论文编码：TP39

首都师范大学
Capital Normal University

首都师范大学本科毕业设计论文

图像轮廓的近似表示方法

Approximate Representation of Image Contour

论文作者：姜捷
院　　系：信息工程学院
专　　业：软件工程
学　　号：1121000203
指导老师：周修庄

完成日期：2016 年 4 月 26 日

摘　要

数字图像处理方兴未艾，其中最重要的方法之一便是图像轮廓的近似表示，从轮廓中提取特征信息进行其它应用。多边形近似是一种高效简洁的图像轮廓的近似表示方法。

本文主要研究图像轮廓的多边形近似算法的优化。PSO 算法从随机解出发，通过迭代寻找最优解，通过适应度来评价解的品质，它比遗传算法规则更为简单，通过追随当前搜索到的最优值来寻找全局最优，具有实现容易、精度高、收敛快等优点。关键点检测法多种多样，以轮廓多边形的内角度数为特征值的一种关键点检测法，具有很高的精度和稳定性。

本文根据问题的复杂性，分别用整数粒子群优化（iPSO）算法对多边形近似最小边数问题进行研究，用多边形内角度数为特征值的关键点检测法对多边形近似最小误差问题进行研究。iPSO 算法是基于 PSO 算法改进的，使之在整数中进行多边形逼近的处理，更加简单，可是算法依然有局部优化不足，结果偶尔跑出可行解空间的缺点，本文在局部优化当中加入了分割与合并技术，以控制解的可行性，结果证明得到的结果全部都在可行域内。不同于 PSO 算法的全局优化，本文用局部优化的关键点检测法来解决多边形近似最小误差问题，结果精确稳定。

关键词：图像轮廓近似；多边形近似；整数粒子群优化算法；关键点检测法

ABSTRACT

Digital image processing is in the ascendant. That approximate the outline of the image is one of the most important processing methods in it. Feature information can be extracted from the profile for additional applications. Polygonal approximation is a simple and efficient image outline approximate representation.

This paper studies optimization algorithm of polygonal approximation of image contours. PSO algorithm starting from random solution, through iterative find the optimal solution, through fitness to evaluate the quality of the solution, which is simpler than genetic algorithm rules, by following the current optimum value to search the global optimal solution. It's easy to implement and has better constringency speed. The results have the advantage of high precision. Dominant-Point-Detection method which regards the interior angles of a polygon as characteristic values has the advantage of high accuracy and stability.

Based on the complexity of the problem, this paper uses integer PSO (iPSO) algorithm to solve the minimum number of edges in polygonal approximation issue, and uses Dominant-Point-Detection method with angle eigenvalues to solve the smallest error in polygonal approximation issue. The iPSO algorithm is improved based on PSO algorithm. So that polygon approximation process in integer simply, but there is still insufficient local optimization, the results occasionally ran out of feasible solution space. The paper added local optimization that split-and-merge-combined technology to control the feasibility of the solution, and the results prove that the solutions obtained are all in the feasible region. Unlike global optimization of PSO, this paper solve the polygonal approximation of minimum error problem with local optimization of Dominant-Point-Detection, and has accurate and stable results.

Key Words: Image-Contour; Polygonal-Approximation; iPSO; Dominant-Point-Detection

目 录

第一章 绪论

1.1 论文的选题背景

数字曲线的表示在图像分析，模式识别和计算机视觉中是一个非常重要的课题。通过在一个具体情境的对象中提取轮廓，遍历轮廓可以得到一系列的候选点。其中包含了对象的大量形状信息，尤其是轮廓的边角信息。数字曲线的有效表示有利于更加便利地进行图像分析，比如形状匹配，对象识别和图像修复[1]。

数字图像处理（Digital Image Processing）是用计算机对图像信息进行处理的一门学科，是利用计算机对图像进行各种处理的技术和方法。20 世纪 20 年代，图像处理首次得到应用。20 世纪 60 年代中期，随电子计算机的发展得到普遍应用。60 年代末，图像处理技术不断完善，逐渐成为一个新兴的学科。数字图像处理主要是为了修改图形，改善图像质量，或是从图像中提起有效信息，另外还可以对图像进行体积压缩，便于传输和保存。

现在，图像处理技术在许多应用领域受到广泛重视并取得了重大的开拓性成就，属于这些领域的有航空航天、生物医学工程、工业检测、机器人视觉、公安司法、军事制导、文化艺术，人工智能等，使图像处理成为一门引人注目、前景远大的新型学科。

图像轮廓可以说是数字处理中经常会用到的参数之一。针对数字图像轮廓提取，目前已经有许多研究者和工程技术人员提出了各种各样的方法。

很多来源于现实世界的图像，比如航空照片，绘画、高清电视网络信号、机器人视觉传输等，都是位图文件格式。位图也称为光栅图或点阵图。位图描述的是每一个像素点的复杂颜色和灰度级变化，还原度比较真实。但是一些高精度的工程和计算领域往往使用更多的是矢量图。矢量图高度抽象，适合用于计算。矢量图是用几何图形的特征数据及其属性来描述图像。将数字位图中的必要信息，比如轮廓，转换成矢量图，将大大有利于进行后续研究。将光栅图像进行分析，识别并重建矢量图的过程就是矢量化。矢量图的最小单位是图元，比如直线、圆弧等基本形状。

对矢量图的几何参数进行存储，只需要存储坐标点，半径，起点和终点等参数，在不损失图像主要信息的前提下大幅度减少了存储空间，还能对图像进行高效灵活的编辑。因此，图像轮廓的矢量近似表示方法具有很大的实际应用价值。

1.2 图像轮廓近似的主要内容与方法

迄今为止，图像轮廓的近似表示有了许多的方法。在图像处理中，目标轮廓线或区域的边界可以表示为一条数字曲线，对其进行有效的描述是进行后续图像分析任务非常重要的一个步骤。将一条复杂的曲线分解成若干相对简单的曲线段，再对每一段用解析曲线进行描述是一种有效而且常用的曲线分析策略。描述曲线段的解析曲线可以是直线段、圆弧或高阶曲线如样条函数和参数集等。而采用直线段构成的多边形近似是一种最简单的曲线描述方法，因为相较于用圆弧或高阶曲线,多边形近似的计算复杂度要低得多[2]。所以，图像轮廓的多边形近似是被研究最多，应用最广的。图像轮廓的多边形近似优化算法主要分为以下两类：

min-ε：给定一个修正值 M，从数字曲线中选取 M 个顶点，从而构造一个多边形，使其与原曲线的误差尽可能的最小。

min-#：给定一个误差容忍范围 ε，从数字曲线中选取尽可能少的顶点，从而构造一个近似误差不超过 ε 的边数最少的近似多边形。

在不同的条件限制下，这两种类型的多边形近似针对不同的目的。min-ε 针对在保持紧密多边形近似的同时，最小化误差，而 min-#则是在保持具体近似准确度的前提下，最小化近似多边形的顶点数或者边数。两种优化方法都需要巨大的搜索空间和计算消耗才能得到优化解。多边形近似就是在保存主要信息的前提下，使存储信息本身的占有空间减少[3]。

对于 min-ε 问题有$\binom{N}{M}$种不同的方法从搜索空间的 N 个曲线点中找到 M 个顶点。

在 min-#问题中，顶点或者边数的数量是可变的，并且搜索空间比 min-ε 问题要大很多。在实际的图片分析应用中，提取出的数字曲线通常有大量的点，从而也导致了搜索空间的巨大。如此一来，寻找到全局优化解也不切实际了。

典型的优化方法中，例如 DP[4]（动态规划）仅仅可以处理小规模问题（例如一百个点），并不适合实际应用中会碰到的大规模问题。因为这个原因，许多基于类似 DPD（主要点缺失）、BPD（断点抑制）或者处在两者之间的局部启发式搜索方法被提出来应用于解决多边形近似问题。

现有的方法大都着眼于求局部的最优解以减少计算上的花费。这些方法中，经典 的算法有顺序跟踪法、拆分法、合并法、拆分与合并法（染色体修复法）以及关

键点检测法。这些方法属于局部的优化方法，其优点是计算速度快，但由于对最初选定的起始点或初始产生 的解的依赖性，其最终得到的解有可能远离全局最优解。

近年来，一些基于全局优化的算法，如遗传算法[5]、禁忌搜索算法、蚁群克隆算法[6]以及微粒群优化算法等用于求解 min-ε 或 min-#两类多边形近似问题并取得了较好的近似效果。但这些方法都存在着收敛速度慢，局部优化能力差的问题。

1.3 本文的研究内容以及目的

数字图像处理在各个领域的应用正在兴起，其作用和意义也逐渐举足轻重。拍照程序进行人脸识别，人工智能认识事物，航空摄像自动导向等等，小到民生，大到国防，都要进行图像处理。图像轮廓近似作为数字图像处理经常用到的技术，必然也是有其重要的实际作用。而研究图像轮廓的近似表达，就是为了更加高效精准快速的处理当前信息化时代的大量数据。

多边形近似也有着很大的缺陷，那就是它的精度可能不够高。对于一条闭合的曲线，当多边形的线段数目与图像轮廓边界点数是一样时，近似是精确的[7]。实际应用中，轮廓的多变形近似的目的是用尽可能少的顶点表示边界的形状。使近似多边形达到顶点数和边数都尽可能的少，是一个具有挑战性的问题。所以对多边形近似的 min-#边数最小和 min-ε 误差最小问题进行研究很有必要。

本文的主要研究内容是对当前图像轮廓的多边形近似的 min-ε 和 min-#两大类问题分别作算法优化的尝试性研究。

作者阅读了国内外有关图像位图转换成矢量图，轮廓提取，多边形近似算法的优化等资料，对相关的学术论文和学位论文也进行了深入的研究。

针对最小误差 min-ε 问题，也是相对 min-#问题简单一些的问题，本文用了关键点检测法[8]进行研究。一般的关键点的特征函数是用线段表示弧的误差程度函数 ISE（积分误差平方和）来表示，或者用弦对应的弧上的点到直线的距离来表示。本文中寻找关键点是用角度来表示。近似多边形中相邻两条边的角度越接近平角，则说明两条边的连接点的重要性越小。

针对最小边数 min-#问题，也是相对更加复杂一些的问题。本文用了更加快速简洁的启发式算法——整数粒子群优化（iPSO）算法来进行研究。相对于遗传算法，该算法更加简单，设置的参数和约束更少。相对于一般的二进制粒子群算法，这种算法可以更加节省存储空间，运算更加有效率。

最后通过实验，加深对两类问题的理解，改进算法的效率。

1.4 论文的章节安排

本文由六个章节组成。

第一章首先论述了图像轮廓的多边形近似的背景和意义，随后介绍了“图像轮廓的近似表示”课题目前研究的内容和方法，并简单地介绍了本文的研究内容和目的，最后介绍了自己在论文完成过程中的准备工作和解决问题的策略。

第二章描述了前期准备工作的形态学图像处理，接着介绍从数字位图中提取轮廓信息将之矢量化的方法改进，最后对多边形近似问题作形式化的描述。

第三章围绕整数粒子群优化（iPSO）算法进行论述，介绍粒子群算法优化（PSO）基本原理，论述了二进制粒子群算法（bPSO）针对图像问题进行的改进与不足，从而引入到整数粒子群（iPSO）的算法介绍中，详细说明 iPSO 算法的改进和针对 min-#问题的应用，并介绍了 iPSO 算法设计思路的算法流图，最后通过实验进行了验证。

第四章围绕关键点检测算法进行论述，首先论述关键点检测的算法原理，然后定义本文中关键点，介绍其优缺点，最后进行实验验证。

第五章对实验结果进行总结，对实验的结果进行比较分析，深入地了解多边形近似问题的关键。

第六章对前期实验目标与后期改进算法的优点进行总结，对目前系统框架的缺陷提出未来研究的展望。

第二章 前期准备与问题描述

2.1 位图转矢量图

本文使用光栅图（位图）为原始图片，从中提取轮廓信息来进行后续处理。光栅图信息量大而复杂，从中识别出图像的轮廓，需要提前进行形态学图像处理，以排除一些其他因素的干扰。

一般形态学图像处理的流程为：首先将光栅图进行灰度处理，然后根据图像的明暗程度，寻找一个平均的灰度级，在此灰度级的基础上，用填洞、膨胀和腐蚀的组合来处理图像的细节，最后转换成二值图像。

针对具体的一幅图像，处理的流程会有很多细微的变化。颜色单一、背景简单的图像，可以直接灰度处理，甚至不用设置特别的灰度级，然后直接转换成二值图像，得到的轮廓已经非常清晰。而背景颜色复杂，轮廓边缘与背景融合的复杂图片，则需要进行细化的处理，才可以得到比较清晰的轮廓。

最后对二值图像进行遍历，提取图像的轮廓信息，将轮廓信息矢量化。

2.2 准确提取图像轮廓

准确提取图像轮廓是进行轮廓多边形近似的基础和前提。依赖于轮廓的准确表述，才可以更好地进行多边形近似。

一般提取图像轮廓，我们可以直接使用 MATLAB 的库函数 contour（）即可。但是为了进一步得到更加细致和准确的轮廓曲线中顶点的坐标信息，本文使用了《Digital Image Processing Using MATLAB》中定义的 boundarise（）函数。

MATLAB 中库函数[C,h] = contour(...)可以得到返回的轮廓矩阵 C 和轮廓对象 h 句柄的数据。其中轮廓矩阵的定义如下：

元胞矩阵元素 C(k)作为一个两行矩阵返回。每个轮廓线具有关联的定义。如果共有 N 条等高线的情节，然后包括 N 条等高线的轮廓矩阵定义 为：

C = [C(1) C(2)...C(k)...C(N)]

每个轮廓线定义遵循以下模式 ：

C(k) = [level x(1) x(2)...

numxy y(1) y(2)...]

第一项 level，表明的等值线绘制轮廓线的位置。下方的 numxy 表示该位置中定义轮廓线的顶点(x，y) 的数目。其余的列包含的数据的每个顶点。如果第一个和最后一个顶点是相同的轮廓线是一个封闭的循环。如果一个特定的轮廓水平面有多个轮廓线，那么矩阵 y(.)可包含为一个水平面里的所有的轮廓曲线信息。

从库函数[C,h] = contour(...)的定义中我们可以看到，contour 函数得到的矩阵 C 是一个元胞矩阵，其中包含了复杂的信息。一般对于平面图，我们使用库函数[C,h] = contour(BW, [0, 0])来得到平面轮廓矩阵。为了得到轮廓曲线的顶点坐标，我们一般使用语句：[C,h] = contour(BW, [0 0]); C_coordinates= C(:,2:C(2,1))。

虽然函数[C,h] = contour(...)可以得到轮廓曲线顶点的坐标信息，但是由于其算法原理是利用线性插值法，从而提取的轮廓与原曲线有很大的误差，得到轮廓边缘并不准确，也并不平滑。还有很重要的一点是，最终得到的坐标不是按照顺时针顺序排列的。这两大缺点将会大大影响后续多边形的近似优化。

为了保证后续图像轮廓的近似表示是在准确的基础上进行的，所以我们采取对二值图像进行全局遍历搜寻准确边界点的方法得到更加精确的边界描述。

本文使用的 boundaries（）函数基本原理是对一幅二值图像的外部边界进行跟踪遍历，图像的背景像素为零，按照顺时针或者逆时针的顺序，8 连接或者 4 连接的方向，全局遍历图像的像素为 1 且连接紧密的曲线，然后依次全部记录下来，最后得到图像中所有对象的轮廓坐标信息，坐标按照遍历时的顺时针或者逆时针顺序存储，依次存入输出的元胞矩阵中，矩阵中每一个元素就是一个轮廓曲线的坐标矩阵。

函数 boundaries（）是以像素点为基本单位进行遍历，不同于 contour（）函数的线性插值法来寻找中间等值线，保证了结果的准确性和曲线的平滑性，为后续的图像轮廓的近似表示提供了可以依赖的基础和保障。

函数 boundaries（）输出的坐标信息是以顺时针或者逆时针的顺序排列的，这种有序性，有利于本文进行图像轮廓的近似表示优化时，按照空间位置顺序优化轮廓的信息。

图 2.1 从左到右依次是：原始图像；contour（）函数提取轮廓图；boundaries（）函数提取轮廓图

2.3 多边形近似问题形式化

为了对多边形近似问题进行统一的数学抽象化表达，本文统一用以下公式来描述多边形近似中用到的变量：

一个闭合曲线 C，其中包含 N 个点，可以表示成顺时针方向的一系列像素点。

$$\mathrm{C} = \{Z_i(x_i, y_i) | i = 1,2,3 \dots N\}$$

其中Z_I表示曲线中的第 i 个点，坐标为(x_i, y_i)，并且$Z_{N+i} = Z_i$。

$A_{ij} = \{Z_i, Z_{i+1}, Z_{i+2} \dots \dots, Z_j\}$表示从$Z_i$到$Z_j$的弧。$L_{ij}$表示从$Z_i$到$Z_j$的直线，即$L_{ij}$是$A_{ij}$的弦。$\mathrm{d}(Z_k, L_{ij})$表示$Z_k$到$L_{ij}$的距离。用弦$L_{ij}$表示弧$A_{ij}$的误差用公式(2.1)计算：

$$e(L_{ij},\ A_{ij}) = \sum_{Z_k \epsilon A_{ij}} d(Z_k, L_{ij})^2 \qquad (2.1)$$

一个多边形 V 近似成曲线 C，可以用一系列直线段表示,m 是 V 中向量的数量。

$$V = \{L_{t_1t_2}, L_{t_2t_3}, L_{t_3t_4} \dots \dots, L_{t_{m-1}t_m}, L_{t_mt_1}\}$$

问题 min-#多边形近似：给定一个允许误差 ε，找到一个近似多边形 C 使得其向量的数量最少，并且平方误差积分（ISE）小于 ε。曲线 C 和多边形 V 之间的平方误差积分（ISE）用公式(1.2)表示：

$$ISE(V,\ C) = \sum_{i=1}^{m} e(L_{t_it_{i+1}}, A_{t_it_{i+1}}) \qquad (2.2)$$

第三章 iPSO 算法与 min-#问题

3.1 粒子群优化(PSO)算法

粒子群优化(PSO)算法是一种基于种群进化的计算技术。PSO 算法是来源于对鸟群、鱼群等动物社交行为的观察发展的。该算法由肯尼迪和艾尔伯特在 1995 年提出。

PSO 算法首先在可行解空间中初始化一群粒子，每个粒子都代表极值优化问题的一个潜在的最优解，用位置、速度和适应度值三项指标来表示该粒子特征，适应度值由适应度函数得到，其值的好坏表示粒子的优劣。粒子在解空间中运动，通过跟踪个体极值和群体极值来更新个体的位置。个体极值是指个体经历位置中计算得到的适应度值最优的位置，群体极值是指种群中所有粒子搜索到的适应度值最优的位置。粒子每一次更新位置，就计算一次适应度值，并且通过比较新粒子的适应度值和个体极值、群体极值的适应度值，更新个体极值的位置和群体极值[9]。

以下简单介绍 PSO 算法和 PSO 算法的二进制版本。

假设在一个 D 维搜索空间中，一个粒子群中包含 N 个粒子，$s \sqsubset R^D$。第 i 个粒子的位置是一个 D 维向量，定义成：

$$x_i = [x_{i1}, x_{i2}, \dots \dots x_{iD}] \in S, x_{i1} < x_{i2} < \cdots < x_{iM}, x_{ij} \in \{1,2, \dots, \mathrm{N}\}$$

速度则定义成：

$$v_i = [v_{i1}, v_{i2}, \dots \dots v_{iD}] \in S。$$

第 i 个粒子在搜索空间中的最佳先前位置定义：

$$P_i = [P_{i1}, P_{i2}, \dots \dots P_{iD}]$$

变量 g 定义整个粒子群中粒子的最佳先前位置的索引序号。下一时刻的第 i 个粒子的速度和位置更新用以下公式：

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_1 \left(p_{ij}(t) - x_{ij}(t)\right) + c_2 r_2 \left(p_{gj}(t) - x_{ij}(t)\right) \quad (3.1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (3.2)$$

其中，t=1,2……是整数，i=1,2……N 是粒子的序号，j=1,2……D 是粒子的维度序号，C_1 和 C_2 是正常数，是认知种群的参数，r_1 和 r_2 则是正态分布在[0,1]的随机数。为了防止粒子群在搜索空间的快速移动，粒子速度更新通常会限定一个范围$[-V_{max}, V_{max}]$。

3.1.1 二进制粒子群优化（bPSO）算法

PSO 算法最初是为了解决连续变量的问题，比如实数范围的问题。为了使 PSO 算法适应二进制的值，肯尼迪和艾尔伯特[10]提出了一种二进制版本的粒子群优化（bPSO）算法。在 bPSO 算法中，速度被当做一个概率向量，位置向量中的一个元素就是一个概率为 1 的数。对于 bPSO，速度不变,位置更新用以下公式：

$$x_{i+1} = \begin{cases} 1 & if\ r < S\left(V_{ij}(t+1)\right) \\ 0 & otherwise \end{cases} \quad (3.3)$$

而

$$S\left(V_{ij}(t+1)\right) = \frac{1}{1+e^{-V_{ij}(t+1)}} \quad (3.4)$$

其中 r 正态分布在[0,1]的随机数。

在处理 min-#问题，也常常用以下公式代替等式(3.4)

$$S\left(V_{ij}(t+1)\right) = \frac{V_{ij}(t+1)}{2V_{max}} + 0.5 \quad (3.5)$$

3.2 整数粒子群优化（iPSO）算法

数字图像中像素的位置坐标并不是实际意义中的物理坐标，而是一个相对的位置概念。这些离散坐标使用整数来表示更加清晰，使用更加方便。PSO 算法本是用于连续变量的，不适用于数字图像的像素坐标表达。改进后的二进制版本表示方法不够直观，使用起来也不方便，且储存信息占用空间大。而本文使用的整数粒子群（iPSO）算法不仅表示更加清晰，存储空间也减少了不少，处理数字图像中像素的位置坐标问题更加方便。比如表示一个正方形的四个顶点的位置，二进制版本表示方法需要使用四个二进制 01 串，非常不直观且有很多冗余信息（固定长度的二进制 01 串中高位的浪费）占用空间；而使用整数表示方法，仅仅需要四个顶点八个整数即可，直观简洁，节省空间。

3.2.1 粒子表示和适应度评估

给定一个数字曲线$C=\{Z_i(x_i,y_i)|i=1,2,3\ldots N\}$有 N 个点，曲线 C 用多边形 V 近似表示：

$$V=\left\{L_{t_1t_2},L_{t_2t_3},L_{t_3t_4}\ldots\ldots,L_{t_{m-1}t_m},L_{t_mt_1}\right\}$$

其中 t1<t2<….<tm，ti $\epsilon[1,N]$，i 是多边形 V 的顶点序号。

粒子 Xi 可以用整数向量表示成：

$X_i=[x_{i1},x_{i2}\ldots\ldots x_{iM}]$其中$x_{i2}<x_{i2}<\cdots\ldots<x_{iM}$，$x_{ij}\in\{1,2,\ldots\ldots N\}$.M 既是$X_i$的维度，也表示多边形 V 中的向量的数量。用$X_i$表示一个候选解以近似表示曲线 C：

$$V=\left\{L_{x_1x_2},L_{x_2x_3},L_{x_3x_4}\ldots\ldots,L_{x_{m-1}x_m},L_{x_mt_1}\right\}$$

在大多数多边形近似的例子中，多边形中被选取的顶点的数量远远小于曲线上的顶点数（即 M<<N）。

3.2.2 速度更新和位置更新

最初的 PSO 算法中速度更新和位置更新公式并不能产生整数。为了将 PSO 算法应用于整数空间，将速度更新和位置更新公式改造成以下公式：

速度更新公式：

$$v_{ij}(t+1)=\begin{cases}\lfloor\check{v}_{ij}(t+1)\rfloor\ mod\ N & \left(\check{v}_{ij}(t+1)\geq 0\right)\\ -\left(\lfloor\check{v}_{ij}(t+1)\rfloor\ mod\ N\right) & \left(\check{v}_{ij}(t+1)<0\right)\end{cases}\qquad(3.6)$$

mod 表示求余，⌊.⌋表示向下取整，公式（3.6）是为了将速度$v_{ij}(t)$从实数范围转换到整数范围内而得到下一刻速度$v_{ij}(t+1)$，其中$v_{ij}>0$表示顺时针，$v_{ij}<0$表示逆时针。

其中：

$$\check{v}_{ij}(t+1)=v_{ij}(t)+c_1r_1\left(p_{ij}(t)-x_{ij}(t)\right)+c_2r_2\left(p_{gj}(t)-x_{ij}(t)\right)\qquad(3.7)$$

位置更新公式：

$$x_{ij}(t+1)=\begin{cases}\check{x}_{ij}(t+1)-N & \left(\check{x}_{ij}(t+1)>N\right)\\ \check{x}_{ij}(t+1) & \left(0<\check{x}_{ij}(t+1)\leq N\right)\\ \check{x}_{ij}(t+1)+N & \left(\check{x}_{ij}(t+1)<0\right)\end{cases}\qquad(3.8)$$

其中：

$$\check{x}_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{3.9}$$

图（3.1）示意了公式（3.8）的过程和计算原理。

不同于 bPSO 算法中对于粒子位置向量的修正，iPSO 算法的位置向量的长度并不是常数值。在等式（3.1）里的$p_{ij}(t) - x_{ij}(t)$需要改为：

$$p_{ij}(t) - x_{ij}(t) = \begin{cases} p_{ij}(t) - x_{ij}(t) & if\ j \le L \\ p_{iL}(t) - x_{ij}(t) & otherwise \end{cases} \tag{3.10}$$

和

$$p_{gj}(t) - x_{ij}(t) = \begin{cases} p_{gj}(t) - x_{ij}(t) & if\ j \le H \\ p_{gL}(t) - x_{ij}(t) & otherwise \end{cases} \tag{3.11}$$

其中 L 和 H 分别是 P_i 和 P_g 的长度。

理想情况下，位置向量$X_i = [x_{i1}, x_{i2} \dots\dots x_{iM}]$遵循规则$x_{i2} < x_{i2} < \cdots\dots < x_{iM}$，$x_{ij} \in \{1,2,\dots\dots N\}$。但是在实际中，位置向量的更新结果可能为违反这种情况。处理这种办法可以进行重新递增排序和删除重复点。

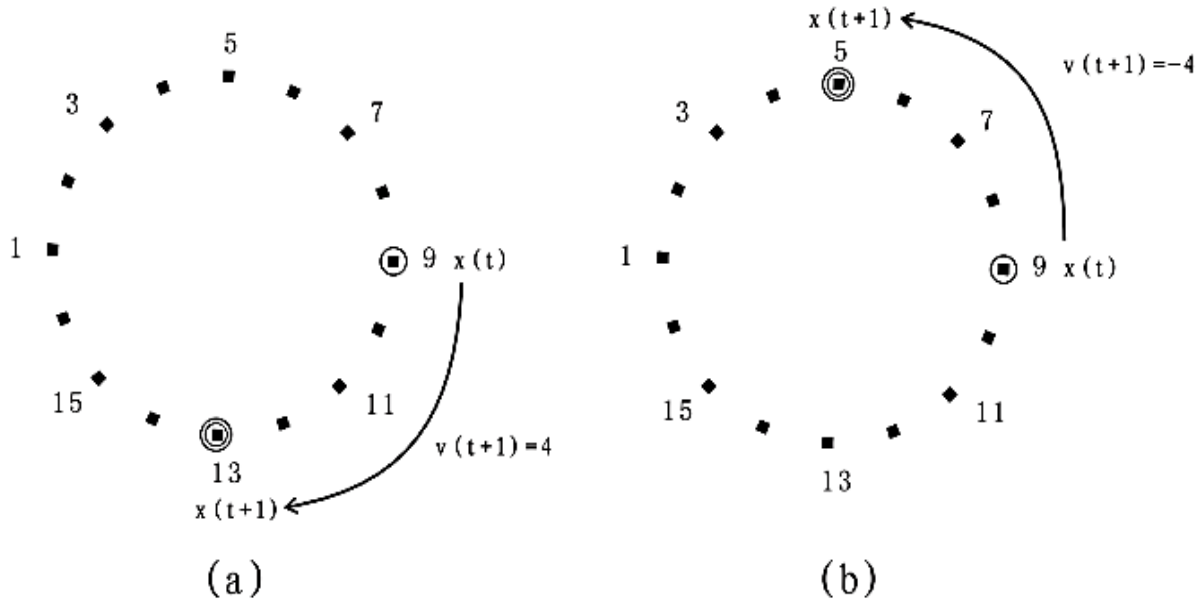

图 3.1 （a）当 v>0 时，点 x 顺时针；（b）当 v<0 时，点 x 逆时针

3.2.3 位置调整

多边形近似是一个约束性优化问题，但是 PSO 算法可能会产生违反约束条件的解。在这些情况下，粒子会跑出可行解的空间，因此必须对这些情况进行约束性处理。尹邦严[11]提出了一种惩罚方法去强制性进行约束。当粒子离开了可行解空间，便对之附加一个跟随违背程度的负数。这种惩罚会使粒子回到可行解空间，但是惩罚函数很难去定义。轻微的惩罚可能还是有粒子离开可行解空间。为了克服这个缺陷，王斌[12]提出了染色体修复方案，利用传统的剪切和组合技术修复一些非法的染色体。当一个粒子离开可行解空间，组合解的近似误差就会超过可以接受的误差范围 ε 。以下几步可以让它回到可行解范围：

1) 顺时针方向遍历多边形，检查和调整每一个顶点，使得在弧上的两个相邻顶点的近似误差尽可能的小。
2) 如果最后的近似误差仍然超出了容忍范围，则使用王斌提出的染色体修复方案。

类似于遗传算法，PSO 有着超强的全局，弱小的局部搜索能力。为了提高 PSO 算法的局部搜索能力，许多研究者提出加入一个局部优化器进去 PSO 算法中，比如爬山算法。

3.2.4 染色体修复算法介绍

染色体修复算法背后的思想其实是用传统的分割与合并技术[13]。

1) 分割技术

传统的分割技术是一种非常简单的多边形逼近近似方法。一种从初始曲线的直线片段开始递归的方法。分割终止的约束函数一般是指定最终应该达到的顶点数目。每一次迭代中，一次分割操作将一个片段从指定的点被分割成两部分。这一个过程反复执行，一直到达到满足约束条件为止。

具体的分割过程是这样的：

假设曲线 C 已经被分成了 M 条弧$\widehat{P_{t1}P_{t2}}$，……，$\widehat{P_{tM-1}P_{tM}}$，$\widehat{P_{tM}P_{t1}}$。P_i表示曲线 C 中的第 i 个顶点，P_j表示多边形 V 中的顶点在曲线 C 中的位置序号。对多边形 V 进行分割的操作是：对于每一个点$P_i \in C$，如果可以找到$P_i \in \overline{P_jP_{j+1}}$，$\overline{P_jP_{j+1}}$是多边形相邻两顶点组成的线段，(i ≠ j 且 i ≠ j + 1)计算距离$\mathrm{D}(P_i) = distance(\mathrm{P_i}, \overline{P_jP_{j+1}})$，

$\mathrm{D}(P_i)$表示点P_i到直线段$\overline{P_jP_{j+1}}$的点线距离。找到一个点：近似多边形的直线与原曲线的顶点的距离的最大值：$\mathrm{D}(P_u)=max\mathrm{D}(P_i)$，$i\in\{1,\dots,M\}$。这样，原来的弧$\widehat{P_JP_{J+1}}$被分成了两部分$\widehat{P_JP_u}$和$\widehat{P_uP_{J+1}}$。然后把$P_u$加入到多边形片段中去。图 2.2 示意了分割技术的过程。

分割技术实际上就是用启发式算法的方式，把新的顶点加入到原来的多边形中，以减少近似误差。

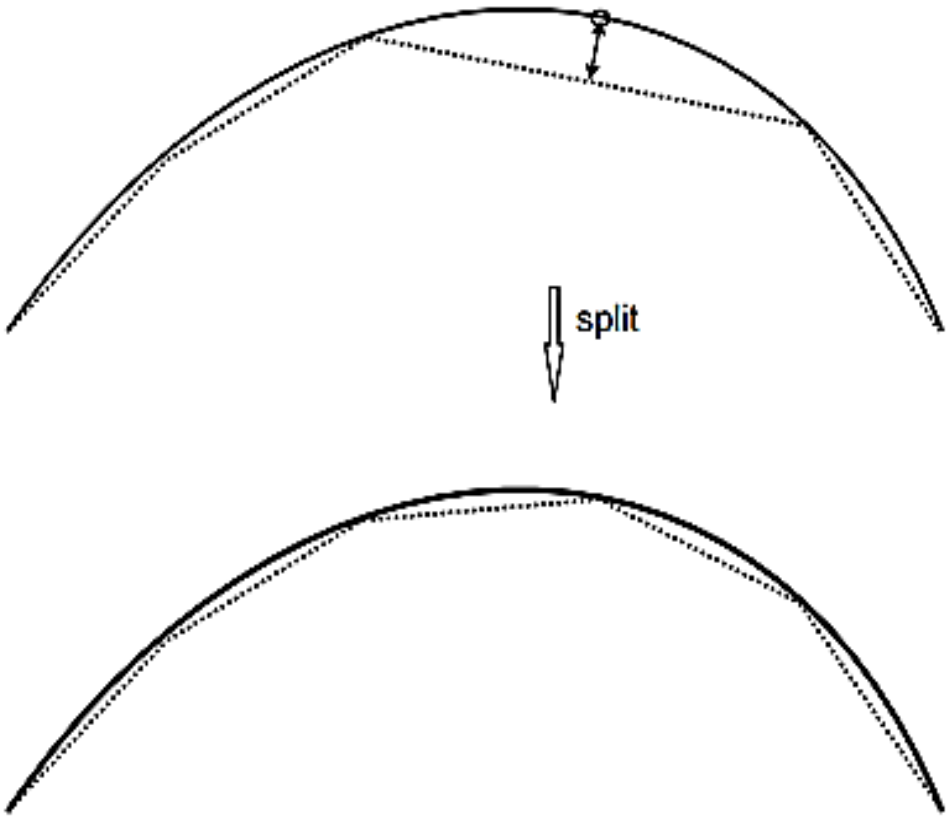

图 3.2 分割操作示意图

2) 合成技术

合成技术也是一种数字曲线多边形近似用到的简单方法。这是一种将初始曲线上的所有顶点都看成近似多边形的潜在顶点的递归方法。在每一次迭代中，合并操作是

把被选中的两个相连线段整合成一条线段。这个合并操作一直进行到满足约束条件为止。

假设多边形 V 有 M 条边$\overline{P_{t1}P_{t2}}$，……，$\overline{P_{tM-1}P_{tM}}$，$\overline{P_{tM}P_{t1}}$，$P_i$表示多边形 V 中的第 i 个顶点。多边形 V 上的合并操作被定义成：

对多边形 V 上每一个点P_i，计算距离$\mathrm{D}(P_i) = distance(\mathrm{P_i}, \overline{P_{i+1}P_{i-1}})$，$P_{i+1}$，$P_{i-1}$是和$\mathrm{P_i}$相邻的两点。$\mathrm{D}(P_m) = min\mathrm{D}(P_i)$，$i \in \{1, \dots, M\}$，找到所有距离中的最小值，将两条相邻线段合并成一条线段，及线段$\overline{P_{m+1}P_m}$和$\overline{P_mP_{m-1}}$合并成了$\overline{P_{m+1}P_{m-1}}$。图 2.3 示意了合并操作的过程。

合并技术可以说是分割操作的逆过程。两个技术相辅相成，达到最优的多边形近似优化的目的。

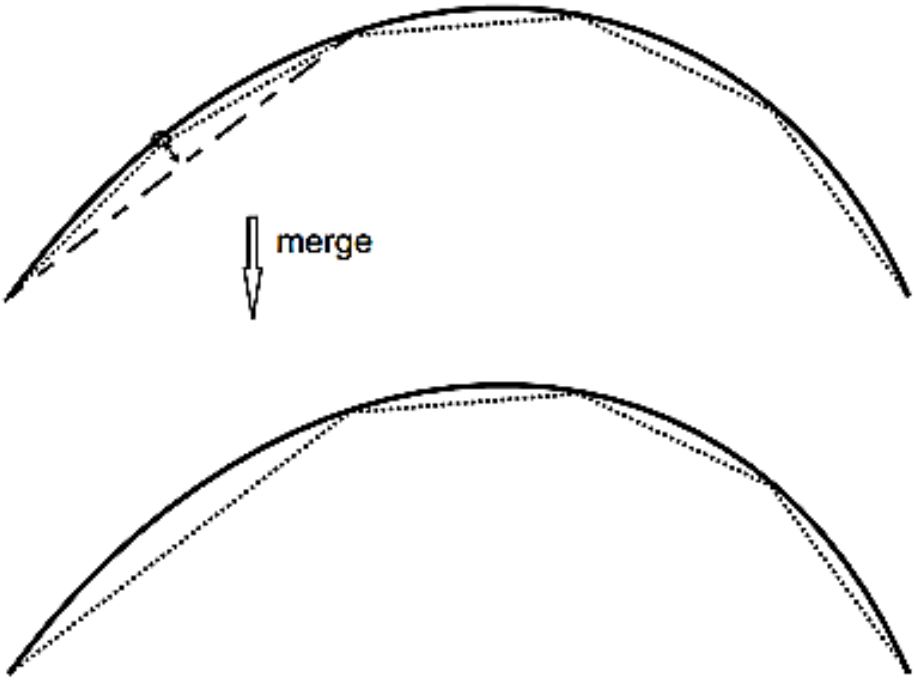

图 3.3 合并操作示意图

3.2.5 算法流

算法首先需要得到准确的图像轮廓的坐标信息，即曲线 C。这是进行后续处理和近似依赖的基础和保障。

然后指定计算中容许的最大平方和误差 ISE，定义 iPSO 算法中需要的常数参量。一般来说 ISE 定义成 10，迭代次数 T 定义为 100，等式（3.7）中的 c1 和 c2 都定义成 2，r1 和 r2 可以在每次迭代中随机，也可以预先正态分布随机定好，如果每次迭代中 r1 和 r2 都不变，收敛快一下，而每次迭代都随机 r1 和 r2 则收敛速度慢一些。算法中首先初始化随机一群粒子，假设每一个粒子就是一个最优解，从中选取个体极值，算法流的简要描述如下：

Algorithm: Integer PSO (iPSO) for polygonal approximation

Input：

Digital curve C

Error tolerance ε

PSO parameters T, c_1, c_2, r_1 and r_2

Output: Optimal polygon approximation P

Begin

Initialize

Position: Random{$X_1(0)$, $X_1(0)$……$X_1(0)$}

Velocity: Random V_i

Best position $P_i(0)=+\infty$

For T 1 to T do

For I 1 to k do

If particle x falls in the infeasible solution space

X = chromosome-repair

Evaluate the fitness value F for x

Update best previous position P_i

Update best position P_g

Local optimize P_g

For I 1 to k do

Update velocity and position

Return p = P_g

End

3.3 iPSO 算法处理 min-#结果

首先对一幅图片进行多次轮廓曲线的多边形近似算法优化实验，通过数据深度分析，纵向对比，总结实验结果，了解算法的一些特征属性。

然后对多幅不同的图片进行多次轮廓曲线多边形近似算法优化实验，通过实验数据，横向对比，总结实验结果，了解算法适应的对象的一些特征属性。

3.3.1 单幅图像纵向对比分析

对单幅图像进行多次实验，发现了算法的一些特征属性。现随机抽取一次 iPSO 算法处理结果，对实验结果加以总结和分析说明。实验结果如表 3.1 所示：算法优化迭代的次数是 100，种群的大小是 20，序号是指种群中粒子的索引序号，M 代表每个粒子的轮廓近似多边形的顶点数，ISE 表示每个粒子代表的解的平方误差和，原轮廓曲线 C 的顶点数是 700。

表格 3.1 iPSO 算法中顶点数 M 与平方误差和 ISE 统计表

序号	1	2	3	4	5	6	7	8	9	10
M	314	305	344	290	322	277	259	301	317	324
ISE	9.479 52699	9.479 52701	9.479 52704	9.479 52700	9.479 52702	9.550 92892	9.697 66862	9.953 12897	9.479 52700	9.697 66861
序号	11	12	13	14	15	16	17	18	19	20
M	205	284	329	167	294	298	329	324	142	320
ISE	9.550 92885	9.622 33076	9.479 52704	9.693 73263	9.479 52700	9.697 66859	9.479 52702	9.479 52700	9.765 13450	9.479 52703

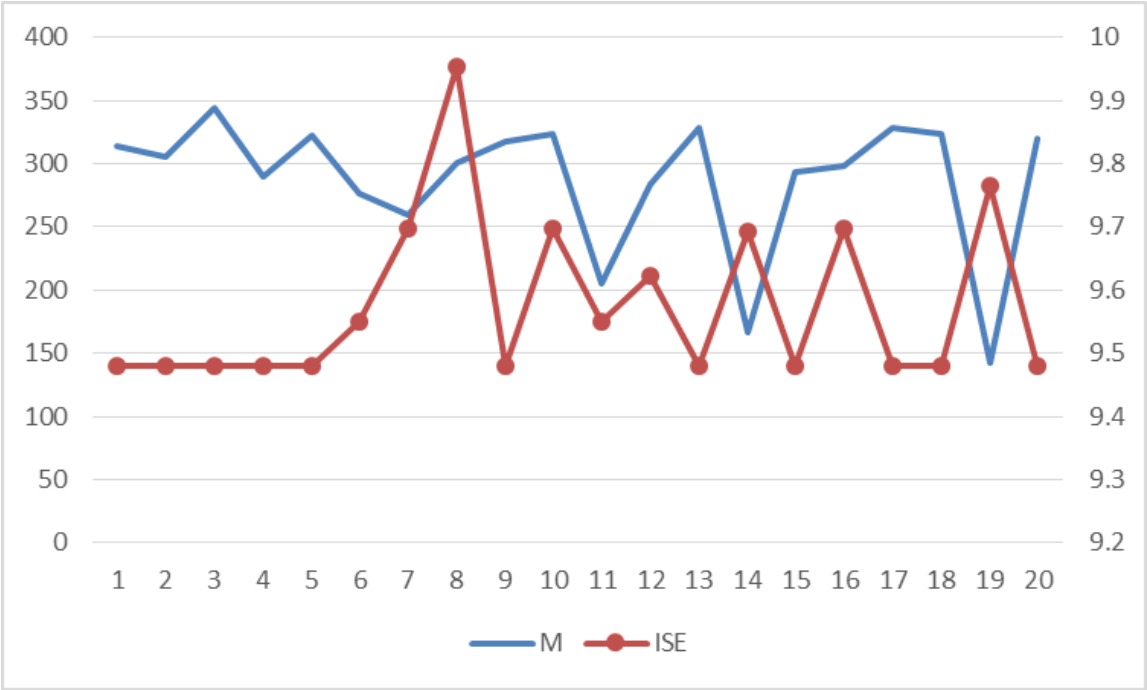

图 3.4 iPSO 算法中顶点数 M 折线图与平方误差积分 ISE 折线图

分析实验数据可以看出：因为初始化种群，其中粒子的位置是随机产生的，所以 iPSO 算法的结果具有随机性。另外粒子的速度也是随机的，因而粒子移动具有随机性，局部优化不够的可能性大大增加。优化后的顶点数 M 和平方误差 ISE 并没有明显的正相关的关系。比如当 iPSO 算法的适应度函数取最小值时，此时粒子的序号为 19，M=142，ISE=9.765135。随机抽取一组误差更大的数据进行对比：粒子的序号为 8，M=301，ISE=9.95312897。两者对比，误差大的一组多边形近似优化的程度明显更好一些，而平方误差 ISE 小一些的，优化的结果中却丢失了很多关键点。结果如图 3.5 和图 3.6 所示。

从折线图、实验数据、实验结果轮廓图分别对比可以看出，iPSO 算法结果具有随机性，不稳定性。但是当进一步约束误差范围，例如任意选取一个误差小于 9.5 的图进行对比，结果具有一定的稳定性，如图 3.7 所示。所以稳定性可以通过误差来控制。

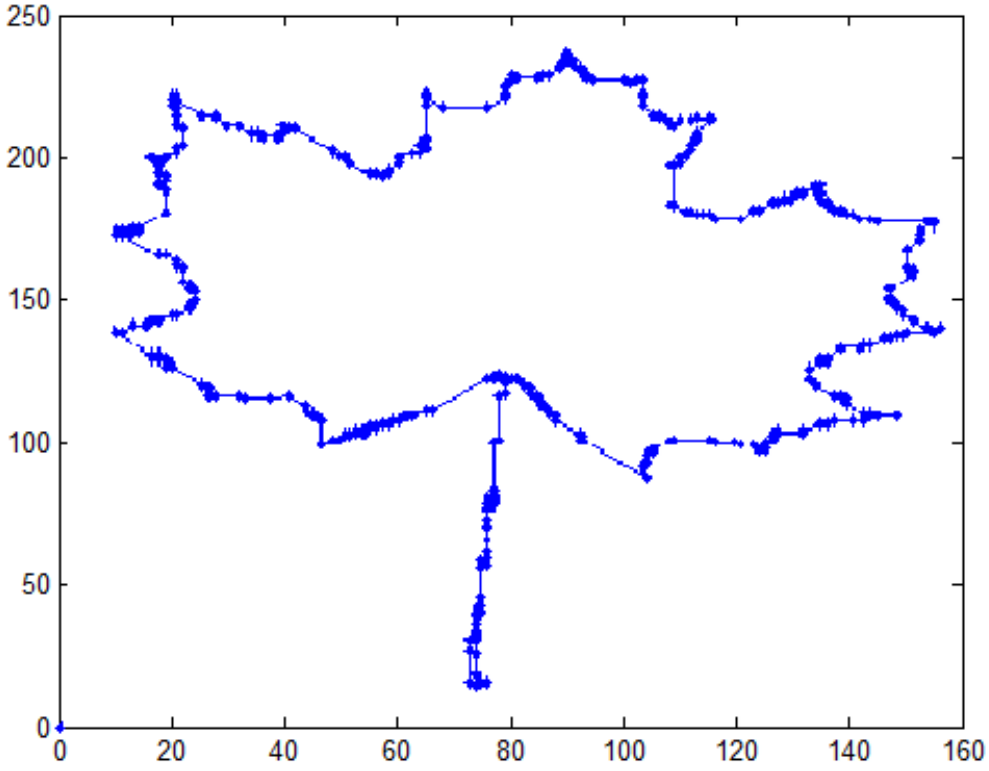

图 3.5　M=301，ISE=9.95312897

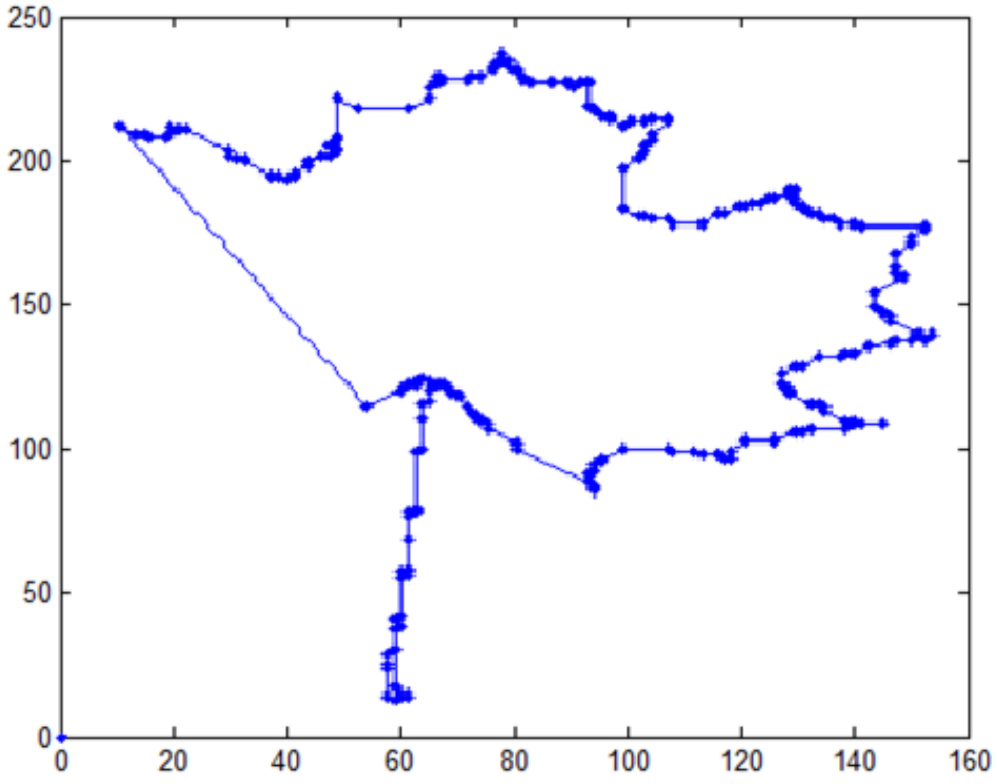

图 3.6　M=142，ISE=9.765135

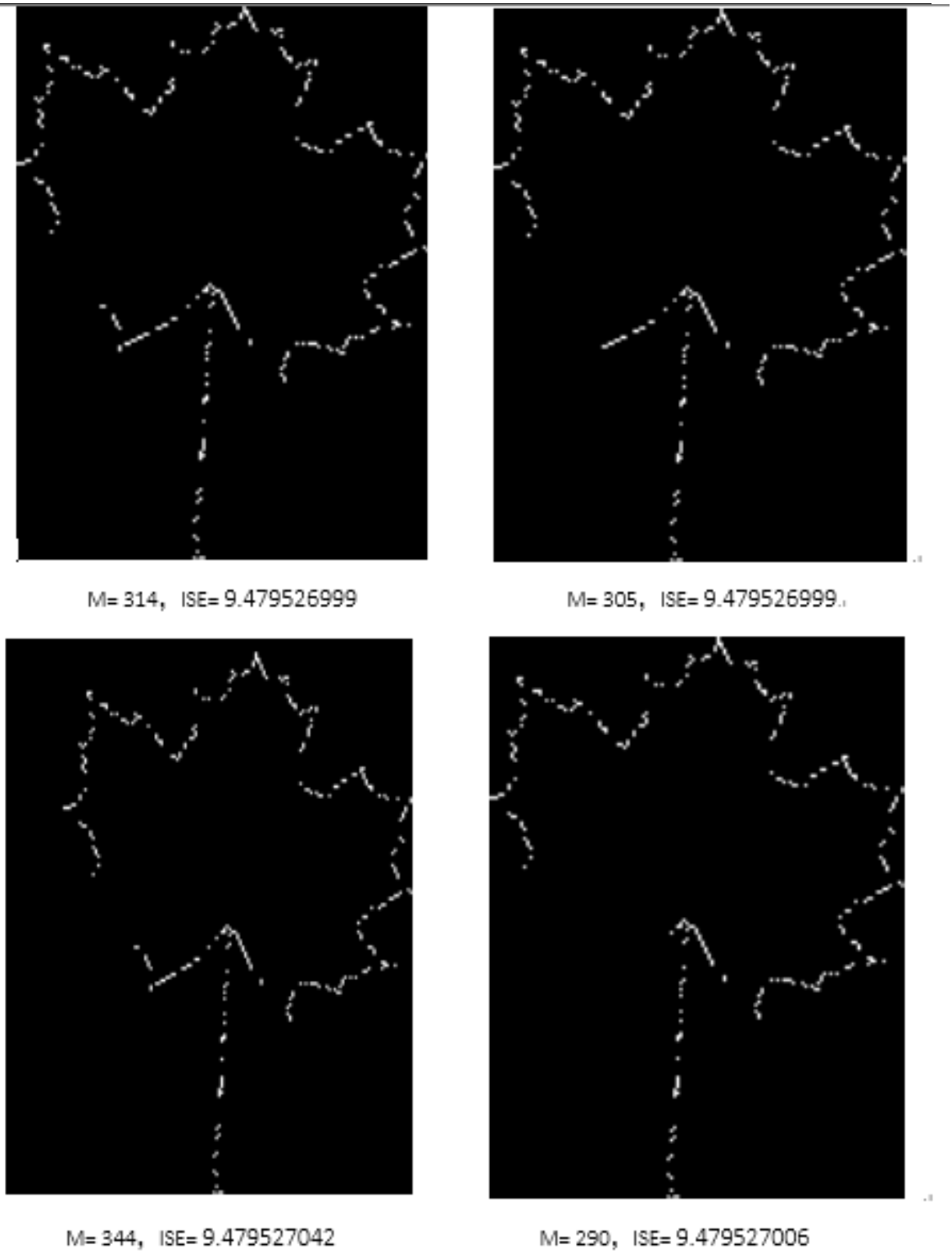

图 3.7　实验中任意选取 ISE 小于 9.5 的轮廓顶点散点图

3.3.2 多幅图像横向对比分析

图 3.8 原始图像-飞机

图 3.9 M=546; ISE=9.47952698126268

图 3.10 原始图像-汽车

图 3.11 初步的二值图像

图 3.12 M=371， ISE=9.47952698126268

图 1.13 茶壶

图 3.14 二值图像

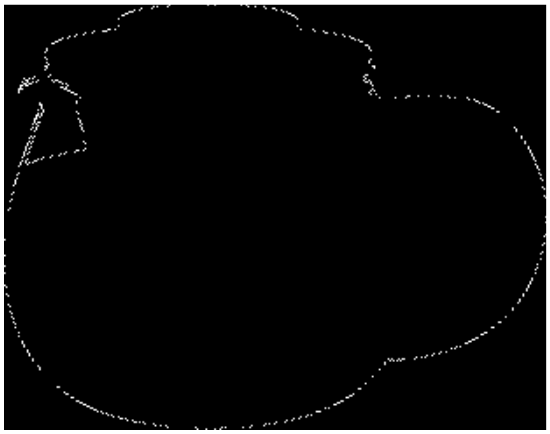

图 3.15 M=510; ISE=9.47952707416525

图 3.16 原始图像-铁塔

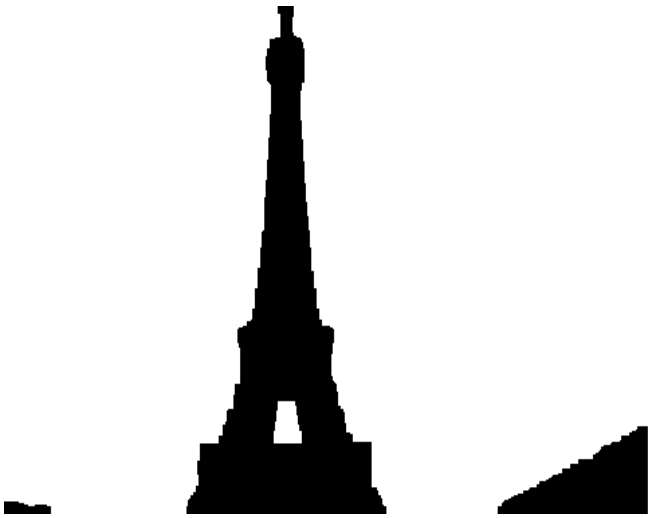

图 3.17 二值图像

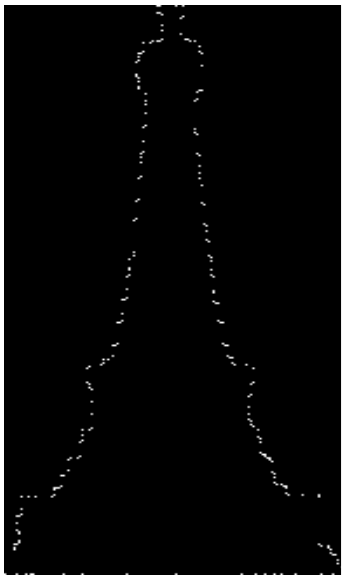

图 3.18 M=234; ISE=9.51802315577175

第四章 关键点检测法与 min-ε 问题

4.1 关键点检测算法原理

图像的特征点检测主要可以分为两类：一类是直接对灰度图像提取特征点，另一类首先对灰度图像进行分割，提取对象的边缘，然后依据轮廓图像的信息提取特征点。

根据实际问题的不同，人们提出了不同的关键点提取法。Sklansky 等人采用最小周长（MPP）来决定关键点[14]。Pavlidis 采用牛顿迭代法，用整体误差作为代价函数，来确定曲线的关键点。还有面积差，线段差和各种描绘子来进行关键点检测。许多属性之间进行排列组合，产生了许多新的方法来进行关键点检测。

轮廓曲线的关键点检测有着丰富的研究内容，研究的热点问题就是多边形逼近算法。当多边形达到误差小，或者顶点数也很少时，依然保留在研究对象中的点就是相对重要一些的点。

本文用关键点检测法对轮廓曲线进行多边形近似的优化。首先是依赖于图像分割和轮廓提取的准确性。然后，本文的研究结果才可以得以保证。一条曲线的描述，集中体现在它的关键点上。我们进行关键点检测，目的就是想以最少的描述来勾勒出对象最主要的信息。

类似于进化论的物竞天择，关键点检测法也是一种基于淘汰机制的算法。当给定一条由 N 个点组成的轮廓曲线 C 时，每一个顶点相对于曲线 C 的重要性是不一样的。

对于 min-ε 问题，当指定了一个近似多边形的顶点数时，我们按重要性从小到大从曲线 C 中淘汰掉重要性小的点，以达到误差的最小化。因为每一次淘汰的点，理论上来说都是重要性最小，也就是误差最小的点，所以当近似多边形的顶点数达到了指定的顶点数时，此时得到的多边形与原来的误差一定是最小的。

对于近似多边形预先设定的顶点数 M 大小的选取，实际过程中并没有特别的要求。一般根据图像轮廓曲线的复杂性而定。轮廓简单的可以将 M 设定小一些，轮廓曲线复杂的，适当将 M 设定大一些。这样可以保证简洁性和准确性之间的平衡。

4.2 本文关键点定义

数字图像处理中，图像的许多属性都可以作为特征值。描绘子、亮度、灰度、链码、直方图、颜色矩、图像的能量值，熵值，傅里叶功率谱值等等都可以作为特征值。处理不同的问题，需要用不同的属性作为特征来进行研究。

处理轮廓的多边形近似，使近似多边形与原曲线的误差最小。作为轮廓最直接的属性之一应该就是轮廓的周长了。针对使轮廓的近似多边形的周长最短，已经有不少人设计出了实际的最短周长算法（MPP）。作者实验过 MPP 算法之后，发现了一名叫皮特的程序员找到了一种更为简单的寻找关键点的方法。

对于图像轮廓的多边形近似，单纯从形状上来看，对于角度接近 180 度平角的相邻的两边，可以把它当一条直线。理论上说就是相邻的两条边，当其角度非常接近平角，这两条线段可以近似成一条线段。两条线段的公共点就是重要性很小的点。

所以本文实验的关键点检测法中，关键点的重要性定义为与多边形某一顶点相连的两条线段所形成的角的角度，其取值范围是[0,π]。

如图 3.1 所示，A,B,C 分别表示相连线段的三个端点的坐标。重要性的值∂的公式表示用公式（3.3），其中 abs（）是绝对值函数，norm（）是范化函数，dir1’是 dir1 的转置矩阵：

$$dir1 = B - A \tag{4.1}$$

$$dir2 = C - B \tag{4.2}$$

$$\partial = abs(acos((dir1' * dir2)/norm(dir1) * norm(dir2)) * norm(dir2) \tag{4.3}$$

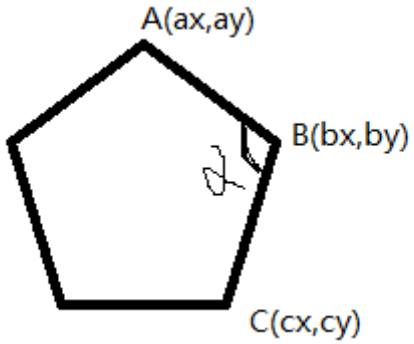

图 4.1 角度计算公式示意图

4.3 关键点算法处理 min- ε 问题结果

4.3.1 单幅图像纵向对比分析

对一幅图像进行多次轮廓的多边形近似算法优化实验。依次设定近似多边形的顶点数从大到小依次改变，记录实验结果，纵向对比分析，了解算法的一些特征属性。

实验中，M 代表设定的近似多边形的顶点数。依次设定了 M 的值为 M=300，M=150，M=75，M=40，M=20，M=10。从图 4.2 到图 4.7，图形逐渐简化，近似度逐渐减低。

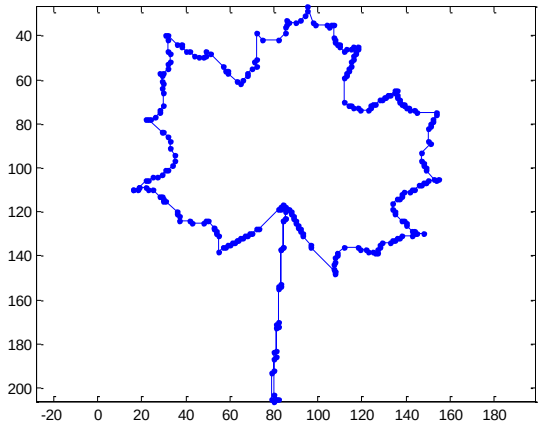

图 4.2 M=300

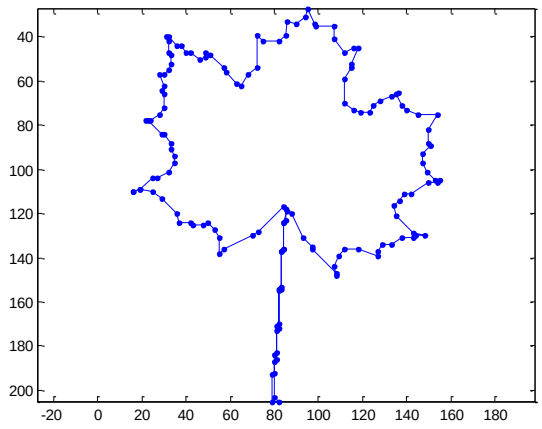

图 4.3 M=150

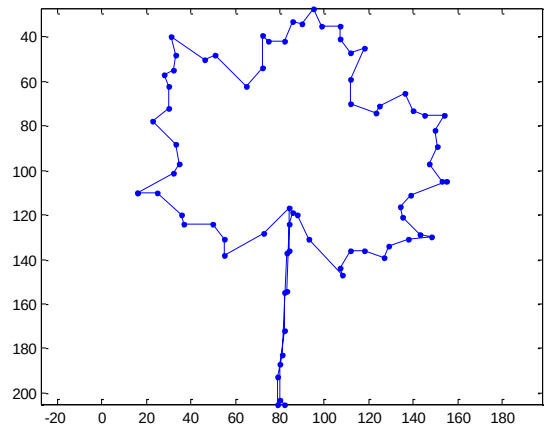

图 4.4 M=75

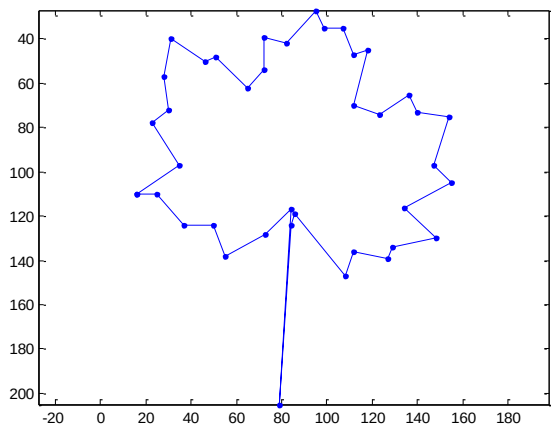

图 4.5 M=40

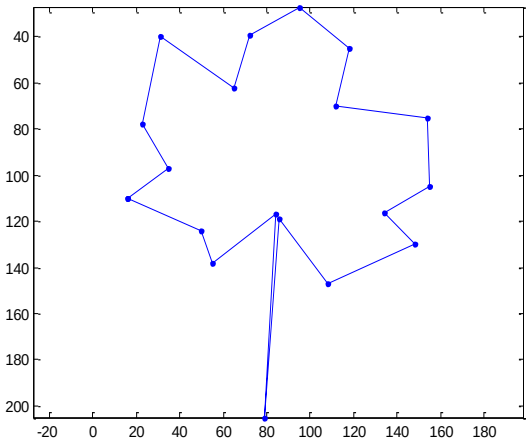

图 4.6 M=20

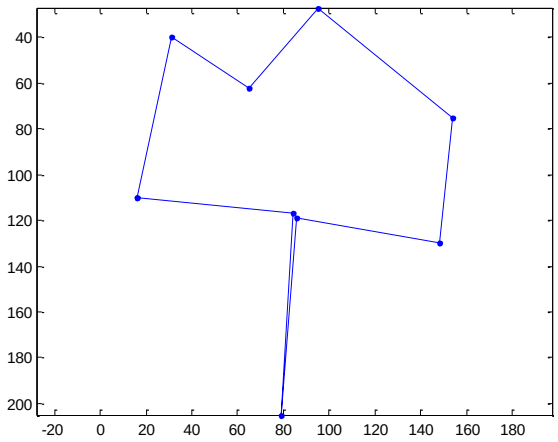

图 4.7 M=10

4.3.2 多幅图像横向对比分析

下列系列实验均取多边形顶点数 M=300，便于保证图像准确度。

1、原始图像为图 3.8，实验结果图如下:

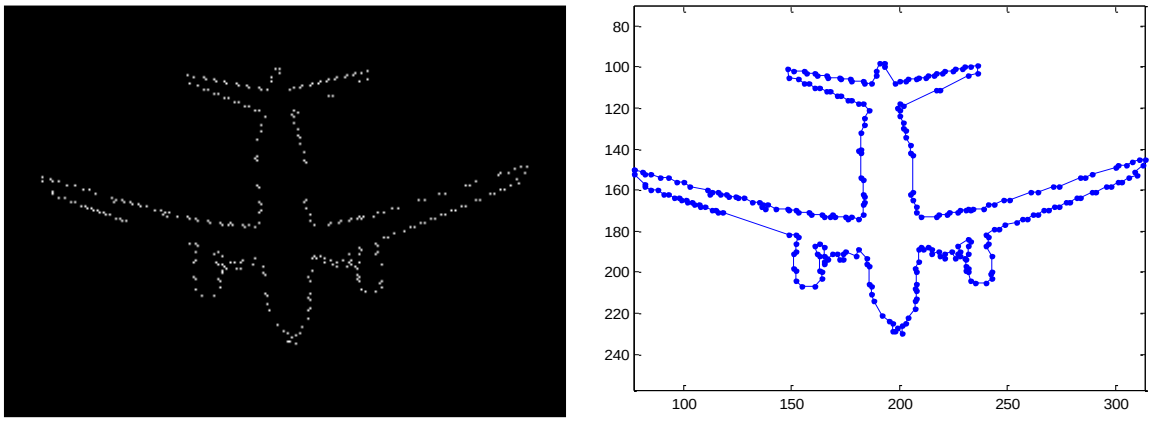

图 4.8 飞机形状 坐标散点图和多边形近似图像

2、原始图像为图 3.10，实验结果图如下：

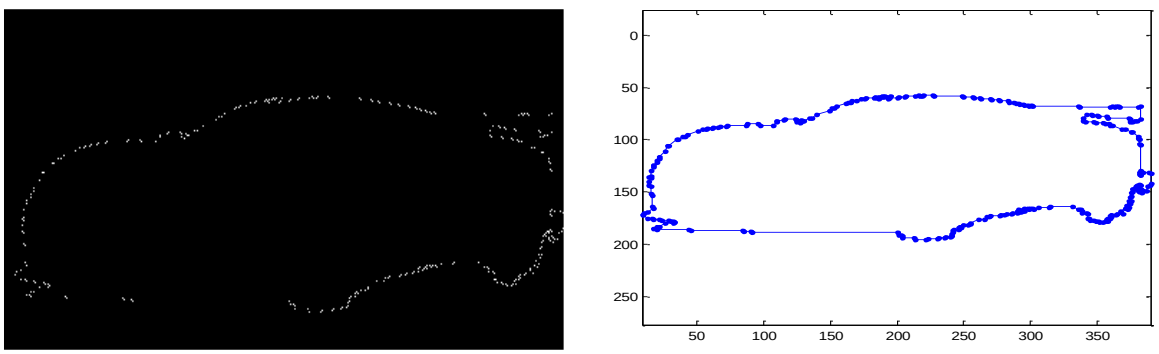

图 2.9 汽车形状 轮廓坐标散点图和多边形近似图像

3、原始图像为图 3.13，实验结果图如下：

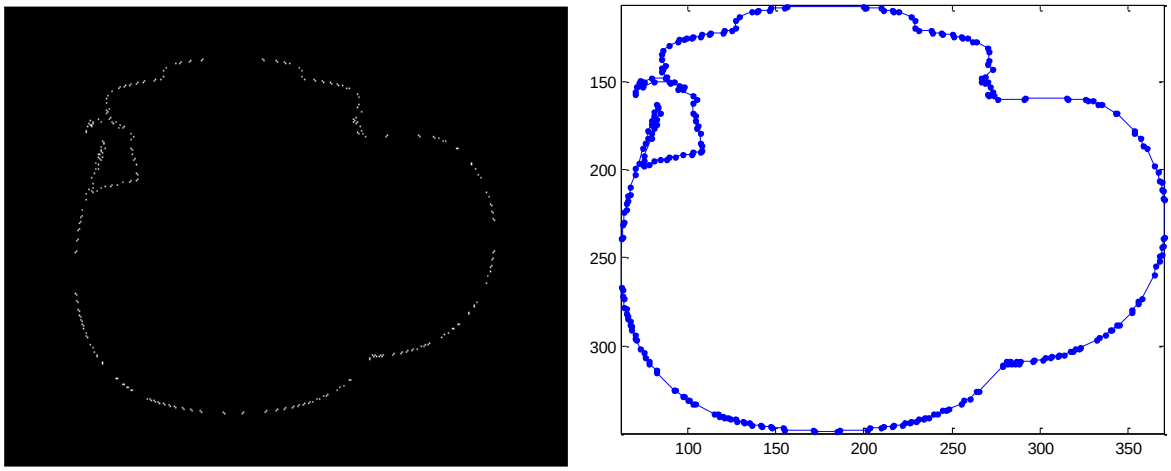

图 4.10 茶壶形状 轮廓坐标散点图和多边形近似图

4、原始图像为图 3.16，实验结果图如下：

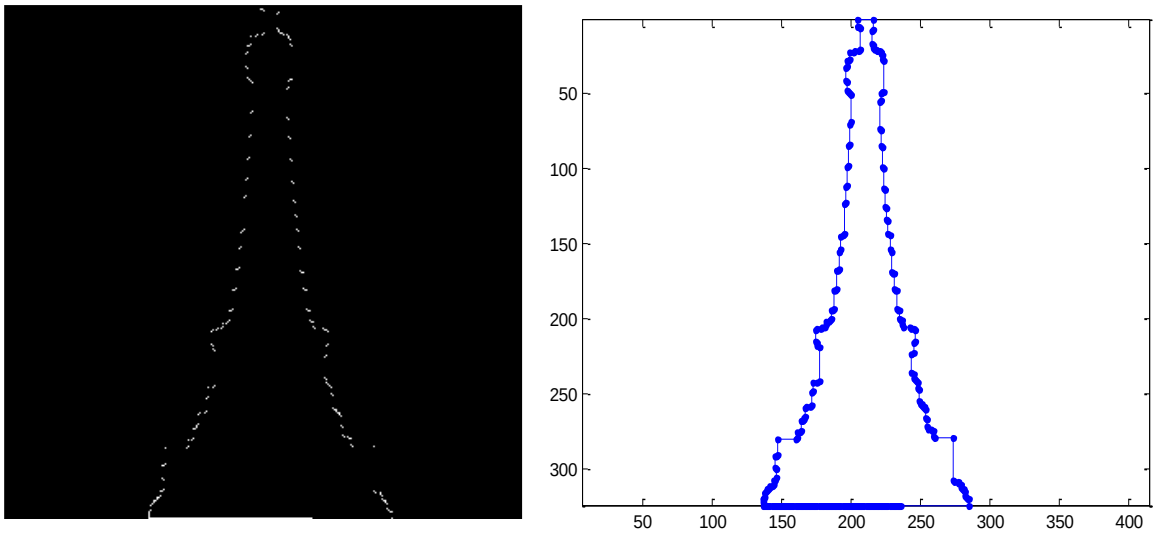

图 4.11 铁塔形状 轮廓坐标散点图和多边形近似图

对比以上几组图像，发现算法只要 M 取得适当值，算法结果有着很高的准确性和优化效果。

在实验过程中也更加深刻理解了提取轮廓的重要性，特别是对于背景复杂的图像，前期要做很多图像的灰度、腐蚀、膨胀，填补空洞等图像形态处理才可以得到理想的轮廓二值图像。

第五章 实验平台及结果评估

5.1 实验平台介绍

MATLAB 是美国 MathWorks 公司出品的商业数学软件，用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境，主要包括 MATLAB 和 Simulink 两大部分。

MATLAB 是 matrix&laboratory 两个词的组合，意为矩阵工厂（矩阵实验室）。是由美国 mathworks 公司发布的主要面对科学计算、可视化以及交互式程序设计的高科技计算环境。它将数值分析、矩阵计算、科学数据可视化以及非线性动态系统的建模和仿真等诸多强大功能集成在一个易于使用的视窗环境中，为科学研究、工程设计以及必须进行有效数值计算的众多科学领域提供了一种全面的解决方案，并在很大程度上摆脱了传统非交互式程序设计语言（如 C、Fortran）的编辑模式，代表了当今国际科学计算软件的先进水平。

MATLAB 和 Mathematica、Maple 并称为三大数学软件。它在数学类科技应用软件中在数值计算方面首屈一指。MATLAB 可以进行矩阵运算、绘制函数和数据、实现算法、创建用户界面、连接其他编程语言的程序等，主要应用于工程计算、控制设计、信号处理与通讯、图像处理、信号检测、金融建模设计与分析等领域。

MATLAB 的基本数据单位是矩阵，它的指令表达式与数学、工程中常用的形式十分相似，故用 MATLAB 来解算问题要比用 C，FORTRAN 等语言完成相同的事情简捷得多，并且 MATLAB 也吸收了像 Maple 等软件的优点，使 MATLAB 成为一个强大的数学软件。在新的版本中也加入了对 C，FORTRAN，C++，JAVA 的支持。

Matlab 是一个高级的矩阵/阵列语言，它包含控制语句、函数、数据结构、输入和输出和面向对象编程特点。用户可以在命令窗口中将输入语句与执行命令同步，也可以先编写好一个较大的复杂的应用程序（M 文件）后再一起运行。新版本的 MATLAB 语言是基于最为流行的 C++语言基础上的，因此语法特征与 C++语言极为相似，而且更加简单，更加符合科技人员对数学表达式的书写格式。使之更利于非计算机专业的科技人员使用。而且这种语言可移植性好、可拓展性极强，这也是 MATLAB 能够深入到科学研究及工程计算各个领域的重要原因。

MATLAB 是一个包含大量计算算法的集合。其拥有 600 多个工程中要用到的数学运算函数，可以方便的实现用户所需的各种计算功能。函数中所使用的算法都是科研和工程计算中的最新研究成果，而且经过了各种优化和容错处理。在通常情况下，可以用它来代替底层编程语言，如 C 和 C++ 。在计算要求相同的情况下，使用 MATLAB 的编程工作量会大大减少。MATLAB 的这些函数集包括从最简单最基本的函数到诸如矩阵，特征向量、快速傅立叶变换的复杂函数。函数所能解决的问题其大致包括矩阵运算和线性方程组的求解、微分方程及偏微分方程的组的求解、符号运算、傅立叶变换和数据的统计分析、工程中的优化问题、稀疏矩阵运算、复数的各种运算、三角函数和其他初等数学运算、多维数组操作以及建模动态仿真等。

MATLAB 自产生之日起就具有方便的数据可视化功能，以将向量和矩阵用图形表现出来，并且可以对图形进行标注和打印。高层次的作图包括二维和三维的可视化、图象处理、动画和表达式作图。可用于科学计算和工程绘图。新版本的 MATLAB 对整个图形处理功能作了很大的改进和完善，使它不仅在一般数据可视化软件都具有的功能（例如二维曲线和三维曲面的绘制和处理等）方面更加完善，而且对于一些其他软件所没有的功能（例如图形的光照处理、色度处理以及四维数据的表现等），MATLAB 同样表现了出色的处理能力。同时对一些特殊的可视化要求，例如图形对话等，MATLAB 也有相应的功能函数，保证了用户不同层次的要求。另外新版本的 MATLAB 还着重在图形用户界面（GUI）的制作上作了很大的改善，对这方面有特殊要求的用户也可以得到满足。

MATLAB 对许多专门的领域都开发了功能强大的模块集和工具箱。一般来说，它们都是由特定领域的专家开发的，用户可以直接使用工具箱学习、应用和评估不同的方法而不需要自己编写代码。领域，诸如数据采集、数据库接口、概率统计、样条拟合、优化算法、偏微分方程求解、神经网络、小波分析、信号处理、图像处理、系统辨识、控制系统设计、LMI 控制、鲁棒控制、模型预测、模糊逻辑、金融分析、地图工具、非线性控制设计、实时快速原型及半物理仿真、嵌入式系统开发、定点仿真、DSP 与通讯、电力系统仿真等，都在工具箱（Toolbox）家族中有了自己的一席之地。

5.2 结果评估

在图像轮廓的近似表示方法的实验中，针对 min-ε 和 min-#问题，分别进行了实验研究。

在整数粒子群优化（iPSO）算法的实验过程中，因为本文的算法中没有加入局部优化器，所以全局优化的结果不错，但是局部优化偶尔并不理想。这使得本来就是随机结果的实验结果中，部分结果细节丢失严重，一部分结果跑出了可行解空间。比如本文实验结果中存在叶子图像中一大部分轮廓是丢失的现象。

当算法结果是理想时，近似结果如图 5.1 右边，M=314，ISE=9.479。

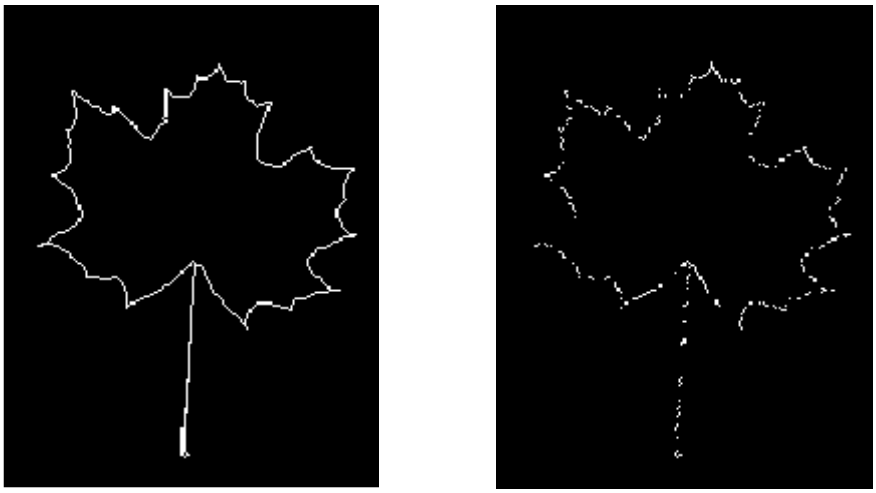

图 5.1 原始轮廓（左边）和理想情况得到的多边形近似的顶点散点图（右边）

在实验过程中，误差偶尔较大，局部缺失信息严重，作者推测是不是因为事先规定的容忍平方误差 ISE 导致的。调整 ISE 继续进行实验验证。

对比一下用不同 ISE 得到的结果，指定 ISE=100 时,得到结果如图 5.2。

图 5.2 iPSO 算法结果，M=343，ISE=98.926

当指定 ISE=10，实际结果如图 5.3 和图 5.4

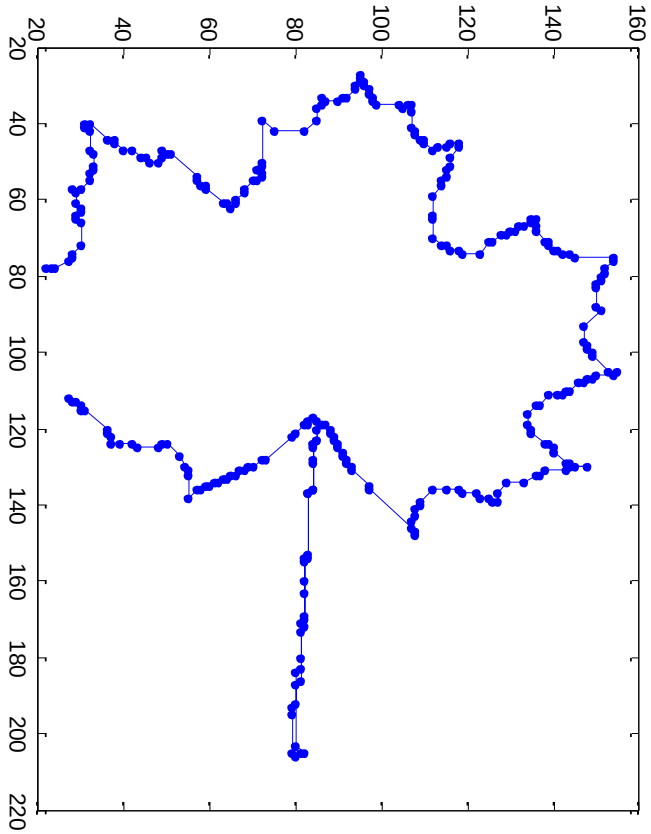

图 5.3 缩小指定 ISE 时，M=311，ISE=9.5509

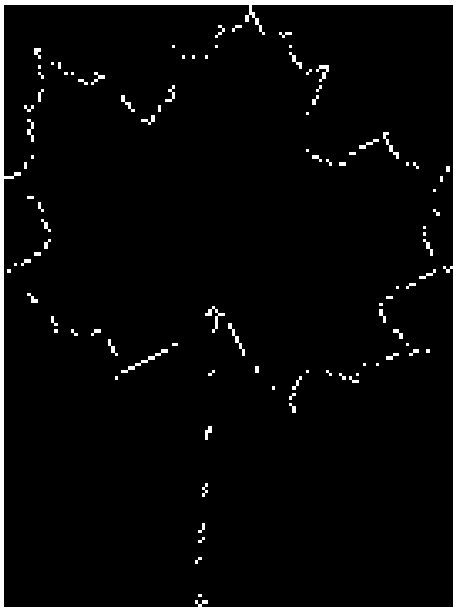

图 5.4，M=334，ISE=9.6977

图 5.3 中未显示的部分即 iPSO 算法中局部缺失的点。反复进行了多次实验，发现结果的不准确和指定的约束变量 ISE 并没有明显的关系。但是缩小 ISE 确实可以是结果更准确，但同时也可能得不到真正的最优解（顶点数最少的多边形近似）。所以单纯用 ISE 作为 iPSO 算法中的约束函数是不够的，应该使用多维的评价体系约束 iPSO 算法。

算法是基于全局优化算法改进的，有一些变异或者最优解的局部变化是不够的。这是这种算法的缺陷，个体最优解不是全局最优。为了改变这种算法的缺陷，可以加入局部的优化器进入算法中，把局部优化算法和全局优化算法进行融合。

在关键点检测法的实验中，实验结果还是比较稳定的。在指定合适的顶点数后，还是能检测到关键点，勾勒出图像的轮廓主要特征来。图 5.6 依次是原图，二值化图，初步轮廓图，和经过关键点检测后的多边形近似特征图。

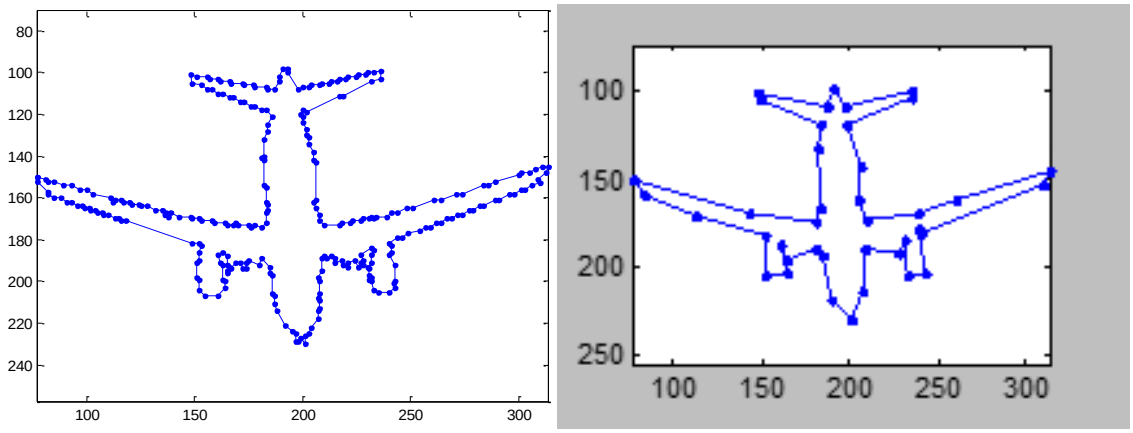

图 5.5 关键点检测法结果图 （左边）M=300，(右边) M=40

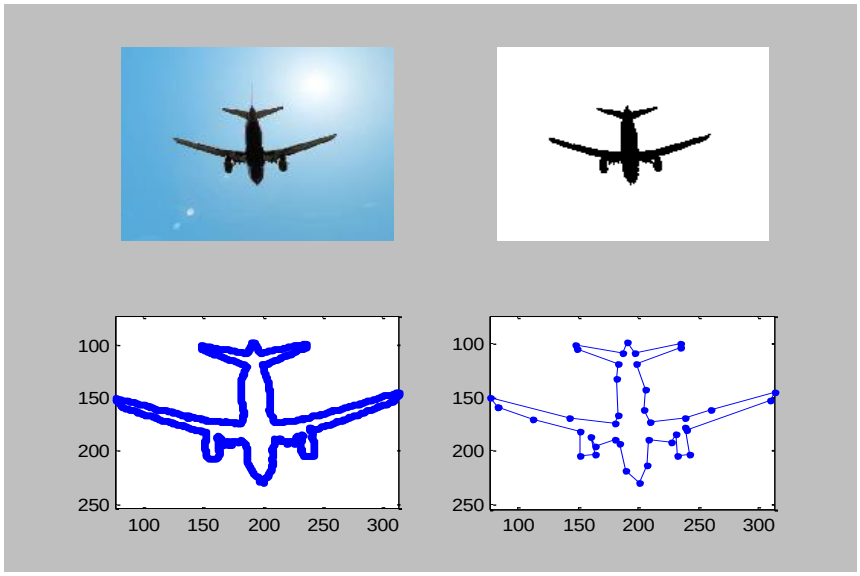

图 5.6 关键点检测法结果对比图

第六章 总结与设想

本文首先分析了图像轮廓的近似表示方法研究与应用的现状与发展趋势，并分析了目前主要的图像轮廓多边形近似算法。对整数粒子群优化（iPSO）算法进行了详细的介绍，该方法属于进化算法的一种，和模拟退火算法相似，它也是从随机解出发，通过迭代寻找最优解，它也是通过适应度来评价解的品质，但它比遗传算法规则更为简单，它没有遗传算法的“交叉”和“变异”操作，通过追随当前搜索到的最优值来寻找全局最优，实现容易、精度高、收敛快。但是在实际操作过程中，该算法的解会在速度和位置更新后，很容易的就跑出了可行解的空间。尽管在算法实现过程中进行了各种约束函数的检测规范，加入了分割和合并技术，但是该算法的解还是会有局部优化不足的缺陷。

全局优化算法更适合解决 min-#问题，也适合解决图形轮廓复杂，曲折多的曲线的简化。相比之下，局部优化更适合解决简单一点的 min-ε 问题，也可以解决顶点不是很多的多边形。

本文实验的多边形内角度数作为特征值的关键点检测法来处理 min-ε 问题，枫叶的边缘很复杂，锯齿形状的边很多，这个时候提前指定的顶点数并不好把握应该选择多大。但是当处理轮廓曲线不复杂的图形时，该算法还是有更加高效的优点，取的点数并不影响图形结果准确度。例如飞机的这种具有大直线轮廓的图案。

在实验过程中，作者也发现了 MATLAB 软件本身的优点和不足。MATLAB 有一些库函数可以直接进行图像处理，对于图像处理是很方便。但是 MATLAB 的循环机制和存储数据形式也导致了其在反复迭代过程中的过程耗时极长。MATLAB 的循环变量是提前都已经存入了循环计数向量中的。比如语句“for i= 1:3 i=i*2； end”输出的结果是 i=[1 4 6]。在实验过程中，特别是 iPSO 算法的实验过程中，一次实验过程足足需要五六分钟。已经不能用运行时间来估算该算法的效率了。

在经过对最小顶点数和最小误差数两种类型的问题研究后，虽然理论上一种算法可以解决这两种问题，但是针对不同情境，确实不同的方法会有不同的效果。而且许多方法融合一起使用，也许能取长补短得到更好地结果。比如在 iPSO 中加入约束函数后结果明显得到了提升。

参 考 文 献

[1] Wang B, Brown D, Zhang X, et al. Polygonal approximation using integer particle swarm optimization[J]. Information Sciences, 2014, 278:311-326.

[2] Horng J H. An adaptive smoothing app roach for fitting digital planar curves with line segments and circular arcs[J]. Pattern Recognition Letters, 2003, 24:565-577.

[3] E.J. Aguilera-Aguilera, A. Carmona-Poyato, F.J. Madrid-Cuevas, et al. Fast computation of optimal polygonal approximations of digital planar closed curves, Graphical Models,2016, 0:1-13.

[4] Kolesnikov A, Fränti P. Polygonal approximation of closed discrete curves[J]. Pattern Recognition, 2007, 40(4):1282-1293.

[5] Huang S C, Sun Y N. Polygonal approximation using genetic algorithms[J]. Pattern Recognition, 1999, 32(8):1409-1420.

[6] Ruberto C D, Morgera A. A New Algorithm for Polygonal Approximation Based on Ant Colony Optimization[C]// Image Analysis and Processing - ICIAP 2009, 15th International Conference Vietri sul Mare, Italy, September 8-11, 2009, Proceedings. 2009:633-641.

[7] Rafael C, Richard E, Steven L, Digital Image Processing Using MATLAB[M], Electronic Industries Press, 2005, 330-340.

[8] Oral L, Ozkan K. Suboptimal optimization method for dominant point detection[C]// Signal Processing and Communications Applications Conference. 2013:1-4.

[9] 史峰, 王辉, 郁磊等. MATLAB 智能算法 30 个案例分析, 北京航空航天大学出版社, 2011, 13(1):130:134

[10] J. Kennedy, R.C. Eberhart. A discrete binary version of the particle swarm optimization[C]// Proc. IEEE Intl. Conf. System, Man, Cybern, 1997, 4104–4108.

[11] Yin P Y. A discrete particle swarm algorithm for optimal polygonal approximation of digital curves[J]. Journal of Visual Communication & Image Representation, 2004, 15(2):241-260.

[12] Wang B, Shu H, Luo L. A genetic algorithm with chromosome-repairing for min−# and min− ε, polygonal approximation of digital curves[J]. Journal of Visual Communication & Image Representation, 2009, 20(1):45-56.

[13] 王斌, 施朝健. 多边形近似曲线的基于排序选择的拆分合并算法[J]. 计算机辅助设计与图形学学报, 2006, 18(8):1149-1154.

[14] Sklansky J, Chazin R L, Hansen B J. Minimum-Perimeter Polygons of Digitized Silhouettes[J]. IEEE Transactions on Computers, 1972, 21(3):260-268.

致　　谢

完成了本科生毕业论文，我的大学本科生涯即将结束。

本研究及学位论文是在我的导师周修庄老师的悉心指导下完成的。他严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深感染和激励着我。在此谨向周修庄老师致以诚挚的谢意和崇高的敬意。

感谢潘巍老师，我在专业课程里并没有学 MATLAB 这门语言和软件，是潘巍老师在大二时额外带领我们做科研项目时，挤出自己的业余时间来教会我们项目小组的同学熟练地使用这门特别的矩阵语言的。

感谢其他老师四年来的谆谆教诲，是你们把我带进计算机这扇大门，领着我走进了一个神奇的世界。这四年里，你们把一个电脑小白教成了一个具备一定的电脑基础的，具有一定专业知识的本科毕业生，更是教会了我许多做人的道理。

感谢大学四年来的同学们，我们一起度过这段青春岁月，快乐时光。

同时也感谢母校给了我们这么好的学习环境。

感谢我的爸爸妈妈，养育之恩！你们的幸福健康是我最大的心愿。

最后再一次感谢所有在毕业设计中曾经鼓励和帮助过我的良师益友，以及在论文中被我引用或者参考著作的作者。