Geometric Computation: Introduction

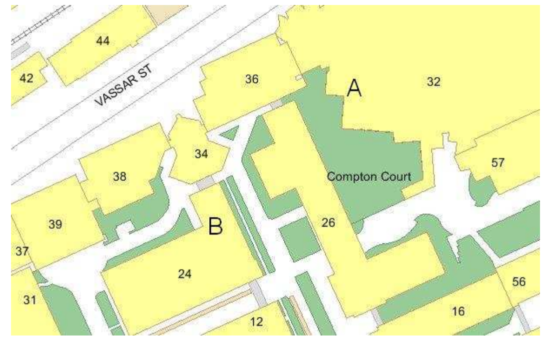
Piotr Indyk

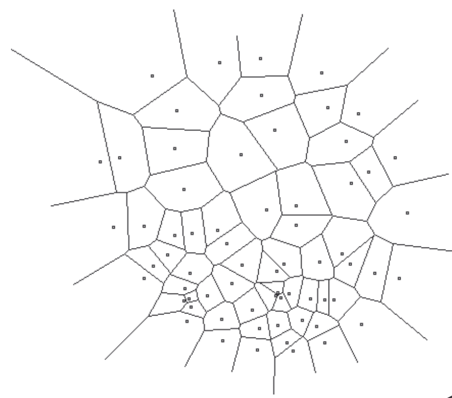
Welcome to 6.838 !

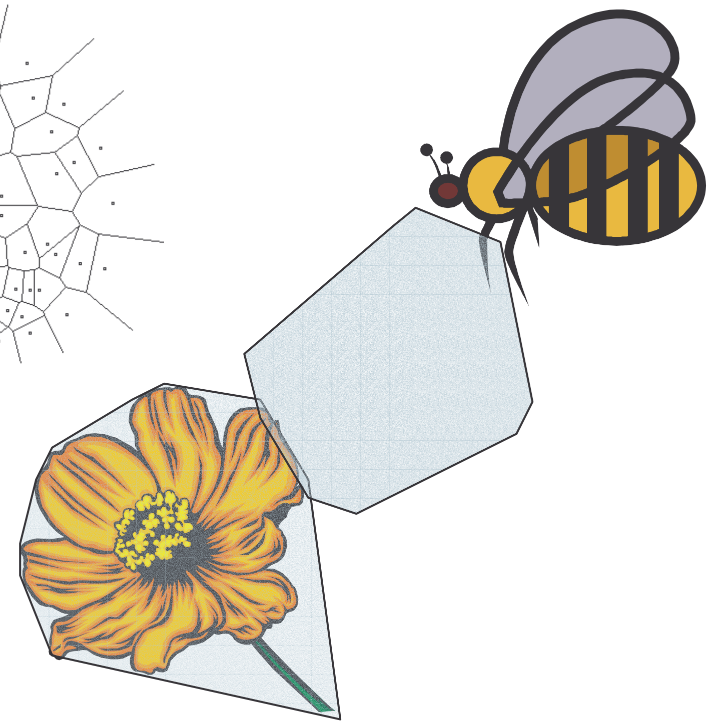
- Overview and goals
- Course Information
- 2D Convex hull
- Signup sheet

Geometric Computation

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







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

Syllabus

- Part I - Classic CG:
  - 2D Convex hull
  - Segment intersection
  - LP in low dimensions
  - Polygon triangulation
  - Range searching
  - Point location
  - Arrangements and duality
  - Voronoi diagrams
  - Delaunay triangulations
  - Binary space partitions
  - Motion planning and Minkowski sum

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

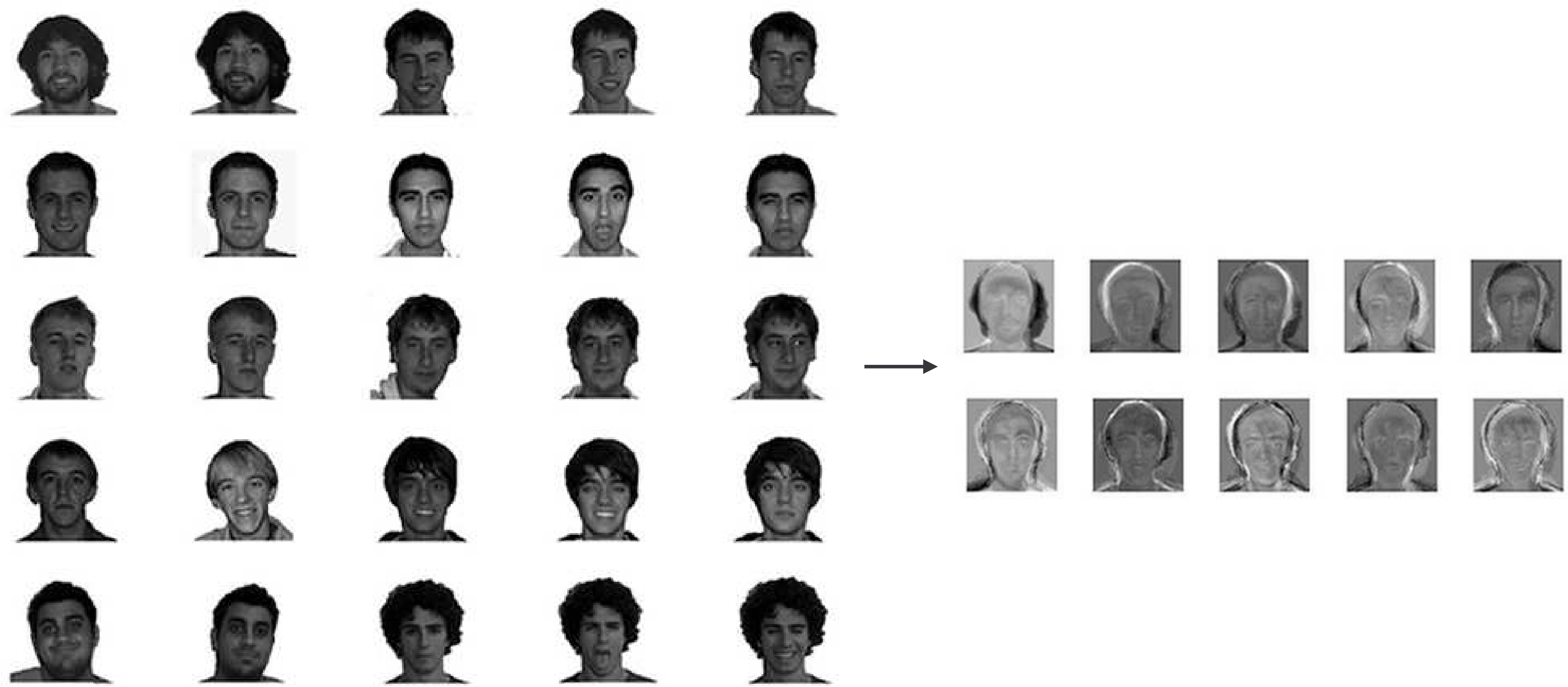
Syllabus ctd.

- Part II - New directions:
  - LP in higher dimensions
  - Closest pair in low dimensions
  - Approximate nearest neighbor in low dimensions
  - Approximate nearest neighbor in high dimensions: LSH
  - Low-distortion embeddings
  - Low-distortion embeddings II

  - Geometric algorithms for external memory
  - Geometric algorithms for streaming data

  - Kinetic algorithms
  - Pattern matching
  - Combinatorial geometry
  - Geometric optimization
  - Conclusions

Higher dimensions - eigenfaces



=?

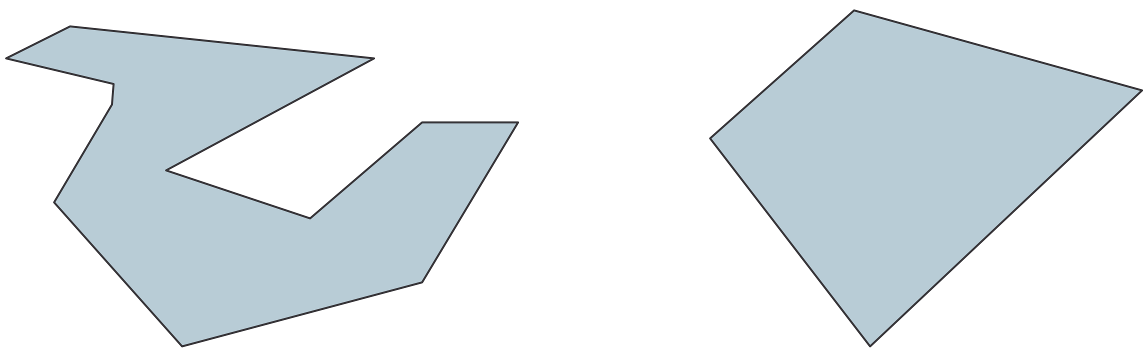
Course Information

- 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 (6.046 level)

Questions ?

Convexity

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

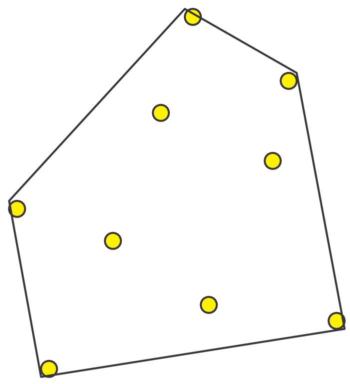


Convex hull

- 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 p of the form

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

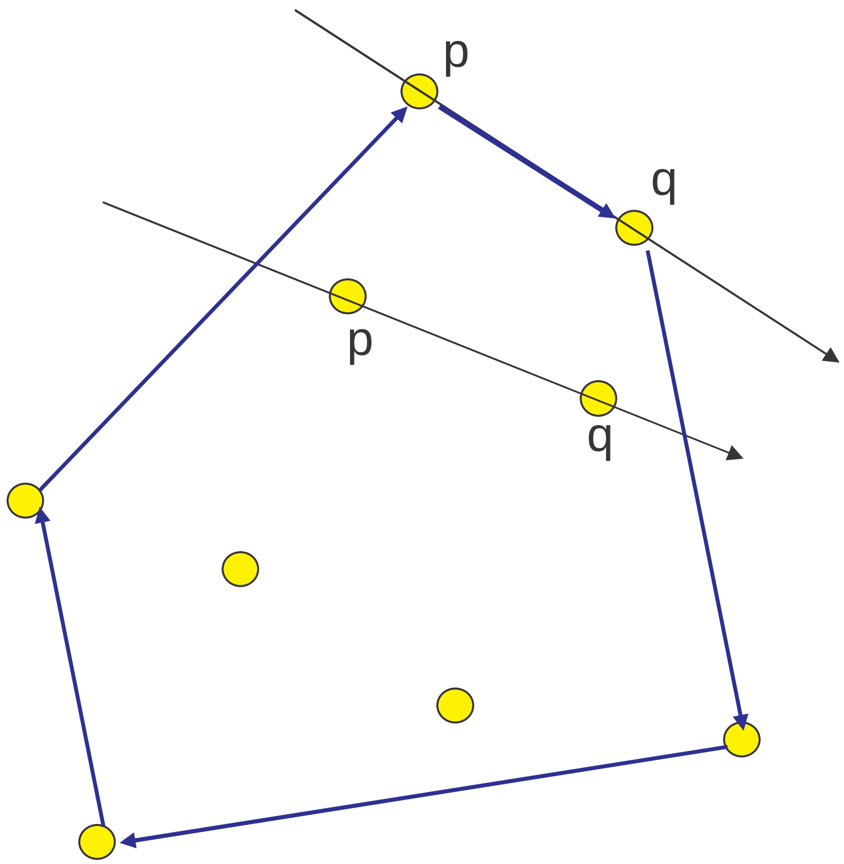
- Definitions not suitable for an algorithm



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)
- Any algorithms ?

Simple approach



Simple approach

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

/* Check if $p \rightarrow q$ forms a boundary edge */

   A. For all points $r \in P-\{p,q\}$:
      – If r lies to the left of directed line $p \rightarrow q$, 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)

Details

- How to test if r lies to the left of a directed line p→q ?
  - Basic geometric operation
  - Reduces to checking the sign of a certain determinant
  - Constant time operation

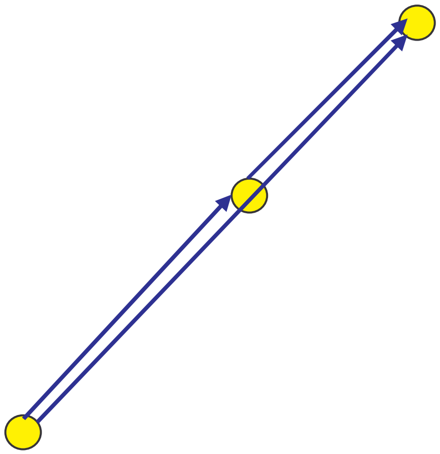
Analysis

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

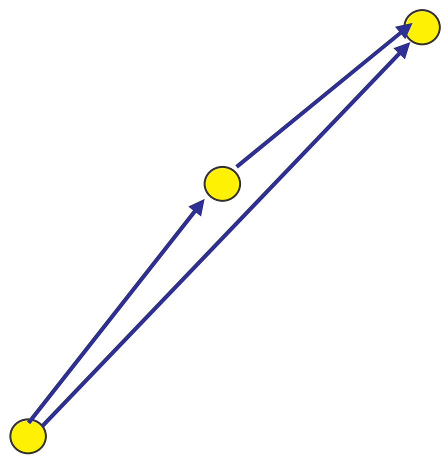
Problems

- Running time pretty high
- Algorithm can do strange things:
  - What 3 points are collinear ? (degeneracy)
  - What 3 points are near-collinear ? (robustness)

Collinear points



Nearly collinear points

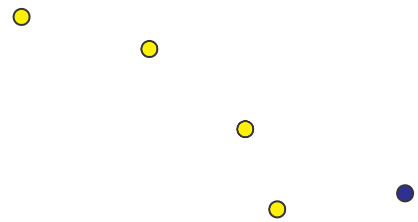


Problems

- Issues:
  - Degeneracy: input is “special”
  - Robustness: implementation makes round-off errors
- Solutions:
  - Degeneracy: correct algorithm design
  - Robustness: higher/arbitrary precision
- Our solution: typically, sweep the issue under the carpet

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)