

长安大学

硕士学位论文

线划图像的细化算法研究

姓名：曹玉龙

申请学位级别：硕士

专业：资源与环境遥感

指导教师：隋立春

20110603

## 摘 要

细化(Thinning)被定义为图像最外层像素的连续移除，直到生成一个连通的、单位像素宽度的骨架，这个骨架的理想状态是图像线条的中轴线。20年前，自从这种方法第一次被提出，细化就成了模式识别领域中重要的处理步骤之一。在测绘领域，二值图像的细化是地图矢量化的重要研究内容，也是地图符号识别的基础，在遥感图像边缘跟踪、分类中，线状图像的细化是重要组成部分。设计一个细化算法的大部分困难在于必须要充分考虑到图像总体的性质，除此之外还要考虑到噪声点的移除和交叉点的选取。要在不同的应用中根据不同的目的选择不同的细化方法，这主要由细化对象的类型决定。因此，以实际应用为基础研究细化方法具有一定的现实意义。

本文在论述细化算法的基本原理和方法、线划图像二值化和预处理的基础上，对影响线划图像细化的因素做了分析，随后，主要针对细化结果的两个方面（锐角和单像素化）做了具体分析。对于锐角细化产生冗余信息的问题，通过分析影响锐角细化效果的因素，提出了在预处理步骤中的改进方法。然后分析了在细化方法中的笔划趋势问题和“肿块”问题，接着对于细化后不能保证骨架为单像素的问题，在细化处理结果上继续对像素进行操作，以保证所有骨架都是单像素。本文对几种主要细化方法进行综合论述，并对 Zhang-Suen 算法进行改进，最后对原始算法和改进后的算法进行实验对比。

**关键字：**细化、预处理、锐角细化、笔划趋势、单像素化

## Abstract

Thinning can be defined as the successive erosion of the outermost layers of a figure until only a connected, unit width framework or 'skeleton' remains. This skeleton runs, ideally, along the medial lines of the limbs of the figure. Since the idea was first developed twenty years ago, thinning has become an important process in pattern recognition. In surveying and mapping field, the binary image's thinning is an important content of the map vector, and also is the base of the map's symbol recognition. In edge tracking and classification of the remote sensing image, the thinning of the threadiness images are important constituent parts. The biggest difficulty in devising a successful thinning algorithm is taking account of global properties of the patterns. Added to this is the difficulty of determining whether a point is an incipient 'noise' and whether, therefore, it should be retained or eliminated. According to different purposes of the applications we must choose different thinning methods, these are due to the types of the thinning objects. Thus, the studying on thinning methods based on practical applications has particular realistic significance.

On the bases of discussing on the fundamental principles of thinning, and the preprocessing prior to thinning, an analysis was made on the factors in the process of thinning the line drawings. And after that, a detail analysis was made on two of the factors of thinning (acute angle and singular pixel). For the error of the acute angle's thinning, an analysis was made about the error factors of thinning, proposing the improved program in the preprocessing, and then analysis the problems of stroke trend and the "Lump" in the thinning methods. For the problem of the singular pixel, proceed with the operator of pixel deletion after the thinning results to guarantee the singular pixel of the skeleton. There's discussing about some primary thinning methods in this paper, then making some improvements in the program of Zhang-Suen. There are some comparisons between the original program and the processed program at last.

**Key words:** Thinning、Preprocessing、Acute Angle Thinning、Stroke Trend、Singular Pixel

## 第一章 绪论

### 1.1 国内外发展及应用现状

细化是将图像的线条从多像素宽度减少到单位像素宽度过程的简称,一些文章经常将细化结果描述为“骨架化”、“中轴转换”和“对称轴转换”<sup>[1]</sup>。在电脑技术发展的早期,人们就意识到用计算机进行图像识别的可能,随之激发了大量的图像数据压缩的方法,从它的初期开始,就有发表近 300 篇关于这个课题的各种方面的论文,其最主要的方法就是减少图像的像素数,之后细化方法被广泛研究与应用<sup>[2]</sup>。很多细化算法的学者对细化概念做了定义,文献[3]将骨架定义为从图像线条的内部传播的波阵面的结果,骨架就是对边波阵面交集的轨迹,它作为一个概括性的理论被大多数学者所接受,图像中最外层的像素被连续的移除直到剩下骨架像素,但同时要保证骨架的连通性。文献[4,5]将骨架定义为图像线条的中心,使用这个骨架化定义可以精确地重建最初始的图像和重新描绘出它们。因为细化是模式识别领域的主要任务或本质要求之一,所以细化或骨架化算法的设计是一个非常活跃的研究领域。

20 年前,自从这种方法第一次被提出,细化就成了模式识别领域的重要处理步骤之一。它已经被用作检查印刷电路板,染色体分析,对地裂缝图的检查和对指纹的分类。它最初应用在光学字符识别中,近年来主要应用在智能复印和传真发送系统中,另外一个应用和发展是在地图数据压缩中。总的来说,这些应用程序使用细化方法将数字图像的线条简化为单位像素宽度的中心线,使对象能够被更简单的数据结构表示来简化数据分析,或者减少数据存储和对传输设备的要求。基本上细化的所有目的为减少冗余信息,留下足够的有用信息来进行拓扑分析、形状分析或者原始对象的还原。对于本来就是线状图形的对象,细化方法允许骨架的大小、形状和表面噪声几乎与原图像保持一致。在这种情况下,细化的目的是改善图像,而不仅仅是消极的进行数据压缩。在模式识别中细化的重要性是显而易见的,它的重要性被过去的 40 年中的大量的细化算法所反映出来。算法的大规模发展支撑了这个观点,到现在为止,细化算法还没有被充分的发展和完善,未来它将比刚开始出现满足一般细化要求时更复杂。

设计一个连续细化算法的大部分困难在于必须要充分考虑到图像的总性质,人们都知道,连通性是一个全局属性,这是计算速度问题的根源。虽然发现并保留骨架线的端点只是个局部问题,但定位它们也是一个全局问题。除此之外,判断线上的一个像素点是不是毛刺噪声,它该被保留还是删除,这是非常重要的。不同算法对这个问题的解

决方法是不同的,有些方法可能导致整个分支的删除或者将噪声细化。对细化一般的要求<sup>[6]</sup>:

- A1:保证细化后骨架的连通性;
- A2:对原图像的细节特征要保持的较好;
- A3:细化的结果是原线条的中心线;
- A4:线条的端点保留完好;
- A5:线条交叉点不能发生畸变;
- A6:细化所用时间不能太长;

从以上 6 点要求中可以发现,前 5 条准则主要集中在对细化质量的评价,最后一条是对细化速度的要求。由于目前常用的细化算法需要不断重复的扫描图像,在算法的计算过程中耗时较多,所以如何在改善细化质量的同时,用最少的迭代次数得到完全细化的图像并且缩短细化时间,这是现在线划图像细化研究的重点。

由于细化的重要性,细化算法一直是人们比较关注的问题,因此出现了比较多的细化算法,无论哪种细化算法,都可能会因为目的和用途的不一样而产生不同的效果。整个细化过程都不能改变原图像的拓扑连接性,也不能有明显的端点缩短,分叉点分离等畸变现象。因此,有大量的复杂问题阻碍了我们设计理想的细化算法。此外,和一个简单图像处理任务一样,如果细化被认为是一个基础算法,那么尽可能的解决基本问题的重要性就非常大了。

## 1.2 细化方法存在的问题

细化是提取能表达图像拓扑结构的骨架像素的方法,为计算机系统进行数据压缩和识别创造条件,但要注意三点:

B1:并非所有形状的图像都可以或者应该细化,细化比较适合由线条组成的物体,如圆环,但实心圆不适合细化。

B2:任何一种细化方法都不能适用所有的情况。

B3:细化是提取骨架的过程,所提取的骨架必须有实质的意义,而不是由所使用的细化算法来定义骨架。

在细化中,理想状态是在对图像像素进行连续移除后形成一个具有连通性的骨架。达到这个目的必须解决三个问题:C1:保持连通性;C2:保留端点;C3:确保像素点是对称的被移除,目的是使算法能各向同性。在理论上和实际工作中,这些问题都是实际存在的,解决方案很不一致,有些甚至存在冲突。

### 1.2.1 保持连通性<sup>[7]</sup>

在串行算法中，图像中的每个点都按顺序被检查，在下一个点被检查之前，任何改动都变成了这个图像被处理的一部分。保持连通性实际上是非常琐碎的，这是因为任何一个叉数值  $\chi=2$  的点都将被删除（叉数被定义为图像 8-邻域中从白色像素点到黑色像素点转变的数量，反过来也成立），同时叉数值  $\chi=0$  或  $\chi>2$  的点必须被保留。对于并行算法来说，上面的  $\chi=2$  的情况对于点的移除也是必须的条件，不幸的是这种情况没有被一些学者所认识到，这导致了在一定情况下骨架断开情况的增加。下面的困难是矩形的连通性问题，与更简单的六边形的叉数相比，矩形叉数必须使用更复杂的公式来表示：

$$\chi_r = \sum_{k=1}^4 (\gamma_{2k-1} \oplus \gamma_{2k+1}) + 2(\overline{\gamma_{2k-1}} \cap \gamma_{2k} \cap \overline{\gamma_{2k+1}}) \quad (1.1)$$

$$\chi_h = \sum_{k=1}^6 (\gamma_k \oplus \gamma_{k+1}) \quad (1.2)$$

式(1.1)与式(1.2)分别表示矩形叉数和六边形叉数，其中“ $\cap$ ”，“ $\overline{\phantom{x}}$ ”，“ $\oplus$ ”这几个符号分别代表了逻辑操作符与、非和异或， $\gamma_k$  是如图 1.1 或图 1.2 所示的 3\*3 窗口内像素的值：

$$\begin{array}{ccc} & \gamma_2 & \gamma_3 \\ \gamma_1 & \gamma_0 & \gamma_4 \\ & \gamma_6 & \gamma_5 \end{array}$$

图 1.1 六边形结构

$$\begin{array}{ccc} & \gamma_2 & \gamma_3 & \gamma_4 \\ \gamma_1 & \gamma_0 & \gamma_5 \\ & \gamma_8 & \gamma_7 & \gamma_6 \end{array}$$

图 1.2 矩形结构

人们注意到经常在矩形叉数中用到的式 1.3 和式 1.4 都没有合适的理论基础也不能提供一个有效的错误概率：

$$\chi_r' = \sum_{k=1}^4 (\gamma_{2k-1} \oplus \gamma_{2k+1}) \quad (1.3)$$

$$\chi_r'' = \sum_{k=1}^8 (\gamma_k \oplus \gamma_{k+1}) \quad (1.4)$$

在平行算法中用条件  $\chi=2$  来进行像素点的删除是不够的，这是另一种困难的情况。很多种不同的算法试图用不同的方法来解决这个问题，大部分的方法被认为很特殊，因为在任何条件下都需要更加详细的条件要求。总而言之，从某种意义上说一个算法如果没有导致一个骨架的断开，它必须在删除像素的步骤上特别谨慎，它能生成一个达到需求的最好的骨架，但是生成骨架的速度会降低。

### 1.2.2 保留端点

保证端点被保留或者识别端点等类似的情况增多了，因为把它们标记出来并保留是很麻烦的，文献[8]中指出对于串行算法来说，主要问题是对于两个像素宽的分支，这些分支趋向消失。在并行算法中出现了一个更基本的问题，在点被确认为端点前，许多边界点被删除的同时，端点也被删除了，并行算法能采用谨慎删除像素的方法来解决这个问题，由于大部分存在的算法采用循环定位端点位置的方法来进行后续的处理，所以在图像中端点被更好的定位了<sup>[9]</sup>。

伴随着端点定位的问题要防止产生“毛刺噪声”。大部分算法在绝对反映目标的形状和抑制所有毛刺噪声之间控制平衡。在一个 3\*3 窗口内，仅有的区分有效点和毛刺噪声点的方法取决于设立一个已知黑色像素的邻接像素的阈值  $\sigma$ ：

$$\sigma = \sum_{k=1}^8 \gamma_k \quad (1.5)$$

式 1.5 中，一方面如果阈值被设为 2，此黑色像素被保留。另一方面，如果阈值被设为 3 或更大，在目标图像拐角处的有效端点有时就被删除了。不论是哪种情况，这些方法看起来都很特殊，很明显在 3\*3 窗口中没有足够的空间来区分噪声点。虽然这种情况可以用 3\*3 窗口连续地操作识别来改善，但对这种连续操作的研究仍然是不令人满意的。

### 1.2.3 对称剥离

最后，确保图像中的像素点被对称地剥离也是相当复杂的<sup>[8]</sup>。在串行算法中，图像东南方向像素的移除存在一个很大的偏差，而这些偏差被连续的边界点标记和连续的剥离操作所避免了（比如传统的前进式光栅扫描）。这些方法有很高的成功率，但是如果每次扫描剥离都是前进式光栅扫描，那么偏差仍然存在，因为在每个被标记的骨架东南方向的像素点中有系统误差，这些误差影响能被东、南、西、北方向连续的像素剥离所除去，但是这在串行算法中不经常执行，因为它降低了处理速度，然而这个步骤经常在并行算法中使用。并行算法中，在图像细化处理的末尾部分有两像素宽的线条，它没有消失，按北、南、东、西的顺序进行移除操作，然后使用一个 5\*5 的操作窗口来确保连通性<sup>[10]</sup>。值得注意的是是一些学者如 Rutovitz<sup>[11]</sup>和 Deutsch<sup>[12]</sup>开始使用比 3\*3 更大的窗口，因此简化了并行细化算法的设计。

关于北、南、东、西不同方向移除方法的变体方法已经发展出来，这些方法包括<sup>[7,9,13]</sup>：

D1: 北、南、东、西按不同顺序的应用操作。

D2: 较少的(北+西,南+东),多一些的(北,东北,东...)像素剥离操作。

以上三部分所做的分析是非常有必要的,算法总是被当作完整的程序包来运行,用以满足一定的任务:例如,剥离和标注端点的步骤有部分相同的。我们已经讨论了很多在算法设计上的“特殊”,因为我们感觉到这个对现存算法的发展构成了一个重大的限制。这是因为特别的设计会增加算法或多或少的缺陷,这些缺陷不容易被矫正;它也使算法缺少对新情况的适应性,比如,对更大噪声级的包容,算法的可移植性,或改进处理速度,最重要的是,它使细化算法的精确性不可预知。

## 1.3 研究的意义和内容

### 1.3.1 意义

用骨架来表示线划图像能够有效地减少数据量,减少图像的存储难度和识别难度。目前线划图(包括纸质地图、线画稿、手绘图等)的存储是非常麻烦的,存储和使用起来都很不方便。例如,一张 A4 大小的线划图存储需要 1M 的容量,另外一些比较严重的问题就是数据的修改、更新和显示。矢量化是解决这些问题的方法,但这些图的宽度经常是大于一个像素的,这会导致矢量化结果有非常大的问题,为了解决这些问题,细化就成了模式识别和矢量化的先决条件。

细化技术的一个主要应用领域是位图矢量化的预处理阶段,相关研究表明,利用细化技术生成的位图的骨架质量受到多种因素的影响,其中包括图像自身的噪声、线条粗细不均匀、端点的确定以及线条交叉点选定等,因而对线划图像进行细化从而生成高质量骨架的方法进行研究具有现实意义。

### 1.3.2 内容

针对目前线划图像预处理和线划图像细化算法的研究现状,为了便于提取正确的线条特征信息以及提高运算效率,对几个影响细化结果的问题进行了研究并提出了一些改进的方法。内容如下:

第一章简单介绍了细化技术,并对该技术在国内外的发展及应用现状做了介绍,最后对实际应用中细化的缺点进行了论述。

第二章主要对线划图像的二值化及预处理步骤进行了介绍。最后,分析了进行预处理的必要性。

第三章首先介绍了细化算法的分类,并对几种现存的主要类型的细化算法进行了比较分析,最后论述了不同类型的算法在实际中的运用。



第四章首先介绍了 Zhang-Suen 算法的基本原理，对此算法的缺点进行了分析，接着讨论了在预处理中和在细化处理结果上改进操作的方法，并对细化过程中的两种特殊情况进行讨论，最后使用一些实际的线划图像进行了比较分析。

第五章对本文的研究内容进行了总结，并对细化算法存在的问题以及今后的发展方向提出了看法。

## 第二章 线划图像的预处理

### 2.1 线划图像的二值化

线划图像在细化之前除了要去噪声外，还要先进行二值化处理。二值化可以将一幅灰度图像转换成黑白二值图像，根据运算范围的不同，线划图像的二值化方法可以分为全局方法和局部方法，全局方法根据线划图像的灰度直方图和灰度空间分布确定一个阈值，由此实现灰度图像到二值图像的转变，比较有代表性的全局算法包括平均灰度法，Otsu 方法，迭代最优算法等。局部方法是通过检查每个像素点的邻域来确定局部阈值，它比全局阈值有着更广泛的应用，典型的局部阈值方法有 Niblack 方法，Bernsen 方法，平均梯度法等。

#### 2.1.1 全局阈值法

期望灰度值法<sup>[14]</sup>：设图像的尺寸为  $M \times N$ ，其灰度取值为  $L_1, L_2, L_3, \dots, L_N$ ，用随机变量  $X$  来表示每个像素点的灰度值。可以用概率分布来描述图像的灰度分布情况，分别设各灰度级出现的概率为式 2.1：

$$p_1 = p(L_1) \quad , \quad p_2 = p(L_2) \quad , \dots, \quad p_N = p(L_N) \quad \text{且} \quad \text{有} \quad \sum_{n=1}^N p_n = 1 \quad (2.1)$$

灰度图像的一个重要的统计量就是灰度期望值，使用它为阈值可以使黑色像素和白色像素的灰度值均等，公式(2.2)即为灰度期望值的计算式：

$$\mu_{threshold} = \sum_{n=1}^N L_n p_n \quad (2.2)$$

该算法对于简单图像的效果好，而且计算复杂度较低，缺点是对于亮度不均匀的线划图像效果较差。

Otsu 方法<sup>[15]</sup>：这种方法又被称为最大类间方差方法，是一种自适应的阈值确定方法。它根据灰度特性，将图像分为目标和背景两个部分，目标和背景的方差越大，说明这两个部分的差别越大，因此类间最大方差的分割意味着错分概率最小。

对于灰度图像  $I(x,y)$ ，目标和背景的分割阈值为  $T$ ，属于目标的像素点数占整个图像像素点的比例是  $\omega_0$ ，其平均灰度值为  $\mu_0$ ；背景像素点数占整个图像像素点的比例为  $\omega_1$ ，其平均灰度值为  $\mu_1$ ，图像总平均灰度值记为  $\mu$ ，类间方差为  $g$ 。

假设图像  $M \times N$  的背景比较暗，图像中像素灰度值小于阈值  $T$  的像素个数记为  $N_0$ ，

像素灰度值大于阈值  $T$  的像素个数记为  $N_1$ ，所以有式 2.3 到式 2.8 成立：

$$\omega_0 = \frac{N_0}{M \times N} \quad (2.3)$$

$$\omega_1 = \frac{N_1}{M \times N} \quad (2.4)$$

$$N_0 + N_1 = M \times N \quad (2.5)$$

$$\omega_0 + \omega_1 = 1 \quad (2.6)$$

$$\mu = \omega_0 \mu_0 + \omega_1 \mu_1 \quad (2.7)$$

$$g = \omega_0 (\mu_0 - \mu)^2 + \omega_1 (\mu_1 - \mu)^2 \quad (2.8)$$

对于灰度图像来说，分别以每个灰度为阈值计算对应的类间方差，其中使类间方差最大化的灰度值即为阈值。该算法对于较简单的线划图像有良好的效果而且有较快的运算速度，所以这种算法的应用非常广泛。

最优阈值法<sup>[16]</sup>：又被称为逼近迭代算法，这种方法的原理是将直方图用两个或多个正态分布的概率密度函数来近似表示的方法，阈值取为对应两个或多个正态分布的最大值之间的最小概率处的灰度值，其结果是具有最小误差的分割。这里的误差包括两部分：将目标误认为背景而被剔除或将背景、噪声归为目标。最优阈值法的处理步骤：

E1: 计算图像的最小灰度值  $Z_{\min}$  和最大灰度值  $Z_{\max}$ ，令阈值初值为式 2.9：

$$T^0 = (Z_{\min} + Z_{\max}) / 2 \quad (2.9)$$

E2: 根据阈值将图像分为目标和背景两部分，求出两部分的平均灰度值  $Z_0$  和  $Z_1$ ，分别计算  $Z_0$  和  $Z_1$  的平均值，用公式 2.10 来表示：

$$Z_0 = \frac{\sum_{I(i,j) \leq T^k} I(i,j)}{\#I(i,j) \leq T^k}, \quad Z_1 = \frac{\sum_{I(i,j) > T^k} I(i,j)}{\#I(i,j) > T^k} \quad (2.10)$$

E3: 计算新阈值，用式 2.11 表示：

$$T^{k+1} = (Z_0 + Z_1) / 2 \quad (2.11)$$

如果  $T^k = T^{k+1}$  或者达到设定的最大迭代次数就结束，否则转向步骤 E2。该算法能较好区分图像的目标和背景，但是会导致图像细微信息的丢失。

### 2.1.2 局部阈值法

Niblack 方法<sup>[16]</sup>：基于局部均值和局部标准差，基本公式如式 2.12：

$$T(x, y) = m(x, y) + k \cdot s(x, y) \quad (2.12)$$

对于图像  $I(x, y)$ , 在  $(x, y)$  处的阈值  $T(x, y)$  由局部均值  $m(x, y)$  和局部标准差  $s(x, y)$  决定,  $k$  表示调整系数, 一般设为 -0.2。在 Niblack 方法中, 窗口大小的选择非常重要, 既要小到能保持足够的局部细节又要大到能抑制噪声。Niblack 方法能很好地保持图像细节, 对于清晰的线划图像能够提供很好的二值化结果, 但是在一些模糊的线划图像中会保留一些不必要的细节。在最初的 Niblack 方法中,  $k$  值是固定的, 但是对于不同的图像, 通常需要根据图像的灰度分布情况自动调整  $k$  的值才能取得较好的结果, 因此后来提出很多改进的算法, 其中一个改进的 Niblack 方法的基本公式为式 2.13:

$$T(x, y) = m(x, y) \left[ 1 + k \left( 1 - \frac{s(x, y)}{R} \right) \right] \quad (2.13)$$

$k$  和  $R$  都是经验常量, 改进的 Niblack 方法使用  $k$  和  $R$  来减少对噪声的敏感度。

### 2.1.3 分解方法

上述的几种二值化方法只能适用于一类线划图像, 对于一些退化较严重的图像, 上面的方法都不能得到理想的二值化结果。

分解方法的大致思想是利用四叉树将灰度图像进行递归分解, 直到可以根据子图像的特征进行分类, 分别对不同的子图像进行不同的处理<sup>[17]</sup>。大致流程为: 首先输入灰度图像  $I(x, y)$  为子图像; 接着计算灰度图像的对比度  $\text{contrast}$ ; 然后如果  $\text{contrast}$  小于设定阈值或者图像大小已经小于  $64 \times 64$ , 则使用四叉树的方法进行分解, 然后从开始执行, 否则执行下一步; 最后根据子图像中笔划的平均梯度获取子图像中的笔划方向, 再通过共生矩阵提取子图像的纹理特征判断子图像的类型, 分为三种: 背景, 模糊子图像块和清楚子图像块, 不同类型的子图像使用不同的二值化方法。

## 2.2 预处理

灰度图像在进行二值化的操作后通常要先进行预处理, 来降低细化的难度。但是, 数据中特有的缺点会导致细化算法破坏信息。这些缺点的数量和重要性对于每个问题和每个应用都是特殊的, 它们对细化算法的影响仅能通过实验来估计。这种数据常见的三种缺点是<sup>[18]</sup>: F1: 在骨架线上出现的孔洞会导致假的圆环出现; F2: 线条上的毛刺; F3: 在分支间的锐角, 这些尖的锐角会引起变形和伪线段。大部分细化算法是在保证形状连通情况下对形状边缘连续的侵蚀, 所有这些算法对以上缺点中的一个或多个很敏感, 连通性准则过分强调小孔洞的重要性, 造成了错误的圆环骨架和多余的分支。边界侵蚀算法的一个特征是周长长一些的图形分支一边移除的像素多于另一边, 这意味着分支的外侧侵蚀速度比内侧侵蚀速度快, 这导致了多余分支和伪信息的出现。这些问题在标准的

细化算法应用之前，可以先运用一些预处理方法，这些预处理方法目的是为了减少要处理数据存在的类似上面提到的问题。

### 2.2.1 膨胀和收缩<sup>[19]</sup>

所谓膨胀就是把连接成分的边界扩大一层的处理。而收缩则是把连接成分的边界点去掉从而缩小一层的处理。若输出图像为  $g(i,j)$ ，则它们的定义为：

$$\text{膨胀: } g(i,j) = \begin{cases} 1 & \text{像元}(i,j) \text{ 为1或其4-/8-邻域的一个像素为1} \\ 0 & \text{其他} \end{cases}$$

$$\text{收缩: } g(i,j) = \begin{cases} 0 & \text{像元}(i,j) \text{ 或其4-/8-邻域的一个像素为0} \\ 1 & \text{其他} \end{cases}$$

收缩和膨胀是数学形态学最基本的变换，数学形态学的应用几乎覆盖了图像处理的所有领域，这里给出利用数学形态学对二值图像处理的一些运算。

令  $E = R^2$  和  $E = Z^2$  分别为二维欧几里德空间和欧几里德栅格。二值图像目标  $X$  是  $E$  的子集。用  $B$  代表结构元素， $B^s$  代表结构元素  $B$  关于原点  $(0,0)$  的对称集合，如式 2.14：

$$B^s = \{-b : b \in B\} \quad (2.14)$$

膨胀和腐蚀变换的定义为式 2.15 和式 2.16：

$$\text{膨胀} \quad X \oplus B^s = \bigcup_{b \in B} X_{-b} = \{z \in E : B_z \cap X \neq \emptyset\} \quad (2.15)$$

$$\text{腐蚀} \quad X \ominus B^s = \bigcap_{b \in B} X_{-b} = \{z \in E : B_z \subset X\} \quad (2.16)$$

膨胀变换  $X \oplus B^s$  是把结构元素  $B$  平移  $z$  后得到  $B_z$ ，使  $B_z$  与  $X$  交集不为空集的所有点  $z$  构成的集合。膨胀是一个扩张的过程。这种变换目标扩张，孔洞收缩。式 2.15 为膨胀变换的结果。腐蚀变换  $X \ominus B^s$  是把结构元素  $B$  平移  $z$  以后得到  $B_z$ ，使  $B_z$  包含于  $X$  的所有点  $z$  构成的集合。腐蚀变换的结果是  $X$  的子集，因此是一种收缩变换。这种变换使目标收缩，使孔洞扩张。式 2.16 为腐蚀变换的结果。膨胀和腐蚀是明可夫斯基加  $X \oplus B$  和明可夫斯基减  $X \ominus B$  的特殊情况如式 2.17 和式 2.18：

$$\text{明可夫斯基加} \quad X \oplus B = \bigcup_{b \in B} X_b \quad (2.17)$$

$$\text{明可夫斯基减} \quad X \ominus B = \bigcap_{b \in B} X_b \quad (2.18)$$

一般情况下，膨胀和腐蚀是不可恢复的运算。先腐蚀再膨胀通常不能使目标  $X$  复原，

而是产生一种新的形态学变换，叫做开运算  $X_B$ ：

$$X_B = (X \ominus B^s) \oplus B = \bigcup \{B_z : B_z \subset X\} \quad (2.19)$$

式 2.19 中  $X_B$  是  $X$  内  $B$  的所有平移  $B_z$  的并集组成。与开运算相对应的是闭运算  $X_B$ ，即先膨胀再腐蚀：

$$X_B = (X \oplus B^s) \ominus B = \bigcap \{B_z^c : B_z \subset X^c\} \quad (2.20)$$

式 2.20 中  $X_B$  是  $X$  外  $B$  的所有平移  $B_z$  的补集和交集。

开运算使目标轮廓光滑，并去掉了毛刺、孤立点和锐化角。闭运算则填平小沟，弥合孔洞和裂缝。

膨胀和腐蚀的反复使用就可检测二值图像中的小成分或清除孔洞。如图 2.1 到图 2.3 为几种效果的对比。



图 2.1 未处理的图像

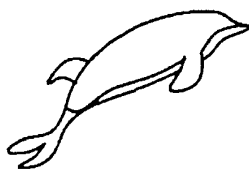


图 2.2 腐蚀处理效果



图 2.3 膨胀处理效果

## 2.2.2 孔洞移除<sup>[18]</sup>

这个处理中，图像被放在一个二值矩阵当中，预处理以扫描仪扫描六种类型的孔洞式样为开始，图 2.4 这三种矩阵将孔洞用黑色像素代替，由图 2.5 所代表的内部孔洞更接近于一个真的圆环而被保留，其它由图 2.4 定义的与线条边缘接近的孔洞都被归为噪声，这些孔洞有两种被移除方法：G1:通过移除黑色像素使孔洞和周围的白色区域合并；G2:用其它的黑色像素来填充。

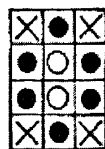
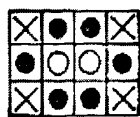
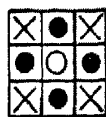


图 2.4 移除孔洞矩阵

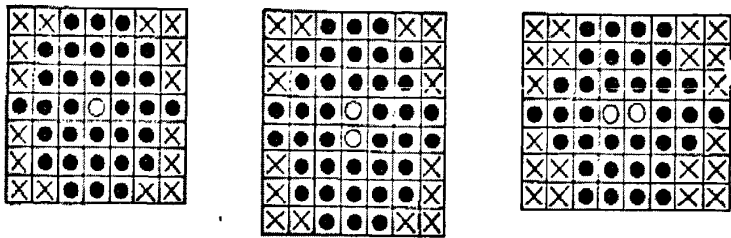


图 2.5 保留孔洞矩阵

如图 2.6 和图 2.7 为孔洞移除前后的对比图像，可以很明显看出移除孔洞后的图像像素点更加连续，为提高后续的细化、矢量化等操作的质量创造了条件。

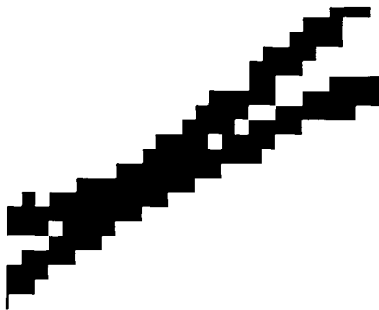


图 2.6 未移除孔洞的图像

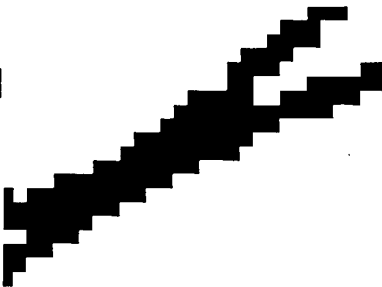


图 2.7 移除孔洞后的图像

2.2.3 去噪<sup>[18]</sup>

在进行腐蚀膨胀和孔洞移除的处理后，接下来的预处理是移除掉所有少于 3（此值的大小是可变的，一般根据实际噪声情况，选择不同的值）个相邻黑色像素的黑像素，这个过程是删除毛刺和孤立的噪声点。如图 2.8 和图 2.9 为去除噪声前后对比的图像，可以看出经过处理后的图像，噪声和毛刺被很好的去除了，降低了后续细化及矢量化处理的难度。



图 2.8 原始图像



图 2.9 去除毛刺和噪声后的图像

## 2.3 本章小结

本章介绍了对细化之前的图像进行二值化和预处理过程中遇到的一些问题，首先介绍了图像的二值化算法，接着介绍了图像预处理的最一般形式膨胀和收缩，然后介绍了图像上孔洞移除的原理，最后对图像进行去噪后的结果进行了分析。通过对各种预处理情况的实验前后进行对比，说明了预处理操作对于图像细化的必要性。



### 第三章 细化方法综合分析

本章中讨论的是广大范围的细化算法，根据算法处理步骤的不同，细化算法分为迭代细化算法和非迭代细化算法。根据检查像素方法的不同，迭代细化算法又分为串行细化算法和并行细化算法。在串行算法中，通过在每次迭代中用固定的次序检查像素来判断是否删除像素，在第  $n$  次迭代中像素  $p$  的删除取决于到目前为止执行过的所有操作，也就是必须在第  $(n-1)$  次迭代结果和第  $n$  次检测像素的基础之上进行像素删除操作；在并行算法中，第  $n$  次迭代中像素的删除只取决于  $(n-1)$  次迭代后留下的结果，因此所有像素能在每次迭代中以并行的方式独立的被检测。非迭代细化算法是不以像素为基础的，它们通过一次遍历的方式产生线条的某一中值或中心线，而不检查所有单个像素，这类算法中最简单的方法是通过扫描确定每段线的中心点，然后把它们连接成一副骨架，这种算法拥有计算机处理速度的优势，但是会有容易产生噪声点的缺陷。

#### 3.1 基本定义

被检测的像素  $p$  是黑色像素，像素所在的  $3 \times 3$  窗口被标志成图 3.1 所示，由  $N(p)$  表示它们整体， $N(p)$  中黑色像素的数目由  $b(p)$  表示。

$p_4$	$p_3$	$p_2$
$p_5$	$p$	$p_1$
$p_6$	$p_7$	$p_8$

图 3.1 图像连通域

在介绍细化算法之前，先介绍一下图像的基本概念。图像的连通性在图像细化和目标识别等处理过程中对边界的建立和区域的选择是一个很重要的基本概念。要确定某两像素间是否存在连通性关系，首先要确定其是否相互接触，严格来讲还要看是否具备相同的灰度值。我们可将二值图像看作是像素点阵的形式，像素点阵形式实际上可以看作是矩形阵列，现在分别介绍 4-邻域及 4-连通，8-邻域及 8-连通的定义。

4-邻域及 4-连通：对于一个  $3 \times 3$  的矩阵而言，如上图 3.1 所示，其中心像素点  $p$  的坐标为  $(x, y)$ ，它有 4 个近邻像素，其坐标分别为  $(x, y+1)$ ， $(x-1, y)$ ， $(x, y-1)$ ， $(x+1, y)$ ，分别为图中的  $p_1$ ， $p_3$ ， $p_5$ ， $p_7$ 。这 4 个像素组成像素  $p$  的 4-邻域，并且称其 4-连通于  $p$ 。

8-邻域及 8-连通：像素点  $p$  有 4 个对角近邻像素，其坐标分别为  $(x-1, y-1)$ ， $(x+1, y-1)$ ， $(x-1, y+1)$ ， $(x+1, y+1)$ ，分别为图 3.1 中的  $p_2$ ， $p_4$ ， $p_6$ ， $p_8$ 。这 4 个像素组

成像素  $p$  的 4 对角近邻像素, 4-对角像素和 4 对角近邻像素共同构成了中心像素  $p$  的 8-近邻像素。

4-连通曲线与 8-连通曲线: 对于一条曲线, 如果其上任意一点都 4-连通于其前一点, 则称该曲线为 4-连通曲线; 如果曲线上任意一点都 8-连通于其前一点, 则该曲线称为 8-连通曲线。

4-连通曲线与 8-连通曲线如图 3.2 和图 3.3 所示:

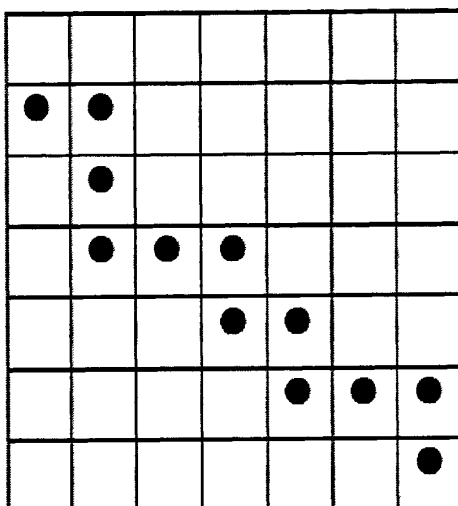


图 3.2 4-连通曲线

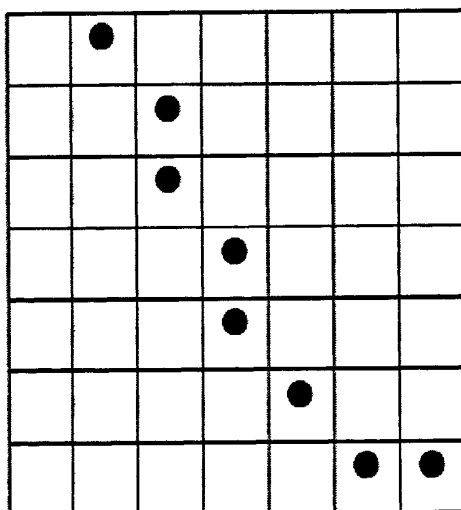


图 3.3 8-连通曲线

相互连通的前景像素点组成的整体称为一个连接成份, 图像中被此分离的连接成份的数目称为连接成份数。之所以介绍连接成份数的概念是因为, 在本文中要利用连接成份数的概念作为细化的判别条件。

在本论文中对线划图像进行细化处理中, 判断是否删除一个前景像素点需要考虑的

是：在  $3 \times 3$  的邻域中除其自身外的 8 个像素点中的连接成份数。如果连接成分数为 1，则说明删除当前像素点不会改变原图像的连通性，若大于 1，删除该像素点就会改变了原图的连通性。

设  $N$  为  $p$  的 8-邻域中的连接成分数，则其由序列  $p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_1$  中 0 到 1 的变化的次数就可以得到连接成份数  $N$ 。连接成分数又被称为叉数，它有两种定义，文献[20]中首先提出了这个对连通性有效的衡量方法，即为从白色像素点到黑色像素点转变的数量，反过来也一样，当  $p$  和周围的像素点按顺时针方向旋转时也是如此，因此叉数  $X_R(p)$  可以定义为式 3.1：

$$X_R(p) = \sum_{i=1}^8 |x_{i+1} - x_i| \quad (3.1)$$

文献[21]中把叉数  $X_H(p)$  定义为当  $N(p)$  的像素点按次序遍历时，从白色像素点到黑色像素点转变的次数定义为式 3.2：

$$X_H(p) = \sum_{i=1}^4 b_i$$

$$b_i = \begin{cases} 1, & x_{2i-1} = 0 \text{ 且 } (x_{2i} = 1 \text{ 或者 } x_{2i+1} = 1) \\ 0, & \text{其它情况} \end{cases} \quad (3.2)$$

如果  $X_H(p)=1$ ，那么  $p$  的删除将不改变图像的 8-连通性。在叉数  $X_H(p)$  和  $X_R(p)$  之间的一个不同点是， $X_H(p)=1$  的情况也会表示  $p$  是一个轮廓点，然而  $X_R(p)=2$  不能确定这种情况，因为如果  $p$  恰好有一白色邻近对角像素点的时候也成立。为了避免在这种情况下删除  $p$ （产生孔洞），就需要其它条件来确定  $p$  是一个轮廓像素。

被标记删除的像素通常叫做简单像素，并且已经验证，在一个简单的 8-连通图像中，一个非孤立的轮廓像素  $p$  只有在  $X_H(p)=1$  的情况下，才是一个简单像素<sup>[22]</sup>。多重像素包括分支的端点，宽度为两个像素的线条和在基于连通性的标准下应该被确定为骨架的像素，这些像素在细化处理过程中将被保留。

根据它们的操作模式和所用像素检测的标准，依照表 3.1，能大体上对许多细化算法进行分类。根据这个综合的图解，串行算法可以通过处理轮廓像素或者光栅扫描来进行操作，并行算法通过使用 4 次、2 次或 1 次迭代算法来进行操作。两种算法都通过与细化所用窗口的匹配方法或只删除简单像素的方法或保留多重像素的方法来找到叉数  $X_R(p)$  或  $X_H(p)$  来保证骨架的连通性。

表 3.1 细化算法分类

操作		像素测试标准			
		$X_R(p)$	$X_H(p)$	窗口匹配	简单或多重像素
串行	轮廓像素	Arcelli[23] Wang[24]		Beun[25] Pavlidis[26] Chu[27]	Pavlidis[28] Arcelli[29]
	光栅扫描	Arcelli[30]	Hilditch[21] Yokoi[31]		Arcelli[32]
并行	4 次子循环		Rosenfeld[33] Hilditch[34]	Stefanelli[35]	Arcelli[36]
	2 次子循环	Deutsch[37] Zhang[38] Chen[39]	Suzuki[40] Guo[41]	Stefanelli[35]	
	1 次子循环	Rutovitz[20] Holt[42]	Chen[43]	Chin[44]	

3.2 串行细化算法

使用串行算法，轮廓像素按确定好的顺序被检测来删除，而且用光栅扫描或轮廓跟踪都可以完成这个操作。如果图像的所有边界和孔洞被跟踪到的话，轮廓跟踪算法可以访问单连通图像的边界像素和多连通图像的边界像素<sup>[22]</sup>。

当检测轮廓像素  $p$  的时候，通常依照  $p$  周围像素的结构来判断删除或保留它。为了避免在一次迭代中连续的删除一整个分支，一个串行算法通常标记那些将被删除的像素，并且在迭代的最后将所有被标记的像素删除，这保证了在每个子循环中只有一个像素层被移除。我们设定像素  $p$  如果满足下列所有条件的话将被删除：

- H1:  $p$  是一个黑色的像素。
- H2: 不是一个孤立点或端点。
- H3:  $p$  是一个轮廓像素，也就是说， $p$  至少有一个白色 4-邻域像素。

使用叉数  $X_H(p)$  的 Hilditch 算法是串行细化算法的起源，图像从左到右、从上到下被扫描，并且在四个附加条件下，给要被删除的像素做标记，条件如下：

- I1:  $p$  的邻域中至少有一个黑色像素是未被标记的。
- I2: 在循环开始时  $X_H(p)=1$ 。
- I3: 如果  $p_3$  是被标记的，在不改变  $X_H(p)$  的情况下将  $p_3$  设为 0。
- I4: 与 I3 相同，用  $p_3$  替换  $p_3$ 。

条件 I1 被设计用来防止对小圆圈形状图像的过度腐蚀，用 I2 来保持连通性，用 I3

和 I4 来保存两个像素宽的线条，有些学者也将这个方法扩展应用到对有灰度级染色体图像的细化。

其他的研究者使用各种各样的方法在二值图像中应用这个算法，文献[25]将文献[45]中的检测像素的不同标准用在轮廓点或边界点的判断上。例如，在  $N(p)$  的第一列中最多有一个黑色像素，并且在  $N(p)$  剩下的像素中至少有三个黑色像素，那么  $p$  就是西面的边界点。既然这样，必须添加一个条件，它可以防止一个内部像素被标记为边界点。在光栅扫描之后，边界点被标记，在这些被标记的点与 6 个  $3 \times 3$  窗口进行对比后（这些模板如图 3.4 所示，包括这些图的 90 度和 180 度旋转，在每一组中被标记为  $x$  或  $y$  的像素最少有一个是不为 0 的），如果它们不是断开点就将其删除，也就是说，这些被标记点的移除不会造成图像的断开。人们发现，端点的消除会产生较少的伪信息，这个过程会一直重复下去，直到没有像素点再被删除或者达到了最高迭代次数，在这之后，一个单独的清除扫描会把所有的不是端点、断点、和内部点的所有黑色像素点都删除。

x	x	x
0	1	0
y	y	y

x	x	x
0	1	x
1	0	x

图 3.4 断点矩阵

上述的算法假定了一个情况是图像在处理前是光滑的，文献[27]中对这个算法有扩展，在每次迭代前先进行去噪处理。在每次迭代中，按照文献[25]中的条件，在保留端点的情况下，满足一般条件的轮廓像素被标记，然后进行像素检测并删除。这个过程中，在确保了骨架线更接近原始图像中心线的同时，也保证了骨架的连通性。

Rutovitz 算法中的叉数  $X_r(p)$  的定义被作为文献[23]、[46]和[30]中的像素点删除条件。在这些算法中，对于轮廓像素定义有一点不同，一个轮廓像素是其 8-邻域内至少有一个白色像素的黑色像素，这个条件和  $X_r(p)$  的使用都要增加一个附加条件 ( $F = p_1 p_3 p_5 p_7 = 0$ )，这样才可以保证当轮廓像素被删除时不会引起孔洞的出现。在保持连通性的基础上删除  $p$  点的完整条件在文献[23]中有详细介绍，简而言之，条件如下：

J1: 如果  $X_r(p)=0$  或者 8,  $p$  不可以被删除；

J2: 如果  $X_r(p)=2$ ,  $F=0$  并且  $p$  不是端点，那么  $p$  是可以被删除的；

J3: 如果  $X_r(p)=4$ ,  $F=0$ , 且  $p$  像素的四个边角像素都是 0, 两个边上都有 1 像素, 那么像素  $p$  是可删除的；

J4: 如果  $X_R(p)=6$ , 且  $p$  像素的四邻域中有一个像素是 0 而其它三个是 1,  $p$  是可删除的;

然而上述条件的使用会出现伪端点和角点腐蚀。所以, 在文献[23]中提出了解决办法, 这个办法可以根据  $N(p)$  中轮廓像素的结构来判断像素  $p$  的删除。当  $N(p)$  的点被顺序遍历时, 叉数  $CN(p)$  代表着从轮廓像素到非轮廓像素的转变数量(反之亦然), 基于  $X_R(p)$ 、 $CN(p)$  和  $p$  的 8-邻域轮廓像素的数量, 这个步骤被推导出来并用于  $p$  的删除。

细化算法中在保护轮廓突起的方法上 Arcelli<sup>[30]</sup>和 Sanniti di Baja<sup>[46]</sup>做了很大的改进, 在这些文献中“突起”这个概念第一次被提出。这些重要的突起被定义成连通轮廓的子集, 当  $X_R(p)=2$  的情况下根据文献[47]的条件, 其它的轮廓像素被连续的移除, 这些重要突起的像素被保留下来。当得到一个保留像素的集合  $S_f$  时, 那么  $S_f$  中每个剩余的像素  $p$  都被赋予一个运算  $e(p)=2(p_5+p_3)+p_1+p_7+1$ 。在上述运算下的非最大值像素被移除, 生成了一个最多只有两个像素宽的骨架, 然后使用步骤 J4 来消除 4-连通性, 这个操作有利于形成 8-连通的骨架。

在这种方法下得出的骨架有不足的地方, 那就是在分叉末端的一端会有毛刺出现。在文献[48]中, 压缩被应用在图像中黑色像素的每一个  $3*3$  窗口中, 然后将灰度图像转换为二值图像, 细化算法就在二值图像上进行处理, 然后骨架扩大到原比例尺。

当轮廓像素  $p$  被连续跟踪并标记时, 多重像素可以被很简单的确认。如果下面条件中有一个是成立的, 那么像素  $p$  就为多重像素:

K1:  $p$  在跟踪过程中被穿过的次数超过一次 (连通数大于 1);

K2:  $p$  在内部没有邻接像素;

K3:  $p$  有至少有一个属于轮廓的 4-邻域, 但是不在  $p$  之前或之后就立即被跟踪到。

由于 K1 包括连通数大于 1 的像素点, K2 包括端点, K3 包括两个像素宽的线条, 所以多重像素的概念是非常有包容性的。然而, 如果轮廓被重复的跟踪, 只有每次跟踪得到的多重像素被保留下来, 结果可能不是一个连通性完好的骨架。所以, 在文献[28]中, 多重像素被叫作骨架, 这些骨架线也许有点粗, 但是这个算法被证明是正确的, 这个算法可以结束运行, 而且得到了连通性完好的骨架。

在文献[26]中, 多重像素的描述按照局部区域被重新定义, 因此在与一系列的模板进行比较后来决定是用并行还是串行算法。为了多重像素的准确性, 这就要求使用图 3.5 还有它的 90 度旋转矩阵来判断。

x	x	x
0	1	0
y	y	y

x	x	x
0	1	x
1	0	x

x	x	z
0	1	d
y	y	z

图 3.5 多重像素 P 的配置

每一组中的像素被标记成 x 或 y，且每一组中必须有非零元素，在第三个窗口中，被标记为 z 的像素至少有一个非零；如果两个都是非零的，标记为 x 或 y 的像素可以有任何值，被标记为 d 的像素是一个轮廓像素。

通过这些算法可以实现对拓扑结构的保存。然而，它需要更多更全面的信息来保存图形的几何特征。为了达到这个目的，用光栅扫描来获得一幅图像的完整骨架几乎是不可能的，反之，一个基于轮廓像素的连续跟踪算法可能会产生更好的结果。后面的方法允许骨架的形状和图像外部的轮廓有一些联系，图像整体性质的保持仅仅通过局部操作是不够的。当然，这些全面的考虑将增加计算的时间和更多的复杂程序，并且在串行算法中很大一部分复杂的地方是花费很大精力去保护细节特征<sup>[23]</sup>。

### 3.3 并行细化算法

在并行细化算法中，只根据上一次的迭代结果来检查像素是否应该被删除。由于这个原因，这些算法非常适合应用在并行处理器上，在这个处理器上像素满足一组条件就会被移除掉。不幸的是，如果仅仅考虑使用 3\*3 的模板，平行算法在保持图像的连通性上会有困难，举例来说，一个两个像素宽的线条经过这样的细化处理可能会完全消失。因此，通常是使用 3\*3 的邻接区域，但是要把每个迭代划分成子迭代或子循环，在每一个子迭代或子循环中只考虑轮廓像素的子集是否被删除，在每次子迭代的最后，剩下来的图像为下一个子迭代进行数据更新。在文献[35]中算法的每个子循环中，每种类型的轮廓像素（北、东、南和西）都被移除了。这些在文献[38]、[39]和[40]中已经被合并为两个子循环，一个子循环移除北方向和东方向的轮廓像素，另外一个子循环删除掉其余方向的轮廓像素。

快速细化算法是一种利用图像 4-邻域特性对二值化后的线划图像进行细化的算法，该算法的描述如下：

L1: 遍历整个图像，找出图像的边界点；

L2: 判断该边界点是否应该被删除，对边界点 p 参考图 3.1 的邻域点，定义两个特征量：nsum 和 tsum，分别如式 3.3 和式 3.4 所示：

$$nsum = \sum_{i=1}^8 p_i \quad (3.3)$$

$$tsum = \sum_{i=1}^8 |p_{i+1} - p_i| \quad (3.4)$$

其中  $p_9 = p_1$ ，如果  $p$  点同时满足  $tsum=2$ ， $nsum \neq 1$  且  $nsum < 6$ ，则可将其删除；

L3: 继续寻找下一个边界点，直到没有可删除的点为止。

快速细化算法的运行速度很快，但是存在一个严重的缺陷，由于这个算法是 4-连通算法，所以该算法所得出的骨架图像很多不是单像素宽度，即细化不彻底。

在文献[20]中提到的并行算法的基本原理，当像素  $p$  满足下列所有条件的时候将被删除：

M1:  $b(p) \geq 2$

M2:  $X_R(p) = 2$

M3:  $p_1 p_3 p_5 = 0$  or  $X_R(p_3) \neq 2$

M4:  $p_7 p_1 p_3 = 0$  or  $X_R(p_1) \neq 2$

这个算法在文献[35]里也被描述过并加入了条件  $b(p) \leq 6$  来保证  $p$  有一个白色的 4-邻域像素，因此  $p$  的删除不会产生一个孔洞，这是一个有一个子循环的算法，它使用一个  $4 \times 4$  的窗口产生了对轮廓噪声不敏感的连通性骨架，但会导致过度腐蚀。

当  $p$  位于两个像素宽的线条上时，并且在  $X_R(p) = 4$  的情况下允许删除像素  $p$ 。这个方法被证明能保持连通性，因为条件 M3 和 M4 不对称的性质，骨架将不能位于中间，因此在文献[26]中引出了下面对像素  $p$  删除的完整条件：

N1:  $X_R(p) = 0, 2, \text{或} 4$

N2:  $b(p) \neq 1$

N3:  $p_1 p_3 p_5 = 0$

N4:  $p_1 p_3 p_7 = 0$

N5: 如果  $X_R(p) = 4$ ，那么满足条件 a) 或 b)：

a):  $p_1 p_7 = 1, p_2 + p_6 \neq 0$  和  $p_3 + p_4 + p_5 + p_8 = 0$

b):  $p_1 p_3 = 1, p_4 + p_8 \neq 0$  和  $p_2 + p_5 + p_6 + p_7 = 0$

N6-N8 是 N3-N5 的 180 度旋转。

人们建议应该使用两个子循环，第一个子循环删除满足条件 N1-N5 的像素，第二个



子循环依照条件 N1、N2 和 N6-N8 进行删除。叉数  $X_R(p)$  被用做一个细化的操作标准，在文献[48]中，一个满足  $X_R(q) \geq 2$  白色像素  $q$  被认为是一个连接点并且被填充，然后一个有 4 个子循环的算法被应用在端点上并且满足  $X_R(p) > 2$  的  $p$  像素被认为是骨架像素，反之非骨架轮廓像素将被删除。

被大量引用的 Zhang-Suen 方法<sup>[38]</sup>，也曾在文献[47]中总结过，是算法中所有条件的一个子集分别在两个子循环中的实现。在第一个子循环中，如果  $p$  满足下列条件的话将被删除：

$$O1: 2 \leq b(p) \leq 6$$

$$O2: X_R(p) = 2$$

$$O3: p_1 p_3 p_7 = 0$$

$$O4: p_1 p_5 p_7 = 0$$

在第二个子循环中，O3 和 O4 被它们的 180 度旋转所代替。因此，第一个子循环删除了在东南边界和西北角像素，第二个子循环删除完全相反方向的像素。这是一个简单而又有效的算法，并且对边缘噪声有免疫性<sup>[39]</sup>；但是两个像素宽的线条可能被严重的腐蚀，并且  $2 \times 2$  正方形像素块将完全消失。为了防止这种结构被删除，人们认为应该将条件 O1 改为  $3 \leq b(p) \leq 6$ 。可想而知，这样的更改又引起了自身保留无关像素的问题，因此在文献[46]中提出了一个方法来将结果细化成单位宽度像素的骨架，但是增加了处理时间。

尽管对线条的过分腐蚀不是一个拓扑问题，在细化过程中  $2 \times 2$  正方形像素块的完全消失会致使图像的拓扑点算法不正确。在文献[49]中显示了算法失效时候的可能结构，文献[38]中的一个改动意味着所有可能的情况都能被正确的处理，当  $X_R(p) = 2$  时，这个修改决定了像素  $p$  的黑色 4-邻域个数。如果  $p$  只有一个或者没有这样邻域的话，将不需要任何变化，如果  $p$  有两个或者三个这样邻域并且  $p$  满足现有的条件，那么在第一个子迭代中如果  $p_1 = 0$  或  $p_7 = 0$ ，或者在第二个子迭代中  $p_3 = 0$  或  $p_5 = 0$  的话，则  $p$  会被删除。

文献[26]中的条件 N1、N2 和 N8 在稍微有些改动的情况下被应用在文献[24]中。一些算法不会删除孤立的像素，这要求  $2 \leq b(p) \leq 6$  而且它不保持 4-连通性，这导致了条件  $p_2 + p_6 \neq 0$  在 N8(a)和 N8(b)中是多余的。一些学者还应用了一个含有两个子循环的平行算法，这个算法在第一个子循环中删除东南边界的点和西北方向的满足条件  $X_R(p) = 2$

的拐角像素点，也删除在  $X_R(p) = 4$  的条件下满足条件 N8 的轮廓像素点，第二个子循环中相反方向的像素被删除。文献[37]中条件 N1-N8 的两个子迭代被应用在文献[39]中。

文献[50]还建议文献[38]中运用减少骨架单像素化的步骤。因此最初被 Rutovitz 所提议、被 Deutsch 所改正的算法已经被许多文章变换了各种各样的形式来应用，它们之间的关系如图 3.6 所示：

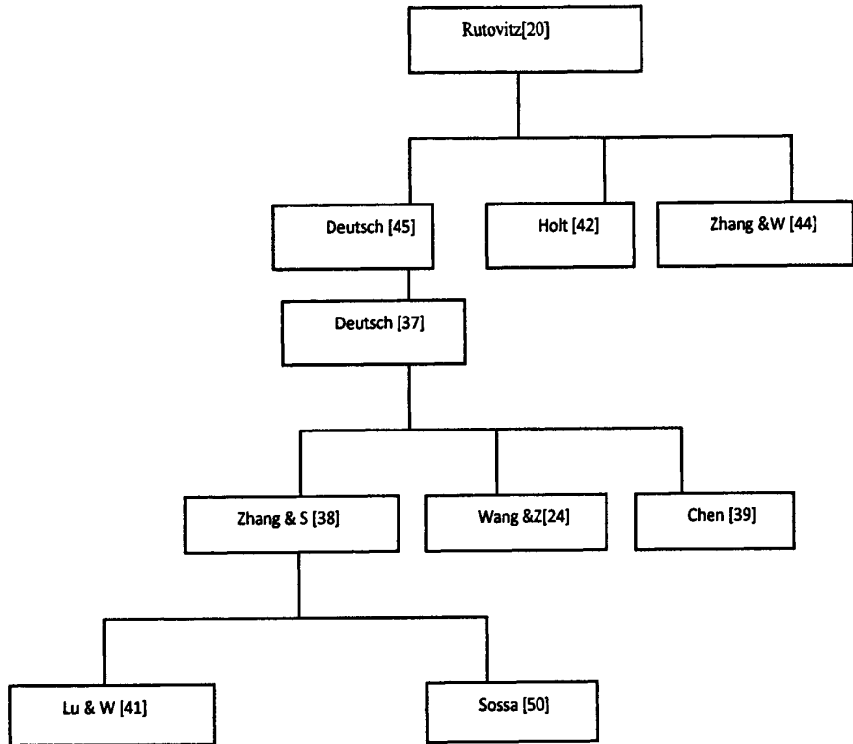


图 3.6 基于 Rutovitz 的细化算法的结构

在并行算法中，用除去一般非端点的轮廓像素操作来保持拓扑关系。为了要克服  $3 \times 3$  窗口局部操作引起的限制，距离变换和细化结合起来的方法开始被使用，距离转换的使用能提供更多的全局性的信息。有学者提出：当距离转换控制着重建能力的时候细化操作能保持连通性，如果要达到这个要求，那么合乎逻辑的做法就是将两种操作结合起来。对每次循环所建议的步骤是，现存图像的像素点的集合  $I$  应该被确定，它的膨胀  $E(I)$  被定义为  $E(I) = \{q | q \in I \text{ 或有一个邻域在 } I \text{ 中}\}$ ，除非它们是局部最大值、非简单像素或非端点，否则轮廓点将会被删除，剩余的骨架像素点被添加到骨架集合中，而且这个过程会在  $I$  中不断地重复。这样的混合算法可能是最有效的，因为距离变换方法一开始在固定

步骤数量中移除掉大部分的轮廓像素，然后在剩下图像像素上应用一个剥皮算法，使最后的骨架为单位像素宽度。

文献[20]和[42]这些只有一个子迭代的算法已经被讨论过了，这些方法都从比  $3 \times 3$  窗口更大的窗口来获取信息以保证连通性。文献[43]和[44]也是这种类型的算法，在文献[44]中，平行算法执行一步操作需要使用 8 个细化模板、2 个恢复模板和 8 个删除模板。这些模板是图 3.7 中的(a)和(a)的 90 度、180 度、270 度旋转，(b)和(b) 的 90 度、180 度、270 度旋转和(c)模板所组成的。

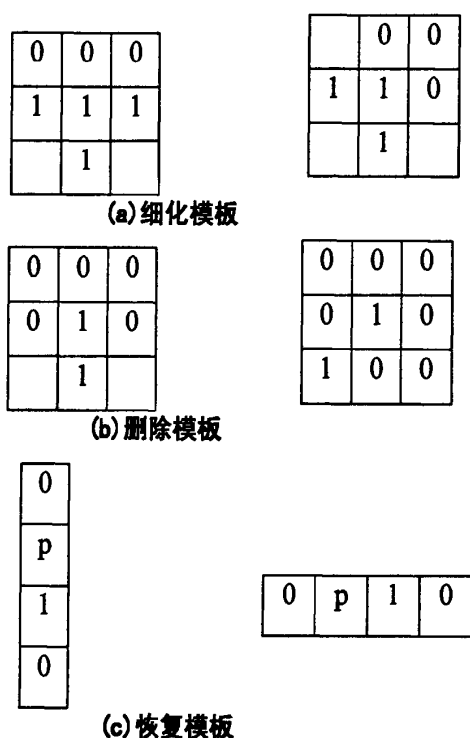


图 3.7 模板种类

这个细化模板把边界点和拐角点删除掉，恢复模板将保存两个像素宽的线条，然而删除模板以缩短分支的代价来换取毛刺噪声的减少。这个算法仅仅把  $3 \times 3$  窗口当做细化和重建模板，在不同方向上仅仅使用了两个循环，在循环后运用一个后期处理来得到一条并不总是单位像素宽的中心线。

在文献[43]中，一个只含有一个子循环的算法被使用，它实际上是使用一个  $5 \times 5$  窗口获取信息，当满足下列条件时像素  $p$  将被删除：

P1:  $X_H(p) = 1$ ,

P2: (a)在 $N(p)$ 中的黑色像素是4-连通的,或者

(b)p 是属于一条两个像素宽线条的。

P2(b)的条件已经在文献[39]中进行了改进,如果  $p$  满足删除条件,那它的 4-邻域也需要被检查判断是否被删除。如果这些像素被删除后连通性没有改变,那么  $p$  是可以被移除的,这些信息被存在查询表中作为查询步骤。除此之外,在算法中加入了一个特殊设计的细化条件,这个算法通过从凹的拐角点处移除掉更多的像素来更好的保持“L”形状的图像,这个条件就是:如果  $p$  的 4 个对角邻域中包含一个白色像素,同时其它三个像素都有黑色的 8-邻域像素,则  $p$  将被删除。在文献[20]中,根据细化条件和保持连通性条件对文献[42]、[43]和[44]进行了比较,文献[43]中的方法是更为优秀的,因为它拥有更加全面的细化条件,并且在保持连通性上花费更少的时间。

对于并行算法来说,近年来大部分的注意力都放在处理速度上,这已经导致了很多在计算时间上的比较或者在迭代和子迭代使用次数上的比较(如文献[40]和[41])。然而一些在迭代次数上的比较并不是完全有效的,因为只有一个步骤的算法,经常用一个比  $3 \times 3$  大的窗口来进行处理,这个造成了对计算中间结果的需要或者关于对邻域像素边界信息的需要<sup>[42]</sup>。基于这个原因,一个一般的骨架需要被定义一次循环,这样的话这些并行算法就可以进行有效的比较,文献[50]表明计算机中的每一个像素都可以计算任何一个  $3 \times 3$  邻域的逻辑函数,并且算法的并行处理速度是以使用这种迭代的次数来衡量的。

总的来说,与串行算法截然相反,并行算法大部分的复杂性来源于当在小的局部邻域中使用并行操作时需要保持连通性,这对于并行细化算法的特点来说尤其是个问题,并且已经通过使用多次子迭代或扩大检查邻域范围来解决。不论发生任何情况,当中间结果已经被准确的计算出来时,应该用一些形式来保持更多的全局结构信息。

### 3.4 非迭代细化方法

在前面的部分中,讨论了通过检验和删除轮廓像素而生成骨架的算法。在这个部分中,要讨论的算法是不基于像素的,它们直接采用一个步骤的方式产生图像的中值线或中心线而不用检查所有的单个像素,因为使用中轴或距离转换来完成骨架化的算法在以前的文章中讨论过,我们主要关心用线条跟踪或运用长度编码的方法来判断中心线,文献[51]中已经讨论了在操作步骤中运用这种算法保留全局特征和连通性是可能的。

这些算法决定了线条中黑色像素的中点并把它们连接成一副骨架。这些方法拥有在计算效率上的优势,但是它们也有自然缺陷,当线条与扫描线近乎平行的时候将会产生噪声分支。这种算法在一些专门的应用中非常有效,这些应用中扫描线近似的垂直于线条。例如在文献[52]中,一个有用的中心线可以通过保持等速扫描方向垂直于这些线条的方法来得到。在其它应用中,扫描方向是 90 度或 45 度轮流可变的。文献[53]中的扫

描沿着  $x$  轴与  $y$  轴两个方向进行并且长度小一些的线条将会被挑选出来。

一些算法包含通过连接有一定方向的线条形成近似的骨架, 例如在文献[54]中 8 个窗口操作被用于 4 个子循环中, 它用来检测和确定图像中的水平、垂直、左对角线和右对角线分支的存在, 同时操作符通过端点的集合也定位拐角点, 这些提取出来的点被连接成一条近似骨架的线段。在文献[55]和[56]中, 通过以上四种局部定位, 边界像素第一次被标记, 对于每一个边界像素, 按照垂直于被标记的像素的方向, 对边界上对边的被标记的相同种类的像素做研究, 然后这些对边的中点被连接起来形成一个骨架。这些方法不适用于一般的应用, 因为这些方法不健壮, 尤其是对线条方向比较多并且线条比较厚(像素多)的情况。

文献[51]中, 目标中的线条被动态矩形窗口跟踪, 这个窗口可以根据线条的宽度自动收缩, 骨架就是连接连续窗口中心点的单位像素宽度的线条。轮廓像素首先被串行检测, 如果  $p$  满足下列情况将被删除:

Q1:  $N(p)$ 至少有一个内部像素,

Q2:  $N(p)$ 至多有三个轮廓像素,

Q3: 在  $N(p) \cup \{p\}$  中轮廓像素被连续的跟踪。

当没有可删除像素的时候, 剩下的图像将被一个  $2 \times 2$  的正方形窗口跟踪, 骨架就是这些方形窗口的轨迹, 必要的时候还可以重复地跟踪。

采用一个不同的方法, 骨架也可以从傅里叶描述符中得到, 如同文献[57]中提到的, 至少对于没有闭合的图像或重叠的图像或有固定宽度的图像是这样, 对于这样的图像来说, 通过跟踪轮廓来获得一条封闭的曲线, 其中傅里叶描述符可以被提取出来, 骨架的傅里叶描述符随之会被确定, 而且骨架可以用简谐波的有限集来构建, 然而复杂的数学方法仅仅可以被用在获取理想图像的骨架上。

### 3.5 本章小结

本章首先对细化方法进行了分类, 然后介绍了迭代细化方法和非迭代细化方法的概念, 然后将迭代细化算法又分为串行细化算法和并行细化算法两类进行论述并对每种算法的优缺点进行了分析, 最后介绍了非迭代细化方法。

## 第四章 改进的 Zhang-Suen 细化算法

### 4.1 Zhang-Suen 算法简介<sup>[38]</sup>

二值图像被定义为一个矩阵  $IT$ ，这个矩阵中的每个像素  $IT(i,j)$  的值为 0 或 1。图像是由像素值为 1 的像素所组成的，图像中的每个线条都是大于或等于一个像素宽的。通过分析矩阵  $IT$  中像素的相邻像素值来对矩阵进行重复的变形。我们假定点  $(i,j)$  周围的像素是  $(i-1,j), (i-1,j+1), (i,j+1), (i+1,j+1), (i+1,j), (i+1,j-1), (i,j-1)$  和  $(i-1,j-1)$ ，如图 4.1 所示：

$P_9(i-1, j-1)$	$P_2(i-1, j)$	$P_3(i-1, j+1)$
$P_8(i, j-1)$	$P_1(i, j)$	$P_4(i, j+1)$
$P_7(i+1, j-1)$	$P_6(i+1, j)$	$P_5(i+1, j+1)$

图 4.1 3\*3 窗口中 9 个像素的指定

在处理图像的时候，第  $n$  次循环中一个点像素值由第  $n-1$  次循环结果后它自己的像素值和这个像素周围八个像素的像素值确定，因此所有图像上的点可以同时被处理。假定使用一个 3\*3 的窗口，每个像素都与自身周围的八个像素联系起来，这个算法只需要进行简单的计算即可。

提取一个图像骨架的方法是将图像的所有轮廓像素移除，只留下属于骨架的像素。为了保持骨架的连通性，我们将每次循环分为两个步骤：在第一个步骤中，轮廓像素  $P_1$  如果满足以下条件将会被删除：

$$R1: 2 \leq B(p_1) \leq 6$$

$$R2: A(p_1) = 1$$

$$R3: p_2 * p_4 * p_6 = 0$$

$$R4: p_4 * p_6 * p_8 = 0$$

$A(P_1)$  是像素  $P_1$  周围八个像素中按  $p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9$  的顺序中 0-1 的数量。

$B(p_1)$  是像素  $P_1$  周围黑色像素的个数：

$$B(p_1) = p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 + p_9$$

如果有任何一个条件不满足，那么  $A(P_1)=2$ ，所以像素  $P_1$  不会被删除。

在第二个步骤中，仅仅 R3 和 R4 的情况发生改变：

$$R3': p_2 * p_4 * p_8 = 0$$

$$R4': p_2 * p_6 * p_8 = 0$$

其它情况都与第一步是一样的。

在第一个步骤中的 R3 和 R4 条件仅仅移除了图像中不属于骨架的东南边界的像素点和西北角的像素点。满足方程 R3 和 R4 的解是  $p_4=0$  或者  $p_6=0$  或者  $p_2=0$ ,  $p_8=0$ 。所以已经被删除的像素  $P_1$  可能是东南边界的像素点或者是西北角的像素点。类似的, 第二个步骤中被删除的像素点可能是西北边界或者东南角的像素点。设立条件 R1 的目的是保护骨架线的端点, 设立条件 R2 的目的是防止删除骨架线上端点之间的点, 这个迭代会一直持续下去直到没有要移除的点为止。这个算法在连通性和噪声免疫上有非常好的效果, 并且删除像素点的算法非常简单。

4.2 细化前锐角问题处理

图像在细化前的预处理中, 一般要进行孔洞移除, 删除孤立噪声点和毛刺噪声等操作, 以提高细化结果的质量。在实际细化过程中, 有的图像含有大量锐角, 而在锐角处的像素比较多的情况下, 细化结果容易产生信息冗余, 即将锐角处看做分支, 针对这种情况, 我们在预处理步骤中采取措施, 以使锐角的细化结果达到令人满意的标准。

4.2.1 锐角处理中模板的选择

在预处理过程中, 通过扫描图像检测出分支间的各个方向的锐角, 然后根据模板进行像素删除操作, 我们先考虑一种 3\*3 大小的向下的锐角细化模板, 如图 4.2 所示:

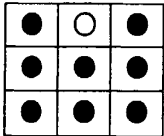


图 4.2 3\*3 锐角细化模板

遍历图像中的每一个像素, 当遇到与模板匹配的像素的时候, 模板的中心像素被删除。此种模板的优点是锐角处的多余像素去除彻底, 缺点是一些与锐角无关的轮廓像素也被删除了, 这是由二值图像的特点决定的, 如图 4.3, 一些轮廓像素点也满足 3\*3 模板的删除条件而被删除了, 它破坏了图像的几何特征, 所以不能选择此类模板作为删除锐角像素的模板。

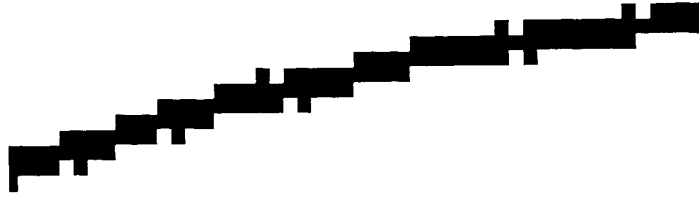


图 4.3 实际线划图像的像素特征

接下来考虑 7\*7 模板，如图 4.4 所示，

●	●	●	○	●	●	●
●	●	●	○	●	●	●
●	●	●	○	●	●	●
●	●	●	●	●	●	●
●	●	●	●	●	●	●
×	●	●	●	●	●	×
×	×	●	●	●	×	×

A1

●	●	○	○	●	●	●
●	●	●	○	●	●	●
●	●	●	○	●	●	●
●	●	●	●	●	●	●
●	●	●	●	●	●	●
×	●	●	●	●	●	×
×	×	●	●	●	×	×

A2

●	●	●	○	○	●	●
●	●	●	○	○	●	●
●	●	●	○	○	●	●
●	●	●	●	●	●	●
●	●	●	●	●	●	●
×	●	●	●	●	●	×
×	×	●	●	●	×	×

A3

●	●	○	○	●	●	●
●	●	○	○	●	●	●
●	●	●	○	●	●	●
●	●	●	●	●	●	●
●	●	●	●	●	●	●
×	●	●	●	●	●	×
×	×	●	●	●	×	×

A4

●	●	●	○	○	●	●
●	●	●	○	○	●	●
●	●	●	○	○	●	●
●	●	●	●	●	●	●
●	●	●	●	●	●	●
×	●	●	●	●	●	×
×	×	●	●	●	×	×

A5

●	●	○	○	●	●	●
●	●	○	○	●	●	●
●	●	○	○	●	●	●
●	●	●	●	●	●	●
●	●	●	●	●	●	●
×	●	●	●	●	●	×
×	×	●	●	●	×	×

A6



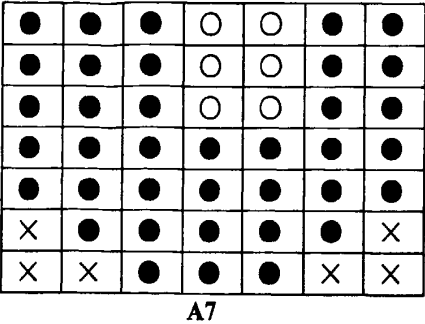


图 4.4 7\*7 锐角细化模板 Ai

遍历图像中的每一个像素，当遇到与模板匹配的像素的时候，模板的中心像素被删除。此种模板的缺点是锐角中一部分多余的像素没有被删除即锐角的多余像素删除不彻底，这导致了经过预处理的锐角在进行细化后仍然会产生冗余信息，所以不能选择此类的模板作为删除锐角像素的模板。同理 9\*9 模板也是如此，也不适合作为像素删除模板。

最后考虑 5\*5 模板来删除锐角中的像素，如图 4.5 所示：

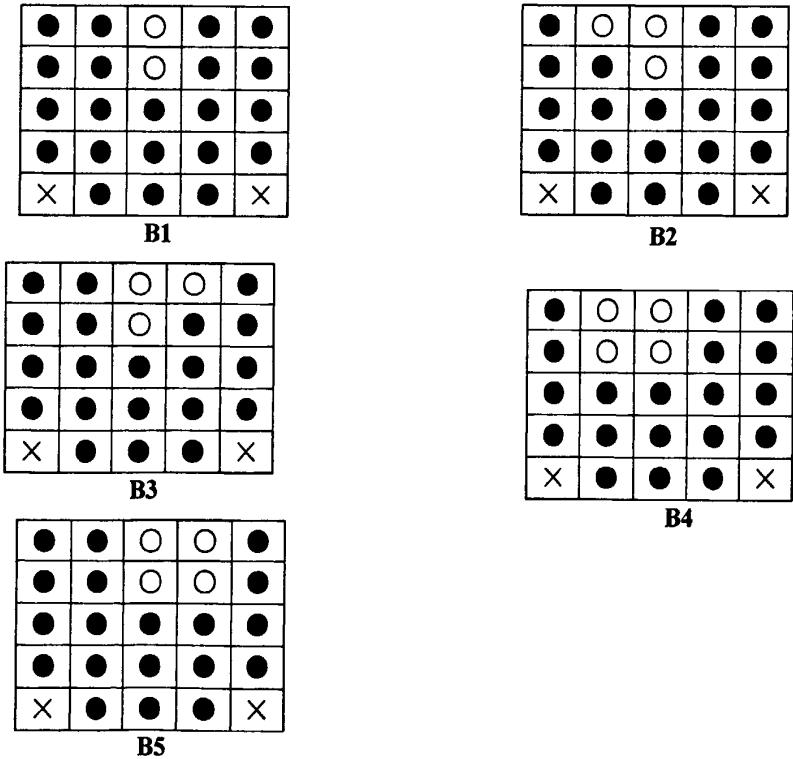


图 4.5 5\*5 锐角细化模板 Bi

经试验证明，5\*5 模板既可以在预处理中最大限度的删除锐角中多余的像素，又能避免删除轮廓像素，在删除锐角像素的同时，又保证了原始图像的几何特征，如图 4.6 所示，锐角内侧的像素被很好的移除了，轮廓像素被完整的保存下来，所以本文中采用

5\*5 模板删除锐角内侧的像素。

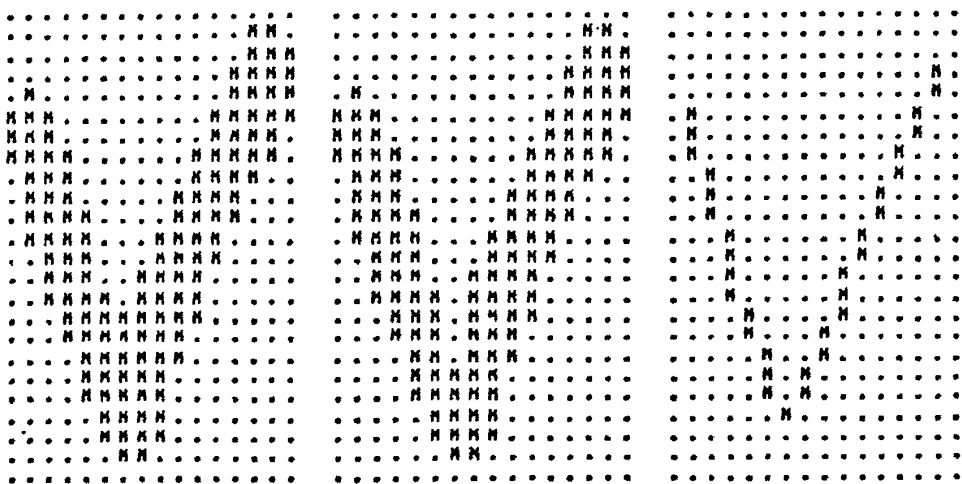


图 4.6 5\*5 细化模板在预处理中删除锐角像素情况

4.2.2 锐角处理改进的具体方法<sup>[18]</sup>

在预处理过程中，通过扫描图形检测出分支间各个方向的锐角，接下来先讨论上下方向锐角的删除模板。 $C_i$  是五种向下的尖角模板（如图 4.7），将  $C_i$  倒置成  $D_i$ ，是五种向上的尖角模板，在检测到合适的模板后，黑色中心像素将被删除，这个过程将多重复两次，第二次和第三次循环仅在这样的条件下执行：首先第一次循环涉及到了删除像素，然后仅当  $C_i$  和  $D_i$  ( $i=1,2,3$ ) 在第二次扫描中被扫描到，最后仅当  $C_1$  和  $D_1$  在第三次扫描中被扫描到，直到将这些模板都扫描完，这个方法通过增加锐角内部边界的长度来弥补不均匀的细化。

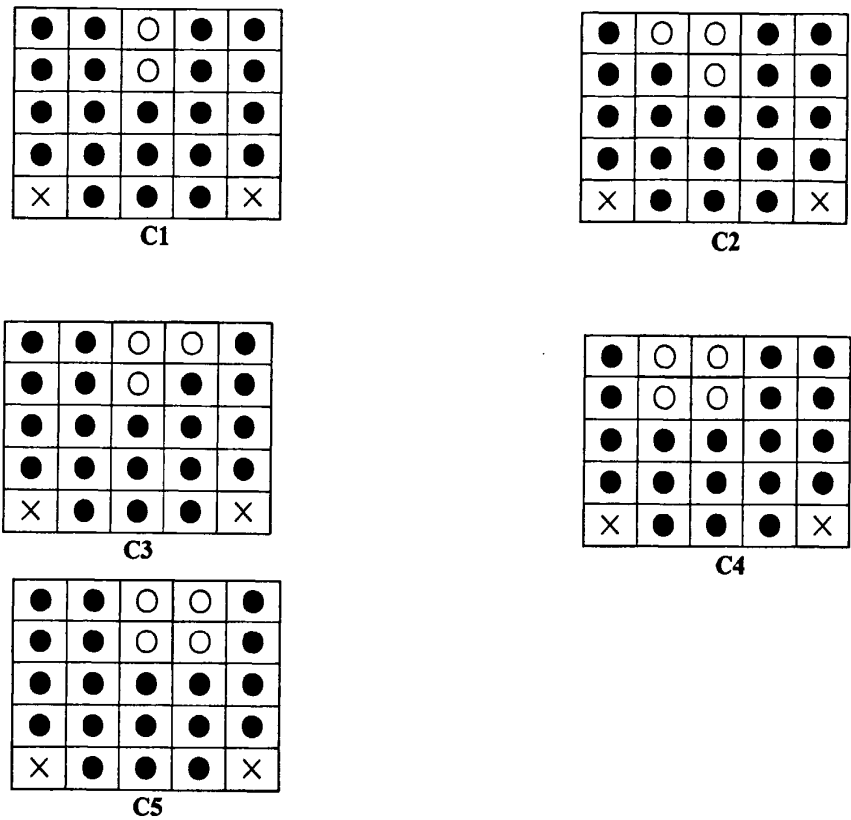


图 4.7 向下角度检测模板  $C_i$

接下来讨论左右方向锐角的删除模板， $E_i$  是五种向右的锐角删除模板（如图 4.8），将  $E_i$  旋转 180 度成  $F_i$ ，它是五种向左的锐角删除模板，在检测到合适的模板后，黑色中心像素将被删除，这个过程也将多重复两次，第二次和第三次循环在这样的条件下执行：首先第一次循环涉及到了删除像素，然后仅当  $E_i$  和  $F_i$  ( $i=1,2,3$ ) 在第二次扫描中被扫描到，最后仅当  $E_1$  和  $F_1$  在第三次扫描中被扫描到，直到将这些模板都扫描完。

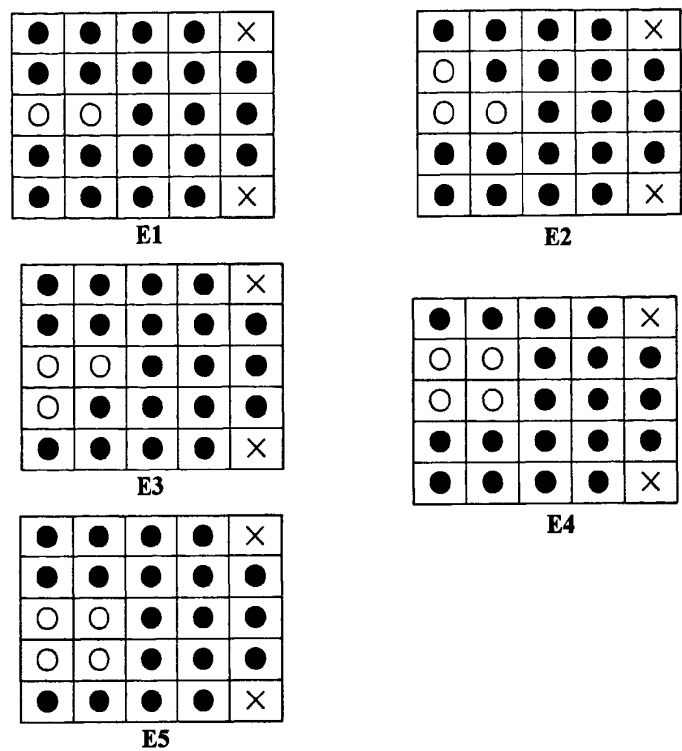
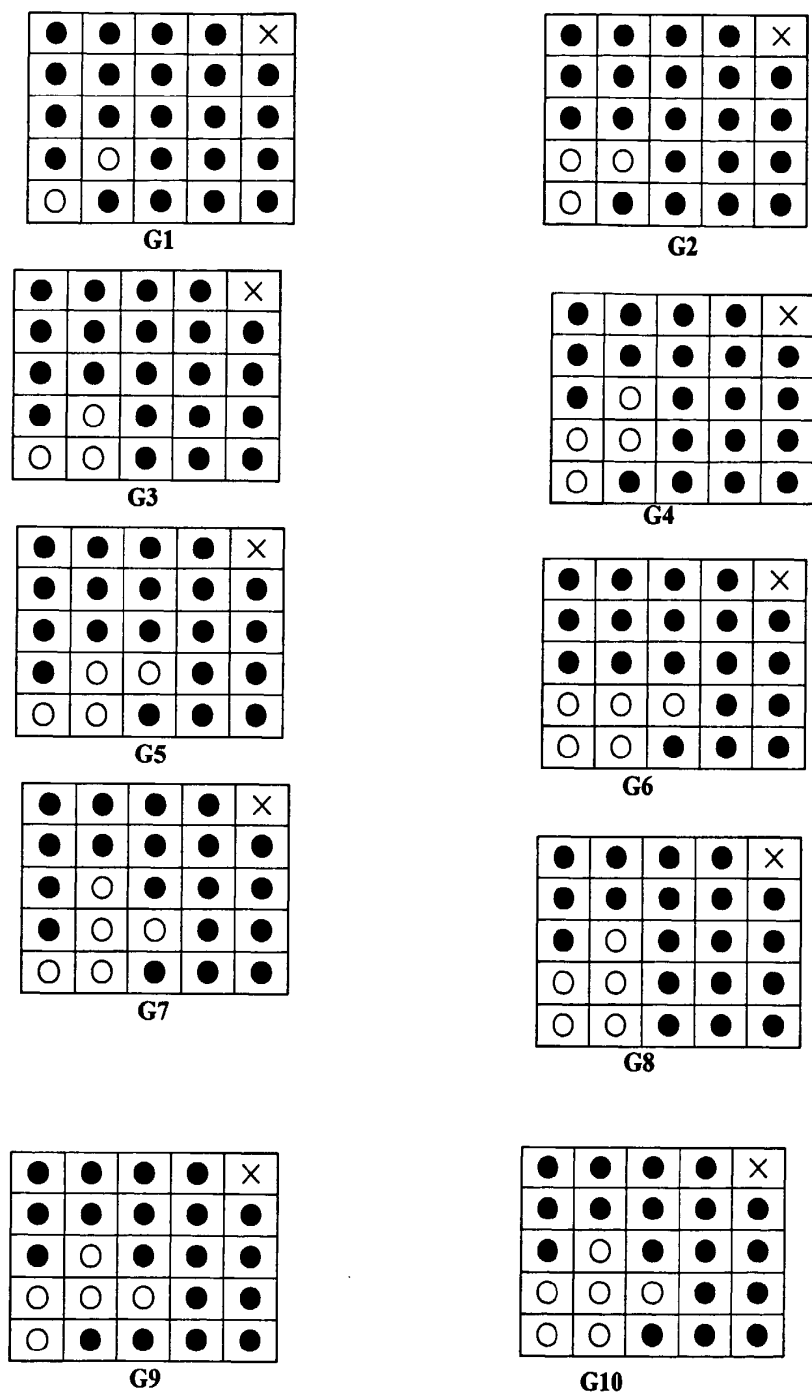


图 4.8 向右角度检测模板  $E_i$

然后讨论的是右上方向锐角的删除模板， $G_i$  是十种方向为右上的锐角删除模板（如图 4.9），将  $G_i$  旋转 180 度成  $H_i$ ，它是十种方向为左下的锐角删除模板，在检测到合适的模板后，黑色中心像素将被删除，这个过程将多重复三次，后三次循环在这样的条件下执行：第一次循环涉及到了删除像素，仅当  $G_i$  和  $H_i$  ( $i=1,2,3,4,5$ ) 在第二次扫描中被扫描到，仅当  $G_i$  和  $H_i$  ( $i=1,2,3$ ) 在第三次扫描中被扫描到，仅当  $G_1$  和  $H_1$  在第四次扫描中被扫描到，直到将这些模板都被扫描完。

图 4.9 右上方向角度检测模板  $G_i$ 

最后讨论的是右下方向锐角的删除模板， $M_i$  是十种右下方向的锐角删除模板（如图 4.10），将  $M_i$  旋转 180 度成  $N_i$ ，它是十种左上方向的锐角删除模板，在检测到合适的模板后，黑色中心像素将被删除，这个过程也将多重复三次，后三次循环仅在这样的条件下执行：第一次循环涉及到了删除像素，仅当  $M_i$  和  $N_i$  ( $i=1,2,3,4,5$ ) 在第二次扫描中被扫描到，仅当  $M_i$  和  $N_i$  ( $i=1,2,3$ ) 在第三次扫描中被扫描到，仅当  $M_1$  和  $N_1$  在第

四次扫描中被扫描到，直到这些模板都被扫描完。

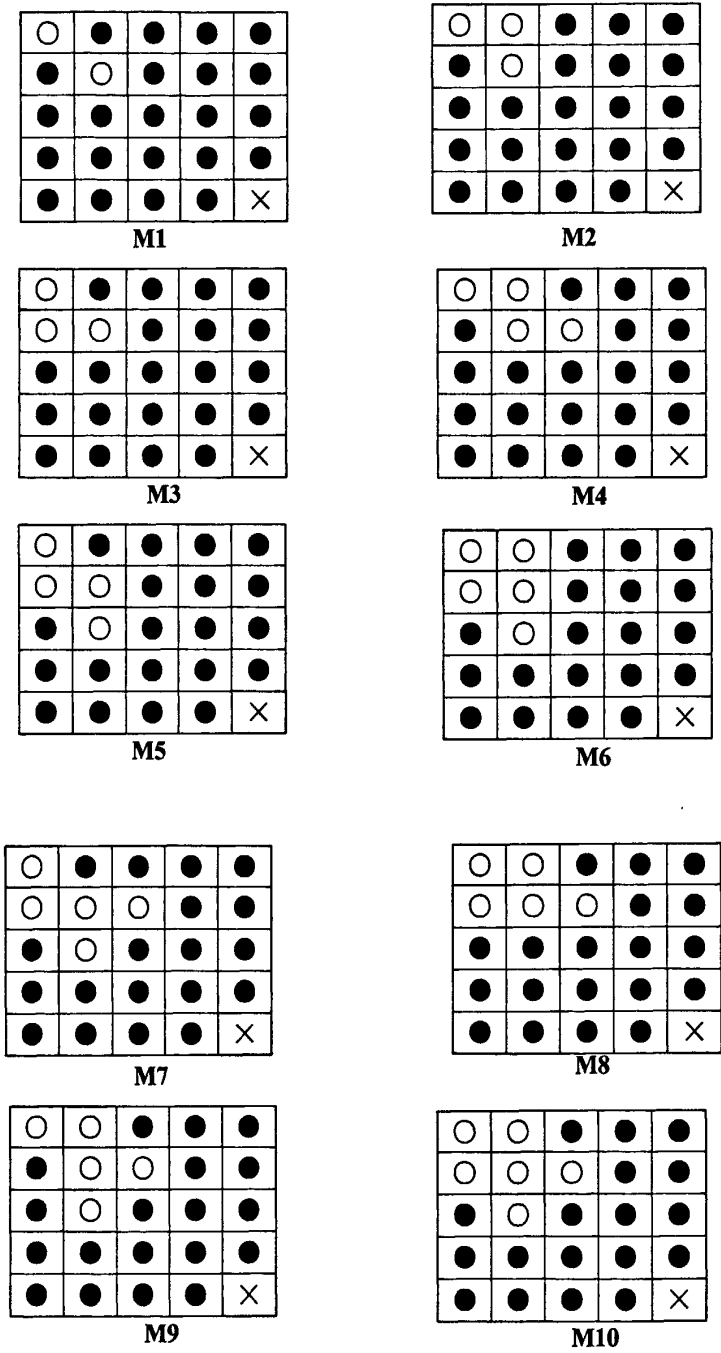


图 4.10 右下方向角度检测模板  $M_i$

在进行锐角像素删除的时候，遍历图像中的每一个像素，并与所有方向的像素删除模板进行匹配，找到相匹配的模板后再进行像素删除操作。

4.2.3 实验结果分析

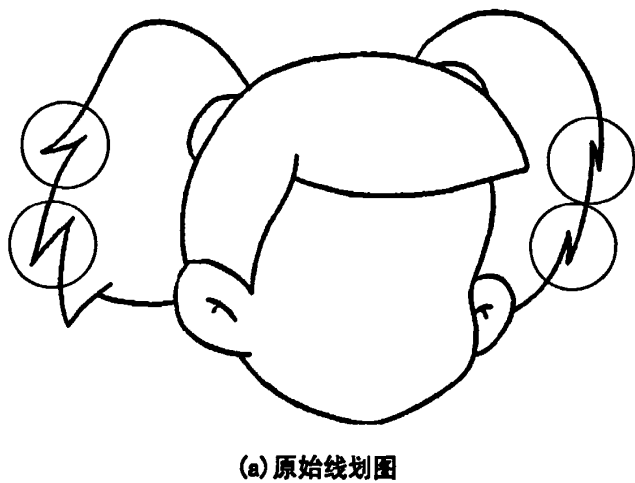
由图 4.11 三幅图对比可知，细化锐角如(a)的情况，经常会造成细化不充分的情况

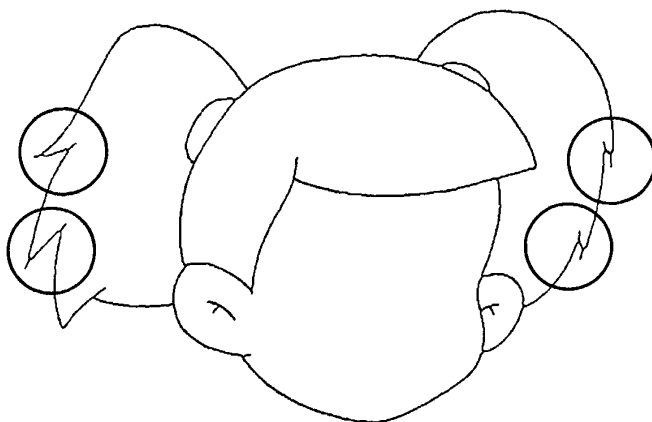
出现，导致信息冗余，使得细化图像失真如(b)，这是由于使用 Zhang-Suen 算法，导致了锐角内角的像素剥离速度小于外角像素的剥离速度。经过在预处理中提前对锐角像素的删除，使锐角细化达到很好的效果如(c)：



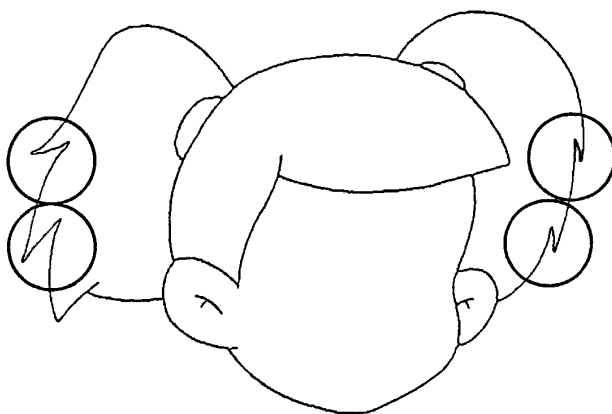
图 4.11 线划图像中局部锐角细化

图 4.12 为实际线划图像的细化效果（改进的地方用圆圈标注），通过对比可知，在实际图像中锐角细化所产生的冗余信息问题被很好的解决了。





(b) 经 Zhang-Suen 算法处理后的细化效果







(c) 改进后的细化效果

图 4.12 含有锐角的线划图像整体细化效果

### 4.3 细化后笔划趋势的改进

#### 4.3.1 改进的笔划趋势分析方法<sup>[58]</sup>

设待细化的二值图像有 $m$ 行 $n$ 列像素点,则整个二值图像可用一个 $m \times n$ 数组表示,前景点标记为1,背景点标记为0,第 $j$ 行 $k$ 列的值的表达式为,先设的值为。由图4.13可以看出,Zhang-Suen算法在决定当前像素点是否保留时,仅考虑了当前点的8-邻域,而没有考虑当前点所在的笔划走向,因此当笔划上有噪声点时,或在笔划的交叉或分叉处就会产生笔划的畸变,特别是在多笔划交叉处畸变就更加严重,如图4.14(b)为图4.14(a)细化后的畸变结果。此种算法如果用于模式识别中,则很可能会导致误识。



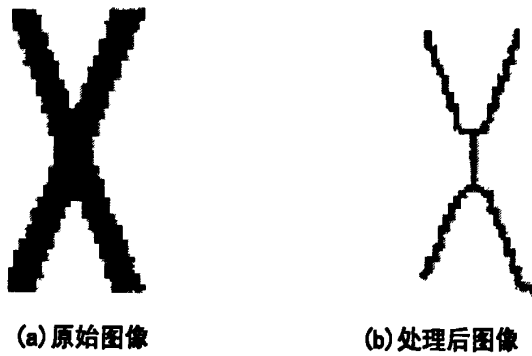


图 4.13 Zhang-Suen 算法的处理结果

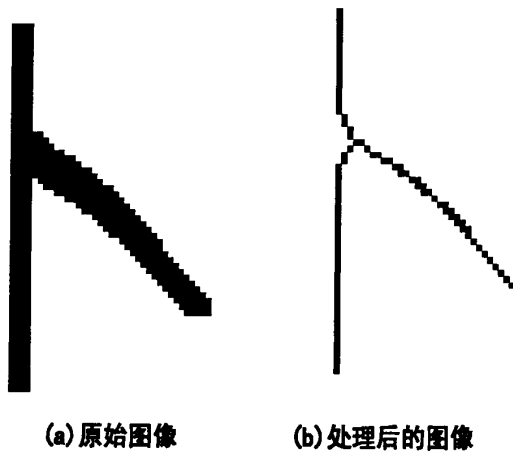


图 4.14 Zhang-Suen 算法细化后的扭曲变形示例

为尽可能避免上述畸变及尽可能减少笔划边缘的噪声点对骨架的影响，需在细化处理时对当前像素点所在笔划走向进行分析并预测，优先保留笔划走向上的点。

本算法即在Zhang-Suen算法的基础上加进笔划走向趋势分析及预测规则，根据当前像素点所在笔划走向分析和预测的结果，决定当前处理点是保留还是删除。符合Zhang-Suen 算法删除条件且不在该笔的笔划走向上的点将被删除，否则将被保留。

判定及预测笔划走向的具体方法如下：

在已用Zhang-Suen算法标记为删除的二值图像中，重新扫描当前点的8-邻域一圈，计算从0 →1 变化次数 $S(P)$ ，由于是从左至右，从上至下扫描，如 $S(P)>1$ 则说明当前点的左上方的点可能已经细化到不大于两个像素宽度的二值图像，此时方可进行笔划走向判定。若 $S(P)>1$ 则分别根据其左、左上、上、右上4 点的8-邻域判断当前所在笔划走向：

S1:若左边点 ( $p_5$ ) 值为1，即=1,然后判断是否满足以下的条件：

- 1) =1;
- 2) 绕其8-邻域扫描一圈从0→1变化次数  $S(p_5)>1$ ;

3) 8-邻域中所有符合 $1 \rightarrow 0 \rightarrow 1$ 变化的两个1之间0的个数不小于2。

若同时满足以上条件，则当前点左边笔划走向为“一”，且其恰在该笔划的延伸方向上，因此保留此点不删除。

S2:若左上点 $p_4$ 值为1，即 $\text{img}[j-1,k-1]=1$ ，再判断是否满足以下条件：

1)  $\text{img}[j-2,k-2]=1$ ;

2) 绕其8-邻域扫描1圈从 $0 \rightarrow 1$ 变化次数 $S(p_4) > 1$ ;

3) 8-邻域中所有符合 $1 \rightarrow 0 \rightarrow 1$ 变化的两个1之间0的个数不小于2。

若同时满足以上条件，则当前点左边笔划走向为“\”，且其恰在该笔划的延伸方向上，因此保留此点不删除。

S3:若上边点 $p_3$ 值为1，即 $\text{img}[j-1,k]=1$ ，再判断其是否满足以下条件：

1)  $\text{img}[j-2,k]=1$ ;

2) 绕其8-邻域扫描1圈从 $0 \rightarrow 1$ 变化次数 $S(p_3) > 1$ ;

3) 8-邻域中所有符合 $1 \rightarrow 0 \rightarrow 1$ 变化的两个1之间0的个数不小于2。

若同时满足以上条件，则当前点左边笔划走向为“|”，且其恰在该笔划的延伸方向上，因此保留此点不删除。

S4:若右上点 $p_2$ 值为1，即 $\text{img}[j-1,k+1]=1$ ，再判断其是否满足以下条件

1)  $\text{img}[j-2,k+2]=1$ ;

2) 绕其8-邻域扫描1圈从 $0 \rightarrow 1$ 变化次数 $S(p_2) > 1$ ;

3) 8-邻域中所有符合 $1 \rightarrow 0 \rightarrow 1$ 变化的两个1之间0的个数不小于2。

若同时满足以上条件，则当前点左边笔划走向为“/”，且其恰在该笔划的延伸方向上，因此保留此点不删除。

上述4步中每一步处理一种笔划走向，即“横”、“竖”、“撇”、“捺”，处于这四种笔划走向上的点被保留，同时删除其8-邻域内不影响连通性的且不处于笔划走向上的点。如果同时有两个或两个以上满足以上条件，表明当前点处于笔划拐角处或交叉处。注意，由于当前待处理点8-邻域内右下角的点已经在上次迭代中处理过，而到本次迭代时右下角的点并没有变化，因此上述算法没有考虑8-邻域内右下角方向点上的笔划。

#### 4.3.2 试验结果分析

图 4.15 是图 4.14 的原始图像经过笔划趋势分析处理后的细化效果图，可以看出，交叉点处的细化效果明显改善（改进的地方用圆圈标注）。

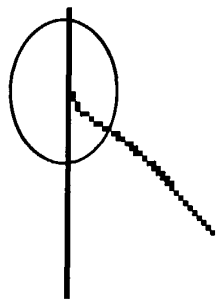
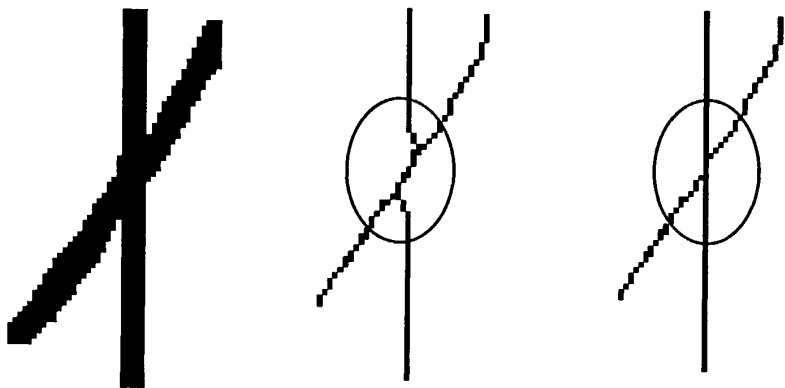


图 4.15 基于笔划趋势分析的三节点细化效果图

经过上述步骤处理后,能够很好地优先保留“横”、“竖”、“撇”、“捺”笔划走向上的点,从而减少畸变的发生,由于其思想是力求保持笔划不偏离原来的方向,所以本算法同时也具有较好的抗噪声能力。图4.16为实际线划图的细化效果(改进的地方用圆圈标注),由细化结果对比可知,通过使用基于笔划趋势预测的细化算法,使得交叉点的细化质量明显提高,能获得比较严格8-连通的且各向对称的中心骨架,并能有效地抑制笔划交叉处的畸变,具有很好的细化效果和较快的处理速度。



(a)原始图像      (b)Zhang-Suen 算法处理过的图像      (c)基于笔划趋势分析的四节点细化效果图

图 4.16 改进后的算法与原始 Zhang-Suen 算法细化结果效果图

4.4 细化后“肿块”问题的处理

4.4.1 问题的提出

图像在经过 Zhang-Suen 细化算法处理后,有时会出现一种特殊的情况,我们称之为“肿块”,肿块即为由四个黑色像素组成的类似正方形的图像块,这种算法不能去除图像中的肿块,如图 4.17 所示,细化的结果中仍然存在肿块(改进的地方用圆圈标注)。

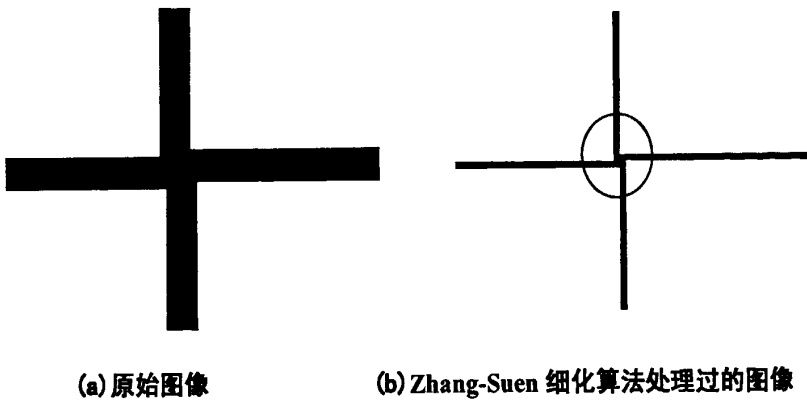


图 4.17 细化处理中的肿块情况

4.4.2 问题的解决

我们假设所有图像在经过 Zhang-Suen 细化算法处理后，都存在着肿块，在这种情况下，我们在算法处理完毕后，再采用一个 5\*5 的模板（如图 4.18）遍历每个像素，这个模板中除肿块外的其它像素都为白色像素，当在处理后的图像中找到与之匹配的像素点后，删除中心像素和左对角线上的黑色像素，采用这种方法的目的就是要保持细化后的骨架都为单像素的，效果如图 4.19 所示：

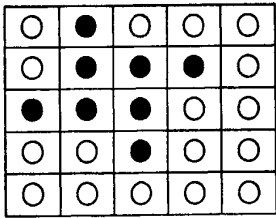


图 4.18 判断肿块模板

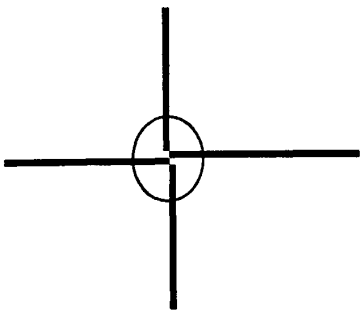


图 4.19 经处理后的肿块效果图

## 4.5 细化后单像素化处理

### 4.5.1 单像素化方法

具体方法是：在 Zhang-Suen 算法的基础上，对细化的结果进行单像素化，并将笔画断开的地方连接起来，由此得到的骨架不仅能保证其是单像素宽的，而且还能保证其连通性。

对细化结果进行单像素化的具体方法是<sup>[59]</sup>：

T1：对细化结果得到的骨架进行扫描，当扫描到黑色像素时，初始化标记 flag，并将其设置为 0，然后进行以下操作：第一步首先对这个黑色像素点的对角线方向进行检测，如果存在黑色像素，则将它压入队列 q 中，同时 flag 的值增加 1，将此黑色像素点变为白色(将其像素值设为 0)。然后第二步如果 flag 的值增加 1，则将此黑像素点的水平和竖直方向的黑像素存入矩阵 D 中，同时此像素置反，但并不将它们压入队列中。如果 flag 的值为 0，那就将当前的像素点的水平和竖直方向黑色像素存入队列中。第三步将 flag 重置成 0，删除当前点，如果队列非空，那么就转到第一步进行下一个点的搜索；若队列为空，则结束目前的区域，到下一个连通的区域进行搜索。

T2：依次处理每个连通的区域，直到扫描完骨架图像。

T3：将要删除点的矩阵 D 在原图中对应的点的像素置为 0，也就是在 Zhang-Suen 算法细化过的骨架上去掉 D 中的点，从而得到单像素宽的骨架。

但经试验可知，单像素化后的骨架有时会保证不了连通性，即骨架出现断开的情况，为了保证骨架的连通性，必须对断开的地方进行连接。

经试验分析，断开点的地方只有一个像素的断裂，针对这种情况，提出了断开点连接的具体方法：U1:对原图增加四条两个像素宽的黑边；U2:扫描原图，如果为黑色像素，那么执行以下操作：第一步对以当前黑色像素点为中心的 8-邻域求和，如果其值不为 2，则执行 U3；如果值是 2，那么这个点可能是断开点或端点；第二步找到与当前黑色像素点相邻的 8-邻域点后，将以它们为中心的 8-邻域设置为 0，然后对以当前像素点为中心的 25-邻域求和，如果其值为 0，那么它为端点，从而执行 U3；如果其值大于 0，则为断开点，将 25-邻域中值为 1 的点的横坐标和中心点的横坐标相加得到 xSum，同理可以得到纵坐标 ySum，然后将 xSum、ySum 求平均值而得到的像素点置为 1，最终将断点连接起来。U3：转到 U2，直到扫描结束。通过此方法，使得细化最终的结果在保持 Zhang-Suen 算法特点的基础上进一步改进了细化的结果。

#### 4.5.2 结果分析

图 4.20 为线划图像的局部线条细化效果,其中(a)为原始线划图像,(b)为 Zhang-Suen 算法处理过的局部线条,(c)为经过在生成的骨架上面再进行单像素化处理所得到的效果图。

由对比可知,细化像素宽度比较大的线条如(a)所示,细化的结果往往不能保证其是单像素的如(b),这会增加图像存储及识别的难度,经过后续的单像素方法处理后,骨架中的多像素问题被很好的解决了。

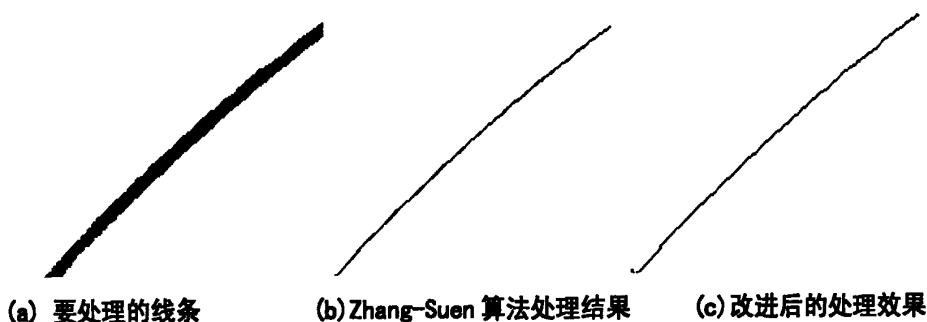
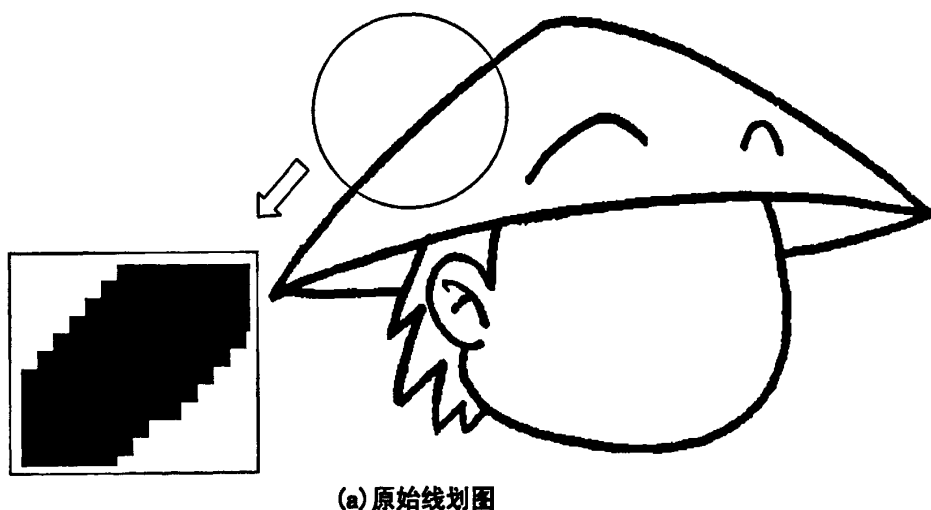
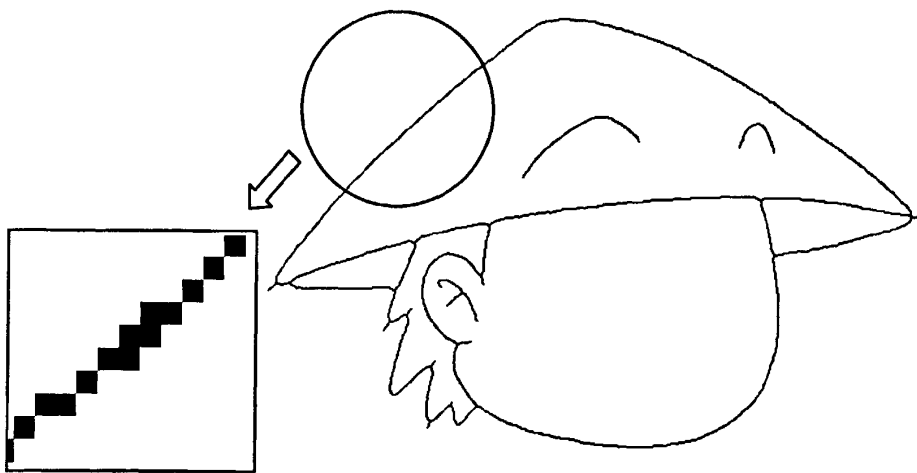


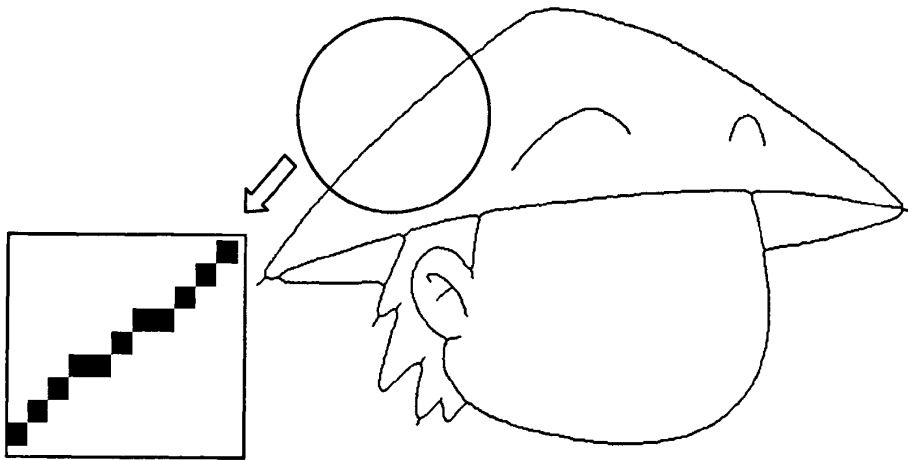
图 4.20 线划图像中局部线条细化

图 4.21 为实际线划图像的细化效果(改进的地方用圆圈标注,并进行了局部放大),通过图像对比可知,在实际图像中线条细化所产生的非单像素宽骨架的问题被很好的解决了。





(b) 经 Zhang-Suen 算法处理后的效果

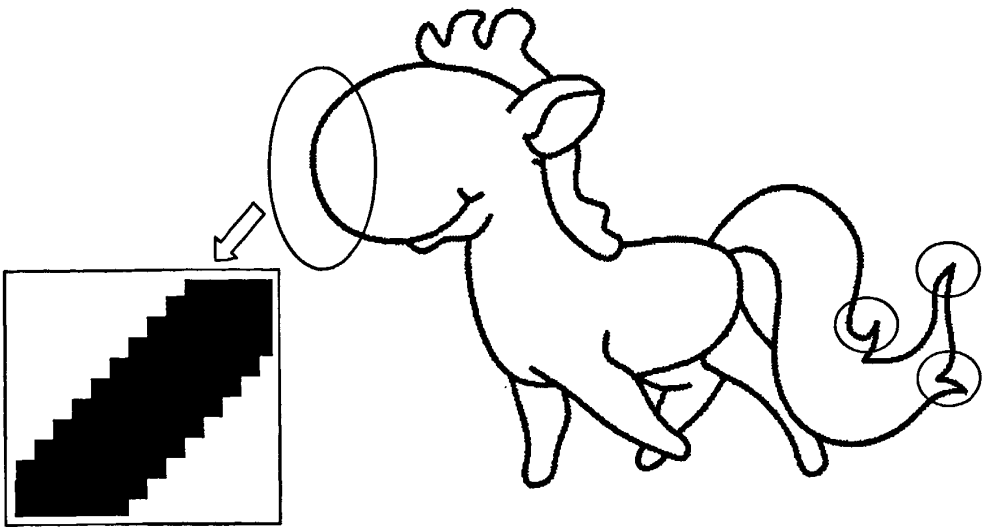


(c) 改进后的细化效果

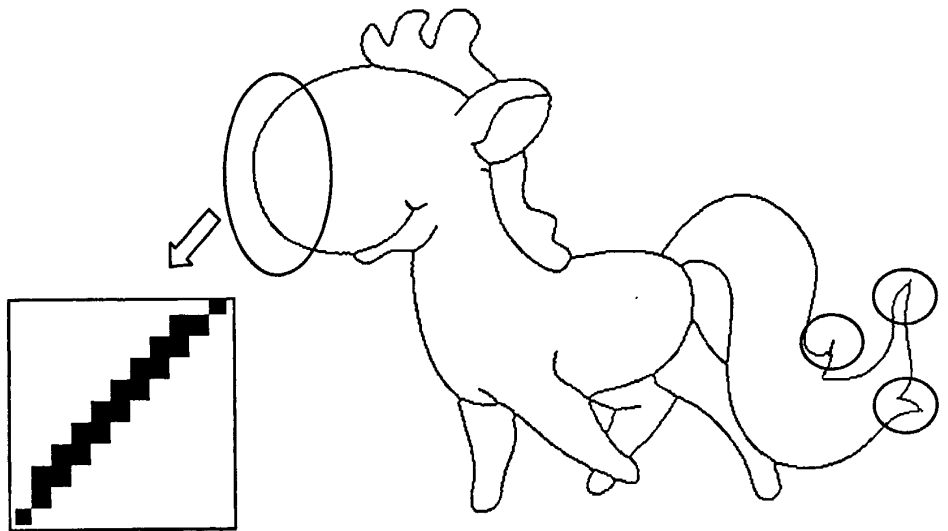
图4.21 线划图像整体细化效果

4.6 实验综合分析和比较

图 4.22 将以上改进的锐角处理与骨架单像素化两种方法融合在一起,通过对比试验可知(改进的地方用圆圈标注,并对单像素化的线条进行局部放大以方便对照),在实际的线划图像中的锐角细化问题和保证骨架为单像素的问题都得到了解决,这对于实际生产中的细化研究具有重要的意义。

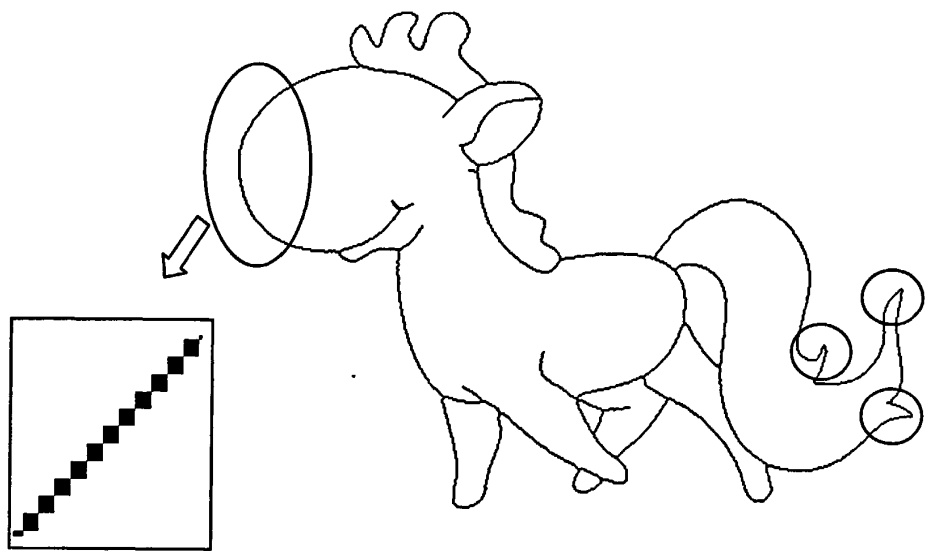


(a) 原始的线划图像



(b) Zhang-Suen 算法的细化效果





(c) 将两种改进方法结合到一起的细化效果

图 4.22 线划图像整体细化效果

4.7 本章小结

本章主要介绍了基于 Zhang-Suen 算法的三种改进方法和一种特殊情况的处理。并且将锐角处理和单像素处理这两种改进方法组合起来进行了比较。通过本章的实验比较可以发现经过预处理中对锐角的像素删除、Zhang-Suen 算法后对细化结果进行的单像素化操作和对笔划交叉处畸变的改进确实改善了细化的质量，达到了预期的目的。

## 第五章 结论与展望

### 5.1 结论

细化技术具有能减少数字图像的数据量、降低图像识别难度、费用低并且处理速度快等优点。随着模式识别技术的不断进步,用细化技术来进行图像的预处理越来越多地得到人们的重视。但是,目前基于该技术生成骨架的误差影响因素还比较多。本文对影响骨架生成的几种因素进行了分析,并且着重在第四章对影响细化效果的图像所固有原因和应用改进方法等相关方面进行了分析和对比实验。通过上述研究,本文主要得出了以下结论:

1、通过对细化前的预处理阶段增加删除锐角像素方法,应用这些主要针对锐角细化的条件,通过改进后的算法和原始算法结果进行对比实验,可以发现锐角细化的质量明显提高。

2、提出了一种基于 Zhang-Suen 算法改进的笔划趋势预测的算法,通过试验可以发现,运用这种算法可以获得比较严格八连通且各向对称的中心骨架,并能有效地改善线条交叉处的形状同时还有较快的处理速度。

3、在数字图像的细化过程中,能否保证细化的结果为单像素,是评判细化算法优劣的一个重要标准。在用 Zhang-Suen 算法进行细化过程中,不能保证细化结果都为单像素,这时需要在细化结果的基础上再进行单像素化操作,以保证细化后骨架的质量。

本文通过对几种主要细化方法进行比较和综合研究,并通过比较分析不同算法的组方法来优化细化结果,最后验证方法的准确性和可靠性。

### 5.2 存在的问题及展望

数字图像的细化处理是目前位图自动矢量化领域研究的重点之一,而细化则是矢量化的主要组成部分。该过程可以有效地去除线划图像中的多余信息,把它变成一幅清晰的点线图,是进行正确可靠的记录拓扑结构的基础。

本文重点对线划图像的细化算法进行研究,在归纳和吸收国内外学者研究成果的基础上,根据线划图像的特点,实现了多种针对线划图像实际特点的改进细化算法,包括:在细化之前的预处理中对锐角处理的改进,试验了一种改进的基于笔划趋势预测的算法和对细化结果进行单像素化的算法。实验结果证明,改进后的算法能够很好地完成工作,并能有效地融合在一起,从而快速、高效地完成线划图像的细化处理工作。

在细化的文献中,对于细化的要求都有一个一般性的统一,这些要求包括了拓扑结

构的保存,几何性质、均质性、可重建性和好的处理速度。重建性,或者说是重新恢复原始图像的能力,是评判一个骨架是否能够精确代表一幅图像的客观标准。均质性或者说在旋转状态下的不变性在迭代算法中是不可能实现的,在串行算法中,像素的检查顺序决定了结果,而在并行算法中,在每次迭代中只删除 1 到 2 条边界,由迭代的顺序决定了骨架。在细化中保持连通性和拓扑关系已经被解决了,在串行算法中,检查黑色像素点的  $3 \times 3$  局部邻域来判断像素的删除是非常有效和快捷的,并行算法把每个循环分为几个子循环或者通过在子循环中考虑一个较大的局部邻域区域来进行像素删除的判定。几何性质的保存是一个更加困难的课题,最主要的困难是如何达到骨架整体连通性和算法简洁性,考虑小的局部特征是需要,但它不能提供整体的特征,我们就需要结构信息去区分真正的噪声点和轮廓点。总的来说,生成一个骨架需要解决问题的集合依赖于应用,有些算法以中心轴转换为基础,并且拥有重建的能力,这些算法对于为贮存而进行的数据压缩或传真传送很合适。然而,用这些方法处理的骨架对轮廓噪声很敏感,线条的毛刺处可能被看成是图像的突起,这是由距离转换的使用所决定的。当然,这些性质让算法有准确的重建原始图像的能力,但这些性质对于一系列可忽略的局部噪声没有用处。对于近年来的应用,依据图像线条的中心线做出的曲线集合来表示图像更加重要,而一些微小的轮廓瑕疵可以忽略。基于删除像素和中心轴转换两种不同的方法可以得到两种不同的骨架,应该根据需要选择不同的方法。

轮廓质量的比较在很大程度上是主观的判断,因为中心线、噪声分支和过度腐蚀的概念还没有被精确地定义过。另外,当一个轮廓分支代表两个轮廓形状的时候,怎么量化和客观表达还不清楚。当一种新的算法被提出的时候,比较是必不可少的,最后的决定是建立在试验基础之上,一个更加客观的对于图像细化质量的评价框架需要继续研究。随着科学的不断发展和技术的不断进步,细化算法作为矢量化、数据压缩、模式识别和机器视觉领域的重要组成部分,必将会在越来越多的领域里得到更加广泛的应用。

## 参考文献

- [1] E.R.Davies, A.P.N.Plummer. Thinning Algorithms: a Critique and a New Methodology[J].Pattern Recognition Society, 1981, Vol.14,NO.1-6
- [2] Louisa Lam, Seong-Whan Lee and Ching Y. Suen. Thinning Methodologies—A Comprehensive Survey[J].Transactions on Pattern Analysis and Machine Intelligence, vol.14,NO. 9, September 1992
- [3] U.Montanari, Continuous skeletons from digitized images[J],J.Ass Comput. 1969, Mach. 16, 534-549
- [4] S. Peleg, A. Rosenfeld. A min-max medial axis transformation[J]. IEEE Trans. Pattern Anal. 1981, Mach, Intell.3,208-210
- [5] J.L. Pfaltz, A. Rosenfeld. Computer representation of planar regions by their skeletons, Communs Ass[J]. Comput.1967,Mach.10,119-125
- [6] 马宁. 指纹图像的二值化与细化研究[D]. 南京理工大学硕士学位论文, 2006 年 6 月
- [7] E. S. Deutsch. Thinning algorithms on rectangular, hexagonal and triangular arrays[J], Communs Ass. comput, 1972, Mach.15, 827-837
- [8] M. Beun. A flexible method for automatic reading of handwritten numerals[J]. Philips Tech. Rev, 33, 89-101, 1973
- [9] R.Stefanelli, A. Rosenfeld. Some parallel thinning algorithms for digital pictures. J.Ass.comput[J]. Mach. 18, 255-264, 1971
- [10] A. Rosenfeld. Connectivity in digital pictures[J]. J. Ass. comput. Mach. 17, 146-160, 1970
- [11]D. Rutovitz. Pattern recognition[J]. J. R. statist. Soc. 129,504-530,1966
- [12] E. S. Deutsch. Thinning algorithms on rectangular, hexagonal and triangular arrays[J]. Communs Ass. comput, Mach .15, 827-837,1972
- [13] C. Arcelli, L. Cordella and S. Levialdi. Parallel thinning of binary pictures[J]. Electron. Lett. 11, 148-149, 1975
- [14] 高永英, 张利, 吴国威. 一种基于灰度期望值的图像二值化算法[J]. 中国图像图形学报, 1999, 6(4): 524-527
- [15]齐丽娜, 张博. 最大类间方差法在图像处理中的应用[J]. 无线电工程, 2006(7)
- [16]N B Rais. Adaptive Thresholding Technique for Document Image Analysis. Prentice Hall, 1986
- [17]Y Chen, G Leedham. Decompose algorithm for thresholding degraded historical document images.

- [18] F. W. M. Stentiford, R. G. Mortimer. Some New Heuristics for Thinning Binary Handprinted Characters for OCR[J]. VOL.SMC-13, NO. 1, January/February 1983
- [19] 贾永红. 计算机图像处理与分析[M]. 武汉大学出版社. 2001 年 9 月
- [20] D. Rutovitz. Pattern recognition[J]. J. Roy. Stat. Soc., vol. 129, Series A, 504-530, 1966
- [21] C.J. Hilditch. Linear skeletons from square cupboards[J]. Machine Intell.(B. Meltzer and D. Michie, Eds.). New York: Amer. Elsevier, 1969, 403-420
- [22] A. Rosenfeld. Connectivity in digital pictures[J]. J. ACM, vol. 17, no. 1, 146-160, 1970
- [23] C. Arcelli and G. Sanniti di Baja. On the sequential approach to medial line transformation[J]. IEEE Trans. Syst. Man Cybern., vol. SMC-8, no. 2, 139-144, 1978
- [24] P. S. P. Wang and Y. Y. Zhang. A fast serial and parallel thinning algorithm[J]. Proc. Eighth Euro. Meeting Cybern. Syst. Res. (Vienna, Austria), 1986, 909-915
- [25] M. Beun. A flexible method for automatic reading of handwritten numerals[J]. Philips Tech. Rev. vol. 33, no. 5, 89-101, 1973
- [26] T.Y. Zhang, C.Y. Suen. A flexible parallel thinning algorithm[J]. Proc. Int. Conf. Patt. Recog. Image Processing (Dallas, TX), 1981, 162-167
- [27] Y. K. Chu and C. Y. Suen. An alternative smoothing and stripping algorithm for thinning digital binary patterns[J]. Signal Processing, vol. 11, no. 3, 207-222, 1986
- [28] T. Pavlidis. A thinning algorithm for discrete binary images[J]. Comput Graphics Image Processing, vol. 13, 142-157, 1980
- [29] C. Arcelli. Pattern thinning by contour tracing[J]. Comput. Graphics Image Processing, vol. 17, 130-144, 1981
- [30] Carlo Arcelli, Gabriella Sanniti di Baja. A thinning algorithm based on prominence detection[J]. Pattern Recogn. vol. 13, no. 3, 225-235, 1981
- [31] S. Yokoi, J. I. Toriwaki, and T. Fukumura. Topological properties in digitized binary pictures[J]. Syst. Comput. Contr, vol. 4, no. 6, 32-39, 1973
- [32] Arcelli, C., Sanniti di Baja, G. A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform[J]. IEEE Trans. Patt. Anal. Machine Intell., vol. 11, no. 4, 411-414, 1989
- [33] Azriel Rosenfeld. A characterization of parallel thinning algorithms[J]. Inform. Contr. vol. 29, no. 3, 286-291, 1975
- [34] C.J. Hilditch. Comparison of thinning algorithms on a parallel processor[J]. Image Vision Comput., vol. 1, no. 3, 115-132, 1983
- [35] R. Stefanelli and A. Rosenfeld. Some parallel thinning algorithms for digital pictures[J]. J. ACM, vol. 18, no. 2, 255-264, 1971

- [36] C. Arcelli. A condition for digital points removal[J]. Signal Processing, vol. 1, no. 4, 283-285, 1979
- [37] E.S.Deutsch Thinning algorithms on rectangular, hexagonal, and triangular arrays. Comm. ACM, vol. 15, no. 9, 827-837, 1972
- [38] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns[J]. Comm. ACM, vol. 27, no. 3, 236-239, 1984
- [39] Y. S. Chen and W. -H. Hsu. A modified fast parallel algorithm for thinning digital patterns[J]. Pattern Recogn. Lett., vol. 7, no. 2, 99-106, 1988
- [40] S. Suzuki and K. Abe. Binary picture thinning by an iterative parallel two-subcycle operation[J]. Patt. Recogn., vol. 10, no. 3, 297-307, 1987
- [41] Z. Guo and R. W. Hall. Parallel thinning with two-subiteration algorithms[J]. Comm. ACM, vol. 32, no. 3, 359-373, 1989
- [42] C. M. Holt, A. Stewart, M. Clint, and R. H. Perrott. An improved parallel thinning algorithm[J]. Comm. ACM, vol. 30, no. 2, 156-160, 1987
- [43] Yung-Sheng Chen, Wen-Hsing Hsu. A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms[J]. Patt. Recogn., vol. 22, no. 3, 267-282, 1989
- [44] R. T. Chin, H. -K. Wan, D. L. Stover, and R. D. Iverson. A onepass thinning algorithm and its parallel implementation[J]. Comput. Vision Graphics Image Processing., vol. 40, 30-40, 1987
- [45] P. Saraga and D. J. Woollons. The design of operators for pattern processing. Proc. IEEE NPL Conf. Pan. Recogn. (Teddington), 1968, 106-116
- [46] C. Arcelli and G. Sanniti di Baja. Medial lines and figure analysis[J]. Proc. 5th Int. Conf Pattern Recogn., 1980, 1016-1018
- [47] J.L. Pfaltz, A. Rosenfeld. Computer representation of planar regions by their skeletons, Communs Ass[J]. Comput.1967,Mach.10,119-125
- [48] C. J. Ammann and A. G. Sartori-Angus. Fast thinning algorithm for binary images[J]. Image Ksion Comput., vol. 3, no. 2, 71-79, 1985
- [49] U. Eckhardt. A note on Rutovitz' method for parallel thinning. Patt. Recogn. Lett., vol. 8, no. 1, 35-38, 1988
- [50] J. H. Sossa. An improved parallel algorithm for thinning digital patterns[J]. Patt. Recogn. Lett., vol. 10, no. 2, 77-80, 1989
- [51] O. Baruch. Line thinning by line following[J]. Patt. Recogn. Lett., vol. 8, no. 4, 271-276, 1988

- [52] J. L. Mundy and R. E. Joynson. Automatic visual inspection using syntactic analysis[J]. in Proc. Int. Conf: Patt. Recogn. Image Processing, 1977, 144-147
- [53] K. Kedem and D. Keret. A fast algorithm for skeletonizing lines by midline technique[J]. in Proc. Int. Comput. Sci. Conf, (Hong Kong), 1988, 731-735
- [54] I. S. N. Murthy and K. J. Udupa. A search algorithm for skeletonization of thick patterns. Comput Graphics Image Processing, vol. 3, 247-259, 1974
- [55] R. M. K. Sinha. Primitive recognition and skeletonization via labeling[J]. in Proc. Int. Conf. Syst. Man Cybern. (Halifax, Canada), 1984, 272-279
- [56] R. M. K. Sinha. A width-independent algorithm for character skeleton estimation[J]. Comput Vision Graphics Image Processing, vol. 40, 388-397, 1987
- [57] E. Persoon and K. S. Fu. Shape discrimination using Fourier descriptors[J]. IEEE Trans. Syst., Man Cybern., vol. SMC-7, no. 3, pp. 170-179, 1977
- [58] 刘桂雄, 申柏华, 冯云庆. 基于笔划趋势分析的二值图像细化方法[J]. 光学精密工程. 2003, 11(5): 527-530
- [59] 吴选忠. Zhang 快速并行细化算法的扩展[J]. 福建工程学院学报. 2006.4(1). 89-92

## 致 谢

在研究生三年中，我学到了很多东西。这离不开我身边老师和同学的帮助。在此谨向他们表达我深深地谢意。

感谢我的指导老师隋立春教授在学习和生活上对我的教导，隋老师严谨的治学态度、渊博的学识和开阔的思维对我在研究生期间的学习起到了很大的作用。

感谢总参测绘研究所的张宝印老师和长安大学的崔建军老师在专业知识和生活上的帮助。

感谢同门王芃芃、李伟、李智霖、王蒙、张硕、朱建峰、郭涛、陈卫。在这三年里他们给了我很多帮助。

感谢舍友俞岱、梁怀翔和袁伟研究生生活期间的帮助和陪伴，使我的宿舍生活丰富多彩。

最后，我要感谢张娟和我的家人，她们一直给我精神上的鼓励、物质上的支持、学习上的督促与激励，对家人的感激之情是我无法用言辞表达的。她们的关怀和激励，是我前进的最大动力。



# 线划图像的细化算法研究

作者：[曹玉龙](#)  
学位授予单位：[长安大学](#)

引用本文格式：[曹玉龙](#) [线划图像的细化算法研究](#)[学位论文]硕士 2011