基于区域正交化分割的平面点集凸包算法

李 可[1, 2] 高清维[1, 2] 卢一相[1, 2] 孙 冬[1, 2] 竺 德[1, 2]

摘 要 为解决实际工程应用中具有超大规模的平面点集的凸包计算问题, 提出了一种基于点集所在区域正交化分割的新算法. 利用点集几何结构的部分极点对平面点集进行正交化分割, 以获取不相干的点集子集簇, 再对所有点集子集分别计算其凸包极点, 最后合并极点得到凸包点集. 在不同层级的正交化分割过程中, 根据已知极点的信息, 逐层舍去对于凸包极点生成没有贡献的无效点, 进而提高算法运行效率. 在与目前常用凸包算法的对比实验中, 该算法处理超大规模的平面点集时稳定性高且速度更快.

关键词 平面点集, 凸包, 正交化分割, 并行算法

引用格式 李可, 高清维, 卢一相, 孙冬, 竺德. 基于区域正交化分割的平面点集凸包算法. 自动化学报, 2022, **48**(12): 2972−2980

DOI 10.16383/j.aas.c190590

A Convex Hull Algorithm for Plane Point Sets Based on Region Normalization Segmentation

LI Ke[1, 2] GAO Qing-Wei[1, 2] LU Yi-Xiang[1, 2] SUN Dong[1, 2] ZHU De[1, 2]

Abstract In order to solve the convex hull calculation problem of ultra-large scale planar point set in practical engineering applications, a new algorithm based on orthogonal segmentation of the region where the point set is located is designed. The partial point of the point set geometry is used to orthogonalize the plane point set to obtain the incoherent point set subset cluster, and then the convex hull poles are calculated for all the point set subsets, and finally the convex hull point set is obtained by combining the poles. In the process of orthogonalization and segmentation in different levels, according to the information of the existing convex hull poles, many of invalid points are discarded layer by layer, which improves the efficiency of the algorithm. In the comparison experiment with the commonly used convex hull algorithm, the proposed algorithm has high stability and speed when dealing with super large-scale planar point sets.

Key words Planar point set, convex hull, orthogonal segmentation, parallel algorithm

Citation Li Ke, Gao Qing-Wei, Lu Yi-Xiang, Sun Dong, Zhu De. A convex hull algorithm for plane point sets based on region normalization segmentation. *Acta Automatica Sinica*, 2022, **48**(12): 2972−2980

一组点的凸包是指包含这些点的最小凸多边形[1], 凸包问题是计算几何和图形学中最基础的问题之一. 平面凸包可以用来处理聚类分析[2−5], 围栏问题和城市规划问题等等. 在 GIS 应用[6] 中, 凸包可以快速获得区域地理的基本轮廓和最有效边界信息, 方便定位与数字化建模. 在模式识别学科的研究中, 凸包的应用对于优化数据结构和降低数据计算规模提供了新思路. 凸包算法[7−10] 在人工智能、人脸识别[11−12] 等前沿领域均有不同程度的应用.

常用的二维平面点集凸包算法有 Graham 算法[13]、Jarvis 步进算法[14] 和穷举算法等, 这些经典算法在处理少量数据时表现出色, 但是当平面点集数据量超过 10^6 数量级时效率较差, 所以不能推广到具有大规模数据集的工程中使用. 为解决实际工程应用中具有超大规模的平面点集的凸包计算问题, 出现了许多快速凸包算法的研究. 文献 [15] 中提出了一种提高排序效率的排序算法, 使得在理论上凸包算法的时间复杂度可以降低为排序算法的时间复杂度. 文献 [1] 提出将二维快速凸包算法与广义超越算法相结合, 当输入包含非极值点时, 算法运行更快, 并且占用内存更少, 但是当使用浮点算法时, 可能会导致严重的错误. 文献 [16−17] 提出了

收稿日期 2019-08-17 录用日期 2020-03-11
Manuscript received August 17, 2019; accepted March 11, 2020
国家自然科学基金 (61402004, 61370110), 安徽省自然科学基金 (2008085MF183, 2008085MF192) 资助
Supported by National Natural Science Foundation of China (61402004, 61370110) and Natural Science Foundation of Anhui Province (2008085MF183, 2008085MF192)
本文责任编委 张化光
Recommended by Associate Editor ZHANG Hua-Guang
1. 安徽大学电气工程与自动化学院 合肥 230601 2. 安徽大学计算智能与信号处理教育部重点实验室 合肥 230601
1. School of Electrical Engineering and Automation, Anhui University, Hefei 230601 2. Key Laboratory of Computational Intelligence and Signal Processing of Ministry of Education, Anhui University, Hefei 230601

基于 GPU 加速的方式来提高凸包问题求解的效率. 文献 [10] 提出了一种计算平面自由曲面精确凸包的高效实时算法, 该算法建立在圆弧构造的近似凸包上, 通过对圆弧进行简单的几何检验, 来确定近似的凸壳线段. 文献 [18] 描述了平面点集凸包的四种空间效率算法, 这些算法的输出与输入位于同一位置, 并且只使用少量的额外内存. 文献 [19] 提出了使用主成分分析法对点集预处理来改善凸包算法效率的递归算法, 但计算过程中进行多次坐标变换, 使得该算法较为复杂. 文献 [20]利用二维平面空间象限的几何和对称特性提出对称凸包算法, 对点集数较大的计算效果有明显的改善.

通过对平面点集所在区域进行正交化分割, 可以得到一种基于点集所在区域正交化分割的新算法, 该算法使得分割得到的点集子集的凸包极点生成过程可以在时间上同步进行. 对于大规模的点集数据, 分割的层次越深, 并行化的程度就越高, 运算的时间花销就会越低. 段落内容安排如下: 第 1 节详细介绍了该算法的原理, 包括点集区域正交化分割的规则和具体过程, 以及单个子集中凸包极点的生成步骤. 第 2 节在特殊情况和平均情况下分析了凸包极点生成算法的时间复杂度. 第 3 节给出一些实验数据, 比较了该算法与其他一些凸包算法的运行效率, 还测试了大规模平面点集数据计算凸包的时间花销. 第 4 节对全文进行了总结, 并提出进一步的研究方向.

1　算法原理

本文算法主要包括点集数据的正交化分割和凸包极点的生成 2 个部分. 1) 对平面点集进行正交化分割, 以获取不相干的点集子集簇. 实验中, 在点集数据的数量为 500 万时, 将正交化分割的层级设置为 5. 2) 对预处理后的点集子集序列进行操作. 首先使用初始凸包极点在序列中设置区间, 根据一定的判断准则, 检索序列子区间中的凸包极点, 并且在检索过程中抛弃掉对于凸包极点生成没有贡献的冗余点, 而检索得到的凸包极点又重新应用于区间设置, 直到所有的凸包极点都被检索到.

1.1　点集数据正交化分割

对于一个有限的二维点集, 由平面几何知识可知, 点集中的部分点可以构成一个凸包 (凸多边形) 将其余所有点包含在内. 一般情况下, 确定一个点是否为凸包极点, 需要保证该点相对于点集中其余所有点, 该点作为凸包极点是成立的. 然而这样确定凸包极点的过程是极其冗余的, 为此提出了对点集数据正交化分割的概念: 将二维点集沿竖直方向分割成一系列互不相关的子集, 每个子集中包含部分凸包极点, 在验证子集中的某个点是否为凸包极点时, 仅需保证该点相对于该子集中的所有点其作为凸包极点成立. 点集正交化分割后, 一个大的凸包求解问题就变成了一簇互不影响的小问题求解.

对点集数据的正交化分割, 可以保证在生成凸包极点的过程中点集数据不被重复使用, 避免计算冗余. 不同层级的分割过程中, 根据已知的凸包极点, 抛弃对于凸包极点生成没有贡献的无效点, 可以减少数据冗余给算法带来的无效操作.

平面点集 $P=\{p_i\,(x_i,\,y_i)\,,\,i=1,\,2,\,3,\,\cdots,\,n\}$, 对 P 进行遍历比较操作获取 4 个极值点: x 方向上的极小值点 p_j 和极大值点 p_k, y 方向上的极小值点 p_m 和极大值点 p_n, 易知这 4 个极值点均为凸包极点. 这 4 个极值点将点集所在平面区域分割成 5 个部分, 如图 1 所示, 点集的 4 个极值点分别位于矩形的 4 条边上, 边角上 4 个直角三角形区域内的点集数据相互独立, 而中间四边形区域内的点集数据是对于凸包极点生成没有贡献的无效点, 可以在算法执行中将其舍弃.

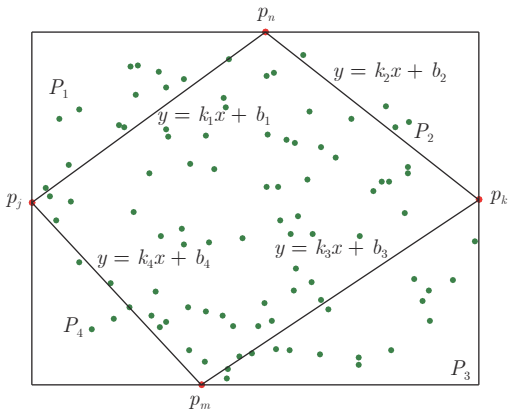


图 1　极值点结构图

Fig. 1　Structure diagram of extreme points

判断点集 P 中的点是否在由 4 个极值点构成的四边形中, 将位于四边形中的无效点从点集 P 中舍去, 具体判断方式是计算点 $p_i\,(x_i,\,y_i)$ 是否满足如下的不等式组:

$$\begin{cases} k_1\times x_i+b_1-y_i\geq 0\\ k_2\times x_i+b_2-y_i\geq 0\\ y_i-k_3\times x_i-b_3\geq 0\\ y_i-k_4\times x_i-b_4\geq 0 \end{cases} \tag{1}$$

式中, k_1 和 b_1 是点 p_j 和点 p_n 所在直线的斜率和截距, k_2 和 b_2 是点 p_n 和点 p_k 所在直线的斜率和截距, k_3 和 b_3 是点 p_k 和点 p_m 所在直线的斜率和截距, k_4

和 b_4 是点 p_m 和点 p_j 所在直线的斜率和截距. 舍去满足不等式组的无效点后, 剩余的点通过简单的区间极值判断, 构成新的点集子集 P_1、P_2、P_3 和 P_4, 分别对应图 1 矩形中的左上角、右上角、右下角和左下角所在直角三角形中的离散点数据.

以 P_1 为例, 对其进行第 2 层级的正交化分割, 如图 2 所示, P_1 中的 2 个极值点 p_j 和 p_n 所在直线 l_{jn} 将矩形区域分成 2 个部分, 下半部分空白区域内的点已经在第 1 次正交化分割的操作中舍去, 而上半部分的点可以继续分割. 计算 P_1 中的点到直线 l_{jn} 的距离, 获取距离直线最远的点 p_{jnl}, 该点也是凸包极点, 以 p_{jnl} 的 x 方向上的值 x_{jnl} 为分界点, 将 P_1 中的点分割成 P_{11} 和 P_{12} 两个子集, P_{11} 中的点分布在图 2 左侧下面的一个直角三角形区域中, 其 x 值小于 x_{jnl}, P_{12} 中的点分布在图 2 右侧上面的一个直角三角形区域中, 其 x 值大于 x_{jnl}. 由图 2 还可以看出, 分割过程中点 p_j、p_n 和 p_{jnl} 围成的钝角三角形区域内的无效点也要被舍去, 这个过程与上文中判断点是否在四边形中基本相同, 只需修改部分不等式即可实现.

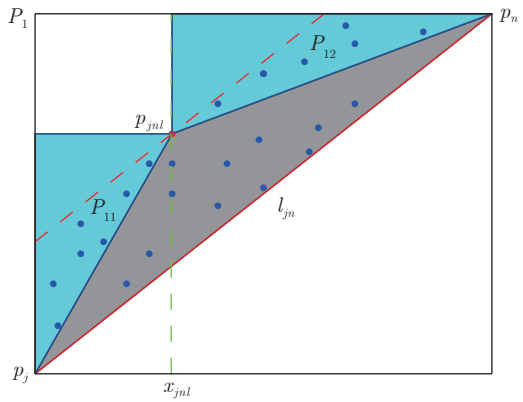


图 2 点集子集第二层级正交化分割图

Fig. 2 Diagram of the second level orthogonalization of the subset of point sets

得到 P_{11} 和 P_{12} 这个层级的全部点集子集后, 再对这些点集子集进行下一个层级的正交化分割. P_{11} 中对应的两极值点为 p_j 和 p_{jnl}, P_{12} 中对应的两极值点为 p_{jnl} 和 p_n, 其分割的步骤与对和 P_1 同一层级上的点集的操作过程完全相同.

点集 P 正交化分割操作全部完成后, 得到最小的点集子集簇 $\{P_i^{sub}, i=1, 2, 3, \cdots, m\}$. 对点集子集进行凸包极点求解前, 需要对其中的点进行 x 方向上的排序, 排序后得到子集序列 $\{S_i, i=1, 2, 3, \cdots, m\}$. 对于序列中 x 值相同的多个点, 由点集 P_1 或 P_3 得到的子集 P_i^{sub} 按 y 值从小到大排序, 由 P_2 或 P_4 得到的子集 P_i^{sub} 按 y 值从大到小排序, 使得每个子集中的首尾两点均为凸包极点. 在大量排序算法的研究中, 许多非基于比较的排序方式已经实现了具有线性时间复杂度的排序算法, 如计数排序, 桶排序等, 实验中采用桶排序对点集进行排序.

点集 P 所在平面区域正交化分割的流程如图 3 所示. 图 3 中仅给出前两个层级正交化分割的过程, 之后层级间的发展与第 2 层的分割形式完全相同. 由图 3 可以看出, 子集的分割操作 B 仅依赖于子集内部的参数, 即子集中直接继承得到的 2 个极值点和距离这两极值点所在直线最远的点, 与其他子集互不相关, 所以点集正交化分割的过程是并行分割过程. 在使用多核 CPU 并行处理的实现中, 首先在主进程中对点集 P 进行 A 操作, 分割出 P_1、P_2、P_3、P_4, 然后在主进程下开辟出 4 个线程, 将 P_1、P_2、P_3、P_4 同时放入到这 4 个线程中. 在每一个线程中, 对对应的子集执行 B 操作, 每个子集被分割成 2 个次子集, 当任意一个线程结束时, 在该线程下再开辟出两个线程并将两个次子集加入进去. 之后执行相同的操作, 直到某条进程中新开辟线程的次数达到预设值, 预设值即为正交化分割的层数. 在得到最次的子集后, 对其执行 C 操作获取最小凸包极点, C 操作即为第 2 节中凸包极点的生成操作. 在图 3中, 如果 P_1 对应的线程中操作已经完成, 而 P_2 对应的线程中操作还在进行中, P_1 对应线程下会立即开辟新的线程进行对 P_{11} 和 P_{12} 的操作, 所以对于一个点集 P, 正交化分割过程花费的时间取决于并行化流程所有进程中花费时间最多的一条进程.

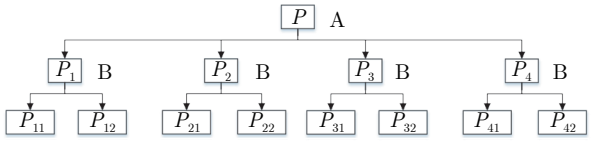


图 3 算法并行化流程图

Fig. 3 Parallel flow chart of the algorithm

1.2 凸包极点的生成

算法 1. 单个子集序列的凸包极点生成算法

预处理. 通过区域正交化分割获取子集序列 S_i.

初始化. 初始化序列 S_i.

1) 遍历起点设置为序列首点, 遍历终点设置为序列尾点, 当前遍历点设置为遍历起点, 最大距离点设置为遍历终点;

2) 最大距离值设置为 0;

3) 设置一个容器, 存储每次计算新得到的遍历终点, 容器最底部存入 1) 中的遍历终点;

遍历终点计算. 利用循环迭代的方式依次计算点集中的

凸包极点;

4) 获取遍历起点与遍历终点, 计算两点所在直线信息, 当前遍历点设置为遍历起点, 并将最大距离更新为 0;

5) 计算当前遍历点到直线的距离. 若为负, 将其从序列中抛弃; 若为正, 比较其与最大距离的值; 若其大于最大距离, 则将该值作为最大距离的值, 并更新该位置作为最大距离点;

6) 当前遍历点沿序列向下一点移动, 若当前遍历点等于遍历终点, 则进入步骤 7); 若不等于遍历终点, 则返回步骤 5);

7) 若最大距离值大于 0, 则将步骤 5) 最后记录的最大距离点位置作为遍历终点并进入步骤 4); 若小于或等于 0, 则将此时的遍历终点作为遍历起点, 并将容器最外面的一个遍历终点抛弃掉. 若此时容器不为空, 则将其最外面的点作为遍历终点并进入步骤 4); 若为空, 则结束.

8) 输出序列中剩余的点, 即为凸包极点.

算法 1 中点到直线距离的计算, 由于计算欧氏距离需要涉及到开方等复杂运算, 所以该算法使用点沿 y 轴方向到直线的距离 Δy. 设目标直线为 $y=kx+b$, 点 (x_0, y_0) 到直线在垂直方向的距离为 Δy, 这个距离乘以常值 $\left(1+k^2\right)^{-0.5}$ 可以得到实际的欧氏距离 ΔL. 由点集子集 P_1、P_2、P_3、P_4 分别得到的最小子集序列 S_i 在计算距离时有略微的不同, 其与凸包极点和直线的相对位置有关, 由图 1 可以看出, 子集 P_1 和 P_2 分割得到的最小子集序列中凸包极点存在于直线的上方, 所以 $\Delta y=y_0-(kx_0+b)$, 而子集 P_3 和 P_4 得到的最小子集序列中凸包极点存在于直线的下方, 所以 $\Delta y=(kx_0+b)-y_0$.

最小点集子集序列的获得是点集所在区域正交化分割得到的结果, 其凸包极点的生成相互独立, 且在实现上步骤相同, 仅有参数的差异, 因而使用并行算法对其处理, 可以极大地降低时间花销, 提高运行效率. 各个子集序列内的凸包极点均得到后, 加以简单合并操作就可以得到完整的凸包点集, 而且这个点集是有序的.

2　时间复杂度分析

对于任意一个点集子集, 由于不同的点集在平面上的分布均不相同, 凸包极点在点集中的分布也是随机的, 所以由算法本身很难计算出具有一般意义的时间复杂度公式, 但是当假定极点分布为一些极端情况时可以得到时间复杂度的上下界. 如图 4 点集遍历示意图, 每一条线段表示一个遍历区间, 半圆曲线表示遍历区间上的遍历过程, 线段上的点表示凸包极点. 点集序列被完整遍历的次数 κ 与点集序列中点的个数 n 的乘积 $\mathrm{O}(\kappa\cdot n)$ 即为本算法的时间复杂度, 下面计算 κ 的取值范围: 假设凸包极点的个数为 h, 当凸包极点集中在遍历终点且遍历过程见图 4 的左图, 点集序列被完整遍历的次数 $\kappa=h-1$ 最大; 当凸包极点集中在遍历起点且遍历过程见图 4 的右图, 点集序列被完整遍历的次数 $\kappa=2$ 最小, 所以该算法的时间复杂度上下限为 $\mathrm{O}(2n)\le\mathrm{O}(\kappa\cdot n)\le\mathrm{O}((h-1)\cdot n)$.

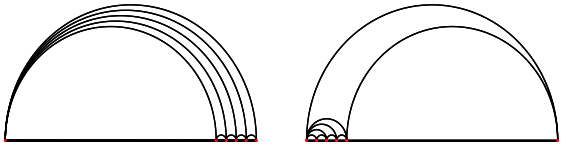


图 4　点集遍历示意图

Fig. 4　Diagram of point sets traversal

实际上只有很特殊的平面点集的凸包才会出现最好或最坏的情况, 因此还需要研究具有近似平均情况的时间复杂度. 如图 5 左图, 假设离散点在其分布区域内是均匀分布的, 且凸包极点在离散点中是均匀对称存在的, 则此时凸包极点的个数为 $2^\lambda+1$, 即任意两个凸包极点之间都会有或者都没有一个新的凸包极点, λ 是任意非负整数. 图 5 右图是极点个数 $h=5$ 时的遍历过程, 图上的数字对应遍历过程的顺序, 设其序列长度为 4 时, 按照遍历顺序, 其需要遍历的区间长度依次为 4 211 211, 这种遍历过程是最为对称的形式, 所以可以得到平均情况下的最好时间复杂度. 对于一个有序的点集序列, 通过归纳法可以得到时间复杂度的具体表达式:

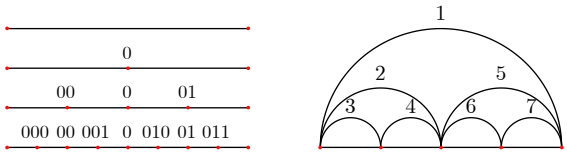


图 5　理想凸包极点分布图 ($h=2, 3, 5, 9$) 和 $h=5$ 的遍历过程

Fig. 5　Ideal convex hull pole map ($h=2, 3, 5, 9$) and a traversal process with $h=5$

1) 当点集中有 2 个凸包极点, 设其序列长度为 1, 则需遍历的区间总长度为 1;

2) 当点集中有 3 个凸包极点, 设其序列长度为 2, 则需遍历的区间总长度为 211;

3) 当点集中有 5 个凸包极点, 设其序列长度为 4, 则需遍历的区间总长度为 4 211 211;

4) 当点集中有 9 个凸包极点, 设其序列长度为 8, 则需遍历的区间总长度为 842 112 114 211 211;

5) 当点集中有 $2^\lambda+1$ 个凸包极点, 设其序列长度为 2^λ, 则需遍历的区间长度和为 $f\left(2^\lambda+1\right)=2^\lambda+$

$2\cdot f\left(2^{\lambda-1}+1\right)$.

令 $2^{\lambda}+1=h$, 可知, 极点个数为 h 时的时间复杂度为 $\mathrm{O}\left(f(h)/2^{\lambda})\cdot n\right)=\mathrm{O}((1+\log_2(h-1)\cdot n)$.

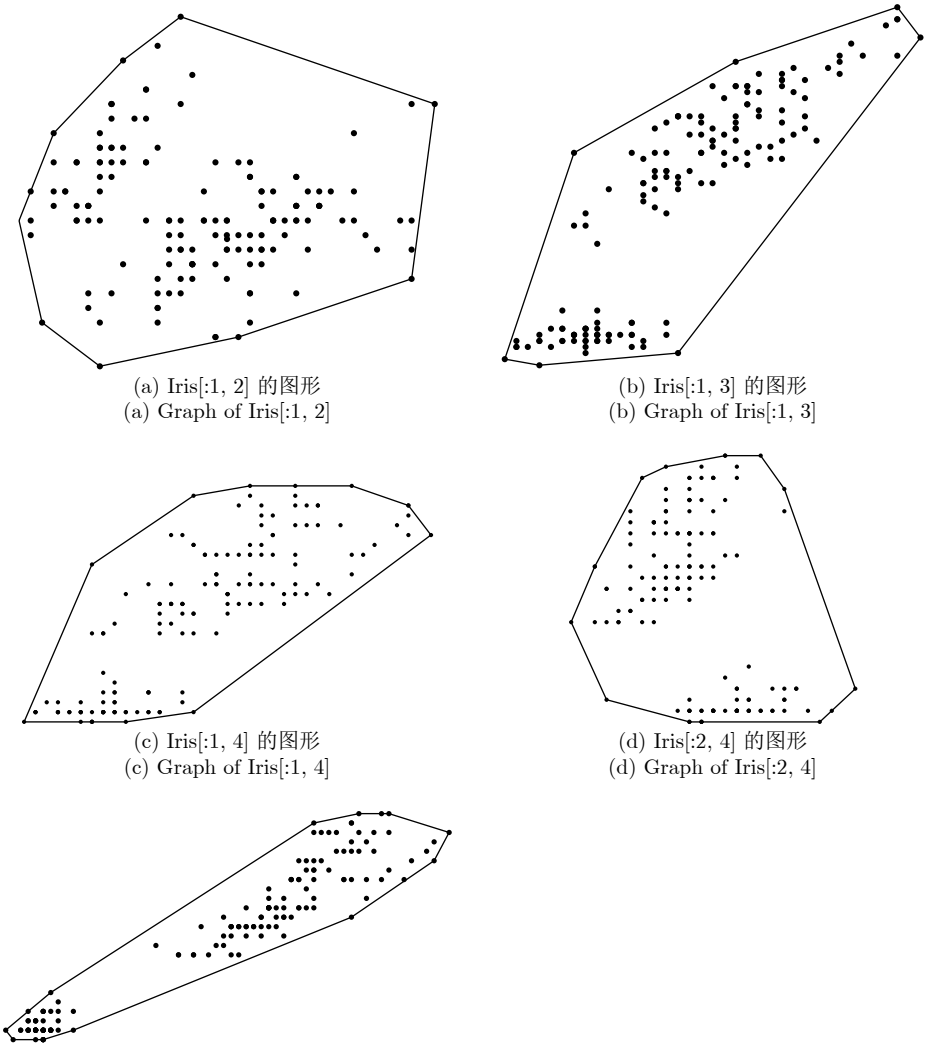
平均情况下最坏的遍历形式如图 4 左图, 其时间复杂度为 $\mathrm{O}\left((h^2+h-4)/(2(h-1))\cdot n\right)$.

凸包求解常用的 Graham 算法, 分治算法和快包法的时间复杂度为 $\mathrm{O}(n\log n)$, Jarvis 步进算法的时间复杂度为 $\mathrm{O}(h\cdot n)$, 通过分析不同的时间复杂度函数可以发现, 基于区域正交化分割算法的时间复杂度在最好和最坏的情况下均优于上述凸包算法的时间复杂度.

3 算法实验

为了验证算法的正确性, 使用该算法设计的程序对 Iris 数据集样本的任意两个特征构成的二维点数据集进行了测试. Iris 数据集的中文名是安德森鸢尾花卉数据集, 该数据集包含 150 个样本, 对应数据集的每行数据, 每行数据包含一个样本的 5 个特征: 花萼长度<1>、花萼宽度<2>、花瓣长度<3>、花瓣宽度<4>和样本的类别信息<5>, 所以 Iris 数据集是一个 150 行 5 列的二维表. 图 6 展示了该算法获得凸包极点构成的凸多边形, 图 6(a) 和图 6(d) 展示了凸包极点分布相对均匀的情形, 图 6(b) 和图 6(e) 展示了凸包极点分布较为特殊的情形.

为获得实验中不同规模点集对应的最佳分割层级数, 对规模在 5000 到 1000 万之间的点集进行了实验. 表 1 展示了不同规模点集在 5 层正交化分割下最小子集对应的点数, 其中每个规模的点集分割



(a) Iris[:1, 2] 的图形
(a) Graph of Iris[:1, 2]

(b) Iris[:1, 3] 的图形
(b) Graph of Iris[:1, 3]

(c) Iris[:1, 4] 的图形
(c) Graph of Iris[:1, 4]

(d) Iris[:2, 4] 的图形
(d) Graph of Iris[:2, 4]

(e) Iris[:3, 4] 的图形
(e) Graph of Iris[:3, 4]

图 6 Iris 数据集的部分凸包图形

Fig. 6 Graphs of partial convex hull of Iris data set

表 1　点集不同层级分割的子集点数

Table 1　The number of points in the subset corresponding to different levels of segmentation

层级	5000	1 万	5 万	10 万	20 万	50 万	100 万	200 万	500 万	800 万	1000 万
1	**438**	**873**	4543	8893	17782	44541	88532	178749	449484	715181	885358
2	65	123	**644**	1213	2404	6120	11861	24559	61273	97727	120068
3	0	16	88	**155**	**302**	**730**	1398	3054	7967	12473	15307
4	0	0	12	22	44	85	**175**	**403**	1032	1630	2053
5	0	0	1	4	5	11	25	55	**146**	**229**	**282**

数据均采用 30 组不同类型点集的计算结果平均值. 先前的实验中, 大量的测试显示当点集点数在 500 万左右时, 正交化层级设置为 5 可以得到最好的实验结果. 依照 500 万规模点集 5 层分割为参考, 分析其第 5 层与第 4 层关系可知, 最后一次分割得到的子集规模需要保证在 1000 以下. 表 1中, 加粗数字对应的层级数即为对应规模下点集的正交化分割层级设置参考. 纵观表 1 中的数据, 点集的规模每增加一个数量级, 对应的层级设置参考也相应的增加 1, 所以任意大小点集的正交化分割层级设置可以得到如下初步结论: 对于规模小于 1000 的点集, 不进行正交化分割, 1 万左右的点集层级设置为 1, 10 万左右的点集层级设置为 3, 100 万左右的点集层级设置为 4, 1000 万左右的点集层级设置为 5, 以此类推, 之后点集规模每增加 1 个数量级, 正交化分割层级增加 1. 由于这个结论是基于大量实验的数值结论, 而非理论推导, 所以在具体的实验中, 层级的选择可以适应性的 $\pm$ 1.

利用计算机随机生成的二维数据点点集, 对 Graham 算法、Jarvis 算法、文献 [19] 的算法和基于区域正交化分割的算法计算凸包极点的运行时间进行了对比测试. 为了使不同算法得到的结果具有可比性, 所以基于区域正交化分割的算法和对比所使用的 Graham 算法、Jarvis 步进算法以及文献 [19] 的算法均是由 C++ 编程实现, 并在 VS2017 软件上运行获取实验数据, 而且没有使用任何的外部加速. 所有的实验均在同一平台上运行, 使用的计算机配置为 Inter(R)i5-4460 CPU, 3.20 GHz, 4 GB 内存. 这些算法分别取点数量相同但类型不同的 10 组点集测试, 每组运行 10 次取平均值. 表 2 展示的是 10 组点集数据运行结果的平均值, 算法 1 是提出的算法只对点集进行正交化分割而没有并行处理的情况. 由表 2 可以看出, 当二维点集数据的数量大于 20 万时, 使用 Graham 算法和 Jarvis 算法已经无法在 1 秒内完成凸包问题的求解, 而没有并行化处理的算法 1 与文献 [19] 算法相比效果略差, 文献 [19] 算法虽然仍能够在 1 秒内完成求解, 但是与基于区域正交化分割的算法相比有一个数量级上的差距.

表 2　各算法运行时间 (s)

Table 2　Runtime for different algorithms (s)

点集点数	算法运行时间				
	Graham	Jarvis	文献 [19]	算法 1	本文算法
5000	0.266	0.022	0.013	0.021	0.008
10000	1.067	0.045	0.027	0.047	0.011
50000	26.999	0.236	0.054	0.076	0.019
100000	108.645	0.539	0.102	0.448	0.028
150000	244.662	0.891	0.329	0.624	0.039
200000	435.077	1.296	0.585	0.752	0.047

为测试基于区域正交化分割的算法对于超大规模点集的计算能力, 表 3 给出了平面点集数据数量级在 10^5 到 10^6 之间时, 该算法和文献 [19] 算法处理凸包问题的运行时间. 表 3 中凸包数量对应的两列数据左侧为文献 [19] 的数据, 右侧为本文算法的数据. 在表 3 数据中, 点集点数为 20 万、50 万、100 万、200 万和 500 万时, 该算法求解凸包问题花费的时间均小于文献 [19] 算法花费的时间; 在点集点数为 50 万时, 文献 [19] 的算法已经无法在 1 秒内完成, 而该算法只需要 0.1 秒的时间; 在点数为 200 万时, 文献 [19] 需要花费 7 秒的时间, 而该算法只需要 0.4 秒的时间; 在点数为 500 万时, 文献 [19] 需要 18.5 秒, 而本文算法可以在 1 秒内完成.

为进一步验证正交化分割凸包算法的有效性, 将其应用到 CT 肺图像的分割实验中. 在近几年的研究中, 对于如何从 CT 图像中准确分割出肺区域, 已有基于分形几何和最小凸包的肺区域分割方法[21], 该方法相较于传统方法更加准确且效率更高. 文献 [21] 方法的步骤是: 首先, 使用阈值法获得肺的初始区域, 并依据 CT 图像上下层结构相似的性质以及肋骨和骨组织的位置关系移除其中的气管、支气管、背景区域等非肺组织. 然后, 利用网格线将肺区域分割成互不重叠的小区域块, 计算各个含边界区块的分形维数并根据边界的全局和局部性质确定分形维数阈值, 通过该阈值选定需要修复的边界区块. 最后, 利用最小凸包方法修复肺边界.

表 3 不同数量级点集的运行时间 (s)

Table 3 Runtime comparison for different orders of magnitude point sets (s)

类别	20 万		50 万		100 万		200 万		500 万	
0	0.586	0.048	1.574	0.102	3.315	0.197	6.964	0.384	18.509	0.989
1	0.580	0.046	1.575	0.104	3.317	0.194	6.962	0.383	18.510	0.965
2	0.589	0.047	1.574	0.103	3.312	0.201	6.962	0.386	18.506	1.021
3	0.579	0.047	1.568	0.107	3.321	0.200	6.960	0.396	18.518	0.949
4	0.581	0.045	1.574	0.106	3.312	0.202	6.963	0.392	18.509	0.959
5	0.589	0.048	1.579	0.108	3.313	0.190	6.958	0.386	18.511	0.962
6	0.585	0.048	1.573	0.105	3.317	0.197	6.988	0.390	18.513	0.970
7	0.585	0.046	1.570	0.103	3.313	0.199	6.966	0.391	18.509	0.957
8	0.591	0.050	1.574	0.105	3.315	0.200	6.963	0.405	18.518	0.959
9	0.586	0.047	1.582	0.105	3.315	0.198	6.968	0.398	18.499	0.961
10	0.584	0.046	1.575	0.105	3.313	0.196	6.962	0.385	18.510	1.019

在文献 [21] 中, 对于由阈值选定的待修复的边界区块, 使用 Jarvis 步进算法作为最小凸包法来对其进行肺边界的修复. 实验时对待修复的边界区块图像进行双线性二次插值操作, 来增加区块中的肺阴影的点数, 并使用提出的算法作为最小凸包法修复肺边界, 而其余的操作均与文献 [21] 的操作相同. 图 7(a) 展示了一幅肺的原始 CT 图像, 图 7(b) 是使用最小凸包法修复肺边界操作前的预处理图, 此时已经移除了非肺组织并对图像进行了二值化操作, 图中肺内部的空洞区域是预处理过程中图像形

(a) 肺的原始 CT 图像
(a) Original CT image of the lung

(b) 预处理后的图像
(b) Preprocessed image

(c) 文献 [21] 恢复的图像
(c) Image recovered from reference [21]

(d) 本文算法恢复的图像
(d) Image recovered by this algorithm

图 7 肺图像的修复

Fig. 7 Restoration of lung image

态学腐蚀膨胀的结果. 图 7(c) 是由文献 [21] 修复得到的图像, 图 7(d) 是提出的算法修复得到的图像. 通过比较可以发现, 在肺图像的两侧外轮廓上 2 种方法修复的结果没有明显的区别, 但是在肺图像中间部位的边界上, 由文献 [21] 算法得到左侧肺图像仍有部分凹陷处未能修复, 在右侧肺的图像中甚至有一个严重的人工伪影, 图 7(c) 中深入到肺内部的一个细长的空白区域. 而使用基于区域正交化分割的算法设计的方法修复的肺图像并没有严重的人工痕迹, 而且肺的轮廓表现得更加光滑流畅. 在时间花费上, 虽然实验过程中对待修复的边界区块加入了插值处理, 但是花费时间仍然小于使用 Jarvis 步进算法作为最小凸包法的文献 [21] 所花费的时间.

4 结束语

对于大规模平面点集凸包问题的求解, 本文提出了基于点集所在区域正交化分割的凸包极点生成算法. 算法前期对点集数据所在区域进行正交化分割, 可以避免数据冗余和无效计算, 后期凸包极点生成的过程中, 通过不断地剔除掉点集中的非凸包极点, 可以提高程序的运行效率. 当点集数据的规模在 500 万时该算法仍然可以在 1 秒内完成凸包问题的求解, 这对于工程应用有着重要的意义. 由大量实验得到的正交化分割层级设置的结论, 也为该算法在实际问题中的应用提供了重要的参考. 一系列的数值实验结果表明, 该算法准确、高效、实用性强, 并且正交化分割的层级参数设置具备很高的鲁棒性.

References

1 Barber C B, Dobkin D P, Huhdanpaa H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 1996, **22**(4): 469−483

2 Kim Y, Kim J. Convex hull ensemble machine for regression and classification. *Knowledge and Information Systems*, 2004, **6**(6): 645−663

3 Ding S G, Nie X L, Qiao H, Zhang B. A fast algorithm of convex hull vertices selection for online classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2018, **29**(4): 792−806

4 Long P M, Servedio R A. Random classification noise defeats all convex potential boosters. *Machine Learning*, 2010, **78**(3): 287−304

5 Chau A L, Li X O, Yu W. Convex and concave hulls for classification with support vector machine. *Neurocomputing*, 2013, **122**: 198−209

6 Sadahiro Y. Number of polygons generated by map overlay: The case of convex polygons. *Transactions in Gis*, 2001, **5**(4): 345−353

7 Chazelle B. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 1993, **10**(4): 377−409

8 Yao A C. A lower bound to finding convex hulls. *Journal of the ACM*, 1981, **28**(4): 780−787

9 Akl S G, Toussaint G T. A fast convex hull algorithm. *Information Processing Letters*, 1978, **7**(5): 219−222

10 Kim Y J, Lee J, Kim M S, Elber G. Efficient convex hull computation for planar freeform curves. *Computers & Graphics*, 2011, **35**(3): 698−705

11 Park M, Park C W, Park M, Lee C. Algorithm for detecting human faces based on convex-hull. *Optics Express*, 2002, **10**(6): 274−279

12 Zhou X F, Jiang W H, Tian Y J, Shi Y. Kernel subclass convex hull sample selection method for SVM on face recognition. *Neurocomputing*, 2010, **73**(10−12): 2234−2246

13 Graham R L. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1972, **1**(4): 132−133

14 Jarvis R A. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 1973, **2**(1): 18−21

15 Andrew A M. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 1979, **9**(5): 216−219

16 Stein A, Geva E, El-Sana J. CudaHull: Fast parallel 3D convex hull on the GPU. *Computers & Graphics*, 2012, **36**(4): 265−271

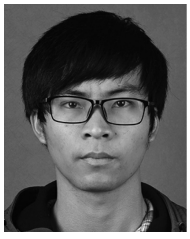
17 Tang M, Zhao J Y, Tong R F, Manocha D. GPU accelerated convex hull computation. *Computers & Graphics*, 2012, **36**(5): 498−506

18 Brönnimann H, Iacono J, Katajainen J, Morin P, Morrison J, Toussaint G. Space-efficient planar convex hull algorithms. *Theoretical Computer Science*, 2004, **321**(1): 25−40

19 Liu Bin, Wang Tao. An efficient convex hull algorithm for planar point set based on recursive method. *Acta Automatica Sinica*, 2012, **38**(8): 1375−1379
(刘斌, 王涛. 一种高效的平面点集凸包递归算法. 自动化学报, 2012, **38**(8): 1375−1379)

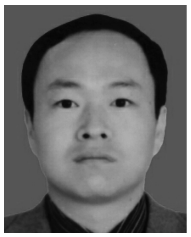
20 Beltran A, Mendoza S. SymmetricHull: A convex hull algorithm based on 2D geometry and symmetry. *IEEE Latin America Transactions*, 2018, **16**(8): 2289−2295

21 Feng Chang-Li, Zhang Jian-Xun, Liang Rui, Dai Yu, Cui Liang. A lung region segmentation method based on the fractal theory and the minimal convex hull method. *Journal of Tianjin University (Science and Technology)*, 2015, **48**(10): 937−946
(冯昌利, 张建勋, 梁睿, 代煜, 崔亮. 基于分形几何和最小凸包法的肺区域分割算法. 天津大学学报 (自然科学与工程技术版), 2015, **48**(10): 937−946)



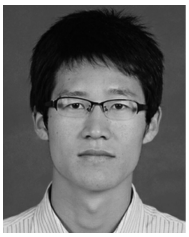
李 可 安徽大学电气工程与自动化学院硕士研究生. 主要研究方向为图像处理. E-mail: likea310@163.com
(**LI Ke** Master student at the School of Electrical Engineering and Automation, Anhui University. His main research interest is image processing.)

高清维 安徽大学电气工程与自动化学院教授. 主要研究方向为数字信号处理, 小波变换与应用, 多尺度几何分析, 混沌与分形理论与应用. 本文通信作者.
E-mail: qingweigao@ahu.edu.cn
(**GAO Qing-Wei** Professor at the School of Electrical Engineering and Automation, Anhui University. His research interest covers digital signal processing, wavelet transform and applications, multi-scale geometric analysis, and chaos and fractal theory and applications. Corresponding author of this paper.)



卢一相 安徽大学电气工程与自动化学院教授. 主要研究方向为图像处理, 信号的多尺度几何分解, 统计信号处理和稀疏表示.
E-mail: lyxahu@ahu.edu.cn
(**LU Yi-Xiang** Professor at the School of Electrical Engineering and Automation, Anhui University. His research interest covers image processing, multi-scale geometric decomposition of signals, statistical signal processing and sparse representation.)

孙 冬 安徽大学电气工程与自动化学院教授. 主要研究方向为图像处理, 模式识别, 非线性动力学信号处理, 计算机图形学和深度学习.
E-mail: sundong@ahu.edu.cn
(**SUN Dong** Professor at the School of Electrical Engineering and Automation, Anhui University. His research interest covers image processing, pattern recognition, nonlinear dynamic signal processing, computer graphics and deep learning.)



竺 德 安徽大学电气工程与自动化学院博士研究生. 主要研究方向为图像处理和模式识别.
E-mail: sundong@ahu.edu.cn
(**ZHU De** Ph.D. candidate at the School of Electrical Engineering and Automation, Anhui University. His research interest covers image processing and pattern recognition.)