:0x4640，屏蔽数 1100 1110 1100 0000:0xcec0；

(2)模板数 000× 111× ×1×× ××××:0x0e40，屏蔽数 1110 1110 0100 0000:0xee40；

(3)模板数×10× 110× ×10× ××××:0x4c40，屏蔽数 0110 1110 0110 0000:0x6e60；

(4)模板数×1×× 111× 000× ××××:0x4e00，屏蔽数 0100 1110 1110 0000:0x4ee0；

(5)模板数×1×× 011× 00×× ××××:0x4600，屏蔽数 0100 1110 1100 0000:0x4ec0；

(6)模板数 00×× 011× ×1×× ××××:0x0640，屏蔽数 1100 1110 0100 0000:0xce40；

(7)模板数×00× 110× ×1×× ××××:0x0c40，屏蔽数 0110 1110 0100 0000:0x6e40；

(8)模板数×1×× 110× ×00× ××××:0x4c00，屏蔽数 0100 1110 0110 0000:0x4e60。

将两个保留模板及其屏蔽数表示成双字节 16 进制数，结果为：

(1)模板数×0×× ×1×× ×1×× ×0××:0x0440，屏蔽数 0100 0100 0100 0100:0x4444；

(2)模板数×××× 0110 ×××× ××××:0x0600，屏蔽数 0000 1111 0000 0000:0x0f00。

下面阐述对各个像素点及其 4×4 邻域的编码

过程:在对原图像进行编码处理时,是按照从左到右、从上到下的顺序进行扫描,如果扫描到的当前像素点的值为 1,则对此点数据进行如下处理,即将当前像素点 $P(x,y)$记为 P_6,并按照图 5 所示的编码方案,将 $P_1,P_2,\cdots,P_{16}$表示成双字节形式,以构成新的图像数据 $I_{16}(x,y)$(即每个像素点及其周围的 4×4 邻域数据用一个双字节数 $I_{16}(x,y)$表示)。这样在新的图像数据 $I_{16}(x,y)$中不仅保留了原图像数据的信息,还引入了当前像素点周围其他 15 个像素点的信息;如果扫描到的当前像素点 P_6 的值为 0,则将其所对应的双字节整数$I_{16}(x,y)$直接置为 0,以便在细化过程中对该点不作处理。

下面仍以图 4 为例来对整幅图像的双字节编码方案做进一步说明:当扫描到像素点 $P(4,4)$时,因 $P(4,4)$值为 1,故可按照图 5 所示的编码方案将 $P_1,P_2,\cdots,P_{16}$转换成双字节数形式为 0010 0111 1110 1110,其表示成 16 进制数的结果为$I_{16}(4,4)=$0x27ee。继续扫描到下一个像素点 $P(4,5)$,该点的值也为 1,这时,只需将表示 $P(4,4)$点的双字节数 $I_{16}(4,4)$左移 4 位,即移出 P_1,P_2,P_3,P_4 点的信息,同时在右边移入 1011(即 $P(3,6)$、$P(4,6)$、$P(5,6)$、$P(6,6)$的信息),这样便得到表示点$P(4,5)$的双字节数 $I_{16}(4,5)=$0x7eeb。当扫描到像素点 $P(4,7)$时,由于 $P(4,7)$值为 0,故将其对应的双字节整数 $I_{16}(4,7)$直接置为 0。依照此法,对整幅图像进行操作,将其转化为用双字节整数表示的新的图像 $I_{16}(x,y)$。在新图像中,每个像素值既包含了原图像当前点的信息,又包含了该点周围 4×4 邻域的其他 15 个点的信息,加之经前述编码处理后已用双字节整数表示了各模板信息及其相应的屏蔽数,在此基础上即可设计出一次处理 16 点信息的并行细化算法。

2.2.2　适于串行机实现的并行细化算法

通过采用前节所述的编码方案,即可将各个 4×4邻域的二值图像用一个双字节整数来表达,同时也将各匹配模板以同样方式进行编码,这样便得到了一个新的双字节形式表达的图像和匹配模板。而这时再对该双字节图像进行处理,则一次可读入相当于原来的 16 个像素点,如再对该双字节数进行一次比较运算,就可以实现一个 4×4 模板的匹配操作,这便是本文并行细化算法的主要思路。

下面仍以图 4 为例,并仍采用 OPTA 算法来具体说明该并行算法:当扫描到点(4,4)时,先经前述的编码方案编码,这时该点的像素值 $I_{16}(4,4)$为 0x27ee,并读入 $I_{16}(4,4)$;然后将 $I_{16}(4,4)$和消除模板所对应的屏蔽数相"与",以提取需要比对的信息,再与该消除模板数相比较,例如当与 2.2.1 节消除模板(1)$M_1(1)=$0x4640 比对时,是先将 $I_{16}(4,4)$和屏蔽数 0xcec0 相"与",得到结果 0x06c0,再将该结果与 $M_1(1)$比对,若结果与 $M_1(1)$不相等,则说明 $I_{16}(4,4)$不和消除模板 $M_1(1)$相匹配。依照此法,再依次和其他消除模板进行比较。如果和任意一个消除模板相匹配,则还要再与两个保留模板进行比对,以判断出该像素点是否可以删除。

当判断出某像素点为可删除像素点时,则要进行删除操作,即在新算法中需将表示该像素点的双字节整数改为 0,以表示该像素点已由前景点删除成为背景点。同时,对于包含该像素点的周围其他双字节数,也需对其进行相应的修改(即该删除点原来在周围双字节数中所对应的比特位是 1,这时需将该比特位改为 0,以表示将其删除)。以 $P(4,4)$点为例,删除此点后,$I_{16}(4,4)$改为 0,对于 $P(4,5)$点所对应的双字节数 $I_{16}(4,5)$原来为 0x7eeb,其中 P_2 代表原 $P(4,4)$点的信息,当 $P(4,4)$点被删除,则 P_2 的值应改为 0,从而 $I_{16}(4,5)$的值也应改为 0x3eeb。同理,对于其他涉及到 $P(4,4)$点信息的邻域点,其双字节数也应做相应的修改。依照上面步骤,继续扫描,直至细化结束。

上述并行细化算法可在通用的 PC 机上实现。由于目前以 Intel 公司 Pentium Ⅱ 及以上芯片为中央处理器的 PC 机都集成了 MMX (MultiMedia eXtension)或 SSE(Streaming SIMD eXtension)技术,同时由于在 MMX 指令中是采用 64 位寄存器(SSE2 中为 128 位),并可用 MMX 指令通过将 4 个双字节整数装入到 64 位寄存器中来并行实现 4 个 4×4 模板的匹配操作(用 SSE2 指令则可并行实现 8 个 4×4 模板的匹配操作),从而可进一步大幅提高该算法的运算速度。

2.3　计算复杂性分析

本文提出的新细化算法的优越性在细化过程中体现得尤为突出,因为在该算法中,完成一个 4×4 模板的匹配运算只需要一个读双字节数的操作及两个双字节数的比较操作;而在原算法中,则需要对 16 个二进制数逐个进行读取和比较操作。

本文提出的新细化算法不仅可以用于 OPTA 算法,而且亦可用于其他模板匹配类的细化算法上(如文献[6]中的细化算法)。

在图像模板细化过程中有保留和删除两种情况，其操作流程如图 6 所示。下面对新算法及原算法的计算复杂性进行具体分析。

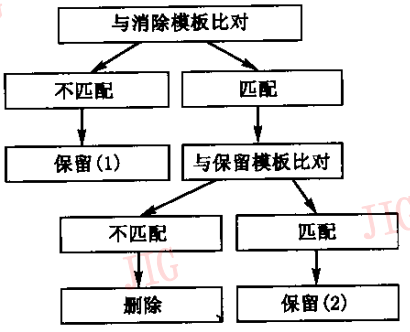

图 6 图像细化操作流程

(1)基于 OPTA 算法的并行细化处理

① 对图 6 中所示的保留(1)操作流程，原算法需 15 次读操作，52 次比较操作；新算法只需 1 次读操作，8 次模板比对操作（其中 8 次“与”，8 次比较）。

② 对图 6 中所示的保留(2)操作流程，原算法需 15 次读操作，60 次比较操作；新算法只需 1 次读操作，10 次模板比对操作(其中 10 次“与”，10 次比较)。

③ 对图 6 中所示的删除操作流程，原算法需 15 次读操作，60 次比较操作，1 次修改数值操作；新算法只需 1 次读操作，10 次模板比对操作(其中 10 次“与”，10 次比较)，10 次修改数值操作(其中 10 次“与”，10 次“存”，10 次指针修改)。

经综合统计可得，对于 OPTA 算法中的保留(1)流程，新算法运算量为原算法的 25.4%；对保留(2)流程，新算法运算量为原算法的 28.0%；对删除流程：新算法运算量为原算法的 67.1%。

(2)基于文献[6]算法的并行细化处理

① 对保留(1)操作流程，原算法需 15 次读操作，52 次比较操作；新算法只需 1 次读操作，8 次模板比对操作(其中 8 次“与”，8 次比较)。

② 对保留(2)操作流程，原算法需 15 次读操作，96 次比较操作；新算法只需 1 次读操作，14 次模板比对操作(其中 14 次“与”，14 次比较)。

③ 对删除操作流程，原算法需 15 次读操作，96 次比较操作，1 次修改数值操作；新算法只需 1 次读操作，14 次模板比对操作(其中 14 次“与”，14 次比较)，10 次修改数值操作(其中 10 次“与”，10 次“存”，10 次指针修改)。

经综合统计可得，对于文献[6]算法的保留(1)流程，新算法运算量为原算法的 25.4%；对其保留(2)流程，新算法运算量为原算法的 26.1%；对删除流程，新算法运算量为原算法的 52.7%。

根据以上分析可知，本文的细化算法可较大幅度地降低运算量。由于在一般情况下，需经过多次迭代方可完整地实现一幅图像的细化过程，故该项节省是比较有价值的。此外，采用本文提出的快速算法进行图像细化，首先需要对图像进行一次预处理(编码)，即将其转化为双字节整数形式，而原来的细化方法的实现则不需要进行此步操作。该预处理过程虽耗费一定的时间，但由于该部分时间相对于整个细化过程来说耗时较小，又由于该图像形式对于后面的特征提取等环节亦将带来简便，故此处未将预处理时间归到细化中来。

由于在新算法中，主要是对双字节数进行读写、比较以及在删除像素点情况下进行修改数值操作，因此当采用 MMX(或 SSE)指令时，虽然对于删除像素的修改操作，并不带来运算上的优势，但对于大量的读写和比较操作，由于采用 MMX 指令一次可完成 4 个双字节整数的运算，故对这些操作的运算效率可再提高近 4 倍(用 SSE2 指令可提高近 8 倍)，因此本文算法还有进一步提高运算速度的潜力。

3 实验结果

为了验证本文算法的效果，对前面所述的基于双字节整数图像的新细化算法进行了大量的计算机实验，同时也与 OPTA 算法和文献[6]的算法进行了实验比较。图 7 为原始待细化的二值指纹图像，图 8(a)、(b)和图 9(a)、(b)分别为采用原算法和新算法的细化结果，表 1 给出了对 8 幅指纹图像采用各算法所用的运算时间。实验采用的 PC 机配置为 PⅢ 733，128M 内存。

图 7 原始待细化指纹图像

(a) 原算法细化结果　　(b) 新算法细化结果

图 8　基于 OPTA 算法的指纹图像细化结果

(a) 原算法细化结果　　(b) 新算法细化结果

图 9　基于文献[6]算法的指纹图像细化结果

表 1　新细化算法和原细化算法运算时间比较

	基于 OPTA 算法			基于文献[6]算法		
	迭代（次）	原算法（ms）	新算法（ms）	迭代（次）	原算法（ms）	新算法（ms）
图像 1	5	13.84	8.19	13	26.97	15.32
图像 2	4	10.17	6.32	9	17.63	10.71
图像 3	4	11.92	7.36	12	24.61	14.33
图像 4	4	11.04	6.59	7	15.65	9.12
图像 5	4	11.21	6.86	8	17.52	10.07
图像 6	4	10.93	6.54	8	17.30	10.05
图像 7	4	10.65	6.49	9	18.07	11.04
图像 8	4	11.70	6.97	7	16.31	9.66

图 8 和图 9 给出的例子和表 1 中的第 7 组数据相对应。由这些结果可以看出，采用新算法得到的细化结果与原算法得到的结果是一样的，但耗费的时间却显著减少。其中，用 OPTA 算法，原算法耗时 10.65ms，新算法耗时 6.49ms；而用文献[6]算法，原算法耗时 18.07ms，新算法耗时 11.04ms.

4　结　论

本文基于现行的模板并行细化算法和根据通用串行机的特点，提出了一种适于在串行机上实现的并行细化算法。由于采用该方法在串行机上对二值图像进行细化，可在不改变原细化方法运算结果的同时，实现 4×4 邻域像素的并行操作，从而可显著提高图像的细化速度。另外，该算法还适用于其他不同模板的并行细化算法。

本文算法还可以通过利用现有 PC 机中的 MMX（或 SSE）技术来进一步提高并行粒度，因而其运算效率还存在大幅提高的潜力，这亦是本文进一步工作的方向。

参 考 文 献

1　Chin R T, Wan H K, Stover D L, *et al*. A one-pass thinning algorithm and its parallel implementation[J]. Computer Vision Graphics Image Processing, 1987, **40**(1):30～40.

2　Hall R W. Optimally small operator supports for fully parallel thinning algorithms[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993, **15**(8): 828～833.

3　Pavlidis T. Algorithms for graphics and image processing[M]. Rockville, Washington D C, USA: Computer Science Press, 1982.

4　Zhang T Y, Suen C Y. A fast thinning algorithm for thinning digital patterns[J]. Communications of ACM, 1984, **27**(3):236～239.

5　GUO Zicheng, Richard W H. Parallel thinning with two-suBiteration algorithms[J]. Communications of ACM, 1989, **32**(3):359～373.

6　冯星奎，李林艳，颜祖泉. 一种新的指纹图象细化算法[J]. 中国图象图形学报，1999，**4**(10)：835～838.

王家隆　1978 年生，2000 年获辽宁大学电子学与信息系统专业学士学位，2003 年获大连理工大学信号与信息处理专业硕士学位。研究方向为数字图像处理与识别。

郭成安　1955 年生，1999 年获美国夏威夷大学电子工程系博士学位，现为大连理工大学电子与信息工程学院教授。主要研究方向为数字图像处理与识别、智能信息处理。