A Minimalist’s Implementation of the 3-d Divide-and-Conquer Convex Hull Algorithm

Timothy M. Chan

Presented by Dana K. Jansens
Carleton University
Simple Polygons

- Polygon = A consecutive set of vertices and edges that form a closed path
- Simple = No edges cross
- Have a *Boundary* and *Interior*
Convex Polygons

For all pairs of points \( x \) and \( y \) inside a polygon, the line segment \( (x, y) \) does not leave the polygon.
2-d Convex Hull

- Given a set $P$ of points in the plane
- The smallest convex polygon that contains all of $P$
3-d Convex Hull

- Given a set $P$ of points in $\mathbb{R}^3$
- The smallest convex **polyhedron** that contains all of $P$
Lower/Upper Hull

- A convex hull can be decomposed into a *lower* and *upper* hull.
- Edges (2d) or Faces (3d) that can be seen from above or below
- Vertical ray (from $+\infty$)
  - enters polygon through *upper* hull
  - leaves through *lower* hull
Lower/Upper Hull

- An edge will either be in the upper hull *or* in the lower hull
- Construct a CH by finding a lower and upper hull and putting them together
Demo
Our Goal

- Given a set of points in $\mathbb{R}^3$
- Construct a 3D lower convex hull
- Strategy:

  *Divide and Conquer*
Divide and Conquer

- Given an input set $S$, of size $n$, solve some global problem on $S$.
- Need two things:
  - Know how to solve the problem for a set of some constant size
  - Know how to merge two solutions of any size together
Merge Sort

- Sorting: How to sort \( n \) numbers?
- What do we need to know:
  - How to sort 1 number
  - How to merge two sorted lists of numbers into one
Merge Sort

- Given sorted lists $A$ and $B$, output a single sorted list $O$ containing all elements of $A$ and $B$.
  - Read the first element of $A$ and $B$
  - Take the smaller of the two
  - Remove it from the input list
  - Add it to the end of $O$
Merge Sort

- Repeat this process until only one sorted sequence is left
We perform similar operations on convex hulls.
Divide and Conquer 3-d Hull

- We need to know two things:
  - How to construct a 3-d lower hull from a constant sized set of points (e.g. \( \leq 5 \) points)
  - How to merge two 3-d lower hulls together
Merging 2-d Convex Hulls

- Given two 2-d lower hulls, separated along the $x$-axis
- Find a bridge edge between the two hulls
- In 3-d, requires finding many bridge faces
Merging 3-d Convex Hulls

- Given two 3-d convex hulls, how do we merge them?
- Project them into 2-d and merge them there
- The third dimension becomes “time”
- 2-d projection point set moves through time
For each point $p_i$, define:

- $p_i'(t) = (x_i, z_i - ty_i)$

- $x$-coord stays the same

- $y$-coord is the 3-d $z$-coord

- $y$-coord changes through time, with a constant velocity related to its 3-d $y$-coord
2-d Projection

- As time $t$ moves from $-\infty$ to $\infty$, the points will move vertically with constant velocity.
- The points start at one extreme and move through to the other.
- Over time, the lower hull of the 2-d point set will change.
2-d Projection

- By the projection, where $y' = z - ty$
  - A point is a vertex of 3-d lower hull \emph{iff}
  - The 2-d projection of the point:
    - lies on a line $y' = sx + b$ with all other points above it
    - is a vertex of the 2-d lower hull for some time $t$
Projection Movie

- Think of the 2-d projection as a movie
- By playing the movie of 2-d projection through time, and watching which vertices join the lower hull
- We can construct the 3-d lower hull
Merging

- We want to be able to merge two 3-d hulls
- By merging, we create a 2-d movie
- So, we are given the 3-d hulls, but also their 2-d movies
- Play their movies back together, and merge them together at each time $t$
Demo
Each point’s 2-d projection moves with constant velocity, and does not change direction.

Thus during one merge step, it will be added to/removed from the 2-d hull at most once each.

Therefore there are $O(n)$ “events”
Quick Analysis

○ The next “event” time for the projection is computed in $O(1)$ time, using the two previous movies

○ This gives $O(n)$ time to perform one merge operation

○ Thus total run time: $T(n) = 2T(n/2) + O(n) = O(n \log n)$
Remarks

- While we describe the algorithm in terms of time..
- The time is precisely a third coordinate, making the algorithm a space sweep.
We can use this algorithm to answer 3-d extreme point queries

Given $s$ and $t$, find a point $(x_i, y_i, z_i)$ minimizing $z_i - sx_i - ty_i$

Remember the 2-d hull at time $t$

Search the lower hull for the point: $O(\log n)$ time
Remarks

- Gives a $O(\log n)$ time query time for 2-d nearest neighbours problem
- Given a point $(a, b)$ in $\mathbb{R}^2$, find $(x_i, y_i)$:
  - Minimize $\sqrt{(x_i - a)^2 + (y_i - b)^2}$
  - Minimize $z_i - 2ax_i - 2by_i$ with $z_i = x_i^2 + y_i^2$
- The data structure uses $O(n \log n)$ space