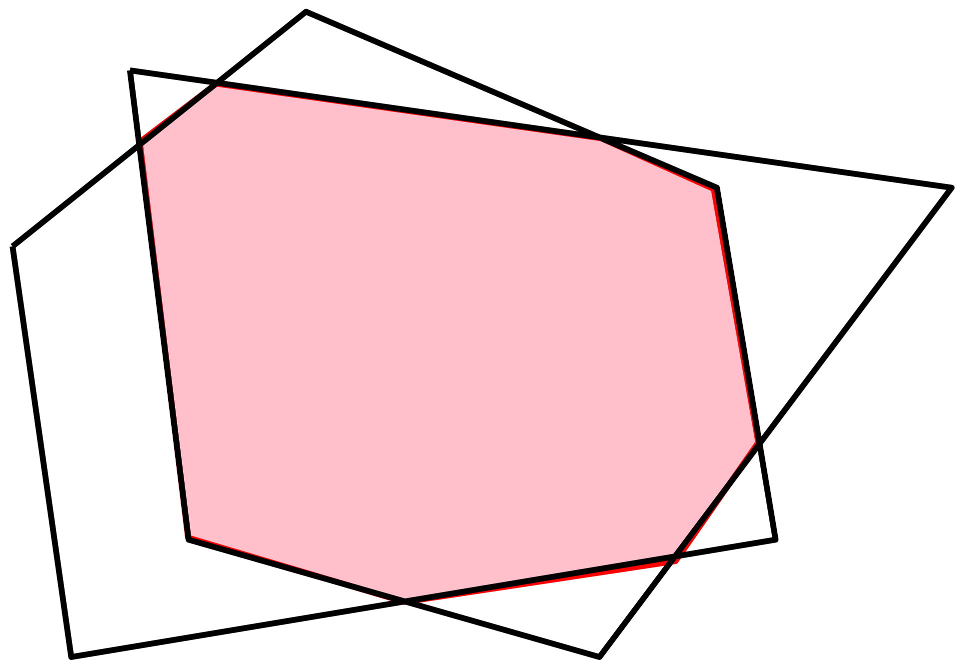
A Simpler Linear-Time Algorithm for Intersecting Two Convex Polyhedra in Three Dimensions

Timothy Chan

(U. of Waterloo)

The Problem

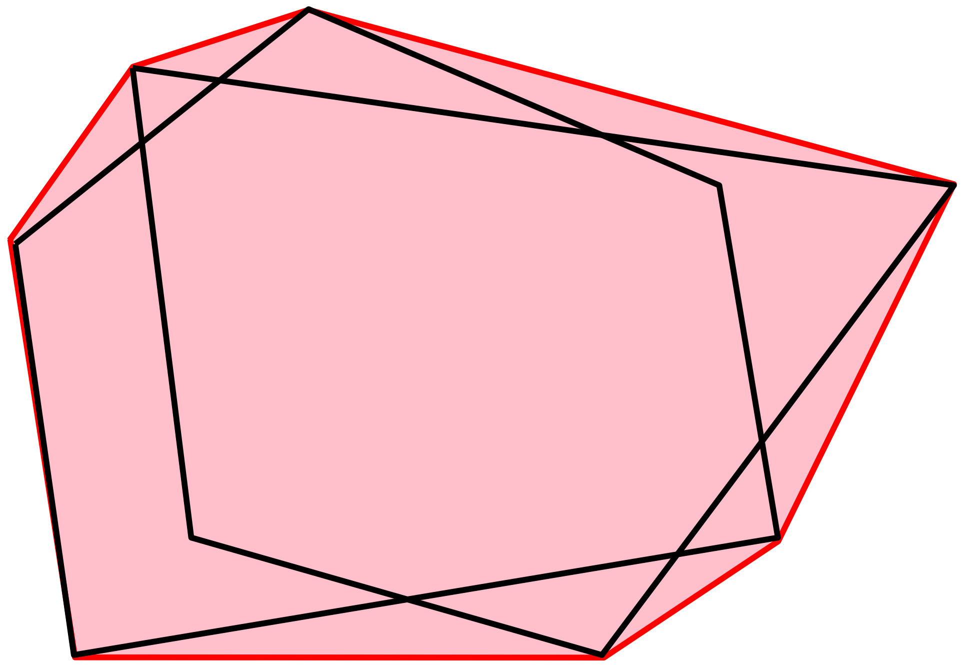
Given 2 convex polyhedra in 3D, compute their intersection



(apologies for pictures in 2D!)

Equiv. Dual Problem

Given 2 convex polyhedra in 3D,
compute their convex hull



Previous Alg'ms: Special Cases

- Shamos&Hoey'75: $O(n)$ time for merging vertical separated 2D Delaunay triangulations
- Preparata&Hong'77: $O(n)$ time for merging vertically separated 3D convex hulls
- Kirkpatrick'79: $O(n)$ time for merging arbitrary 2D Delaunay triangulations

Previous Alg’ms: General Case

- Chazelle’89: $O(n)$ time for arbitrary 3D convex polyhedra. . .

  but it’s complicated (25-page paper!)

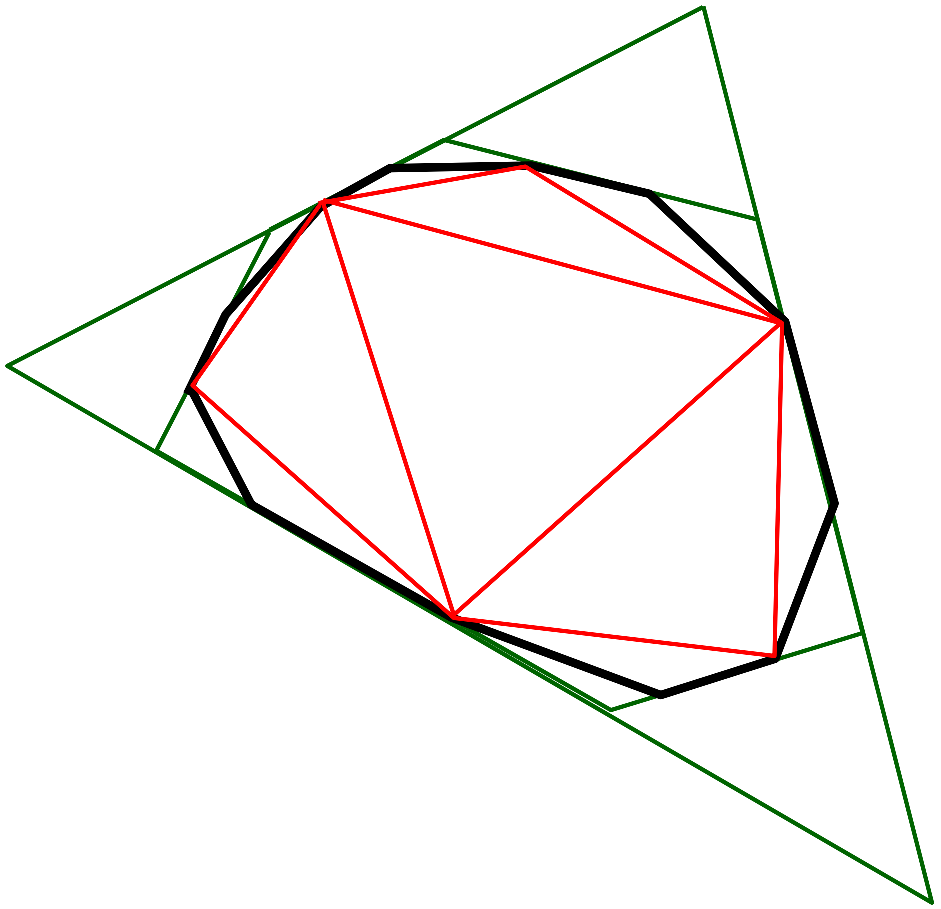
As Chazelle put it. . .

(http://www.cs.princeton.edu/˜chazelle/linernotes.html)

“There is a genre of geometric problems whose solutions are conceptually trivial yet fiendishly tricky to implement. . . Oddly enough, intersecting two n-vertex convex polyhedra is not. In fact, I know few geometric problems whose exploration offers such breathtaking vistas. Picture stitching together the two polyhedra’s boundaries as the act of hiking in a valley surrounded by tall mountains. The valley, as it happens, consists of disjoint parts that can be reached only by climbing the neighboring peaks. Bad idea! Ah, but from a dual perspective, you see, the structure of the mountain peaks very much resembles that of the valley. So say hello to primal–dual trekking. Soon, though, we need tunnels across the mountains to use as shortcuts, a job we subcontract to the Dobkin–Kirkpatrick hierarchy. Alas, we can’t dig too deeply into the mountains lest we get hit with an extra log factor, so the hiking metaphor stops there. . . ”

Chazelle's Approach

- Navigate in the Dobkin–Kirkpatrick hierarchies of the given convex polyhedra... in both primal & dual space



“. . . Salvation comes in the shape of a ‘cloning’ recursion. What’s that? In the world of algorithms, it is useful to distinguish among three types of recursion: binary search gives your the nonbranching kind; quicksort gives you the no-clone branching sort (no data replication); linear selection (median finding) gives you the cloning branching variety (with data replication). This last flavor is the tastiest of them all because it features two competing exponential growths (as in fractional cascading). Confession time: I’ve always had a weakness for linear selection and its sky-high coolness-to-simplicity ratio. I’ll admit that RSA and FFT have enviable ratios, too, but both owe their sparkle to algebra whereas median finding is pure algorithmic sizzle. Just as Hume is said to have interrupted Kant’s ‘dogmatic slumber,’ so linear selection interrupted the sorting snoozefest I once endured in Algorithms 101.”

Chazelle's Approach (Cont'd)

- Recurrence:

$$T(n) = 4\,T(\delta n) + O(n) \;\Rightarrow\; T(n) = O(n)$$

[Martin'91 (Master's thesis): "improves" this to $T(n) = 2\,T(\delta n) + O(n) \;\Rightarrow\; T(n) = O(n)$]

New Alg'm

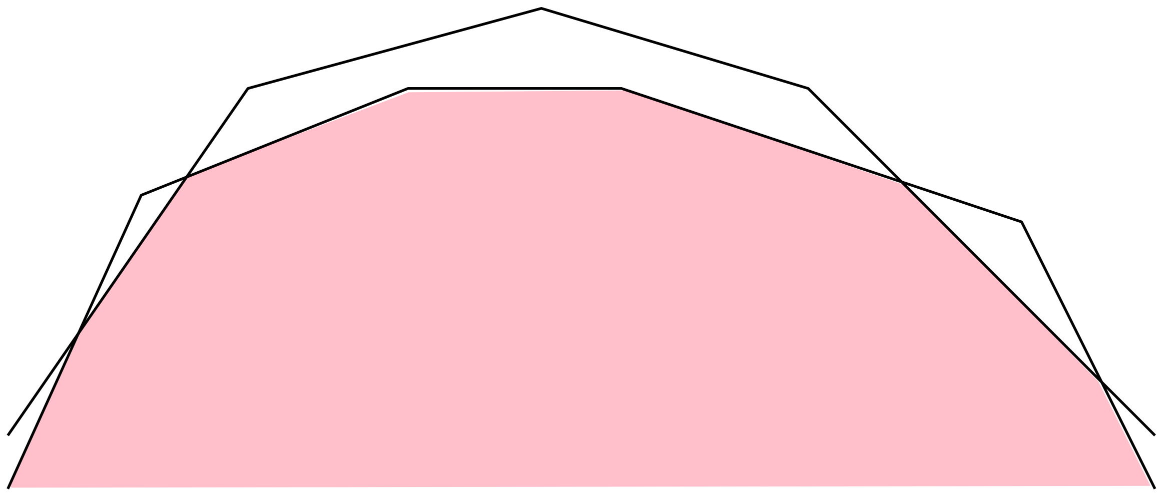
- Much simpler (6-page paper!)
- More "mundane" recurrence:

$$T(n) = T(\delta n) + O(n) \;\Rightarrow\; T(n) = O(n)$$

- Also use Dobkin–Kirkpatrick hierarchies but stays in primal space
- Borrow more "modern" ideas from randomized geometric alg'ms (canonical triangulations, conflict lists, . . .)

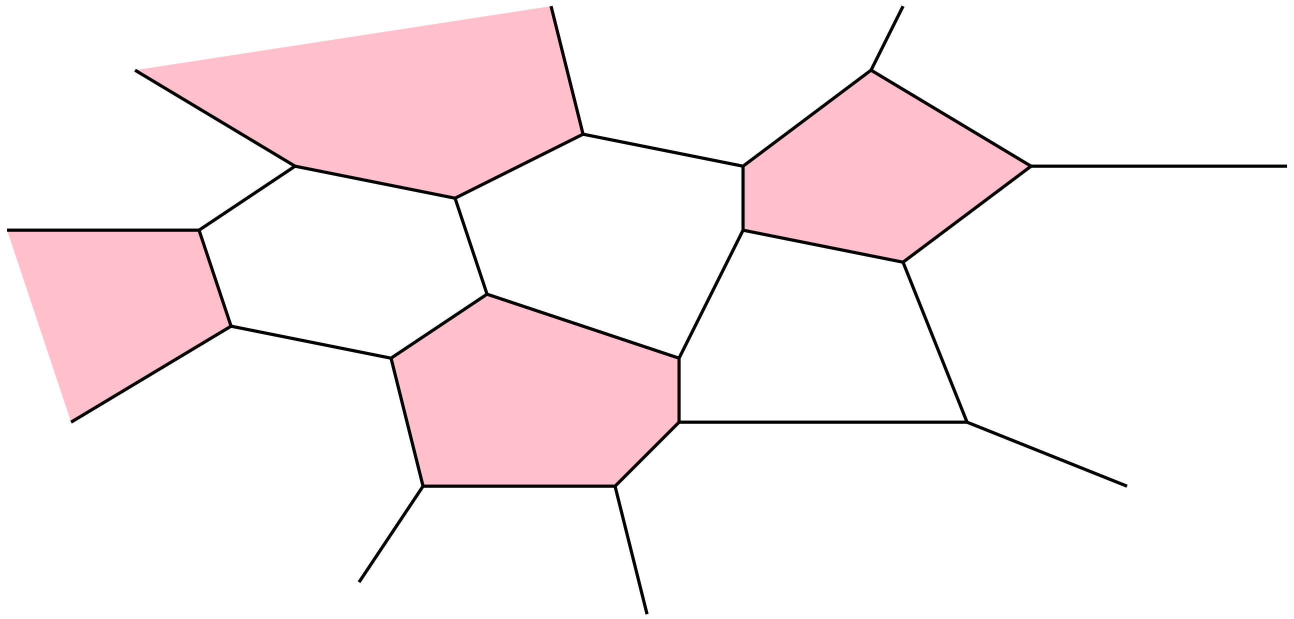
Setup

- Work with lower envelope $\mathcal{P}(A)$ of n (nonredundant) planes A
- Problem: given $\mathcal{P}(A)$ and $\mathcal{P}(B)$, compute $P = \mathcal{P}(A) \cap \mathcal{P}(B)$



Idea 0: Dobkin–Kirkpatrick

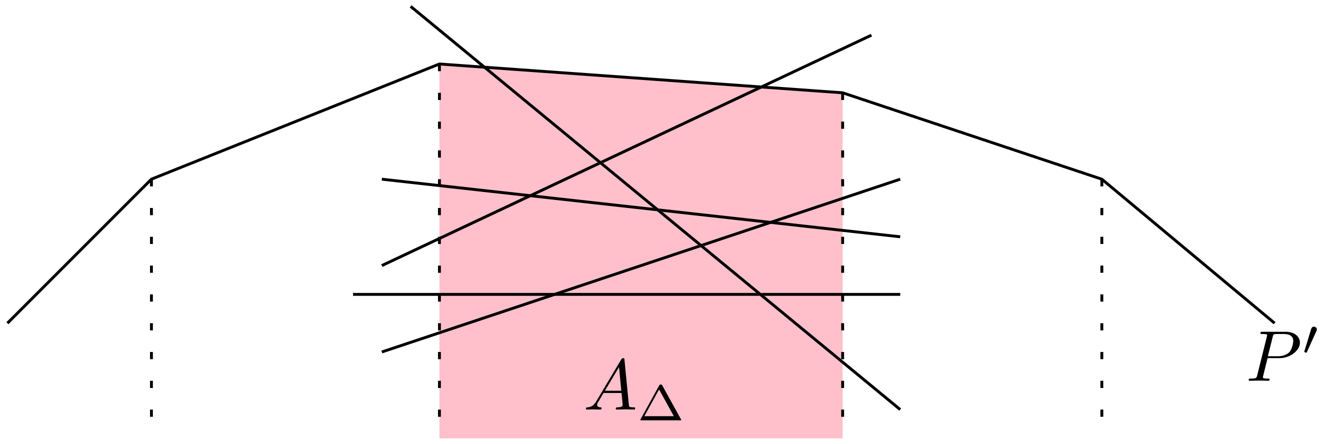
- Find independent set I of faces of $\mathcal{P}(A)$ with $|I| \geq n/24$, face size ≤ 11
- Set $A' = A \setminus I$ (B' similar)
- Recursively compute $P' = \mathcal{P}(A') \cap \mathcal{P}(B')$



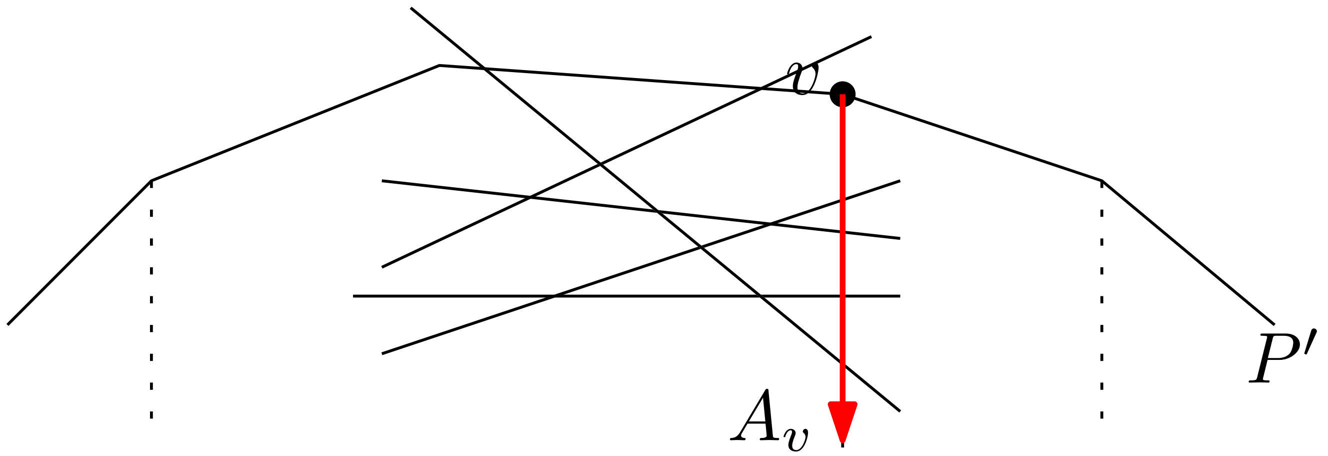
Idea 1: Conflict Lists

- Consider canonical triangulation $\mathcal{T}(P')$
- For each cell Δ of $\mathcal{T}(P')$:
  - generate conflict list

    $A_\Delta = \{$ all planes of A intersecting Δ $\}$ (B_Δ similar)
  - compute $\mathcal{P}(A_\Delta) \cap \mathcal{P}(B_\Delta)$ inside Δ in $O(1)$ time
- Glue results to get $P = \mathcal{P}(A) \cap \mathcal{P}(B)$ & done!

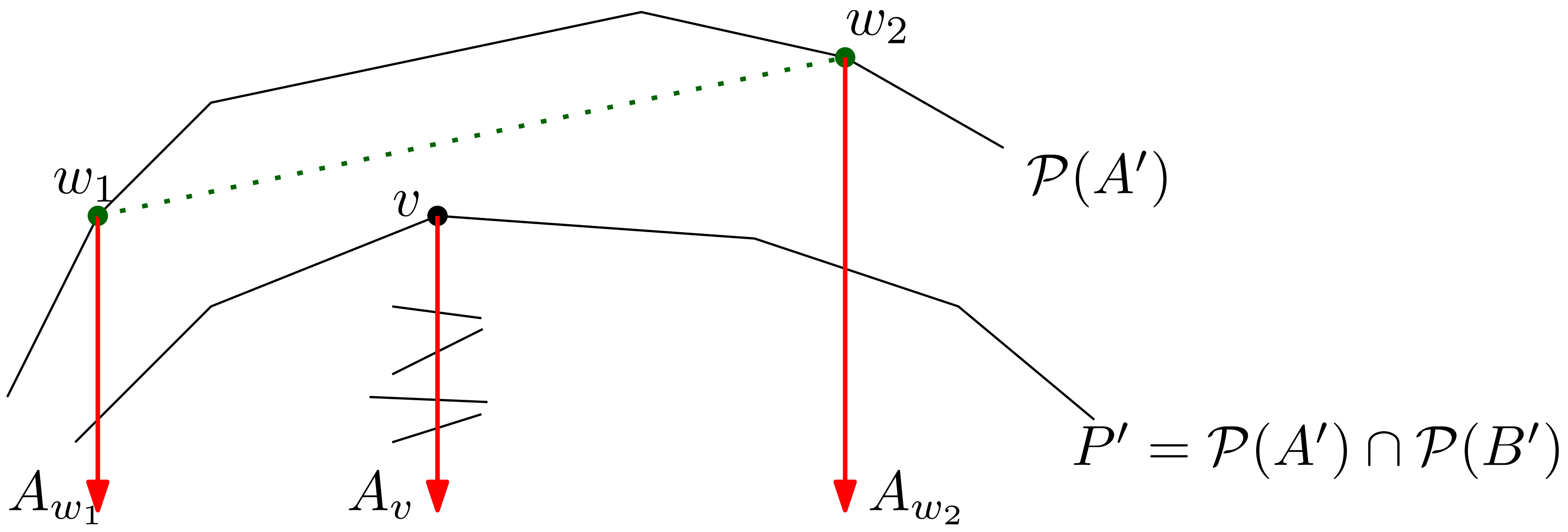


- But how to generate conflict lists?
- Suffices to generate the vertex-conflict lists
  $A_v = \{$ all planes of A below v $\}$
  for the 3 vertices v of Δ…



Idea 2: Using Witnesses

- For each vertex v of P':
  - v is below $\mathcal{P}(A') \quad \Rightarrow \; \exists$ vertices w_1, w_2, w_3 of $\mathcal{P}(A')$ s.t. v is below $\triangle w_1 w_2 w_3$
  - Assume we are given these "witnesses" w_1, w_2, w_3
  - Can generate the vertex-conflict list A_v from $A_{w_1}, A_{w_2}, A_{w_3}$, which are trivial, in $O(1)$ time



- But how to find witnesses?
- Suffice to describe how to get new witnesses for P from old witnesses for P'...

Idea 3: Finding New Witnesses

- For each vertex v of P:
  - say v is in cell Δ of $\mathcal{T}(P')$
  - search for new witnesses for v "locally" among the following candidate vertices:
    - $*$ old witnesses for the 3 vertices of Δ (# ≤ 9)
    - $*$ vertices of $\mathcal{P}(A)$ on $\{$ planes of I that are below old witnesses of the 3 vertices of Δ $\}$ (# $\leq 9 \times 11$)

    in $O(1)$ time [see paper for correctness proof...]
- Really done!

Finale

- Recurrence:

$$T(n) = T(\tfrac{23}{24}n) + O(n) \;\Rightarrow\; T(n) = \boxed{O(n)}$$

- Moral of the story: old “solved” problems may still be worth a second look. . .

- An open problem: merge 2D additively weighted Voronoi diagrams in $O(n)$ time?