

任意两个多边形的求交算法*

张全伙 曾晓帆 范慧琳 余 坚

(华侨大学计算机科学系, 泉州 362011)

摘要 对 A. Mangen 的算法进行改进,使之在计算机辅助排样应用中效率更高,通用性更强.

关键词 排样,多边形,求交算法

分类号 TP 391.72;TS 941

所谓排样,是指在一个平面区域(通常是一个矩形)内如何合理放置样片,使得所用材料最少且各样片之间不存在覆盖现象. 计算机辅助排样则是利用计算机自动求取某种放置方式,以达到提高生产效率和材料利用率的目的. 在服装行业中的裁剪和机械行业中的零件加工,都存在着排样问题. 现在,计算机程序可借助于图形工作站来帮助排样人员进行自动排样. 计算机辅助排样较常采用的有三种方法:(1)人机交互法. 操作者利用显示器和计算机边对话边配置样片,使浪费布料尽量少,这种方法类似于人工排样;(2)二步法. 排样过程分为自动排样和人机交互排样两步. 以矩形包络法为例,先由计算机自动求取样片的最佳包络矩形. 然后对这些包络矩形进行自动优化布局,最后在此基础上对实际形状的样片进行人机交互调整;(3)黑盒法. 只要用户事先输入样片数据和排样条件,计算机自动给出一个较佳排样结果,整个排样过程相当于一个“黑盒”. 由于样片之间是否存在覆盖不能明眼看出,也不能人为防止. 因此,检测两个样片之间是否相交,并精确计算出其相交面积是实现自动排样的核心.

1 算法预处理

由于求解两个多边形的相交多边形需大量有效的计算机资源,为减少计算量和避免一些不必要的检测,可对算法进行. 离散化处理和包围盒检测等预处理.

1.1 离散化处理

通常,一个样片模型是通过数字化被输入到计算机中的. 一个模型的轮廓线是一条封闭曲线,它可以用一个近似多边形来表示. 为消除原始模型和数字化之间产生错误轮廓线,多边形是由足够多顶点组成的. 在求解一个模型和另一个模型的所有交线时需进行大量计算. 所谓离散化处理就是把那些点较密集又较平缓的曲线用直线段代替,以简化初始模型. 离散化处理类似于曲线光滑技术的逆过程. 是从表征多边形点集中找出能反映多边形边界总体特征的特征点,并由这些特征点组成一个简化的多边形轮廓,其它非特征点则消去不予考虑. 特征

* 本文 1994-08-13 收到

点求取由对曲线段分组和组间省略两步实现. 我们采用下面两种分组方法. (1)把相邻边夹角在一定范围内的点或可容忍幅度下的一些细微振荡点归为一组(图 1). (2)计算偏量 d_i 和跨度 L_i 的比值来确定某点的弯曲度(图 2), 其中 P_{i-1}, P_i, P_{i+1} 是多边形上的三个连续点.

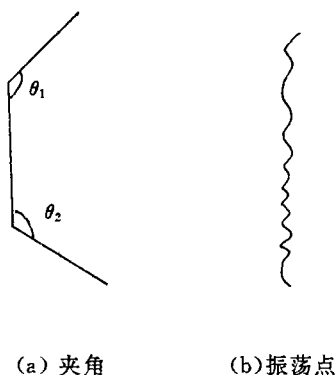


图 1 夹角和振荡点

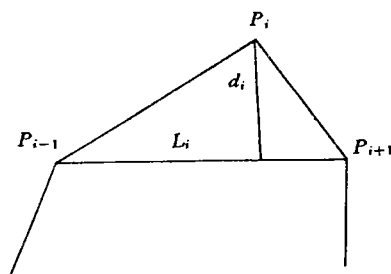


图 2 偏量与跨度

当一个多边形点集被分成若干组时,按如下方法略去同一组内的非特征点. 假设点 P_1, P_2, \dots, P_n 被归为同一组, d_i 是点 $P_i (i=2, 3, \dots, n-1)$ 的偏量, L 是 P_1 到 P_n 的跨度, 求出

$$d = \max(d_2, d_3, \dots, d_{n-1}).$$

如果 d/L 小于某定值, 则点 P_2, P_3, \dots, P_{n-1} 作为非特征点而被忽略; 否则将该组从 d 处分开成两个组并重复以上过程.

1.2 包围盒检测

所谓多边形包围盒是指平面系中该多边形的最小外部包围矩形. 假定两个多边形的包围盒分别为 box1 和 box2 , 并求两个包围盒的交, 即 $\text{box} = \text{box1} \cap \text{box2}$. 如果 box 为空, 则表明两个多边形根本不相交, 就没有必要执行求交算法.

2 求交算法的实现

显然, 两个多边形相交部分上的点必定位于两个多边形的内部或其边界上, 求交算法旨在找出所有这些点的有序集合. 因此要有一个算法能够检测一点 (x_0, y_0) 是在一个多边形 P 的内部、外部还是其边界上.

2.1 点与多边形的位置关系

点与多边形的位置关系已有多种解法, 比较常见的是“奇偶法”和“Miding-Number”法, 我们对 Warnock 算法进行推广, 得到了一种判断点与多边形位置关系的有效方法. Warnock 算法用于确定窗口与多边形的位置关系, 我们把窗口缩小成一点时, 就得到了点与多边形位置关系的判别准则. 考虑点 (x_0, y_0) 与多边形 P 的位置关系(图 3). 以点 (x_0, y_0) 为原点建立坐标系, 把平面分成 4 个区域, 每个区域依次赋予区号 0, 1, 2, 3. 多边形每边所跨越的区域数可取其两端点所在区号之差, 并按下面算法计算: $\Delta\alpha = (\text{终点区号}) - (\text{起点区号})$; 若 $\Delta\alpha > 2$, 则 $\Delta\alpha = \Delta\alpha - 4$; 若 $\Delta\alpha < -2$, 则 $\Delta\alpha = \Delta\alpha + 4$; 若 $\Delta\alpha = \pm 2$, 则该边在纵轴处分为两段, 对每段分别处

理. 当所有 $\Delta\alpha$ 求出后, 累计各边的 $\Delta\alpha$ 值, 得到: $|\Sigma\Delta\alpha|=4$, 点 (x_0, y_0) 在 多边形内; $|\Sigma\Delta\alpha|=0$, 点 (x_0, y_0) 在 多边形外. 现以图 3(a) 为例: $\Delta\alpha_{12}=0-3=-3<-2=-3+4=1$; $\Delta\alpha_{23}=1-0=1$; $\Delta\alpha_{34}=2-1=1$; $\Delta\alpha_{54}=2-2=0$; $\Delta\alpha_{15}=3-2=1$; $\Sigma\Delta\alpha=1+1+1+0+1=4$, 故点 (x_0, y_0) 在 多边形内.

2.2 标识点类型

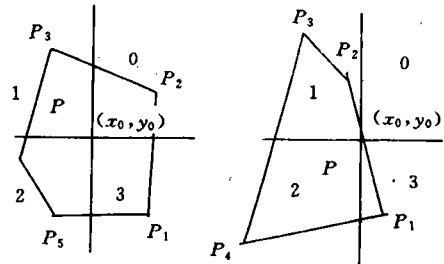
两个多边形相交的结果多边形, 其顶点只有三种不同类型: a 类点: 是 P_1 的顶点且在 P_2 内; b 类点: 是 P_2 的顶点且在 P_1 内; c 类点: 是 P_1 中一条边和 P_2 中一条边的交点. 其它的一些顶点是一个多边形的顶点, 但不在另一个多边形中, 它们不属于相交部分, 标记为 d 类点和 e 类点. 其中: d 类点: 是 P_1 的顶点但不在 P_2 中; e 类点: 是 P_2 的顶点但不在 P_1 中. 如图 4 所示, P_1 和 P_2 是两个相交的多边形, 它们的顶点具有如下类型: a 类点: 3, 7; b 类点: 14, 17; c

类点: 2, 4, 6, 8, 13; d 类点: 1, 5; e 类点: 9, 10, 11, 12, 15, 16. 由于 a, b, d, e 类点都是已存在的顶点, a, b 类点和 d, e 类点之间的差别, 通过点与多边形位置关系的判别就可得到确定. 算法的困难在于要求出所有 c 类点, 并且要使 P_1 和 P_2 的相关 c 类点之间形成链接.

我们通过逐条检测一条边与另一多边形所有的相交情况来求解 c 类点. 假设 t_i, u_i 是 P_1 的顶点, t_j, u_j 是 P_2 的顶点, 对所有成对的边 $(t_i - u_i, t_j - u_j)$ 进行比较: 若两条边互相平行, 则无交点; 若两条边互相覆盖, 则将覆盖部分上的点标识为 c 类点, 并插入该多边形的结构集合中. 一个 c 类点可能是已存在的顶点, 其相应边不作插入, 但顶点的类型将改为 c 类点. 此外, 多边形中一条单独的边可能含有多个 c 类点, 由于多边形的边均为直线, 只要比较这两点之间的距离, 就能够对相邻 c 类点进行正确的排序. 一个多边形上两个相邻 c 类点之间的所有点总是同一类型, 只要它们之中一个点的类型得以确定, 就不必每遇到一个顶点就调用“点类型检测”过程. 如果在一个 c 类点之间遇到的是一个 a 或 b 类点, 则后面的必定是一个 d 或 e 类点. 但被一个 c 类点分离的一个多边形的两条相继边可以定位于另一个多边形内(图 4 中 c 类点 2 就是这种情形). 因此, 当沿着多边形边界移动时, 每遇到一个 c 类点都必须调用“点类型检测”过程.

2.3 相交部分搜寻

在建立两个多边形 P_1 和 P_2 的数据结构时, 每个顶点设置一个属性, 以标识每个顶点的类型 a, b, c, d 或 e 类点. 算法将从某一个多边形的任一个顶点开始, 沿多边形边界移动, 直至建立起所有由 a, b, c 类点组成的相交部分. 算法的主要步骤描述如下. (1) 沿多边形边界移动, 找出属于相交部分的第一点 F , 它必定是 a, b 或 c 类点. 找到起点 F 后, 必须找到属于相交部分的一个新的第一个点, 并且这个点成为当前点 t . (2) 如果当前点 t 是 a 或 b 类点, 则它必定是正在移动的多边形的内部点. 按同样的方向移动同一个多边形, 就可找到下一个当前点



(a) 点在 P 内 (b) 点在 P 外

图 3 点与多边形位置关系

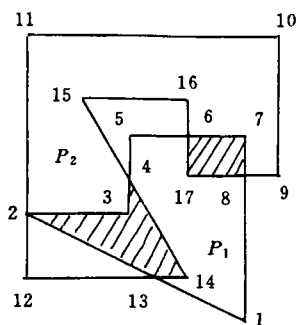


图 4 两个相交多边形

u . 但点 u 为起点 F 时, 算法将转到下述步骤(4). (3) 如果当前点 t 为 c 类点, 由于它是两个多边形的十字交点, 至少存在四个移动方向, 算法必须依次检查除刚移动的边的反方向外的其它三个方向, 以确定下一步的正确移动方向. 由于所有相关 c 类点都存在关联, 算法可方便地对各个方向进行检测. 首先, 从点 t 开始检查该方向先前是否遇见过, 这样可避免一些不必要的检测. 其次, 还要保证将要移动的边属于相交部分. 一条边是否属于相交部分只检查其两端点是不够的. 如图 5 所示, 点 t_1, t_2 都是 c 类点, 但边 $t_1 t_2$ 却不属于相交部分. 对于这种情形, 只要取该边上中点进行检查. 若它属于相交部分, 则整条边也属于相交部分. 但是如果这两个多边形的边出现重叠时, 按前面的测试, 这条边必然属于相交部分, 从图 6 看出, 事实并不一定如此. 解决这种情况的办法是先绕过它, 试一试其它方向. 当其它方向均不成功的话, 再来考虑. 这可通过检测该方向和另一多边形从点 t 开始的前后边方向是否一致来解决. 由上可知, 算法每遇到一个 c 类点, 依次按跳到另一多边形向前向后方向以及正在移动的多边形的向前方向三个方向作上述检测. 检测中, 如果该边此方向未遇见过, 且属于相交部分. 又不与另一多边形从点 t 开始的向前向后方向相同时, 算法找到下一点 u . 否则, 算法仍按同样的顺序测试各方向. 但此时对该边移动方向和另一多边形向前向后不再进行方向上的相等比较. 同时, 对相同方向的另一多边形边置为“已遇见过”, 以避免可能引起的回溯. 此时, 若 u 仍未找到, 发现当前点 t 等于 F , 无法生成相交部分的一条边(图 7), 算法退回到第(1)步, 重新寻找合适的起点 F . (4) 如果前面所有测试均不成功, 表明当前部分已完全被移动了. 当前点 t 等于 F , 说明当前相交部分已建立在内存中. 算法将继续寻找其它相交部分. 这里, 有一种情况需要说明, 就是相交结果可能含有几个具有共同点 F 的部分(图 8(a)), 算法将关闭当前部分, 并继续搜寻下一个相交部分. 图 8(b)中两个多边形有重叠边, 这种情况可看成有两个相交部分, 其中一个退化的多边形, 其面积为 0, 可以忽略. 共同起点 F 的几个相交部分(5) 在前面各步中, 找到一个合适的下一个点 u , 则该边 $t-u$ 方向置成“已遇见过”. 如果 u 与 F 相同, 相交结果当前部分已建立起来, 这是当几个相交部分相交于点 F 时发生的. 算法将以 u 为当前点, 即 u 赋予 t , 继续寻找下一个相交部分. 当 P_1 上所有可能的起点都被找到后, 算法已找到所有相交部分.

2.4 点类型计数

在求交过程中, 首先对一些特殊情况进行简单检测, 可以避免不必要的后续操作. 例如在标识点类型的同时, 对各类型点进行计数, 可以检测出两多边形的几种位置关系.

显然, 如果 c 类点个数为 0, 这两个多边形只可能是包含或分离. 此时, 若 a, e 类点总数为 0, 则 $P_1 \subset P_2$; 若 b, d 类点总数为 0, 则 $P_1 \supset P_2$. 其它情况 P_1 和 P_2 分离. 如果 c 类点个数不为 0, 两多边形才有可能相交. 此时, 若 a, b, d, e 类点皆不存在, 表明两多边形互相包含, 即 $P_1 = P_2$; 若不存在 b, d 类点, 而存在 a 类点(相应地必存在 c 类点)时, 则 $P_1 \subset P_2$; 若不存在 a 类点和 e

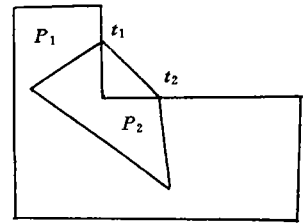


图 5 整边是否属
相交部分检查

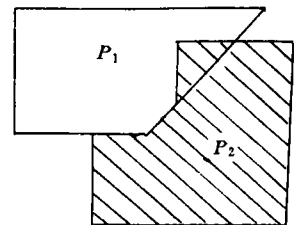


图 6 重叠边情况

类点, 而存在 b 类点(相应地必存在 d 类点)时, 则 $P_1 \supset P_2$.

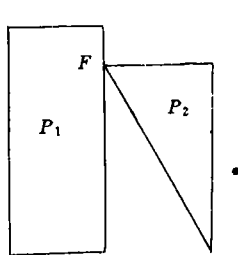
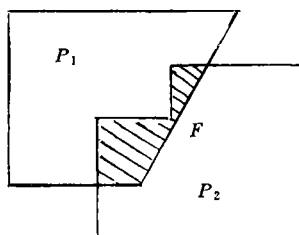
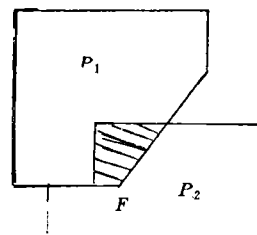


图 7 无法生成相交部分的一条边



(a) 相交两部分



(b) 一部分退化

图 8 共同起点 F 的几个相交部分

3 多边形面积计算

当两个多边形相交部分的结构已建立在计算机主存时, 其面积 S 可由下面公式计算, 即

$$S = \frac{1}{2} * \left| \sum_{i=0}^{n-1} (x_{i-1} - x_i) * (y_{i-1} + y_i) \right| = \frac{1}{2} * \left| \sum_{i=0}^{n-1} x_i * (y_{i-1} - y_{i+1}) \right|.$$

这个公式可用来计算含有 n 个顶点 (x_i, y_i) 的一个相连部分的面积, 并且不管该多边形的凹凸性及 x 坐标如何定位, 但顶点编号务须细必. 其实, 计算一个封闭区域的面积是很容易的, 只要对该区域填充时附带设置一个计数器, 以统计填充的像素点数, 就能计算其面积. 这是基于象点不同于几何点, 象点是具有大小的.

参 考 文 献

- 1 Mangel A, Lasudry N. Search for the intersection polygon of any two polygon. Application to the Garment Industry, 1991, (10): 195~208
- 2 饶运清, 刘延林, 段正澄等. 计算机辅助排样系统研制. 计算机辅助设计与图形学学报, 1994, 6(1): 72~74
- 3 罗杰斯 D F 编. 计算机图形学的算法基础. 梁友栋等译. 北京: 科学出版社, 1987. 273~296
- 4 张全伙, 房蓉晖, 范慧琳. 服装 CAD 的排料处理. 华侨大学学报(自然科学版), 1994, (3): 348~352

An Algorithm for Computing the Intersection Polygon of Two Arbitrary Polygons

Zhang Quanhua Zeng Xiaofan Fan Huilin Yu Jian

(Dept. of Computer Science, Huaqiao Univ., 362011, Quanzhou)

Abstract An improvement is made on the algorithm, with the result that it can be more efficiently and commonly applied to the computer-aided layout in tailoring.

Keywords layout in tailoring, polygon, intersection algorithm