

CM0304 Graphics

II Geometric Modelling

II.7 Convex Hull Finding Algorithms

Xianfang Sun

F. C. Langbein

School of Computer Science
Cardiff University



Version 2.3

Overview

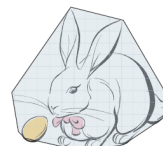
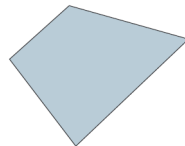
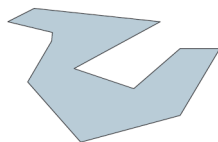
- Convexity
 - Convex hull
- Convex hull finding
 - Naive algorithm
 - Incremental convex hull algorithm

Xianfang Sun F.C. Langbein, CM0304 Graphics – II Geometric Modelling; II.7 Convex Hull Finding Algorithms

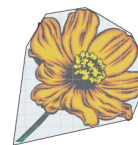
1

Convexity

- A set is *convex* if every line segment connecting two points in the set is fully contained in the set



- Sometimes it is useful to make a shape convex
 - Natural method for *shape simplification*
 - Better *approximation* than bounding box
 - *Collision detection*



- Algorithm illustrates many techniques and issues

Xianfang Sun F.C. Langbein, CM0304 Graphics – II Geometric Modelling; II.7 Convex Hull Finding Algorithms

2

Convex Hull

➤ *Convex hull* of a set of points P

- *Smallest convex set containing P*

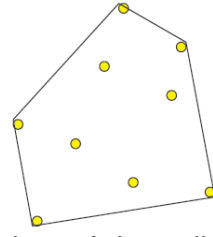
➤ How to find it using maths?

- Take all line segments between all point pairs
- Add all line segments between all points of these line segments, ... and so on until no more new points are added

➤ Take all *convex combinations* of the points in P :

$$\sum_{p \in P} c_p p, \quad c_p \geq 0, \quad \sum_{p \in P} c_p = 1$$

➤ This is not suitable for an algorithm



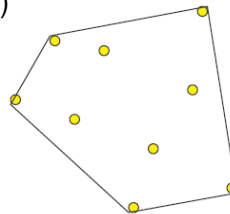
Computational Problem

➤ Given n points P in \mathbb{R}^2 , find the convex hull $H(P)$

- $H(P)$ is a *convex polygon*
- $H(P)$ *cannot have more than n vertices*
- Seek the points of $H(P)$ in *clockwise* order

➤ Design a *fast* algorithm for this problem

➤ Assume all points are *distinct* (otherwise sort and remove duplicates)



Naive Approach

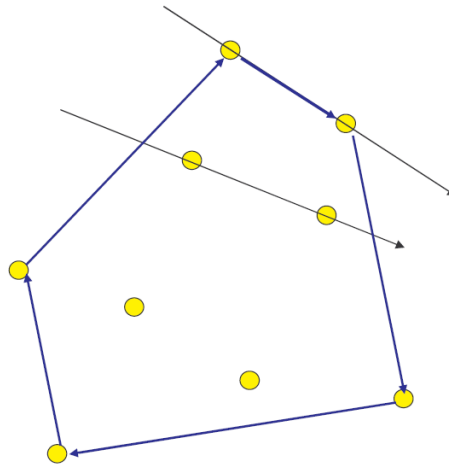
➤ All edges of the convex hull polygon are *line segments* between two points in P

- A directed line segment is an edge if there are *no points to the left of it*

➤ Naive algorithm: *check all line segments*

1. **For** all pairs (p, q) of points in P , **do**
// Check if line segment $p \rightarrow q$ forms a boundary edge
 - A. **For** all points $r \in P - \{p, q\}$, **do**
 - i. **If** r lies to the left of directed line $p \rightarrow q$, go to 2
 - B. Add (p, q) to the set of edges E
2. **Endfor**
3. Order edges in E to form the boundary of $H(P)$

Example



Algorithm Details

- ▶ Test to check if r lies to the left of a directed line $p \rightarrow q$
 - Recall polygon orientation / sidedness
 - Line equation in 2D: $l^t x - d = 0$
 - l is orthogonal to line $p \rightarrow q$ (points to the left)
 - d is shortest distance between line and origin
 - r is to the left if $l^t r - d > 0$
 - ➡ Constant time operation
- ▶ Order the edges in E
 - Sort them by concatenating edges with common vertices

Algorithm Analysis

1. For all pairs (p, q) of points in P , do
 - // Check if line segment $p \rightarrow q$ forms a boundary edge
 - A. For all points $r \in P - \{p, q\}$, do
 - i. If r lies to the left of directed line $p \rightarrow q$, go to 2
 - B. Add (p, q) to the set of edges E
 2. Endfor
 3. Order edges in E to form the boundary of $H(P)$
- ▶ Outer loop (1-2): repeat loop body for all n^2 point pairs: $O(n^2)$
 - ▶ Inner loop (A): repeat loop body for all n points except two: $O(n)$
 - ▶ Overall for loops: outer loop repeats inner loop (multiply): $O(n^3)$
 - ▶ Naive approach for 3 takes $O(n^2)$ time, can be improved to $O(n \log n)$
 - ▶ Loops (1-2) determine time order of algorithm: $O(n^3)$ (expensive!)

A Faster Algorithm

- Make algorithm *incremental* to increase efficiency
 - Add a point a time to convex hull polygon
 - First sort points in x coordinate
 - Try adding the points in sorted sequence
- Process points from left to right
 - Not all points on the convex hull polygon follow this order
 - First process points from *left to right* for *upper hull*
 - Then process points from *right to left* for *lower hull*

Incremental Convex Hull Algorithm

1. Sort P by x coordinates: p_1, p_2, \dots, p_n
2. Create a list $L_{\text{upper}} = (p_1, p_2)$
3. **For** $i = 3$ to n , **do**
 - A. Append p_i to L_{upper}
 - B. **While** L_{upper} contains more than two points **and** the last three points in L_{upper} do not make a right turn, **do**
 - i. Delete the middle of the last three points from L_{upper}
4. Create a list $L_{\text{lower}} = (p_n, p_{n-1})$
5. **For** $i = n-2$ to 1 , **do**
 - A. Append p_i to L_{lower}
 - B. **While** L_{lower} contains more than two points **and** the last three points in L_{lower} do not make a right turn, **do**
 - i. Delete the middle of the last three points from L_{lower}
6. Remove first and last point from L_{lower} to avoid duplication
7. Concatenate L_{lower} and L_{upper} to obtain result

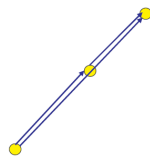
Correct?

- Points with the same x coordinate cause problems
- Solution: *lexicographic order*
 - First sort by x coordinate
 - If two points have the same x coordinates, sort by y coordinate in addition

Incremental Convex Hull Animation

Colinear Points

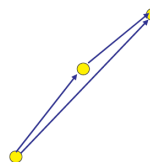
- ▶ Algorithm does weird things for colinear points



- ▶ Introduce *left/right turn speciality*
 - Three points which lie on a straight line have to be treated as if they make a left turn
 - Avoids that the middle point of three colinear points is added to the convex hull

Nearly Colinear Points

- ▶ Algorithm does weird things for nearly colinear points



- ▶ Handling nearly colinear points is *highly non-trivial*
 - Requires higher precision
 - Or arbitrary precision methods
- ▶ Our approach: sweep it under the carpet

Algorithm Analysis

1. Sort P by x coordinates: p_1, p_2, \dots, p_n
2. Create a list $L_{\text{upper}} = (p_1, p_2)$
3. **For** $i = 3$ to n , **do**
 - A. Append p_i to L_{upper}
 - B. **While** L_{upper} contains more than two points **and** the last three points in L_{upper} do not make a right turn, **do**
 - i. Delete the middle of the last three points from L_{upper}
- 4.-5. Lower hull similar to 2-3
- 6.-7. Prepare and report results

- *Sorting* in 1: $O(n \log n)$
- *Outer loop* 3: repeat once per point: $O(n)$
- Inner loop 3.B: ?

Time Order

- Time order of *inner loop*
 - For each execution of the outer loop, the inner loop is executed once
 - For any extra execution of the inner loop, a point is deleted
- As each point can be *deleted only once*, the total number of extra execution over *all* outer loops is bounded by n
- Hence, the time order of the *loops* is $O(n)$
- So overall, *sorting* is *most expensive* and determines the time order of the algorithm: $O(n \log n)$
 - Note, if we have sorted points from some other source, the order can be reduced to $O(n)$

Summary

- What is a convex set and what is the convex hull of a set? What are convex combinations of points?
- Explain a naive algorithm to find the convex hull. What is its principle? List the pseudo-code. What is its time order?
- Describe a fast incremental algorithm to find the convex hull. What is its principle? List the pseudo code. What is its time order? What part of the algorithm determines the time order?
- How can we handle colinear points? What problems arise from nearly colinear points?