# 计算几何实验报告

## ——凸包的实现

## 二维凸包的实现

二维凸包的实现方法有很多种，算法复杂度为$O(n\log n)$的有 GraHam's scan 算法。这种算法在最坏的情况下是最优的。但是如果凸包极点的个数$h$很小时，我们可以获得更好的算法，使算法的时间复杂度更小。如：Jarvis's march 算法可以在$O(nh)$构造凸包。本次实验我实现了一种更优的输出敏感(output sensitive)的凸包算法，算法的复杂度为$O(n\log h)$。

### 算法描述

1. 首先对原始点进行分组。选择参数$1 < m < n$, 将原始点分成$\lceil n/m \rceil$组，每组的点最多为$m$个。
2. 对每个分组采用 graham's scan 算法计算凸包。每个分组的复杂度为$(m\log m)$,共为$O(\lceil n/m \rceil(m\log m)) = O(n\log m)$
3. 扫描$\lceil n/m \rceil$个分组凸包，对当前点$p_k$计算每个凸包的切线或支撑线,对凸包进行包扎。如图 1 示。这部分的时间复杂度为$O((n/m)\log m)$
4. 判断凸包是否构造成功，如不成功，则重复 1-3 步。

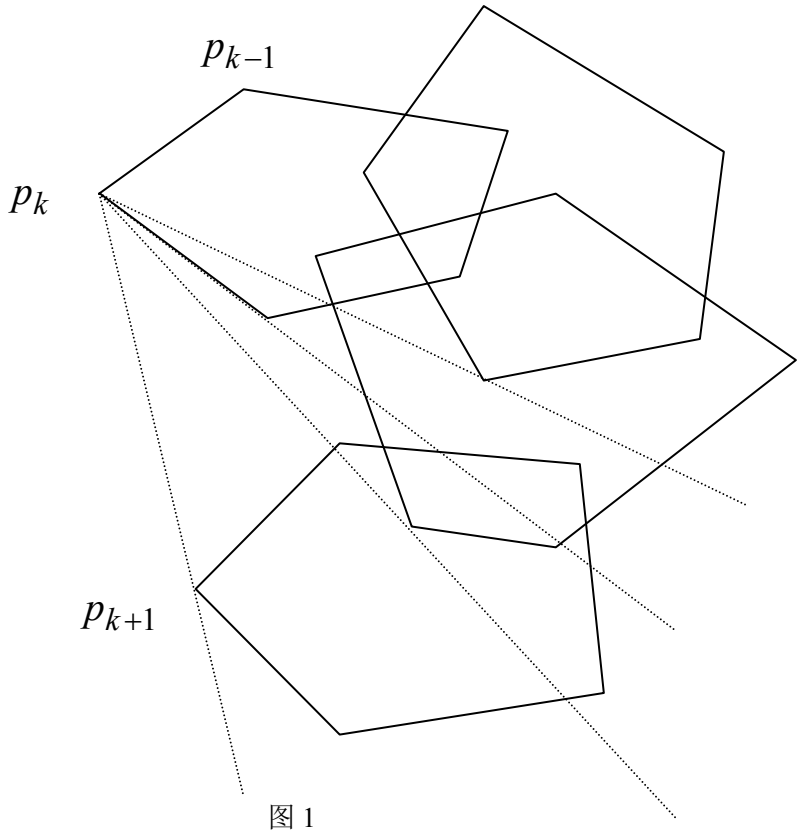



图 1

以下为算法的伪码描述：

Algorithm Hull2D(P , $m$ , H)　　where　$P \subset E^2$　$3 \le m \le n$ , and $H \ge 1$

1 . Partition P into subsets $p_{1,}...p_{\lceil n/m \rceil}$ each of　size at most $m$

2　for　i = 1, . . . , $\lceil n/m \rceil$ do

3　　compute $conv(P_i)$ by　Gramham's scan and store its vertices in an array in ccw order

4　$p_0 \leftarrow (0,-\infty)$

5　$p_1 \leftarrow$ the rightmost point of　P

6　for k = 1. . .　H do

7　　for　i = 1 , . . . , $\lceil n/m \rceil$ do

8　　compute the point $q_i \in P_i$　that maximizes $\angle p_{k-1}p_kq_i$ $(q_i \neq p_k)$

by performing a binary search o the vertices of $conv(P_i)$

9　　$p_{k+1} \leftarrow$ the point $q$　from $\{q_1,\cdots,q_{\lceil n/m \rceil}\}$ that maximize $\angle p_{k-1}p_kq$

endfor

if $p_{k+1}$ = $p_1$ then return the list $\{p_1, \cdots, p_k\}$

endfor

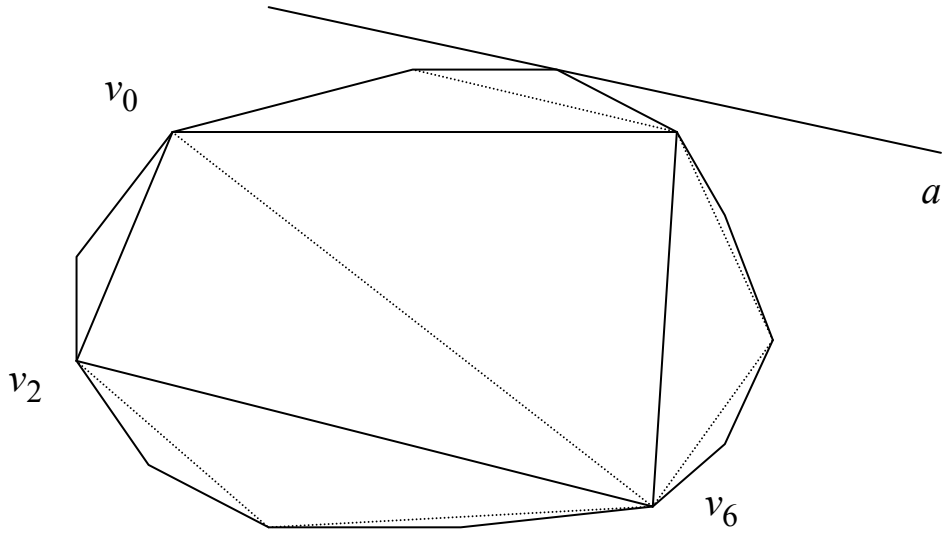
10 return INCOMPLETE

Algorithm Hull2D(P), where $P \subset E^2$

1. for i = 1,2 , ... do
2. $L \leftarrow$ Hull2D(P, m ,H) where m = H = min $\{2^{2^t}, n\}$
3. if $L$ = INCOMPLETE then return $L$

## 算法关键

该算法主要有两部分组成，第一部分为用 Gramham's scan 算法进行分组，构造凸包，并将将极点按逆时针方向排序。第二部分是对分组的凸包包扎。Gramham's scan 算法比较简单，这里就不再详述。对分组的凸包包扎需要一些技巧，首先在整个点集 P 中找到最右边的点，记为 $p_1$，以 $p_1$ 为基点，扫描每块凸包，求 $p_1$ 点到各个凸包的切线或支撑线。扫描每个凸包时，为了保证复杂度要求，我们需要利用一种等级结构。 如图 2 示，这样,原凸包可以表示成一个凸包序列 $p_0 \supset p_1 \supset p_{2,} \cdots p_k$。




$p_0$ 为原凸包本身，$p_k$ 由 $p_{k-1}$ 的相隔的点组成。扫描凸包找切线其实就是从凸包外一点 $a$ 发出一条射线，相对 $a$ 按逆时针方向旋转所接触到的第一个点与 $a$ 的连线。这个过程是个递归的过程，假设扫描 $p_k$ 得到符合条件的点 $v_i$，再在 $p_{k-1}$ 中比较 $v_i$ 和其相邻的点

$v_{i+l}, v_{i-l}$ 从中找到符合条件的点。在比较 $v_{i+l}$， $v_i$， $v_{i-l}$ 时，我采取的是构造凸包时判断左拐的算法（因为我们是按逆时针方向包扎）。

具体算法如下

Initial： l = 1 , P 为凸包的极点集，m 为 P 中极点的个数

```
BiSearch(m,l,P,a)
begin
    if   2* l   > m-1
         if   Left(a, p0 , pl )
         then return 0
         else return l
    else
        n = BiSearch( m, 2*l, P, a)
        if   Left(a,  pn , pn−l )
        then   tmp = n
        else   tmp = n-l
        if   Left(a,  ptmp , pn+l  )
        then return   tmp
        else return   n+l
end
```

在特殊情况，即 a 本身是凸包 P 的极点时，用左拐规则无法判断，此时我们比较 a 是否为 $p_n, p_{n-l}$ $p_{n+l}$ 如果是则返回相应的序号。

分析算法我们可以看出，算法本身和折半查找很类似，算法复杂度也类似为 $O(\log m)$

## 算法的复杂度分析

从伪码描述的算法的可以看出，我们事先是无法知道 $h$，所以我们采用试探的算法。以 H 去猜测 $h$,当 H 等于或刚大于 $h$ 时，算法结束。

在每一次试探中，分组计算凸包的复杂度为 $O(n \log m)$ 对分组的凸包进行包扎的复杂度为 $O((n/m)\log m)$ ,在包扎时，我们需要 H 步，所以算法的复杂度为 $O(n\log+ H((n/m)\log m)) = O(n(1+H/m)\log m)$ ,通过选择 m=H, 复杂度为 $O(n\log H)$=$O(n2^t)$

在试探 H 的过程中，我们需要重复$\lceil \log\log h \rceil$步，而且第 *t* 次循环的复杂度为$O(n2^t)$

所以，最终的复杂度为$O(\sum_{t=1}^{\lceil \log\log h \rceil} n2^t) = O(n2^{\lceil \log\log h \rceil+1}) = O(n\log h)$

# 三维凸包的实现

按照凸包的思想，我们也可以做到$O(n\log h)$的算法，我们同样也采取 Divide and Conquer 算法进行分组凸包，并将它们按照等级结构存储。然后，对分组的凸包进行包扎。

为了实现等级结构存储，我们需要设计复杂的数据结构。由于时间和精力的限制，我没有实现此算法，而实现了一种较为简单的增量凸包算法。它的算法复杂度为$O(n^2)$，最好情况下为$O(nh)$。

算法思想：

Algorithm:

Initilize $H_3$ to tetrahedron ( $p_0, p_1, p_2, p_3$ )

for i = 4, . . . n - 1 do

for each face $f$ of $H_{i-1}$ do

Compute volume of tetrahedron determined by $f$ and $p_i$

Mark $f$ visible iff volume < 0

if no face are visible

then Discard $p_i$ ( it is inside $H_{i-1}$ )

else

for each border edge of $H_{i-1}$ do

Construct cone face determined by *e* and $p_i$

for each visible face $f$ do

Delete $f$

Update $H_i$

此算法较为简单，且数据结构也容易实现。在这里，就不再详述了。

# 参考文献


1.T.M.Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions . Discrete&Computational Springer-Verlag , 1996

2. B. Chazelle and D. P. Dobkin. Intersection of convex objects in two and three dimensions. J. *Asso*c. *Comput*. *Mac*h., 34:1–27, 1987.

3. D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra: a unified approach. *Pro*c.1*7th Interna*t. *Collo*q. *on Automat*a, *Language*s, *and Programmin*g, pp. 440–413, Lecture Notes in Computer Science, vol. 443. Springer-Verlag, Berlin, 1990.

4. H. Edelsbrunner. *Algorithms in Combinatorial Geometr*y. Springer-Verlag, Berlin, 1987.

5. J. O'Rourke. *Computational Geometry in* C. Cambridge University Press, Cambridge, 1994.

6. F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commu*n.
*AC*M, 20:87–93, 1977.