基于凸多边形的凸壳算法*)

张显全　刘丽娜　唐振军
(广西师范大学计算机科学系　桂林 541004)

摘　要　确定平面点集的凸壳问题在计算机图形学、图像处理、CAD/CAM、模式识别等众多领域中有广泛的应用。本文根据凸多边形的性质构建了一种新的基于凸多边形的凸壳算法，该算法利用 x、y 坐标的极值将凸多边形分为几个段，应用凸壳顶点有序性，分段计算凸壳的顶点而得到凸壳。理论分析和实验结果表明，该算法运行速度快效率高，具有较强的实用性。

关键词　凸壳，单调段，计算几何

A Convex Hull Algorithm Based on Convex Polygon

ZHANG Xian-Quan　LIU Li-Na　TANG Zhen-Jun
(Department of Computer Science, Guangxi Normal University, Guilin541004)

Abstract　Convex hull problem is one of the fundamental problems in computer graphics, image processing, CAD/CAM and pattern recognition. In this paper, the properties of the convex hull are investigated and a new algorithm for the convex hull based on its properties is presented. The algorithm divides the convex polygon into several segments by the extreme points, and then computes the convex hull points of every segment. The theoretical analysis and experimental results show that the proposed algorithm is in correct and high efficiency.

Keywords　Convex hull, Monotonic segment, Computation geometry

1　前言

平面点集的凸壳定义为包含点集的最小凸集，即以点集中部分点为顶点的一个凸多边形，对该凸多边形的任意一条边，点集中所有不在该边上的点都在该边的同一侧。计算平面点集的凸壳问题是计算几何的基本问题之一，在计算机图形学、图像处理、CAD/CAM、模式识别等众多领域中有广泛的应用。虽然目前已有不少凸壳算法，但人们始终在致力于寻求更好的方法。

在给定平面点集凸壳算法中，Chand 和 Kapur[1] 最早提出了“包挠法”，利用点集的几何性质逐面构造凸包的边界，该凸壳算法的时间复杂度为 $O(n^2)$，时间复杂度较高。Jarvis[2] 提出了时间复杂度为 $O(mn)$的平面点集凸壳算法，其中 m 为凸壳顶点的个数，其时间复杂度也较高。R. L. Graham[3] 给出了一个确定平面点集凸壳算法，首先确定点集 y 坐标最小的点，计算该点到其他点连线与水平线的夹角，按角的大小进行排序，然后得出凸壳上的点；Preparata 和 Hong[4] 把分治技术应用于求解凸壳问题，把点集分成大小近似相等的两个子集，递归求出两个子集的凸壳，通过计算两个凸壳的并求出最终的凸壳；周培德等[5] 证明了线段集的平面凸包的时间复杂度为 $O(n \log n)$，然后将线段转换成平面简单多边形链，通过简单多边形链计算凸壳；崔国华等[6] 研究了排序与凸壳之间的内在联系，建立了以排序算法为基础利用双动线检测方法确定平面点集凸壳；金文华等[7] 利用排序算法与简单多边形凸包算法相结合的方式确定平面点集凸壳；T. Chen[8] 通过取点对计算由点对确定直线的斜率，找出斜率的中值，用中值把平面点集分为两个部分，应用递归算法求出凸壳，另一种算法是对点集进行分组，分别求出每组的凸壳，然后通过计算凸多边形的并求出整个点集的凸壳；Hervé Brönnimann 等[9] 对 T. Chen 算法进行了改进，对凸壳算法的存储空间进行了研究，这些计算平面点集凸壳的算法[3~9]，其时间复杂度均为 $O(n \log n)$，已证明 $O(n \log n)$是所有平面点集凸壳算法时间复杂度的下限[10]。

在进行图形设计时需计算几个凸多边形的凸壳，设两个凸多边形的顶点分别为 m 和 n，若重新对点集进行排序，则复杂度为 $O((m+n)\log(m+n))$，由于凸多边形顶点本身的有序性，本文对凸多边形的性质进行了研究，构建了一种新的基于凸多边形的凸壳算法，首先计算凸多边形的 8 个极值点，根据这些极值点把凸多边形划分为 4 个单调段，计算凸壳时只须对两个凸多边形中对应的段分别进行计算，最终可得到两个凸多边形的凸壳。该算法简单，运行的效率高，在最坏情况下，算法时间复杂度为 $O(m+n)$，具有较好的应用价值。

2　凸壳的性质

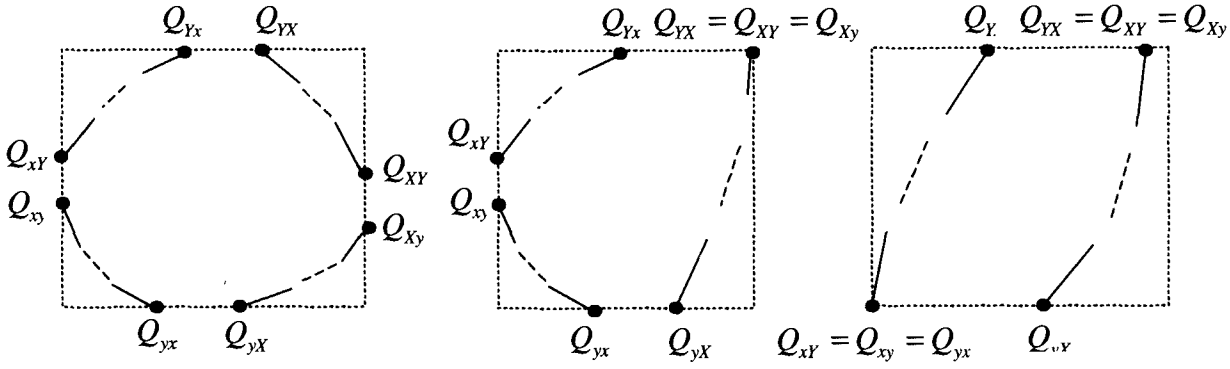
凸壳是一凸多边形，对于凸多边形的任意一条边，所有不在该边上的顶点都在该边的同一侧，具有任意两个顶点间的线段都在多边形内部，凸壳的内角小于 180 度等性质。设 $Q=\{q_1,q_2,q_n\}$为凸壳上的顶点(按顺时针进行编号)，在点集 Q 的横坐标最小的集合中，记纵坐标最大的点为 Q_{xY}，纵坐标最小的点为 Q_{xy}；在点集 Q 的横坐标最大的集合中，记纵坐标最大的点为 Q_{XY}，纵坐标最小的点为 Q_{Xy}；在点集 Q 的纵坐标最小的集合中，记横坐标最大的点为 Q_{yX}，横坐标最小的点为 Q_{yx}；在点集 Q 的纵坐标最大的集合中，记横坐标最大的点为 Q_{YX}，横坐标最小的点为 Q_{Yx}，第一个下标表示该坐标的极值，

*)广西自然科学基金(0447035)资助课题。**张显全**　副教授，主要研究方向：图形图像处理；**刘丽娜**　硕士研究生，主要研究方向：图形图像处理；**唐振军**　硕士研究生，主要研究领域：图像处理。

第二个下标是表示在第一下标的极值集合中另一坐标的极值，大写表示极大值，小写表示极小值。凸壳顶点集的编号从 $Q_{xY}=q_1$ 开始顺时针方向依次进行，对这些点进行定义：

定义 1 凸壳中 Q_{xy}，Q_{xY}，Q_{Xy}，Q_{XY}，Q_{yx}，Q_{yX}，Q_{Yx}，Q_{YX} 这 8 个点称为该凸壳的极值点。其中 Q_{xy} 与 Q_{xY}、Q_{Xy} 与 Q_{XY}、Q_{yx} 与 Q_{yX} 和 Q_{Yx} 与 Q_{YX} 称为同类极值。

对于 8 个极值点，分别过 Q_{yx}、Q_{Yx} 做 y 轴垂线，过 Q_{xy}、Q_{Xy} 做 x 轴垂线，把平面分为矩形内部和外部，凸壳上的点都在矩形内部，如图 1 所示。



(a) 4 个单调段　　(b) 3 个单调段　　(c) 2 个单调段

图 1　凸壳的极值点和单调段

对于 $Q_{xY}Q_{Yx}$ 段（顺时针方向编号）的顶点集，设该段凸壳上的顶点为 $q_1,q_2,\cdots,q_n(n\geqslant 2)$，$q_i$ 的坐标为 (x_i,y_i)，其中 $q_1=Q_{xY}$，$q_n=Q_{Yx}$，则对顶点 q_i、q_{i+1} 两个相邻点有 $\begin{cases}x_i<x_{i+1}\\ y_i<y_{i+1}\end{cases}$。假设该结论不成立，因为 $q_1=Q_{xY}$，所以 q_2 横坐标和纵坐标都单调增加，设 $q_1,q_2,\cdots,q_i$ 横坐标和纵坐标都单调增加，假设在 q_{i+1} 点结论不成立，则有三种情况：(1) $\begin{cases}x_i\geqslant x_{i+1}\\ y_i<y_{i+1}\end{cases}$；(2) $\begin{cases}x_i\geqslant x_{i+1}\\ y_i\geqslant y_{i+1}\end{cases}$；(3) $\begin{cases}x_i<x_{i+1}\\ y_i\geqslant y_{i+1}\end{cases}$；对于情况(1)，可得 $q_{i-1}q_i$ 与 q_iq_{i+1} 的夹角大于等于 180 度，与凸壳内角小于 180 度相矛盾；对于情况(2)和(3)，因而有 $y_i\geqslant y_{i+1}$；又因为 $q_n=Q_{Yx}$，所以 $y_{i+1}<y_n$，则必存在三点 $q_{k-1},q_k,q_{k+1}(k\geqslant i)$ 使得：$\begin{cases}y_{k-1}\geqslant y_k\\ y_{k+1}>y_k\end{cases}$（若一直递减，则不能到达 $q_n(Q_{Yx})$），从而 $q_{k-1}q_k$ 与 q_kq_{k+1} 所成的多边形的内角大于等于 180 度，与凸壳内角小于 180 度矛盾，所以在 $Q_{xY}Q_{Yx}$ 段凸壳上顶点的横坐标和纵坐标都单调增加，同样对其他段有如下凸壳单调性定理：

定理 设 $q_i(x_i,y_i)$、$q_{i+1}(x_{i+1},y_{i+1})$ 是凸壳中相邻两点，在 $Q_{xY}Q_{Yx}$ 段有 $\begin{cases}x_i<x_{i+1}\\ y_i<y_{i+1}\end{cases}$；在 $Q_{YX}Q_{XY}$ 段有 $\begin{cases}x_i<x_{i+1}\\ y_i>y_{i+1}\end{cases}$；在 $Q_{Xy}Q_{yX}$ 段有 $\begin{cases}x_i>x_{i+1}\\ y_i>y_{i+1}\end{cases}$；在 $Q_{yx}Q_{xy}$ 段有 $\begin{cases}x_i>x_{i+1}\\ y_i<y_{i+1}\end{cases}$。

定义 2 凸壳上 $Q_{xY}Q_{Yx}$、$Q_{YX}Q_{XY}$、$Q_{Xy}Q_{yX}$ 和 $Q_{yx}Q_{xy}$ 段上的有序顶点集（含两个极值点）称为单调段。

若凸壳中存在极值相等，共有三情况，一类如图 1(a)由 4 个单调段组成，另一类如图 1(b)由 3 个单调段组成，第三类如图 1(c)由 2 个单调段组成。因而在求凸壳时可根据单调段的性质对点进行判断，确定是否是凸壳上的顶点，减少运算量，提高运算效率。

3　有序点集凸壳计算

在求凸壳时，如果点集是有序的，则可使算法简单，两个凸多边形原有单调段上的顶点本身是有序的，因而可用有序点集的方法计算两个凸多边形的凸壳，下面以 $Q_{xY}Q_{Yx}$ 段为例说明如何判断一个点是否是凸壳顶点的算法，其他单调段类似；设过 A、B 两点的直线方程为：$f(p,A,B)=0$（p 为直线的动点），$r_1,r_2,\cdots,r_k$ 为凸壳 $Q_{xY}Q_{Yx}$ 段上的临时顶点，$r_1,r_2,\cdots,r_k,p$ 有序（以 x 坐标为关键字，如图 2 所示），且点 p 的纵坐标小于 Q_{Yx} 的纵坐标，p 是否是临时凸壳上的顶点可通过其与直线 $\overline{r_kQ_{Yx}}$ 的位置关系判断，若 $f(p,r_k,Q_{Yx})>0$，则 p 为新临时凸壳上的顶点，否则 p 不是凸壳上的顶点。

若 p 为新凸壳上的顶点，则需判断 $r_1,r_2,\cdots,r_k$ 中那些点仍是临时凸壳上的顶点，从 r_k 开始向后进行回溯处理，若存在 $r_s(s\leqslant k)$ 为新临时壳上的顶点，则 $r_1,r_2,\cdots,r_{s-1}$ 为临时壳上的顶点；这是因为：$r_1,r_2,\cdots,r_k$ 是临时凸壳上的点，则对所有不在直线 $\overline{r_ir_{i+1}}(1\leqslant i<k)$ 上的点均在该直线的同侧，在 $r_1,r_2,\cdots,r_k$ 中，若 $r_s(s\leqslant k)$ 为新临时凸壳下标最大的顶点，则 r_s 和 p 是新凸壳上两个相邻的顶点，因而所有的点均在直线 $\overline{r_sp}$ 的同侧，若有一顶点 $r_j(j<s)$ 不是新凸壳上的顶点，则 p 与 r_s 一定在直线 $\overline{r_jr_{j+1}}$ 的两侧（若在同侧，则 r_j 是新凸壳上的顶点），因为 $f(r_s,r_j,r_{j+1})<0$，所以 $f(p,r_j,r_{j+1}>0$；从而有 $f(r_t,r_j,p)<0(t=j+1,j+2,\cdots,k)$，即 $r_{j+1},r_{j+2},\cdots,r_s,\cdots,r_k$ 在直线 $\overline{r_jp}$ 的同一侧，因而 r_s 不是临时凸壳上的点，与已知矛盾，所以 $r_1,r_2,\cdots,r_s,p$ 均为临时凸壳上的顶点。

从上可知，从 r_k 开始向后进行回溯处理只须计算 $f(p,r_{i-1},r_i)(i=k,k-1,\cdots,2)$，若存在 i 使得：$f(p,r_{i-1},r_i)<0$，则 $r_s(s=i)$ 已找到，否则 $r_2=p$。对其他段，有类似的结论。

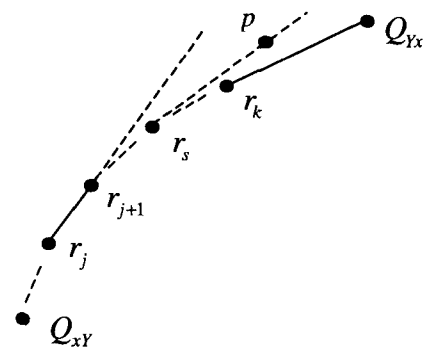


图 2　单调段凸壳的计算

在计算有序点集凸壳时首先需判断点集中的点 p 是否是凸壳上的点，若 p 不是凸壳时上的点，从点集中去除，若点 p 是凸壳上的点，需进行回溯处理，找到第一个凸壳上的顶点即可。

4 两个凸多边形的凸壳算法

对两个凸多边形的凸壳，可通过凸多边形的极值求出并集的极值，通过极值点确定单调段，分别计算每个单调段的顶点，从而得到整个凸壳。

4.1 确定两个凸多边形的单调段

设两个凸多边形的顶点集分别是 $P=\{p_1,p_2,\cdots,p_m\}$ 和 $Q=\{q_1,q_2,\cdots,q_n\}$，求 $P\cup Q$ 的凸壳时如果通过点集排序进行计算，则复杂度为 $O((m+n)\log(m+n))$，显然复杂度较高。因为 P 和 Q 是两个凸多边形的顶点本身有序，所以可应用这一特性求 $P\cup Q$ 的凸壳。首先求 8 个极值点，设 P、Q 的极值点分别为 $Q1_{xy}$，$Q1_{xY}$，$Q1_{Xy}$，$Q1_{XY}$，$Q1_{yx}$，$Q1_{yX}$，$Q1_{Yx}$，$Q1_{YX}$ 和 $Q2_{xy}$，$Q2_{xY}$，$Q2_{Xy}$，$Q2_{XY}$，$Q2_{yx}$，$Q2_{yX}$，$Q2_{Yx}$，$Q2_{YX}$，点集 $P\cup Q$ 的极值点为 Q_{xy}，Q_{xY}，Q_{Xy}，Q_{XY}，Q_{yx}，Q_{yX}，Q_{Yx}，Q_{YX}，则 $P\cup Q$ 的极值点可通过 P、Q 的极值点计算，Q_{xy} 是 $Q1_{xy}$，$Q1_{xY}$，$Q2_{xy}$，$Q2_{xY}$ 中 x 坐标最小的集合中 y 坐标最小的点；Q_{xY} 是 $Q1_{xy}$，$Q1_{xY}$，$Q2_{xy}$，$Q2_{xY}$ 中 x 坐标最小的集合中 y 坐标最大的点；同理可求出其他极值点，如图 3 所示，通过这些极值点确定了点集 $P\cup Q$ 的单调段。

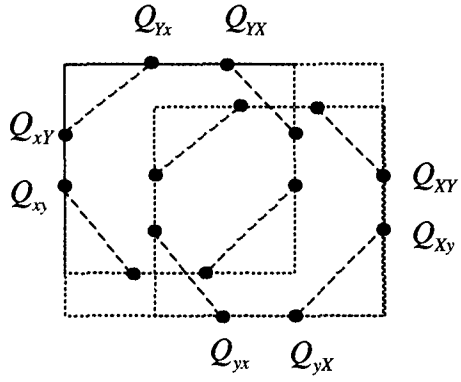


图 3 两个凸多边形的极值及单调段

4.2 单调段凸壳的算法

求 $P\cup Q$ 的凸壳通过计算它的四个单调段来进行，每个单调段的点是否是凸壳上的点只与两个凸多边形中同一类型的单调段有关，与其他单调段无关，因此求两个多边形的凸壳时可分别计算凸多边形对应单调段上的点，然后依次取单调段上的点就可得到整个凸壳。下面以 $Q_{xY}Q_{Yx}$ 单调段进行说明，对于 $Q_{xY}Q_{Yx}$ 段，只与 $Q1_{xY}Q1_{Yx}$ 和 $Q2_{xY}Q2_{Yx}$ 两个单调段有关，与其他段无关。设 $Q1_{xY}Q1_{Yx}$ 和 $Q2_{xY}Q2_{Yx}$ 段凸壳上的顶点为 $p_i(px_i,py_i)(i=1,2,\cdots,T)$ 和 $q_j(qx_j,qy_j)(j=1,2,\cdots,K)$，$P\cup Q$ 的凸壳 $Q_{xY}Q_{Yx}$ 单调段顶点为 $r_1,r_2,\cdots,r_L$（$2\leqslant L\leqslant K+T$），$r_t(t=1,2,\cdots,L)$ 的坐标为 (rx_t,ry_t)，Q_{xY} 点是 $P\cup Q$ 的凸壳上 x 坐标最小的点，该点一定是凸壳上的顶点，设 $r_1=Q_{xY}$，分别从 p_1 和 q_1 开始计算凸壳的顶点，找出 p_i 使得 $py_i>ry_1$，若不存在满足条件的点，根据凸壳的单调性定理 $Q1_{xY}Q1_{Yx}$ 段上所有点都不是 $P\cup Q$ 的凸壳上的点，因而 $Q_{xY}Q_{Yx}$ 与 $Q2_{xY}Q2_{Yx}$ 相同；同理找出 q_j 使得 $qy_j>ry_1$，若不存在则 $Q2_{xY}Q2_{Yx}$ 上所有点都不是 $P\cup Q$ 的凸壳上的点，因而 $Q_{xY}Q_{Yx}$ 与 $Q1_{xY}Q1_{Yx}$ 相同；若存在点 p_i 和 q_j 满足以上条件，如果 $px_i\leqslant qx_j$ 则 $r=p_i$，$i=i+1$，否则 $r=p_j$，$j=j+1$，r 成为临时凸壳上的点。这样依次取 p_j 和 q_j 中 x 坐标小的点，使得这些点的 x 坐标有序，用有序点集凸壳计算方法判断该点是否是临时凸壳上的点，若是临时凸壳上的点则对 r 进行回溯处理，若不是则用同样的方法找出下一个有序点，直到 Q_{Yx} 点为止，这样就可求出该段凸壳上的所有的顶点。具体算法如下。

STEP 1：$r_1=Q_{xY}$，$i=1$，$j=1$，$s=2$；

STEP 2：IF($py_i\leqslant ry_1$)goto STEP 3；
ELSE goto STEP 4；//找到需处理的点

STEP 3：IF($i<T$) $i=i+1$，goto STEP 2；
ELSE goto STEP 11；//到最后一个点

STEP 4：IF($qy_j\leqslant ry_1$)goto STEP 5；
ELSE goto STEP 6；//找到需处理的点

STEP 5：IF($i<K$) $j=j+1$，goto STEP 4；
ELSE goto STEP 12；//到最后一个点

STEP 6：IF($px_i\leqslant qx_j$) $p=p_i$，$i=i+1$，goto STEP 7；//找 x 坐标小的点
ELSE $p=p_j$，$j=j+1$，goto STEP 7；

STEP 7：IF($p=Q_{Yx}$)goto STEP 10；//找到该段凸壳上最后一个点
IF($f(p,r_{S-1},Q_{Yx}<0)$goto STEP 6；//不是凸壳上的点
ELSE goto STEP 8；//是凸壳上的点，开始回溯

STEP 8：IF($s=2$ *or* $f(p,r_{S-2},r_{S-2})<0$) $r_S=p$，$s=s+1$，goto STEP 6；//到端点或回溯结束
ELSE goto STEP 9；//继续回溯

STEP 9：$s=s-1$，goto STEP 8；

STEP 10：$r_s=Q_{Yx}$，$r_1,r_2,\cdots,r_s$，为凸壳上的点，结束；

STEP 11：$r_i=p_i(i=1,2,\cdots,T)$，$s=T$，结束；

STEP 12：$r_i=q_i(i=1,2,\cdots,K)$，$s=K$，结束；

同理可求出其他段的凸壳上的点，顺时针连接可得整个凸壳。

4.3 两个凸多边形的凸壳算法

对两个多边形的凸壳，首先从它们的极值点中计算 Q_{xy}，Q_{xY}，Q_{Xy}，Q_{yx}，Q_{yX}，Q_{Yx}，Q_{YX} 这 8 个极值点，通过这些极值点计算凸壳每个段的顶点，依次取这四个段上的点可得凸壳。具体算法如下。

STEP 1：在极值点集 $Q1_{xy}$，$Q1_{xY}$，$Q1_{Xy}$，$Q1_{XY}$，$Q1_{yx}$，$Q1_{yX}$，$Q1_{Yx}$，$Q1_{YX}$ 和 $Q2_{xy}$，$Q2_{xY}$，$Q2_{Xy}$，$Q2_{XY}$，$Q2_{yx}$，$Q2_{yX}$，$Q2_{Yx}$，$Q2_{YX}$ 中计算 8 个极值点 Q_{xy}，Q_{xY}，Q_{Xy}，Q_{XY}，Q_{yx}，Q_{yX}，Q_{Yx}，Q_{YX}；

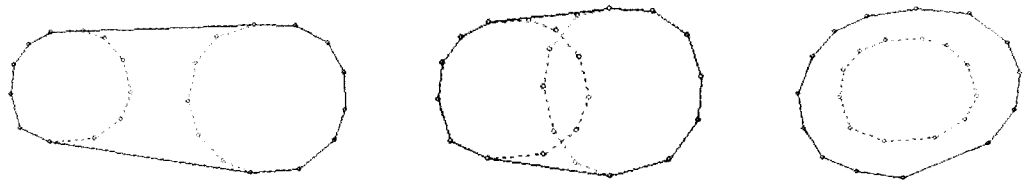
STEP 2：由极值点确定凸壳的 4 个单调段，应用单调段上凸壳算法计算每段上凸壳的顶点；
在 $Q1_{xY}Q1_{Yx}$ 和 $Q2_{xY}Q2_{Yx}$ 段上计算 $Q_{xY}Q_{Yx}$ 上凸壳的顶点；
在 $Q1_{YX}Q1_{XY}$ 和 $Q2_{YX}Q2_{XY}$ 段上计算 $Q_{YX}Q_{XY}$ 上凸壳的顶点；
在 $Q1_{Yx}Q1_{xY}$ 和 $Q2_{Yx}Q2_{xY}$ 段上计算 $Q_{Yx}Q_{xY}$ 上凸壳的顶点；
在 $Q1_{yx}Q1_{xy}$ 和 $Q2_{yx}Q2_{xy}$ 段计算 $Q_{yx}Q_{xy}$ 上凸壳的顶点；

STEP 3：按顺序 $Q_{xY}\rightarrow Q_{Yx}$，$Q_{YX}\rightarrow Q_{XY}$，$Q_{Xy}\rightarrow Q_{yX}$，$Q_{yx}\rightarrow Q_{xy}$ 取出每个段上的点，对相同的极值点只取一次，可得两个凸多边形并的凸壳。

5 复杂度分析及实验结果

在本文的凸壳算法中，对于顶点分别为 m 和 n 的两个凸多边形，寻找 8 个极值点的时间复杂度分别为 $O(m)$ 和 $O(n)$，由于对凸多边形进行分段处理，每段上凸壳顶点的计算只与两个同类的单调段有关，与其他段无关，每个点处理一次，因而对两个凸多边形的所有顶点进行处理的时间复杂度为 $O(m+n)$，对临时凸壳上的点进行回朔处理的时间复杂度为 $O(m+n)$，所以整个算法最坏时的时间复杂度为 $O(m+n)$，本文算法的时间复杂度低。

对本文的算法进行大量的实验，都能正确计算出两个凸多边形的凸壳。对于多个凸多边形的凸壳算法，可计算两个凸多边形的凸壳，再与另外的凸多边形进行计算。图 4 为采用本文算法生成的凸壳(虚线为凸多边形，实线为本文算法求出的凸壳)，图 4(a)为不相交的二个凸多边形的凸壳，图 4(b)为相交的二个凸多边形的凸壳，图 4(c)为两个凸多边形包含情况的凸壳。



(a) 二个凸多边形不相交　(b) 二个凸多边形相交　(c) 二个凸多边形包含

图 4　凸多边形的凸壳

结论　通过极值确定了凸壳的四个段，凸壳的每段都是单调的，根据这一性质计算两个凸多边形的凸壳算法简单可行；对每个段分别进行计算，减小了点集的规模，提高了运算速度，另一方面利用了凸多边形顶点本身有序的特点，不需进行排序，从上几个方面进行处理降低了计算复杂度；理论分析和实验表明本文算法的性能好，具有较强的实用性。下一步将应用凸壳的性质对平面点集的凸壳问题进行研究。

参 考 文 献

1 Chand D R, Kapur S S. An algorithm for convex polytopes [J]. JACM, 1970, 17(1): 78～86
2 Jarvis R A. On the identification of the convex hull of a finite set of points in the plane [J]. Information Processing Letters, 1973, 2(1): 18～21
3 Graham R L. An efficient algorithm for determine the convex hull of a finite linear set [J]. Information Processing Letters, 1972, 1(1): 132～133
4 Preparata F P, Hong S J. Convex hulls of finite sets of points in two and three dimensions [J]. CACM, 1977, 20(2): 87～93
5 周培德. 寻求平面上线段集的凸壳的算法[J]. 工程图学学报, 2003, 23(2): 116～119
6 崔国华，洪帆，余祥宣. 确定平面点集凸包的一类最优算法[J]. 计算机学报, 1997, 20(4): 330～334
7 金文华，何涛，刘晓平，等. 基于有序简单多边形的平面点集凸包快速求取算法[J]. 计算机学报, 1998, 21(6): 533～539
8 Chan T. Optimal output-sensitive convex hull algorithms in two and three dimensions [J]. Discrete Comput Geom, 1996, 16(3): 361～368
9 Brönnimann H, Iacono J, Katajainen J, et al. Space-efficient planar convex hull algorithms [J]. Theoretical Computer Science, 2004, 321(1): 25～40
10 Yao A C. A lower bound to finding convex hulls [J]. J ACM, 1981, 28(4): 780～787

(上接第 120 页)

功能，其中采集编码都用软件方法实现。由于监控服务器既要负责多路音视频的存储，又要提供远程监控数据的检索功能，因此对磁盘的访问很频繁，磁盘调度问题就成为监控服务器的性能瓶颈。

实验 1　没有调用本算法。监控服务器同时采集 2 路音视频数据，实时监控画面基本流畅，远程数据检索基本达到性能要求；监控服务器同时采集 4 路音视频数据，实时监控画面延迟明显，偶尔有跳帧现象，远程数据检索反映很慢，还出现检索不成功的情况。

实验 2　采用本算法。监控服务器同时采集 2 路音视频数据，实时监控画面流畅，远程数据检索响应很快；监控服务器同时采集 4 路音视频数据，实时监控画面基本流畅，没有出现跳帧现象，远程数据检索响应基本正常，没有出现检索不成功的情况。

通过实验可以看出，本算法的应用，基本解决了在以软件方法实现的监控系统中，由于存储大量数据而造成的实时显示画面延迟大，远程数据检索响应慢的问题；同时也提高了监控服务器提供服务的路数。

结论　本文提出了一种流媒体服务器磁盘调度策略，给出了算法主要模块探测模块、负载监测模块和自适应管理模块的具体实现。该策略应用到基于软件压缩的多路视频监控系统中，视频服务器同时提供实时视频远程服务和历史视频的远程查询服务功能，在发送给客户端视频数据的同时，还要保存服务端采集到的音视频数据，这就会产生大量的数据流，因此，造成对磁盘的频繁访问。而且视频服务器还要接受一些控制命令，如果不及时响应，会造成监控数据的丢失。通过应用本调度算法，很好地解决了这个问题，使得访问音视频流数据的实时请求不会错过截止时间，也不会造成控制命令等尽力服务请求长时间得不到服务。自适应管理策略的应用，增加了视频服务器的易管理性，也增强了它的稳定性。

参 考 文 献

1 Aurrecoechea C, Campbell A T, Hauw L. A survey of QoS architecture [J]. Multimedia System, 1998, (6): 138～151
2 丁峰，邓勇，沈钧毅. 一种在分布式系统中支持 QoS 的方法研究[J]. 小型微型计算机系统, 2000, 22(1)
3 Nerjes G, Muth P, Paterakis M, et al. Incremental scheduling of mixed workloads in multimedia information servers[J]. Multimedia Tools and Applications, 2000, 11: 9～33
4 石柯. 支持 QoS 的操作系统框架的研究[J]. 计算机工程与应用, 2002(7): 110～112
5 Alvarez G, Keeton K, Merchant A, et al. Storage Systems Management[C]. Tutorial presented at ACM Sigmetrics 2000, Santa Clara, CA, June 2000
6 Pradhan P, Tewari R, Sahu S, et al. An Observation－based Approach Towards self－managing Web Servers[C]. In: Proc. of ACM/IEEE IntlWorkshop on Quality of Service (IWQoS), Miami Beach, FL, May 2002
7 Chesire M, Wolman A, Voelker G, et al. Measurement and Analysis of a Streaming Workload[C]. In: Proc. of the USENIX Symposium on Internet Technology and Systems (USEITS), San Francisco, CA, March 2001
8 Revel D, McNamee D, Pu C, et al. Feedback Based Dynamic Proportion Allocation for Disk I/O[J]: [Technical Report CSE-99-001]. OGI CSE, January 1999
9 Hennessy J. The Future of Systems Research[J]. IEEE Computer, August 1999. 27～33
10 Sundaram V, Shenoy P. A Practical Learning-Based Approach for Dynamic Storage Bandwidth Allocation[J]. In: Proc. of ACM/IEEE Intl Workshop on Quality of Service (IWQoS), 2003, LNCS 2707. 479～497