第 7 卷(A 版) 第 7 期
2002 年 9 月

中国图象图形学报
Journal of Image and Graphics

Vol. 7(A), No. 9
Sep. 2002

一种基于 Delaunay 三角化的手写体文字细化方法

杨义军 孟祥旭 杨承磊 汪嘉业

(山东大学计算机科学与技术学院, 济南 250100)

摘 要 为了对手写体文字进行快速准确的识别,基于 Delaunay 三角化方法,提出了一种新的文字图象细化算法.该算法首先通过对文字图象边界的近似多边形进行 Delaunay 三角化,同时把其分成一系列保持拓扑关系的三角形;然后根据三角形的类型生成不同的局部骨架;最后连接生成整幅文字图象的骨架.由于该算法充分利用了图象的全局和局部信息,因此具有速度快,效果好等优点.

关键词 细化 骨架 Delaunay 三角化 手写体

中图法分类号: TP391.43 **文献标识码**: A **文章编号**: 1006-8961(2002)09-0938-07

An Algorithm for Thinning Hand-written Text Based on Delaunay Triangulation

YANG Yi-jun, MENG Xiang-xu, Yang Cheng-lei, WANG Jia-ye

(*School of Computer Science, Shandong University, Jinan* 250100)

Abstract Based on Delaunay triangulation, This paper describes a novel algorithm of thinning text images for fast nicety recognition. The polygonal approximation of images' boundaries is decomposed into Delaunay triangles that represents the topological features of the object. All triangles are classified into three types that generate different local skeletons which are connected to form the whole text images skeletons. Taking full advantage of text images' whole and local information, this algorithm has the advantages of fast speed and nice effect.

Keywords Thinning, Skeleton, Delaunay triangulation, Hand-writing

0 引 言

在自动文字识别[1]、指纹识别[2]、染色体分析[3]和自动线路板检测[4]等识别系统中,采用细化方法计算图象的骨架是一个十分重要的预处理步骤.因为骨架包含了文字图象特征的最有效数字化信息,所以能对文字图象进行有效地描述.早在 1967 年,Blum 就给出了著名的"模拟森林着火"骨架定义[5]:假设有一块平面多边形区域的森林,如果从森林的边界同时点火,且大火以相同速度向内部燃烧,则大火熄灭点的集合就称为该区域的骨架.后来,Pavlidis 给出一个标准化的定义[6]:

"令 $\boldsymbol{R}$ 表示一个平面多边形区域内点的集合,$\Gamma_{\boldsymbol{R}}$ 表示其边界边的集合,P 是 $\boldsymbol{R}$ 内任意一点,M 表示 $\Gamma_{\boldsymbol{R}}$ 上到 P 距离最近的点,即对于$\forall\ T\in\Gamma_{\boldsymbol{R}}$,总存在 $|PM|\leq|PT|$.如果在 $\Gamma_{\boldsymbol{R}}$ 上,存在到 P 距离最近的点数多于一个,即

$$|\{M\,||PM|\leq|PT|,\forall\ T\in\Gamma_{\boldsymbol{R}},M\in\Gamma_{\boldsymbol{R}}\}|>1$$

则称 P 为平面多边形区域 $\boldsymbol{R}$ 的一个骨架点.所有骨架点的集合就称为这个平面多边形区域 $\boldsymbol{R}$ 的骨架".

Davies 等在 1981 年就指出图象骨架的不唯一性[7],但是,一种好的细化算法产生的骨架不仅应该能够保持原来图象的形状信息,且能同时达到具体应用所要求的精度,而且它应该能够快速地产生具有连通性的、只有一个像素宽的骨架.

迄今为止,已有很多细化算法产生,大致可归纳为迭代算法和非迭代算法两大类.其中,迭代算法是

基金项目:国家自然科学基金项目(69973028);山东大学青年基金
收稿日期:2002-06-13;改回日期:2002-07-08

首先根据骨架的特性来制定一些限制条件，再通过由外到里逐步去除边缘点来求得图象的骨架（见图 1 (a)）. Lam 等在 1992 年回顾了 15 种迭代细化算法，发现其中的 Nachache 和 Shighal 方法是最快的，且产生的骨架质量最好[8]. 但迭代算法存在的问题是运算速度较慢，在一般情况下，即使采用了并行算法[9~12]，其运算速度还是比较慢.

(a) 细化算法　(b) 基于图象划分的骨架算法　(c) 基于三角化的骨架算法

图 1　已有算法图示

非迭代的细化算法要比迭代算法快得多，因为非迭代的细化算法是在一个和分辨率无关的固定数目的步骤下完成. 这类算法[13~19]获取骨架的最常用技术是先利用图象距离的变化来求得可能组成骨架的像素；然后根据骨架的特性来选择其中的子集，其中许多算法都是利用基于区域边界的逼近多边形来进行骨架计算. 如 Feng 和 Pavlidis 根据内角的凸凹形状先把边界多边形分解成比较简单的、不可分割的部分，然后以此来计算图象的骨架[15]. 这种分解的优点是，不仅具有平移和旋转无关性，而且基于这种分解的图形非常容易生成，可对以后的识别有帮助，但这种方法不能直接用来细化文字图象，而需要进一步的分解；杨承磊、孟祥旭等提出了通过把整幅图象分割成简单多边形来求解图象骨架的算法（见图 1(b)）[16~18]，该算法首先把图象分成许多三角形、四边形等简单多边形，并用无向图记录它们之间的位置关系；然后针对不同类型的简单多边形，采用不同的求解方法来计算骨架，由于该算法充分利用了图象的全局和局部信息，因此不仅速度比较快，效果比较好，而且很好地保存了图象骨架拓扑特性，但由于该算法不具备平移和旋转无关性，因此在分割算法方面还有待改进；Melhi 等提出了用一般三角化方法来求解图象骨架的方法（见图 1(c)）[19]，该算法是通过对图象边界多边形进行三角化来获得具有拓扑特性的骨架，但由于该算法主要是通过对多边形的每个顶点都先计算出其可能的对角线，然后根据三角形各边长度和的大小来决定如何分割，其时间复杂度比较高 $O(n^3)$，且三角化的结果不太理想，从而影响了所求骨架的质量.

为此，本文提出了一种基于 Delaunay 三角化的手写体文字图象快速细化算法，以用于解决当前已有算法存在的一些问题. 为了便于算法的描述，假设手写体文字图象是二值图象，且已经经过了一些去噪声处理.

1 算法描述

手写体文字图象（见图 2）一般包括几个独立的部分，而每一部分的边界又可分为外边界和内边界两种类型. 在进行文字图象细化时，首先计算这类图象边界的逼近多边形；然后基于边界多边形的顶点进行 Delaunay 三角化，并对生成的三角形进行调整，从而得到该边界多边形的比较理想的三角化结果；接着根据每一个三角形的拓扑结构来计算局部骨架；最后这些局部骨架的集合就构成了整个文字图象的骨架. 对于存在内边界的情况，则需要先把有内边界的文字图象的某一部分切断，以形成一个没有内边界的图象，然后再按无内边界的情况进行图象的骨架化处理.

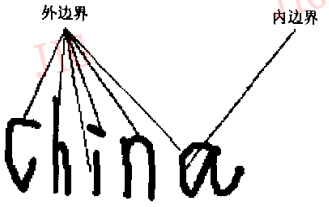


图 2　具有内边界和外边界的文字图象

1.1 边界多边形的计算

在进行边界多边形计算时，首先用一个时间复杂度为 $O(n)$ 的算法[20]来计算图象的边界多边形. 由于边界多边形各边的长度变化非常大，为了产生一个更精确的骨架，有必要用文字图象笔划的宽度（d）来对边界多边形各边进行分割，以形成较短的边，而且对于每一个文字都需要估计一个宽度参数 d. 由于大部分文字都包括大量的垂直或水平笔划，所以可以通过水平和垂直扫描来估计文字笔划的宽度.

在图象扫描过程中，扫描线与图象相交可得到一系列线段，其每一条线段就叫一个游程[17]，一般可用一个频率数组来记录所有游程出现的次数，其中，频率数组的下标对应游程的长度，频率数组的元素就记录不同长度的游程出现的次数. 在图象扫描过程中，每得到一个游程，就使相应频率数组元素的值加 1.

在整个扫描过程结束后,频率数组中数值最大的元素的下标就可以看作是该文字笔划的宽度 d.

算法如下:

```
Begin
    频率数组的每一元素置为 0;
    For 每一条扫描线
      算出扫描线和文字图象得到的所有游程.
      For 每一个游程
          频率数组[游程的长度]++;
      End for
    End for
    找到值最大的数组元素,元素的下标就是估计值 d.
End
```

一旦文字笔划的宽度被估计出来,那么边界多边形上所有长度大于 d 的边就被分成许多长度为 d 的小边. 如果分割所得的最后一条边的长度小于 d,则把它合并到上一条边中.

1.2 含内边界的多边形 Delaunay 三角化

迄今为止,已经有了很多用于多边形三角化的算法[21~23],但由于这些算法很少考虑三角化的"最小角最大"原则,而基于点集的 Delaunay 三角化[23]却能很好地满足这一原则,所以本文以基于点集的 Delaunay 三角化算法为基础来研究实现可满足"最小角最大"原则的多边形三角化算法,以便生成更合理地结果.

在对不含内边界的多边形进行 Delaunay 三角化时,算法首先以边界多边形的所有顶点为"生成子"[23]来进行 Delaunay 三角化;由于生成的三角形有可能和文字图象相交,所以在三角化以后,还需对生成的三角形进行处理,以保证最后生成的三角形完全在文字图象内部或者外部;然后根据三角形面积的符号把外部三角形去掉;最后将剩余的每个三角形(在文字图象内部),根据拓扑结构将其细化成一条边或者 3 条边,即形成文字图象的骨架.

为了处理的方便,这里按逆时针方向对边界多边形上的顶点,由小到大进行编号. 例如,图 3(a)给出了图 2 中第 1 个文字图象的边界多边形顶点的编号.

1.2.1 基于顶点集合的 Delaunay 三角化

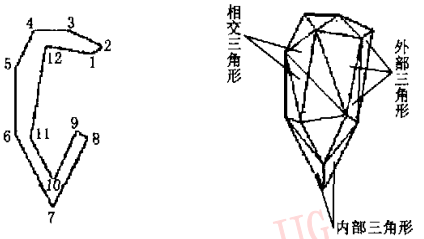
在对多边形的顶点集合进行 Delaunay 三角化时,采用了根据三维凸包找相邻的 Delaunay 三角形的递增算法[23]来生成如下 3 种类型的 Delaunay 三角形(见图 3(b)):

(1)内部三角形 Delaunay 三角形完全在边界多边形内部;

(2)外部三角形 Delaunay 三角形完全在边界多边形外部;

(3)相交三角形 Delaunay 三角形和边界多边形相交.

由于边界多边形分割后,最终得到的所有三角形必须在该多边形的内部,所以对于以上 3 种类型的三角形来说,其所得到的内部三角形是最终结果的一部分,应当保留;外部三角形完全在边界多边形外部,应当删除;而对于相交三角形,由于其和边界多边形相交,因此需要把其剖分成一些完全在文字图象内部的三角形和一些完全在文字图象外部的三角形,以便使得剖分以后的三角形或者在文字图象内部,或者在文字图象外部.



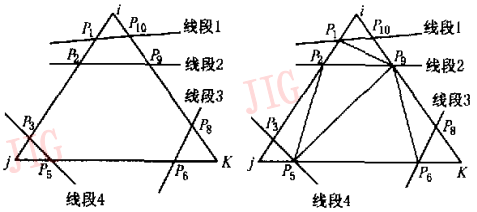
(a) 边界多边形顶点的编号 (b) Delaunay 三角化结果

图 3 边界多边形顶点排序及 Delaunay 三角化

这里需要特别一提的是,由于使用前面文字图象宽度把边界多边形的边分割得比较短,所以 Delaunay 三角化后得到的大多数都是内部三角形或外部三角形,由于相交三角形很少,即需要进一步剖分的三角形很少,因而不至于对整个算法的速度影响较大. 另外,为了方便算法的描述,后面有些例子是基于对分割前的边界多边形进行操作得到的结果.

1.2.2 相交三角形的剖分

由于边界多边形的边不会自交,且 Delaunay 三角形互不覆盖,所以 Delaunay 三角形与边界多边形相交的情况一般如图 4(a)所示.



(a) 相交情况 (b) 剖分结果

图 4 与边界多边形相交的 Delaunay 三角形及其剖分

假设一个 Delaunay 三角形的顶点分别为 i、j、k. 在进行相交三角形剖分时，首先计算三角形和边界多边形的交点，并按照 $i\rightarrow j\rightarrow k\rightarrow i$ 的顺序放到一个数组里. 这样数组中，点的存放顺序为 $i\rightarrow$ 在 i 和 j 之间的交点 $\rightarrow j\rightarrow$ 在 j 和 k 之间的交点 $\rightarrow k\rightarrow$ 在 k 和 i 之间的交点 $\rightarrow i$ 点. 由此可见，在该数组中存放的点有：Delaunay 三角形的端点和 Delaunay 三角形与边界多边形边的交点（算法中简称端点和交点）两类，其中，后者可以直接通过边界多边形的边到达另一个交点.

下面以图 4(a)为例，介绍一般相交三角形的剖分方法. 剖分时，首先以 $i\rightarrow p_1$ 作为起始边，从 p_1 点通过线段 1 到达 p_{10}点，再从 p_{10}点继续到达 i 点，这样就得到一个凸的多边形 iP_1P_{10}；然后可以很容易把它进行 Delaunay 三角化，并将生成的三角形添加到三角形序列里. 这时，该凸多边形中，只有边 $p_1\rightarrow p_{10}$是新边（不位于三角形边界上，其余的边都位于三角形边界上），接着，以 $p_1\rightarrow p_{10}$为起始边，重复上面操作，完成其余部分的剖分. 具体算法如下：

```
struct pointarraystruct{
  CPoint point;
  int  label; //标记是端点还是交点
  int  next; //next 记录一条边界多边形的边与 Delaunay 三角形相交的另一个交点的序号
}pointarray[Maxpointnum]; // Maxpointnum 记录三角形端点和交点的最大可能数目
```

三角形端点和所有交点按照 $i\rightarrow j\rightarrow k\rightarrow i$ 的顺序放到数组 pointarray 里.

如果是 Delaunay 三角形的端点，则置 label 为 0，next 置为 -1；如果是交点，则置 label 为 1，next 置为一条边界多边形的边与 Delaunay 三角形相交的另一个交点的序号.

```
Procedure DivideTriangle(P,V)
BEGIN
  L=∅;Q←P;//L 为生成的凸多边形的顶点集合;
  While Q ! = V Do
  Begin
    If Q 为交点 and pointarray[Q.next]> Q
    Then DivideTriangle(Q,Q 的下一个点);
    L=L∪{Q};
  If Q 为 Delaunay 三角形的端点 or Q 为交点 and Q.next<Q
    Then Q pointarray[m+1];
    Else Q=pointarray[Q.next];
  End
  L=L∪{Q};
```

对集合 L 进行 delaunay 三角化，同时把新生成的 Delaunay 三角形添加到三角形序列中：

```
END
```

为了便于删除边界多边形外部的三角形，赋予与每一对由边界多边形的边和 Delaunay 三角形的边形成的交点对应的边界边端点序号. 对图 4(a)进行剖分和 Delaunay 三角化处理以后，生成的三角形如图 4(b)所示.

1.2.3　外部三角形的删除

对相交三角形处理后，生成的所有三角形或者在边界多边形的内部，或者在边界多边形的外部. 为了删除边界多边形外部的三角形，需要利用三角形面积的正负概念[23]：

对于一个给定的$\triangle ijk$，它的 3 个顶点都有一个编号. 这里假定 $i<j<k$，i,j,k 3 点坐标为 $i(a_0a_1)$，$j(b_0b_1)$，$k(c_0c_1)$. 则其面积 S 计算公式为

$$\begin{vmatrix} b_0 & a_1 & 1 \\ b_0 & b_1 & 1 \\ c_0 & c_1 & 1 \end{vmatrix} = a_0b_1-a_1b_0+a_1c_0-a_0c_1+b_0c_1-c_0b_1=2\times S$$

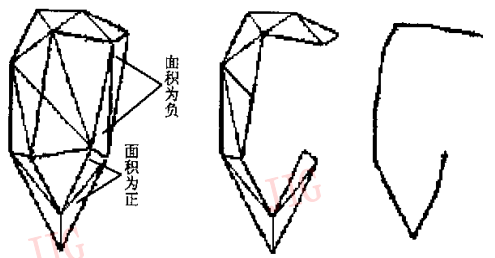
如果三角形的顶点编号顺序是逆时针方向，则三角形面积为正；如果编号顺序是顺时针方向，则三角形面积为负.

由于边界多边形的顶点是按逆时针方向由小到大的顺序进行编号的，因此对其三角化后得到的三角形顶点也都有编号，如果按由小到大的编号顺序来确定三角形顶点的顺序方向，那么可得到如下判断：

(1)内部三角形的面积为正；

(2)外部三角形的面积为负.

据此，即可进行外部三角形删除，即对于所有的三角形，如果其面积为正，则表明此三角形为内部三角形，应保留；如果三角形的面积为负，则为外部三角形，删除之. 图 5 给出了字母“C”的三角化及骨架化结果.



(a) Delaunay 三角剖分结果　(b) 删除了外部三角形后的剖分结果　(c) 骨架化结果

图 5　字母“C”的三角化及骨架化结果

1.2.4 骨架计算

对于最终所得到的三角形,把其在边界多边形边上的边称为外部边,在边界多边形内部的边称为内部边.这样由于三角形 3 条边的类型不同,因此就决定了三角形在整个图象中的位置特点.归纳起来,这些三角形一般分为如下 3 种类型(图 6):

(1)终端三角形 具有两个外部边的三角形,其只有在文字图象的开始和结束才能找到;

(2)连接三角形 具有一条外部边的三角形;

(3)分支三角形 没有外部边的三角形,其只有在文字图象分叉的地方才能找到.

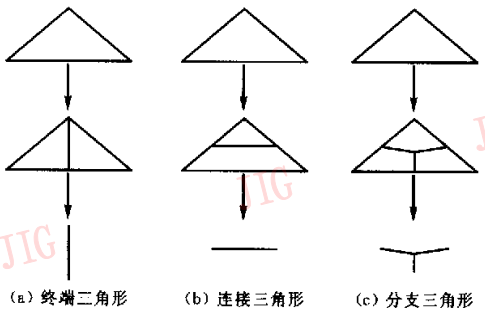


图 6 三角形的 3 种类型及其细化结果

(图中粗边为外部边)

这 3 类三角形的骨架定义如下:

(1)终端三角形骨架 三角形内部边的中点和对应顶点的连线(图 6(a));

(2)连接三角形骨架 三角形两条内部边的中点的连线(图 6(b));

(3)分支三角形骨架 三角形重心和 3 条边的中点的连线(图 6(c)).

1.3 含内边界的边界多边形的三角化

实际上,很多文字在结构上比图 2 中的例子要复杂得多,其边界多边形有时包含多个内边界,如图 7(a)所示.这种内边界可以采用文献[19]中的方法来先去掉,即可以在包含内边界的文字图象的最窄处截掉一小段(见图 7(a)圆圈处),先形成只有一个封闭轮廓的外边界;然后再按前面所介绍的算法进行骨架计算;最后可采用文献[19]的方法去除掉毛刺和恢复交叉点.另外,在对终端三角形求骨架时,如果发现两条外部边的比值大于 2 或者小于 1/2,可用内部边的中点和较短外部边的中点的连线来作为三角形的骨架,这样可消除大多数骨架在文字笔划末端的误差.例如,对于图 7(a)的字母,其左、右部各有一个内边界.其中对于右部的内边界,可找出其最窄的部分(图中圈 1 中标出处,圈 2 中标出的为圈 1 原样放大图,圈 3 中标出为边界处理示意图).通过比较发现,内边界上的 WQ 和外边界上的 VP 是相距最近的,且四边形 $VPQW$ 是完全在文字图象内部.在进行内边界删除时,首先,把这个四边形分解成两个连接三角形 PVQ 和 VQW,并加入到三角形的序列里;然后,将内边界上所有顶点添加到外边界的顶点 V 和 P 之间,以形成一个更长的外边界,这样就去掉了这个内边界.由于外边界形成的多边形的顶点编码顺序是逆时针的,而内边界形成的多边形顶点是顺时针的,所以添加过程是把内边界上由 W 点开始按顺时针到 Q 点结束的点序列加到点 V 的后面,再与 P 相连就形成了一个更长的外边界,这样就删除了这个内边界.图 7(b)给出了去掉一个内边界后的情况.在这里,连接外边界和内边界的边 VW 和 PQ 应该被标记成内部边,以便后面能够正确地进行骨架计算.重复上面的过程,就可以去掉所有的内边界,删除所有内边界后的结果如图 7(c)所示.所有的内边界去掉以后,就可以对没有内边界的多边形采用前面 1.2 节中的方法进行三角化,形成的骨架如图 7(d)所示.

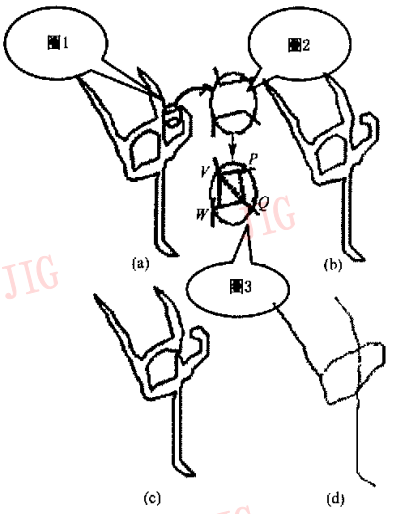
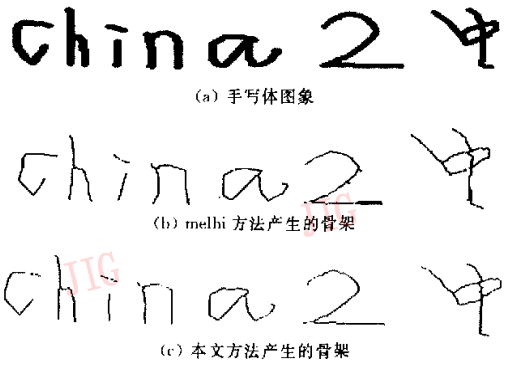


图 7 包含内边界的字母及简化处理

2 实验结果与算法复杂性分析

图 8 给出了几个手写体文字图象用 Melhi 方法

和本文方法生成的骨架. 显然，用本文方法要比用 Melhi 方法生成的骨架光滑得多.



(a) 手写体图象

(b) melhi 方法产生的骨架

(c) 本文方法产生的骨架

图 8　本文方法和 Melhi 方法生成的手写体文字骨架

对于 Melhi 方法，由于其主要是通过对多边形的每个顶点都先计算出其可能的对角线，然后根据三角形各边长度和的大小再决定如何分割，所以其时间复杂度为 $O(n^3)$，而本算法采用的 Delaunay 三角化算法的时间复杂性为 $O(n^2)$，由于其产生的 Delaunay 三角形数为 $O(n)$，故对每个三角形的处理时间复杂性为 $O(n)$，其总的时间复杂性为 $O(n^2)$. 由此可见，本文方法要比 Melhi 方法快得多.

3　结　论

本文基于 Delaunay 三角化方法，提出了一种新的文字图象细化算法. 该算法主要是通过对文字图象边界的近似多边形进行 Delaunay 三角化，即首先把其分成一系列保持拓扑关系的三角形；然后根据三角形的类型生成不同的局部骨架；最后连接生成整幅文字图象的骨架. 该算法不仅与图象分辨率无关，而且在细化时间和细化骨架的质量上都要优于已有算法. 此外，由于骨架本身就是用无向图结构存放的，所以便于识别等后续处理.

虽然本文方法是用来进行手写体文字的识别，但也可用于指纹、工程图等带状图象的骨架计算.

笔者将在此工作的基础上，充分利用文字图象的宽度等信息，研究新的方法，进一步降低算法的时间复杂性.

参 考 文 献

1　Abuhaiba I, Holt M, Datta S. Processing of binary images of handwritten text documents [J]. Pattern Recognition, 1996, 29(7):1161～1177.

2　Luk A, Leung S, Lee C *et al*. A two-level classifier for fingerprint recognition[A]. In: Proc. of IEEE Internat. Symp. on Circuits and Systems[C], Singapore, 1991:2625～2628.

3　DiezHiguera J, DiazPernas F, LopezCoronado J. Neural network architecture for automatic chromosome analysis[A]. In: Proc. of SPIE Applications of Artificail Neural Networks in Image Processing[C], San Jose, CA, USA, 1996:85～94.

4　Ye Q, Danielsson P. Inspection of printed circuits boards by connectivity preserving shrinking[J]. IEEE Trans. Pattern Anal. Machine Intell., 1988,10(5):737～743.

5　Blum H. A transformation for extracting new description of shape[A]. In: Wathen-Dunn W ed. Model for the perception of Speech and Visual Form[C]. Cambride, Massachusetts USA: MIT Press, 1967:362～380.

6　Pavlidis T, Ali F. Computer recognition of handwritten numerals by polygonal approximation[J]. IEEE Trans. Systems Man Sybernet, 1975,5(6):610～614.

7　Davies E R, Plummer A P. Thinning algorithms: a critique and a new methodology[J]. Pattern Recognition, 1981,14(1):53～63.

8　Lam L, Lee S, Suen C Y. Thinning methodologies—— A comprehensive survey[J]. IEEE Trans. Pattern Anal. Machine Intell., 1992,14(9):869～885.

9　Arcelli C, Cordella L, Leviadi S. Parallel thinning of binary pictures[J]. Electronic Lett., 1975,11(7):148～149.

10　Rosenfeld A. A characterization of parallel thinning algorithms [J]. Inform. Contr. 1975,29(3):286～291.

11　Zhang T Y, Suen C Y. A fast parallel algorithm for thinning digital patterns[J]. Commun. ACM, 1984,27(3):236～239.

12　Datta A, Parui S. A robust parallel thinning algorithm for binary images [J]. Pattern Recognition, 1994, 27 (9): 1181～1192.

13　Pavlidis T. A vectorizer and feature extractor for document recognition [J]. Comput. Vision Graphics Image Process. 1986, 35(1):111～127.

14　Lin J, Chen Z. A Chinese-character thinning algorithm based on global features and contour information [J]. Parttern Recognition, 1995,28(4):493～512.

15　Feng H F, Pavlidis T. Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition [J]. IEEE Trans. Comput., 1975,24(6):636～649.

16　杨承磊，孟祥旭. 一种新的快速细化算法的设计与实现[J]. 工程图学学报，1998,(3):87～93.

17　杨承磊，孟祥旭，李学庆等. 基于无向图的图象整体骨架表示模型及其算法[J]. 计算机学报，2000,23(3):293～299.

18　杨承磊，孟祥旭，李学庆等. 带状图象交叉区域的骨架求解算法[J]. 计算机辅助设计与图形学报，2000,12(9):677～681.

19　Melhi M, Ipson S S, Booth W. A novel triangulation procedure for thinning hand written text[J]. Pattern Recognition Letters, 2001,22(10):1059～1071.

20　Yuan J, Suan C Y. An optimal $O(n)$ algorithm for identifying

line segments from a sequence of chain codes [J]. Pattern Recognition, 1995, 28(5): 635～646.

21 Tarjan R, Wyk V. An $O(n\log\log n)$-time algorithm for triangulating a simple polygon [J]. SIAM J. Comput, 1988, 17(1): 143～178.

22 Kirkpatrick D, Klawe M, Tarjan R. Polygon triangulation in $O(n\log\log n)$ time with simple data-structures [A]. In: Proc. Sixth Ann. Symp. on Computational Geometry [C], Berkeley, CA, USA, 1990: 34～43.

23 Joseph, O'Rourke. Computational geometry in C [M]. Cambride, England: Cambridge University Press, 1994.

杨义军 1979 年生，硕士研究生. 主要研究领域为人机交互与虚拟现实、图象处理.

孟祥旭 1962 年生，教授，博士生导师. 主要研究领域为人机交互与虚拟现实、CAD、图象处理.

杨承磊 1972 年生，在职博士研究生，副教授. 主要研究领域为人机交互与虚拟现实、图象处理、计算几何.

汪嘉业 1937 年生，教授，博士生导师. 主要研究领域为计算机图形学、计算几何、人机交互与虚拟现实等.