Computational Geometry

- algorithms dealing with geometric objects, such as points, lines, segments, planes, circles, ellipses, polygons, maps, ...
- applications in CAD (Computer-Aided Design), Geographic Information Systems, Virtual Reality, Planing, Optics, ...
- overview of lectures
  - concepts: convex hulls, Delaunay triangulations, Voronoi diagrams, polygons
  - techniques: sweep, divide-and-conquer, incremental constructions, randomized incremental construction
  - implementation issues: geometric primitives

Books

- [dBKOS97], a standard text book on computation geometry
- [Kle97], the standard German text book
- [MN99, Chapters 9 and 10] treats algorithms, implementations, and checking
- [Mul94, Ede87, O'R94, Meh84, BY98], more books

[BY98] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, Cambridge, 1998.

[dBKOS97] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 1997.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.

[Kle97] R. Klein. *Algorithmische Geometrie*. Addison-Wesley, 1997.

[Meh84] K. Mehlhorn. *Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry*. Springer, 1984.

[MN99] K. Mehlhorn and S. Näher. *The LEDA Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999. 1018 pages.

[Mul94] K. Mulmuley. *Computational Geometry*. Prentice Hall, 1994.

[O'R94] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.

A Geometric Primitive: The Orientation Function

For three points p, q, r in the plane, define

$$orientation(p,q,r) = \begin{cases} +1 & \text{if } (p,q,r) \text{ is a left turn} \\ -1 & \text{if } (p,q,r) \text{ is a right turn} \\ 0 & \text{if the points are collinear} \end{cases}$$

collinear = there is a line passing through p, q, and r

A Formula for the Orientation Function

Lemma 1

$$orientation(p,q,r) = sign \det \begin{pmatrix} 1 & 1 & 1 \\ p_x & q_x & r_x \\ p_y & q_y & r_y \end{pmatrix} = sign \det \begin{pmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{pmatrix}$$

- $p = (0,0)$, $q = (1,0)$, $r = (0,1)$, $orientation(p,q,r) = +1$,

$$\det \begin{pmatrix} 1 & 1 & 1 \\ & & \\ & & \end{pmatrix} =$$

- $p = (0,0)$, $q = (1,0)$, $r = (\cos\alpha, \sin\alpha)$, $orientation(p,q,r) = 0$ if $\alpha \in \{0,\pi\}$, , if $0 < \alpha < \pi$, , if $\pi < \alpha < 2\pi$

$$\det \begin{pmatrix} 1 & 1 & 1 \\ & & \\ & & \end{pmatrix} =$$

A Formula for the Orientation Function, II

Proof:

- if $p = q$, $orientation(p, q, r) = 0$ = value of determinant
- so assume $p \neq q$.

$$\det\begin{pmatrix} 1 & 1 & 1 \\ p_x & q_x & r_x \\ p_y & q_y & r_y \end{pmatrix} = \det\begin{pmatrix} 1 & 1 & 1 \\ 0 & q_x - p_x & r_x - p_x \\ 0 & q_y - p_y & r_y - p_y \end{pmatrix} = \det\begin{pmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{pmatrix}$$

$$= ||q-p|| \cdot ||r-p|| \cdot \det\begin{pmatrix} \cos\alpha & \cos\beta \\ \sin\alpha & \sin\beta \end{pmatrix}$$

$$= ||q-p|| \cdot ||r-p|| \cdot (\cos\alpha \sin\beta - \cos\beta \sin\alpha)$$

$$= ||q-p|| \cdot ||r-p|| \cdot \sin(\beta - \alpha)$$

- $\sin(\beta-\alpha) = \begin{cases} 0 & \text{if } \beta \in \{\alpha, \alpha+\pi\} \\ +1 & \text{if } \alpha < \beta < \alpha+\pi\text{, i.e., left turn} \\ -1 & \text{if } \alpha+\pi < \beta < \alpha+2\pi\text{, i.e., right turn} \end{cases}$

Convex Hulls

- a set C is *convex* if with every two points p and q the entire line segment $\overline{pq}$ belongs to C.
- the *convex hull* of a set S is the smallest (with respect to set inclusion) convex set containing S (notation $CH(S)$)

Further Properties

- For $S = \{p, q, r\} \in I\!R^2$: $CH(S)$ = the triangle with vertices p, q, and r.
- Alternatively, $CH(S) = \alpha p + \beta q + \gamma r$, where $\alpha, \beta, \gamma \geq 0$ and $\alpha + \beta + \gamma = 1$.
- For any $S \subseteq I\!R^2$: $CH(S) = \left\{ \sum_{0 \leq i \leq 2} c_i x_i \; ; \; x_i \in S,\; c_i \geq 0,\; \sum_i c_i = 1 \right\}$

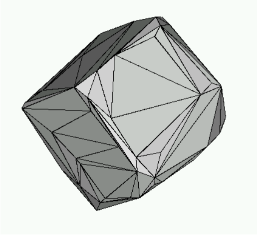
Convex Hull Problem

Version 1: Given a point set S, determine the extreme points of S.

Version 2: Given S, determine the complete boundary description of $CH(S)$.

2d determine extreme points and their cyclic order along the boundary of $CH(S)$ (alternatively: determine vertices and edges and their incidences)

3d determine vertices, edges, and facets and their incidences



Applications

- Mixing natural products to obtain normalized products
- Statistics: Area/Shape of Hull says sth about properties of a point set.
- Gift wrapping.
- 3D Plotters
- Equivalence to other geometric problems (e.g. Delaunay Triangulations)

2d Convex Hulls via Sweep

Compute *upper hull* and *lower hull* separately.

Upper hull = extreme points above the connection between the lexicographycally smallest and largest points.

```
// permute p such that p[0..returnValue] is the upper hull of p
int upperHull(vector<Point>& p) {
  sortLexicographically(p);
  int j = 1;
  for (int i = 2;  i < p.size();  i++) {
    // invariant: p[0..j] is upper hull of p[0]..p[i-1]
    while (!right_turn(p[j-1], p[j], p[i])) j--;
    j++;
    swap(p[j], p[i]);
  }
  return j;
}
```


Running Time

Theorem 1 *The running time of the sweep algorithm for convex hulls is* $\Theta(n \log n)$.

Proof:

- it takes time $O(n \log n)$ to sort the points lexicographically.
- the time to add a point is $O(1 + \text{number of nodes removed from the hull})$.
- any point can disappear from the hull at most once
- hence, the total time to add all points is linear.
- the running time of the algorithm is never better than $n \log n$ since it takes $\Omega(n \log n)$ time to sort the points.

∎

A Lower Bound

Theorem 2 *It takes time* $\Omega(n \log n)$ *in the worst case to compute the convex hull of* n *points.*

Proof:

- Recall: it takes time $\Omega(n \log n)$ to sort n real numbers (or even to determine whether n number are pairwise distinct)
- given n real numbers x_i, $1 \le i \le n$, construct the points $p_i = (x_i, x_i^2)$.
- this takes linear time.
- the points p_i, $1 \le i \le n$, lie on the parabola $y = x^2$.
- we have n extreme points iff the x_i are pairwise distinct.
- given the hull, it takes linear time to determine the number of extreme points.

∎

Detailed References

- deBerg et al: pages 1 – 9 discuss the sweep algorithm
- Klein: pages 159 – 174 discuss the problem definition, the lower bound, the incremental algorithm, the randomized incremental algorithm, sweep algorithm
- LEDAbook: pages 637 – 651 discuss sweep algorithm, incremental construction, randomized incremental construction, implementation details, running times.

2d Hulls by Divide-and-Conquer

- if S contains at most three points, do the obvious.
- sort the points lexicographically
- take the median of S (according to lexicographic order) and use it to split S into two sets S_1 and S_2 of (about) equal size.
- compute $CH(S_1)$ and $CH(S_2)$ recursively
- combine the two hulls by constructing the common tangents
- we will see in the exercises that the tangents can be constructed in time $O(n)$, $n = |S|$.
- total time $T(n) = O(n) + 2T(n/2) + O(n) = O(n \log n)$.

Convex Hulls by Incremental Construction

- consider the points in arbitrary order
- construct hull of the first three points, a triangle
- in order to add a point p
  - determine whether p is inside the current hull (see below)
  - if not, determine all edges visible from p and construct the tangents from p
  - add the tangents, do not (!!!) remove the edges which disappear from the hull, but make them point to the two new edges (reuse *pred*- and *succ*-pointers)
- algorithm is online, i.e., there is no need to know all points at the beginning, and can be made to work in any dimension (see CMS, CGTA 93)

Details of the Walk

- o is a point in the interior of the first triangle
- current hull is available as a circular list
- edges removed from the hull point to their replacements
- let e be an edge of first triangle intersected by the segment $\overline{op}$ (stop, if there is none)
- while e is not an edge of the current hull do
  - if none of the replacement edges of e is intersected by $\overline{op}$, stop.
  - let e be the replacement edge intersected by $\overline{op}$ (I ignore the possibility that $\overline{op}$ passes through the common vertex)

Lemma 2 *If p is outside the current hull, one of the replacement edges is intersected by $\overline{op}$*

Running Time

- the running time of incremental construction is $O(n^2)$, since we visit at most i edges when we insert the i-th point and $\sum_{1 \le i \le n} i \le n^2$
- in the worst case the running time of incremental construction is quadratic
- if the points are inserted in random order the running time is $O(n \log n)$.

Expected Running Time of IC

Theorem 3 *The expected running time of the incremental algorithm for convex hulls is $O(n \log n)$ if all insertion orders are equally likely.*

Assumption of the theorem is satisfied,

- when the points in S are generated according to a probability distribution for points in the plane.
- when the points in S are randomly permuted before the incremental construction process is started: *Randomized incremental construction (RIC).*

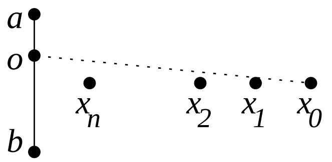
- RIC was invented by Ken Clarkson and Peter Shor in the late 80s
- very powerful paradigm
- the books by Boissonnat-Yvinec and Mulmuley center around the method
- we assume for the analysis that no three points in S are collinear; this is called a *general position assumption*; LEDAbook gives a proof for the general situation

RIC: Analysis

- the time for an insertion is
  1. the time for the walk
  2. the time spent to find all visible edges of the current hull
  3. the time spent to add the tangents and to make the “removed” hull edges point to their replacements

- let k be the number of visible edges of the current hull
- time for (2) and (3) is $O(1 + k)$.
- every edge is removed at most once from the hull
- thus, the total time for (2) and (3) is $O(n)$.

Analysis of Walk, A Special Case



assume: a and b are inserted first, x_0 is inserted last, and $x_1, \ldots, x_n$ are inserted in random order

- let o be a point on the segment $\overline{ab}$
- how many *constructed edges* will the segment $\overline{ox_0}$ intersect?
- edges (a, x_i) and (x_i, b) are constructed **iff** x_i is inserted before $x_1, \ldots x_{i-1}$
- probability for this to happen is $\frac{1}{i}$.
- expected time of walk for x_0 is

$$\sum_{1 \le i \le n} \frac{1}{i}$$

Analysis of Walk, General Case

We analyse the cost of the walk along $\overline{op}$ under the following assumptions:

- the first three points inserted are fixed (and hence o is well-defined)
- the remaining points are inserted in random order.

- how much action is there on the segment $\overline{op}$?
- run the movie backwards (Raimund Seidel) and ask yourself: how often does the edge of the current hull intersected by $\overline{op}$ change?
- the edge changes if one its endpoints is removed.
- when the current hull is the hull of i points, the probability for this to happen is $2/i$.

- the expected cost of the walk is therefore

$$\sum_{1 \le i \le n} \frac{2}{i}$$

- since the expected cost of each walk is $O(\log n)$, the expected cost of randomized incremental construction is

$$O(n \log n)\ .$$

- **Remarks:** running time of RIC is $O(n)$ in the best case.
- RIC is fast when number of extreme points of S and every random subset of S is small

IC versus RIC

				IC		RIC	
K	n	Gen	V	Random	Sorted	Random	Sorted
S	4000	0.29	18	0.09	0.27	0.11	0.13
S	8000	0.64	23	0.16	0.76	0.22	0.21
S	16000	1.34	29	0.33	2.53	0.42	0.41
D	4000	0.27	59	0.1	0.45	0.11	0.1
D	8000	0.59	66	0.17	1.26	0.23	0.2
D	16000	1.25	87	0.43	3.48	0.5	0.41
C	4000	9.32	4000	0.32	15.57	0.34	0.37
C	8000	18.87	7995	0.7	65.93	0.75	0.71
C	16000	37.62	1.599e+04	1.47	253.4	1.53	1.57

S = n points in the square with side length $2R$, $R = 16000$

D = n points in the disk with radius R centered at the origin

C = n points (approximately) on the circle with radius R centered at the origin

random = points in random order

sorted = lexicographically sorted input

RIC randomly permutes input and then calls IC

Sweep versus RIC

				Sweep		RIC	
K	n	Gen	V	Random	Sorted	Random	Sorted
S	20000	1.72	25	1.68	1.54	0.55	0.55
S	40000	3.77	29	3.6	3.26	1.26	1.43
S	80000	7.92	31	7.72	6.98	2.06	2.07
D	20000	1.62	106	1.75	1.59	0.55	0.56
D	40000	3.49	109	3.76	3.33	1.17	1.25
D	80000	7.32	152	8	7.02	2.42	2.58
C	20000	47	1.999e+04	1.82	1.67	2.13	2.12
C	40000	94.68	3.994e+04	3.96	3.57	4.46	4.41
C	80000	188.8	7.979e+04	8.6	7.78	10.31	10.04

columns and rows have the same meaning as on preceding slide

observe how well RIC does on inputs where the number of extreme points is small.