Geometric Computation: Introduction

Piotr Indyk

Welcome to 6.838 !

- Overview and goals
- Course Information
- Syllabus
- 2D Convex hull
- Signup sheet

Geometric Computation

- Geometric computation occurs everywhere:
  - Geographic Information Systems (GIS): nearest post office, map overlays
  - Simulation: collision detection
  - Computer graphics: visibility tests for rendering, lighting simulations
  - Computational drug design: spatial indexing
  - Computer vision: pattern matching
  - Robotics: motion planning, map construction and localization
  - …

Computational Geometry

- Started in mid 70's
- Focused on design and analysis of algorithms for geometric problems
- Many problems well-solved, e.g., Voronoi diagrams, convex hulls
- Many other problems remain open

Course Goals

- Introduction to Computational Geometry
  - Well-established results and techniques
  - New directions

Course Information

- 3-0-9 H-level Graduate Credit
- Grading:
  - 4 problem sets (see calendar):
  - In each PSet:
    - Core component (mandatory): 6.046-style
    - Two optional components:
      - More theoretical problems
      - Java programming assignments
  - Can collaborate, but solutions written separately
  - No midterm/final J
- Prerequisites: understanding of algorithms and probability

Syllabus

- Part I - Classic CG:
  1. 2D Convex hull
  2. Segment intersection
  3. LP in low dimensions
  4. Polygon triangulation
  5. Range searching
  6. Point location
  7. Arrangements and duality
  8. Voronoi diagrams
  9. Delaunay triangulations
  10. Convex hulls in 3D
  11. Binary space partitions
  12. Motion planning and Minkowski sum

  Use “Computational Geometry: Algorithms and Applications” by de Berg, van Kreveld, Overmars, Schwarzkopf (2nd edition).

Syllabus ctd.

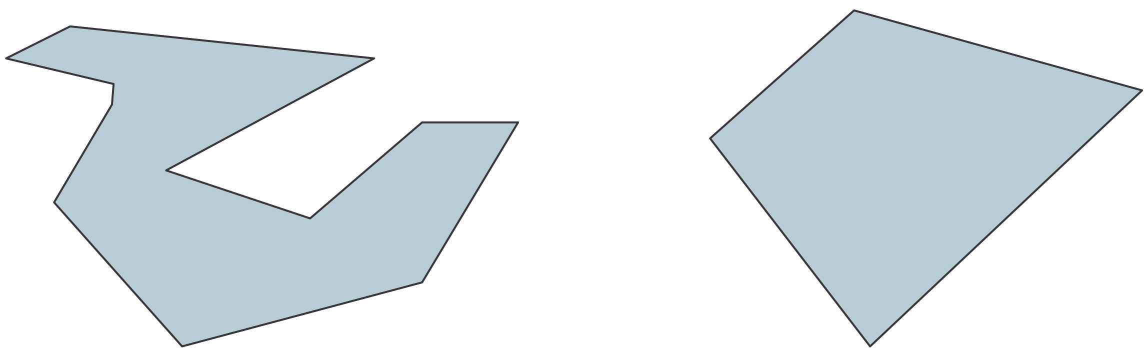
- Part II - New directions:
  13. Folding
  14. Quad-trees
  15. Kinetic algorithms
  16. LP in higher dimensions
  17. Closest pair in low dimensions
  18. Approximate nearest neighbor in low dimensions
  19. Approximate nearest neighbor in high dimensions: LSH
  20. Low-distortion embeddings
  21. Low-distortion embeddings II
  22. Geometric algorithms for external memory
  23. Geometric algorithms for streaming data
  24. Combinatorial geometry
  25. Exciting topic X
  26. Conclusions

Fall’03 vs Fall’01

- Less Computer Graphics
- More Computational Geometry
- Talks given by the instructor
- PS4 will happen
- Thanks to Seth Teller and the students

Convexity

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

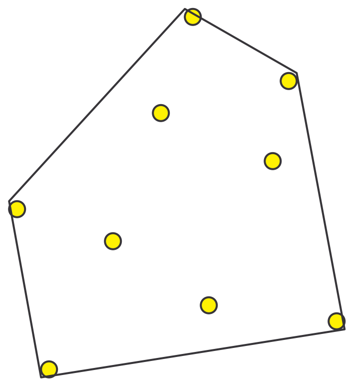


Convex hull

- What is a convex hull of a set of points P ?
  - Smallest convex set containing P
  - Union of all points expressible by a convex combination of points in P, i.e. points of the form

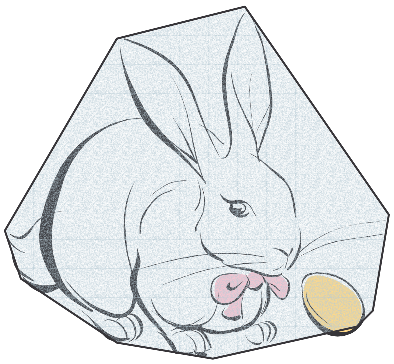
$$\Sigma_{p\in P}\, c_p{*}p,\ c_p\geq 0,\ \Sigma_{p\in P}\, c_p=1$$

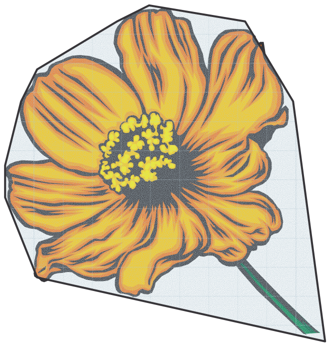
- Definitions not suitable for an algorithm



2D Convex Hull

- Motivation:
  - Collision detection
  - Natural method for shape simplification
  - Relation to Voronoi diagram (in 3D)
  - Illustrates many techniques and issues





Computational Problem

- Given $P \subset R^2$, $|P|=n$, find the *description* of CH(P)
  - CH(P) is a convex polygon with at most n vertices
  - We want to find those vertices in clockwise order
- Design fast algorithm for this problem
- We assume all points are distinct (otherwise can sort and remove duplicates)

Naive approach

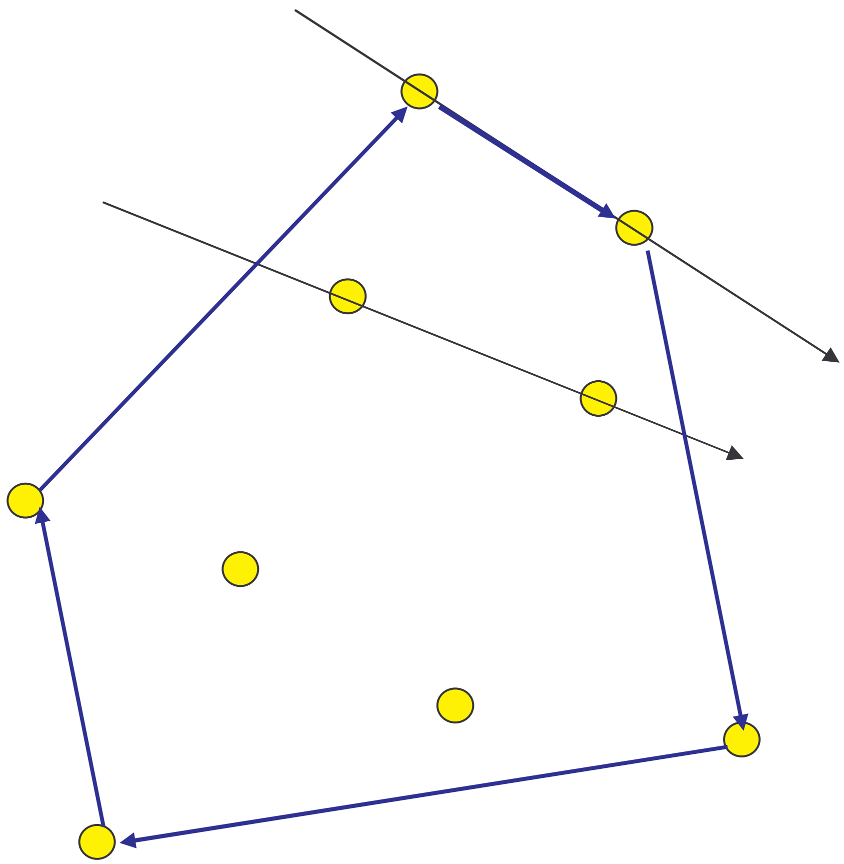
1. For all pairs (p,q) of points in P

/* Check if p→q forms a boundary edge

   A. For all points $r \in P-\{p,q\}$:
      - If r lies to the left of directed line p→q, then go to Step 2
   B. Add (p,q) to the set of edges E

2. Endfor
3. Order the edges in E to form the boundary of CH(P)

Example



Details

- How to test if r lies to the left of a directed line $p \rightarrow q$?
  - Basic geometric operation
  - Reduces to checking the sign of a certain determinant
  - Constant time operation
- How to order the edges in E ?
  - Sort

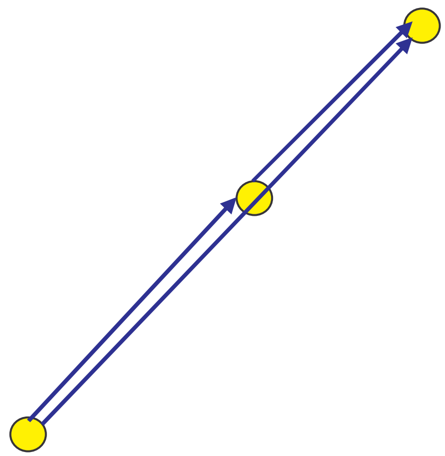
Analysis

- Outer loop: $O(n^2)$ repetitions
- Inner loop: $O(n)$ repetitions
- Total time: $O(n^3)$

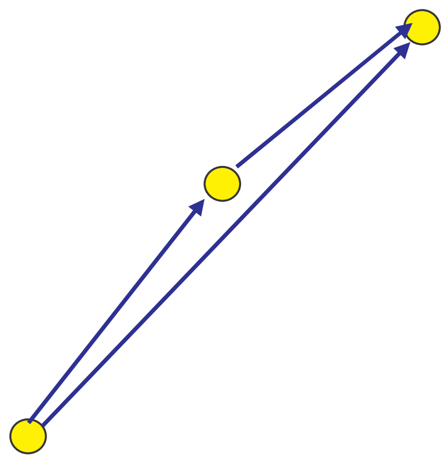
Problems

- Running time pretty high
- Algorithm does weird things:
  - What 3 points are collinear ? (degeneracy)
  - What 3 points are near-collinear ? (robustness)
- The last issue highly non-trivial
- Many ways of dealing with it:
  - Higher precision
  - Arbitrary precision
- Our approach: sweep it under the carpet

Collinear points

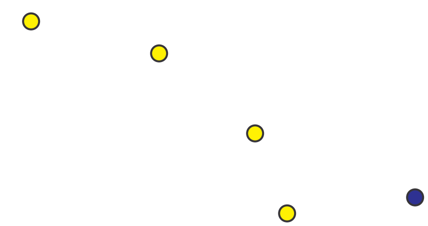


Nearly collinear points



Andrews algorithm

- Convexify(S,p)
  - While $t=|S|\geq 2$ and p left of line $s_{t-1}\rightarrow s_t$, remove s_t from S
  - Add p to the end of S
- Incremental-Hull(P)
  - Sort P by x-coordinates
  - Create $U=\{p_1\}$
  - For i=2 to n
    - Convexify(U,p_i)
  - Create $L=\{p_n\}$
  - For i=n-1 downto 1
    - Convexify(L,p_i)
  - Remove first/last point of L, output U and L



Animation

[Daniel Vlasic's CH Animation]

Issues

- Points with the same x-coordinate
- Modification: Sort by x and then by y
- Solves the degeneracy problem
- Robustness:
  - Still an issue
  - But the algorithm outputs closed polygonal chain

Analysis

- Sorting: O(n log n)
- Incremental walk: O(n)
- Altogether: O(n log n)