OI-OPPA. European Social Fund
Prague & EU: We invest in your future.

# CONVEX HULL IN 3 DIMENSIONS

**PETR FELKEL**

**FEL CTU PRAGUE**

**felkel@fel.cvut.cz**

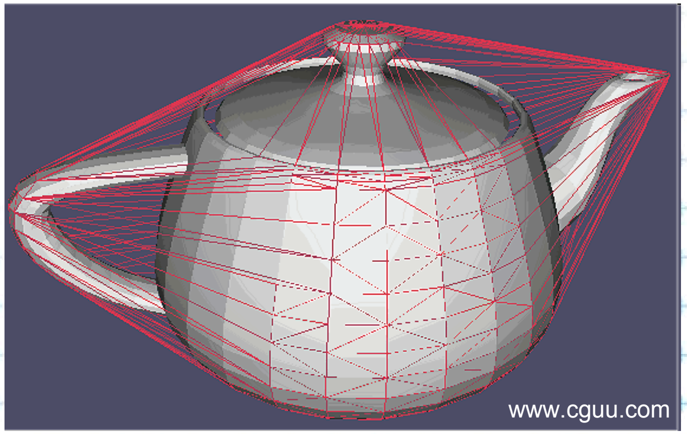
https://cw.felk.cvut.cz/doku.php/courses/a4m39vg/start

**Based on [Berg], [Preparata], [Rourke] and [Boissonnat]**

**Version from 8.11.2012**

# Talk overview

- Lower bounds for convex hull in 2D and 3D
- Other criteria for CH algorithm classification
- Recapitulation of CH algorithms
- Terminology refresh
- Convex hull in 3D
  - Terminology
  - Algorithms
    - Gift wrapping
    - D&C Merge
    - Randomized Incremental



# Lower bounds for Convex hull

- O(*n* log *n*) in $E^2$, $E^3$
- O(*n h*), where h is number of CH facets
  – output sensitive algs.
- O(*n*) for sorted points and for polygon
- O(log *n*) for new point insertion in online algs.

# Other criteria for CH algorithm classification

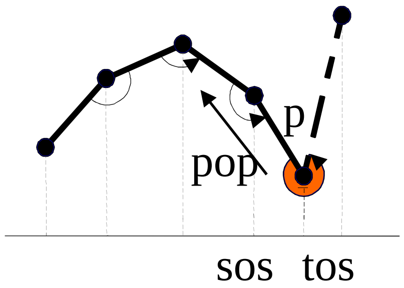
- Optimality – depends on data order (or distribution)

  In worst case x In expected case
- Output sensitivity – depends on the result
- Extendable to higher dimensions?
- Off-line versus on-line
  - Off-line – all points available, preprocessing for search speedup
  - On-line – stream of points, new point $p_i$ on demand, just one new point at a time, CH valid for $\{p_1, p_2, \ldots, p_r\}$
  - Real-time – points come as they “want”

    (not faster than optimal constant O(log $n$) inter-arrival delay)
- Parallelizable
- Dynamic – points can be deleted

# Why to search other convex hull algorithms?

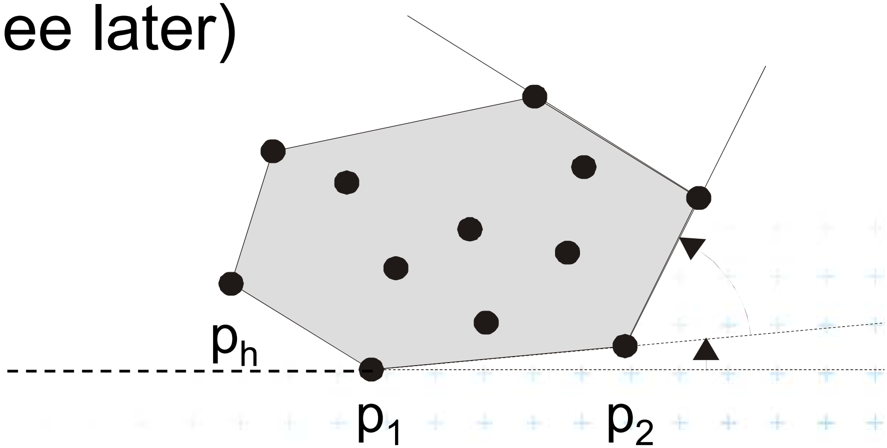
- Graham scan
  O($n$ log $n$) time and O($n$) space is
  - optimal in worst case
  - not optimal in average case (not output sensitive)
  - only 2D
  - off-line
  - serial (not parallel)
  - not dynamic




$O(n)$ for polygon (will be discussed in seminar [9])

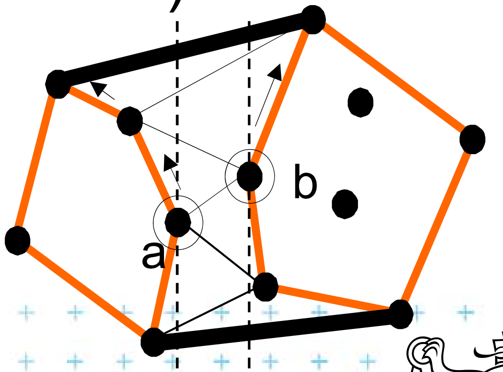
# Jarvis March – Gift wrapping

- *O*(*hn*) time and O(*n*) space is
  - not optimal in worst case $O(n^2)$
  - may be optimal if *h* << *n* (output sensitive)
  - 3D or higher dimensions (see later)
  - off-line
  - serial (not parallel)
  - not dynamic



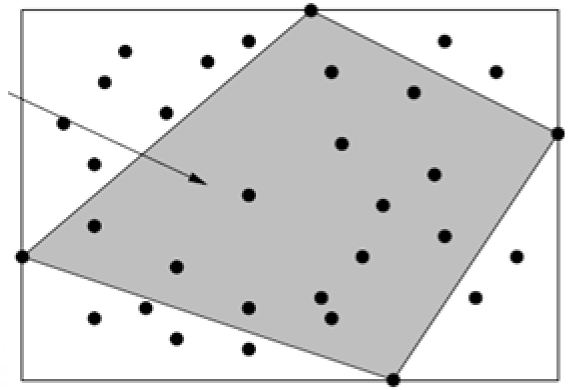
# Divide & Conquer

- O($n$ log $n$) time and O($n$) space is
  - optimal in worst case (in 2D or 3D)
  - not optimal in average case (not output sensitive)
  - 2D or 3D (circular ordering), in higher dims not optimal
  - off-line
  - Version with sorting (the presented one) – serial
  - Parallel for overlapping merged hulls
    (see Chapter 3.3.5 in Preparata for details)
  - not dynamic



# Quick hull

- O($n$ log $n$) expected time, O($n^2$) the worst case and O($n$) space *in 2D* is
  - not optimal in worst case O($n^2$)
  - optimal if uniform distribution
    then $h$ << $n$ (output sensitive)
  - 2D, or higher dimensions [see http://www.qhull.org/]
  - off-line
  - serial (not parallel)
  - not dynamic



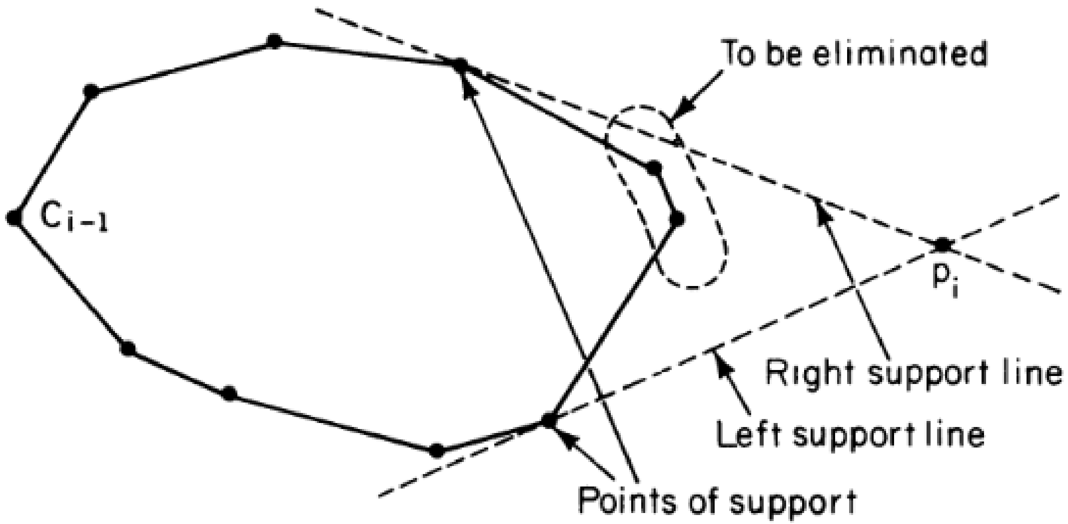
[Mount]

# Chan

- *O*(*n* log *h*) time and O(*n*) space is
  - optimal for *h* points on convex hull (output sensitive)
  - 2D and 3D --- gift wrapping
  - off-line
  - Serial (not parallel)
  - not dynamic

# Preparata's on-line algorithm

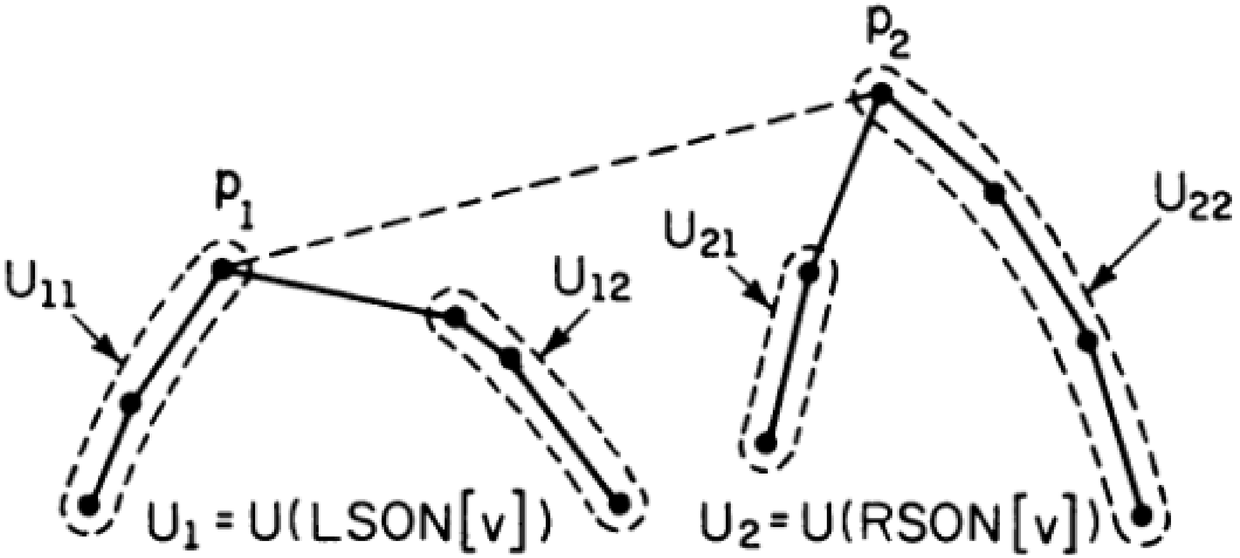
- New point *p* is tested
  - Inside –> ignored
  - Outside –> added to hull
    - Find left and right supporting lines (touch at supporting points)
    - Remove points between supporting points
    - Add *p* to CH between supporting lines




[Preparata]

# Overmars and van Leeuven

- Allow dynamic CH
  (on-line insert & delete)
- Manage special tree with all intermediate CHs
- Will be discussed on seminar [7]



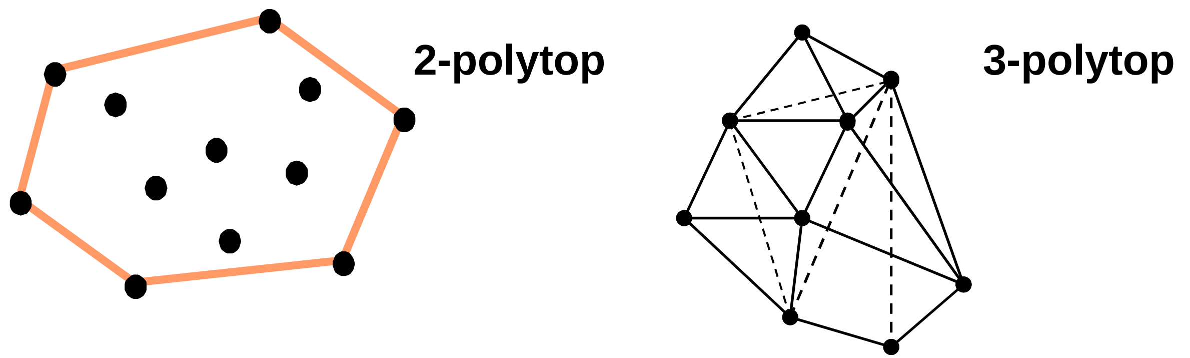

[Preparata]

# Convex hull in 3D

- Terminology
- Algorithms
  1. Gift wrapping
  2. D&C Merge
  3. Randomized Incremental

# Terminology

- Polytope (d-polytope)
  = convex hull of finite set of points in $E^d$



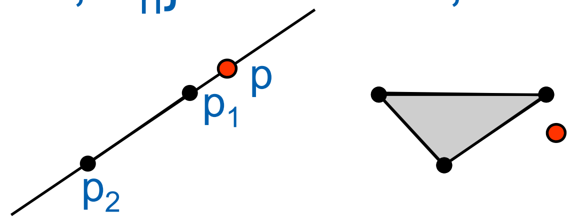

- Simplex (k-simplex, d-simplex)
  = CH of $k$ + 1 affine independent points


= "Special" Polytope with all the points are on the CH

# Terminology (2)

- Affine combination
  = linear combination of the points $\{p_1, p_2, \ldots, p_n\}$
  whose coefficients $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ sum to 1, and $\lambda_i \in R$

$$\sum_{i=1}^{n} \lambda_i p_i$$



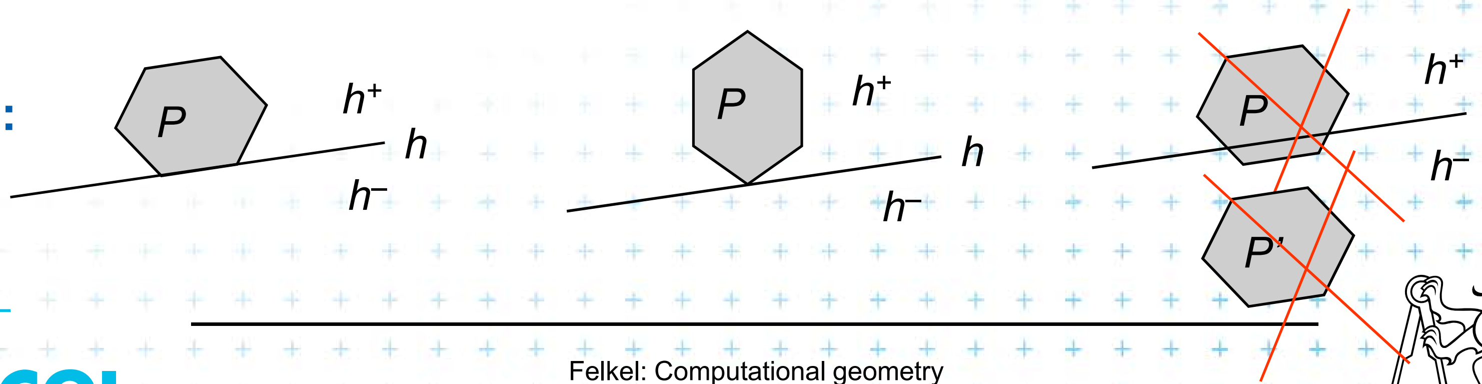

- Affine independent points
  = no one point can be expressed as affine combination of the others


- Convex combination
  = linear combination of the points $\{p_1, p_2, \ldots, p_n\}$
  whose coefficients $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ sum to 1, and $\lambda_i \in R^+_0$
  (i.e., $\forall i \in \{1, \ldots, k\}$, $\lambda_i \geq 0$)

# Terminology (3)

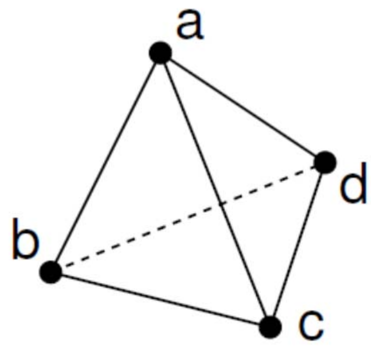
- Any (d-1)-dimensional hyperplane $h$ divides the space into (open) halfspaces $h^+$ *and* $h^-$, *so that* $E^n = h^+ \cup h \cup h^-$
- *Def:* $\overline{h^+} = h^+ \cup h$, $\overline{h^-} = h^- \cup h$ (closed halfspaces)
- Hyperplane supports a polytope $P$ (Supporting hyperplane)
  - if $h \cap P$ is not empty and
  - if $P$ is entirely contained within either $\overline{h^+}$ or $\overline{h^-}$

**In 2D:**



# Faces and facets

- Face of the polytope
  = Intersection of polytope *P* with a supporting hyperplane *h*
  - Faces are convex polytops of dimension *d* ranging from 0 to d – 1
  - 0-face = vertex
  - 1-face = edge
  - (d – 1)-face = facet




Proper faces:
Vertices: a,b,c,d
Edges: ab, ac, ad, bc, bd, cd
Facets: abc, abd, bcd

In 3D we often say *face*, but it means correctly a *facet*

# Proper faces

- Proper faces

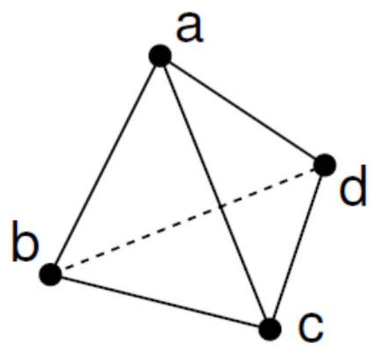
  = Faces of dimension *d* ranging from 0 to *d* – 1

- Improper faces

  = proper faces + two additional faces:

  – {} = Empty set = face of dimension -1

  – Entire polytope = face of dimension *d*




Improper faces:

Empty set {}

Vertices: a,b,c,d

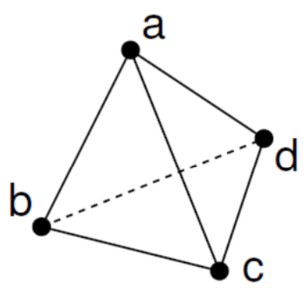
Edges: ab, ac, ad, bc, bd, cd

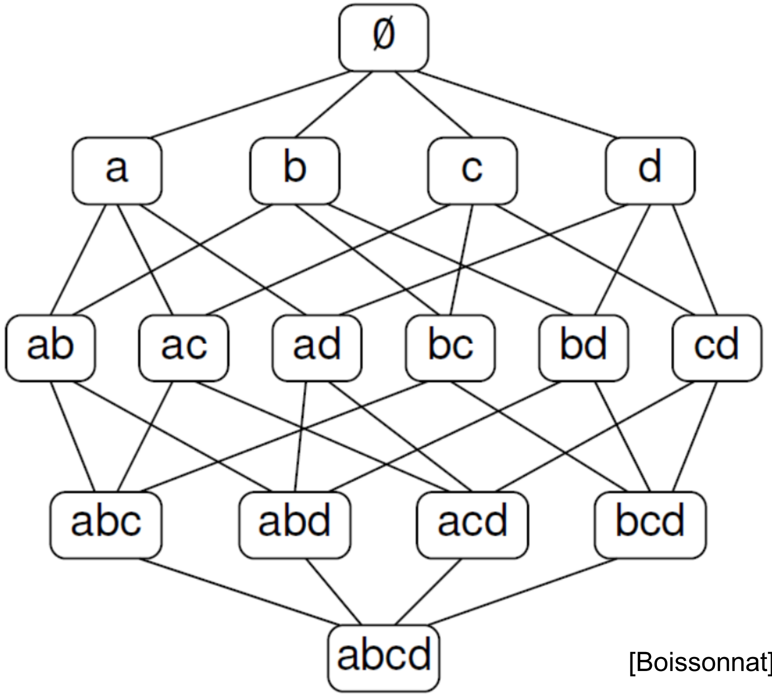
Facets: abc, abd, bcd

Entire polytope: abcd

# Incident graph

- Stores topology of the polytope
- Ex: 3-simplex:




[Boissonnat]

**Dimension**

**-1**

**0**

**1**

**2**

**3**

- D-simplex is very regular face structure:
  - 1-face for each pair of vertices
  - 2-face for each triple of vertices

# Facts about polytopes

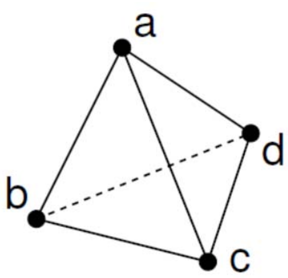
- Boundary o polytope is *union of its proper faces*
- Polytope has *finite number of faces (next slide)*. Each face is a polytope
- Polytope is *convex hull of its vertices (the def)* (its bounded)
- Polytope is the *intersection of finite number of closed halfspaces* $\overline{h^{+}}$ (conversely not: intersection of closed halfspaces may be unbounded => called polyhedron or unbounded polytope)

# Number of faces on a d-simplex

- Number of *j*-dimensional faces on a *d*-simplex
= number of (*j*+1)-element subsets from domain of size (*d*+1)

$$\binom{d+1}{j+1} = \frac{(d+1)!}{(j+1)!(d-j)!}$$

- Ex.: Tetrahedron = 3-simplex:




  – facets (2-dim. faces) $\binom{3+1}{2+1} = \frac{4!}{3!1!} = 4$

  – edges (1-dim. faces) $\binom{3+1}{1+1} = \frac{4!}{2!2!} = 6$

  – vertices (0-dim faces) $\binom{3+1}{0+1} = \frac{4!}{1!3!} = 4$

# Complexity of 3D convex hull is O(n)

- The worst case complexity → if all *n* points on CH

=> use 3-simplex for complexity derivation

1. has all points on its surface – on the Convex Hull
2. has usually more edges E and faces F than 3-polytope
3. has triangular facets, each generates 3 edges, shared by 2 triangles => $3F = 2E$  2-manifold

- $V - E + \boxed{F} = 2$ … Euler formula for V = n points

$V - E + 2E/3 = 2$

$V - 2 = E / 3$

$E = 3V - 6, \quad V = n$

$E = O(n)$

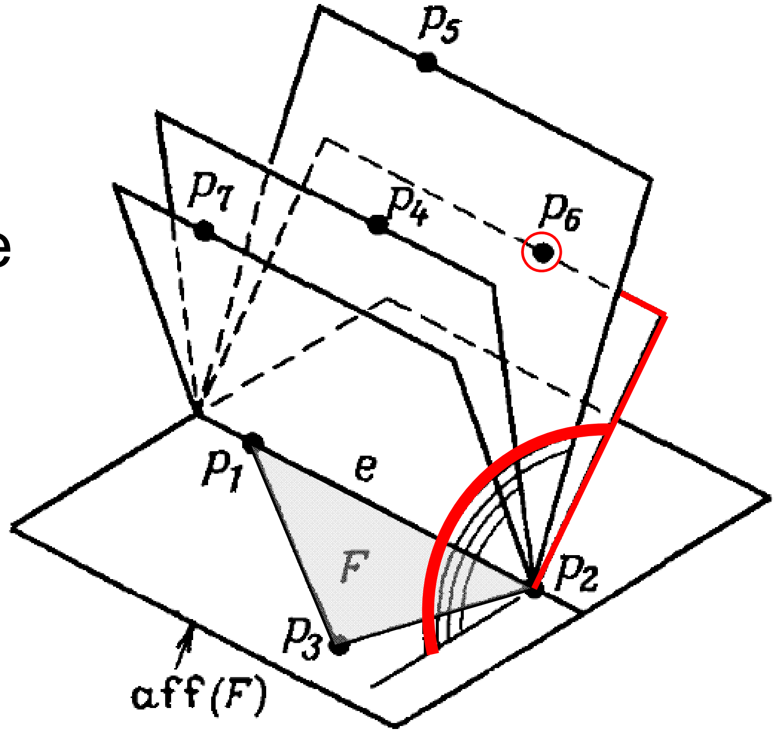
$F = 2\boxed{E} / 3$

$F = 2V - 4$

$F = O(n)$

# 1. Gift wrapping in higher dimensions

- First known algorithm for n-dimensions (1970)
- Direct extension of 2D alg.
- Complexity O(nF)
  - F is number of CH facets
  - Algorithm is output sensitive
  - Details on seminar, assignment [10]



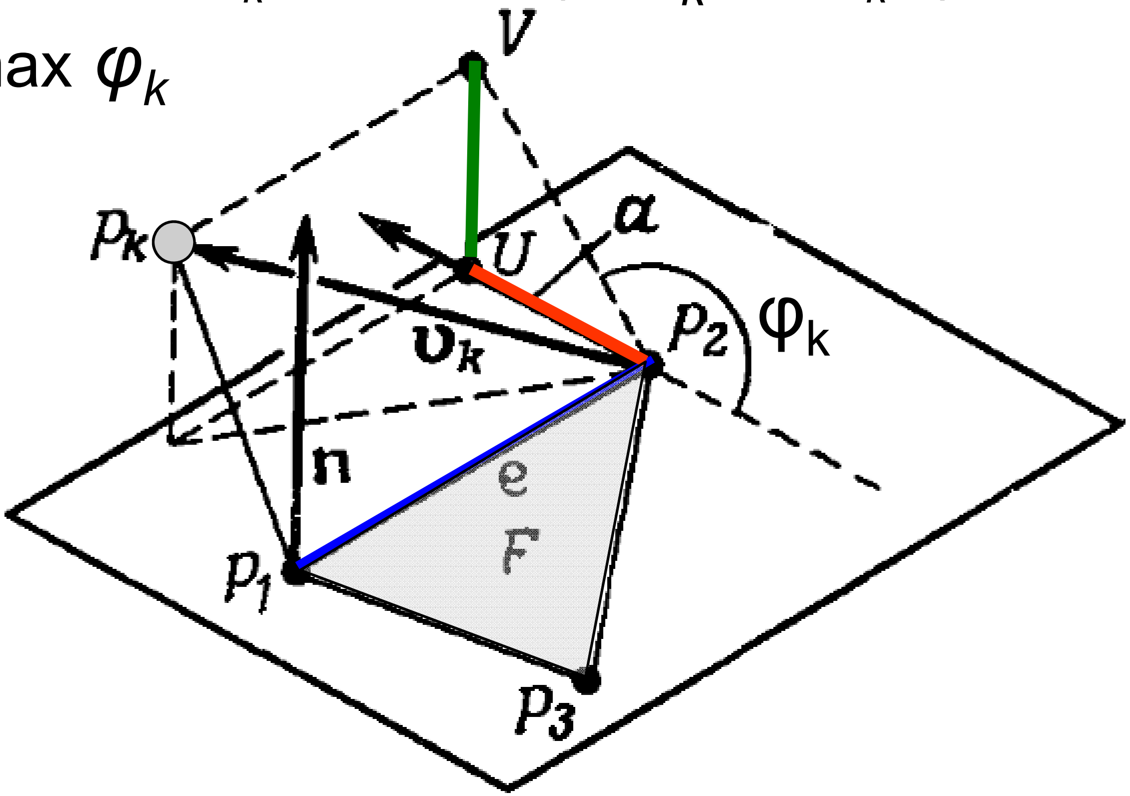

[Preparata]

# The angle comparison [Preparata 3.4.1]

Cotangent of the agle $\varphi_k$ between halfplanes $F$ and $ep_k$
$= -\ |UP_2|\ /\ |UV|$ , where $|UP_2| = \boldsymbol{v}_k.\boldsymbol{a}$ and $|UV| = \boldsymbol{v}_k.\boldsymbol{n}$

For each $P_k$ compute $\varphi_k = \text{arcctan}(-\ \boldsymbol{v}_k.\boldsymbol{a}\ /\ \boldsymbol{v}_k.\boldsymbol{n})$,

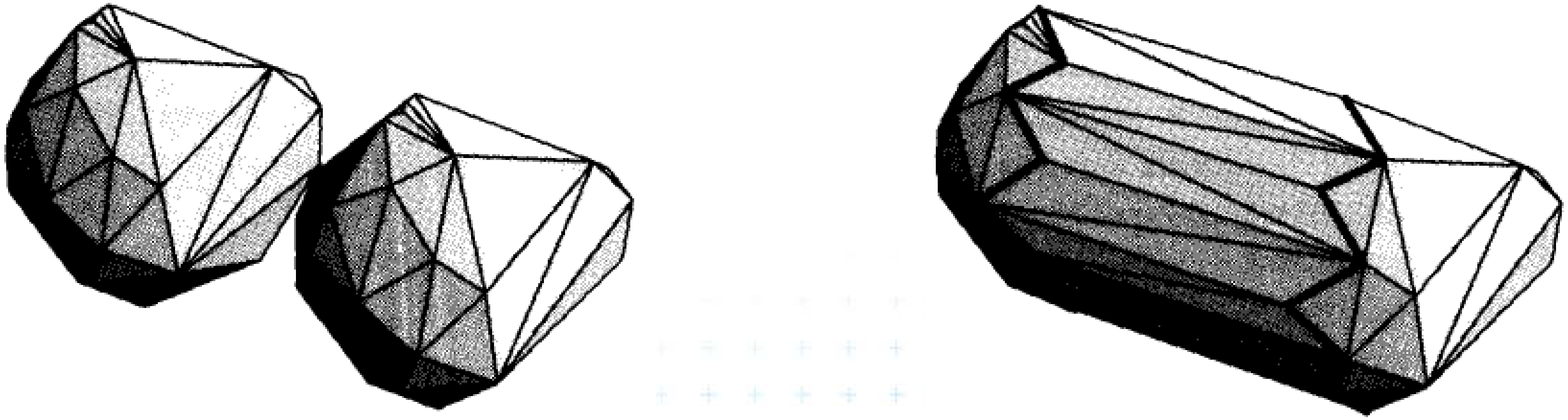
The angle is max $\varphi_k$




[Preparata]

# 2. Divide & conquer 3D convex hull [Preparata, Hong77]

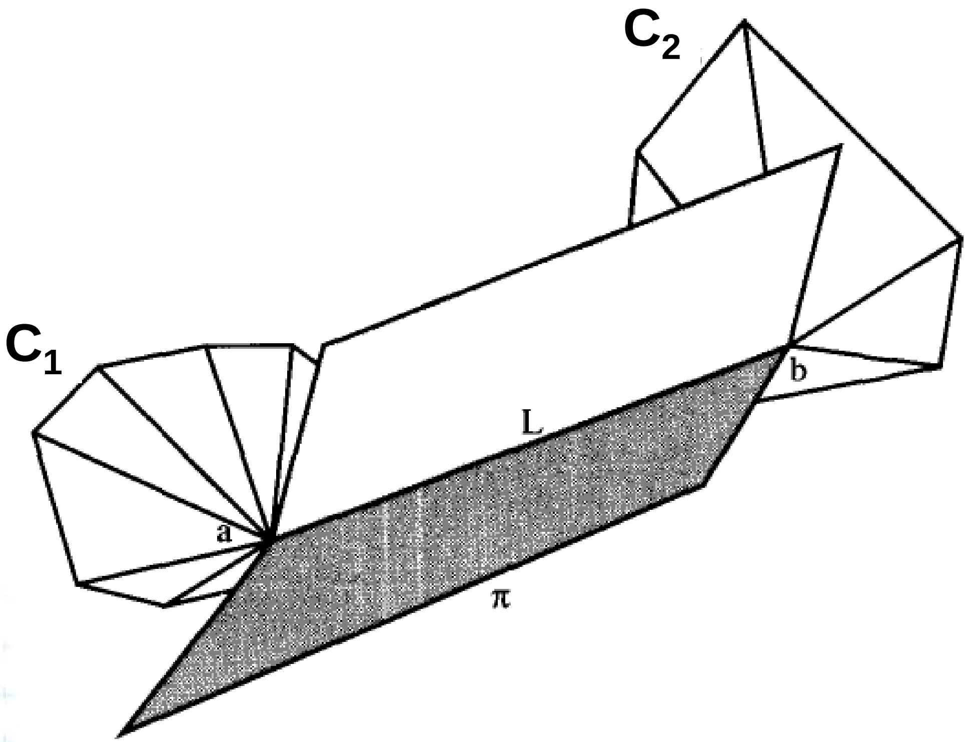
- Sort points in x-coord
- Recursively split, construct CH, merge
- Merge takes O(n) => O(*n* log *n*) total time



[Rourke]

# Divide & conquer 3D convex hull [Preparata, Hong 77]

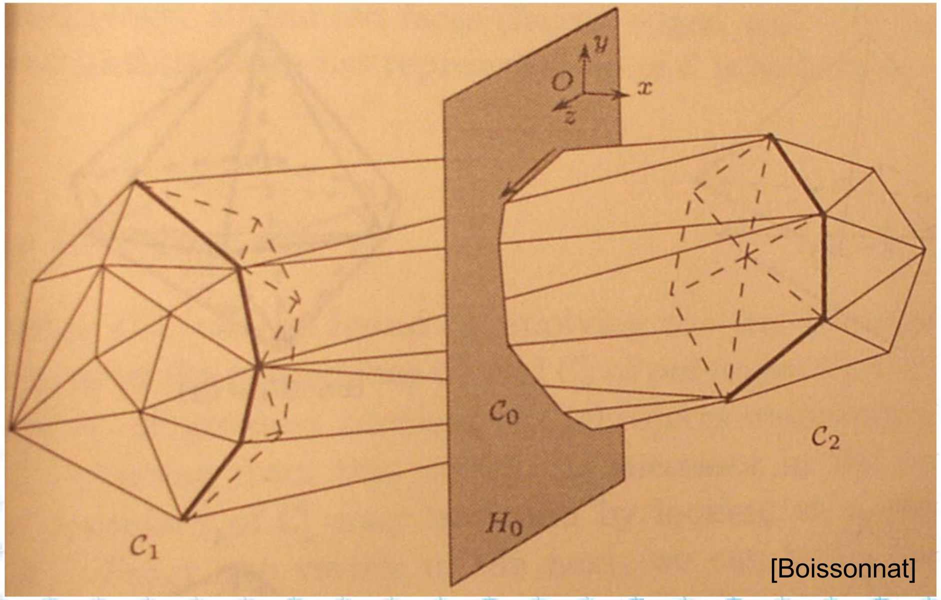
- Merge($C_1$ with $C_2$) uses gift wrapping
  - Gift wrap plane around edge *e* – find new point *p* on $C_1$ or on $C_2$ (neighbor of *a* or *b*)
  - Search just the CW or CCW neighbors around *a, b*




[Rourke]

# Divide & conquer 3D convex hull [Preparata, Hong 77]

- Performance O(*n* log *n*) rely on circular ordering
  - In 2D: Ordering of points around CH
  - In 3D: Ordering of vertices around 2-polytop $C_0$ (vertices on intersection of new CH edges with separating plane $H_0$) [ordering around horizon of $C_1$ and $C_2$ does not exist, both horizons may be non-convex and even not simple polygons]




[Boissonnat]

  - In ≥ 4D: Such ordering does not exist

# Divide & conquer 3D convex hull [Preparata, Hong 77]

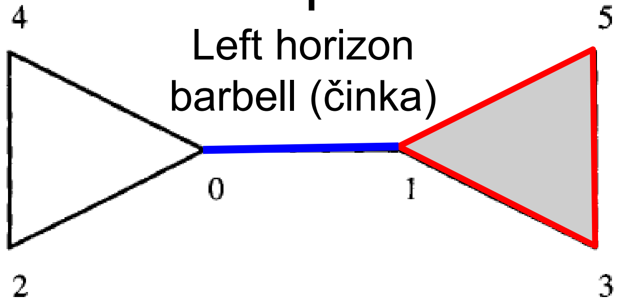
Merge($C_1$ with $C_2$)

- Find the first CH edge *L* connecting $C_1$ with $C_2$
- *e* = *L*
- While not back at *L do*
  - store *e* to *C*
  - Gift wrap plane around edge *e* – find new point *P* on $C_1$ or on $C_2$ (neighbor of *a* or *b*)
  - *e* = new edge to just found end-point *P*
  - Store new triangle *eP* to *C*
- Discard hidden faces inside CH from *C*
- Report merged convex hull *C*

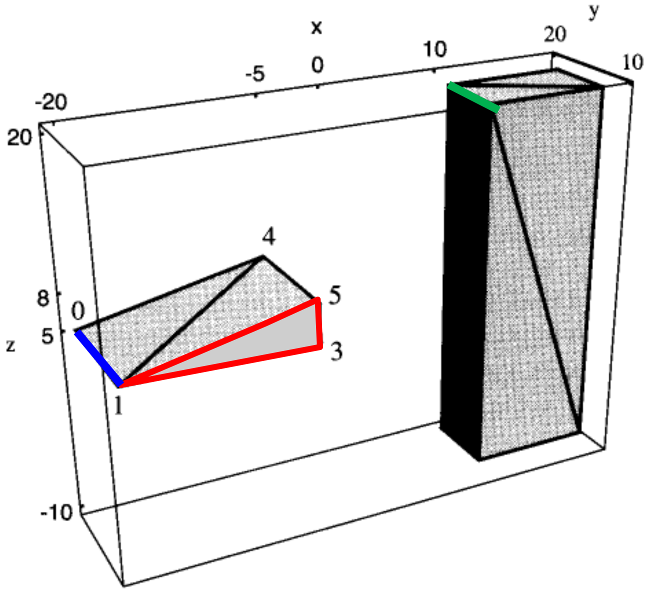
# Divide & conquer 3D convex hull [Preparata, Hong 77]

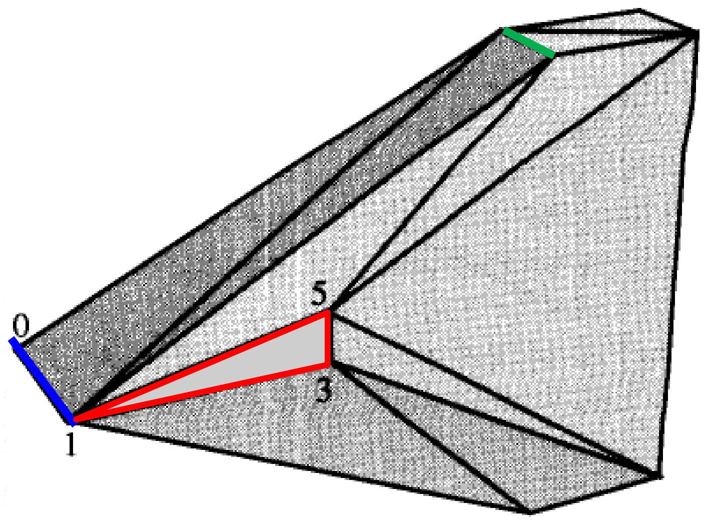
- Problem of gift wrapping [Edelsbrunner 88]
  - The edges on horizon do not form simple circle but a “barbell” 0,2,4,0,1,3,5,1

Left horizon barbell (činka)




**Do not stop here!**



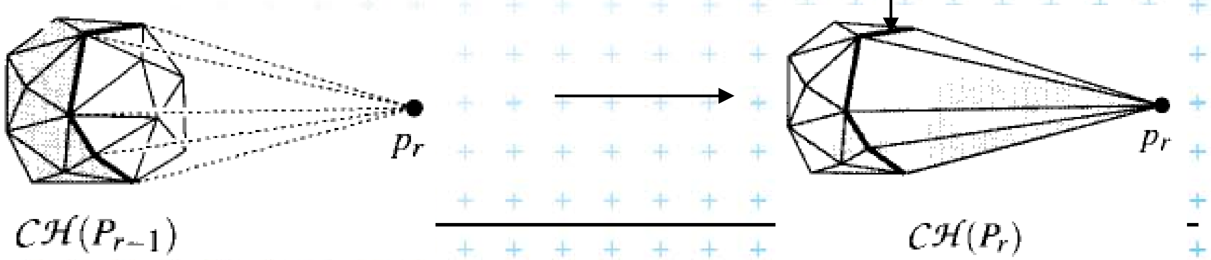




[Berg]

# 3. Randomized incremental alg. principle

1. Create tetrahedron (smallest CH in 3D)
   - Take 2 points $p_1$ and $p_2$
   - Search the 3rd point not lying on line $p_1p_2$
   - Search the 4th point not lying in plane $p_1p_2p_3$ …if not found, use 2D CH
2. Perform random permutation of remaining points $\{p_5, \ldots, p_n\}$
3. For $p_r$ in $\{p_5, \ldots, p_n\}$ do add point $p_r$ to CH($P_{r-1}$)

   Notation: for $r \geq 1$ let $P_r = \{p_1, \ldots, p_r\}$ is set of already processed pts
   - If $p_r$ lies inside or on the boundary of CH($P_{r-1}$) then do nothing
   - If $p_r$ lies outside of CH($P_{r-1}$) then
     - find and remove visible faces
     - create new faces (triangles) connecting $p_r$ with lines of horizon




[Berg]

# Conflict graph

- Stores unprocessed points with facets of CH they see
- Bipartite graph

  points $p_t$, $t > r$ … unprocessed points

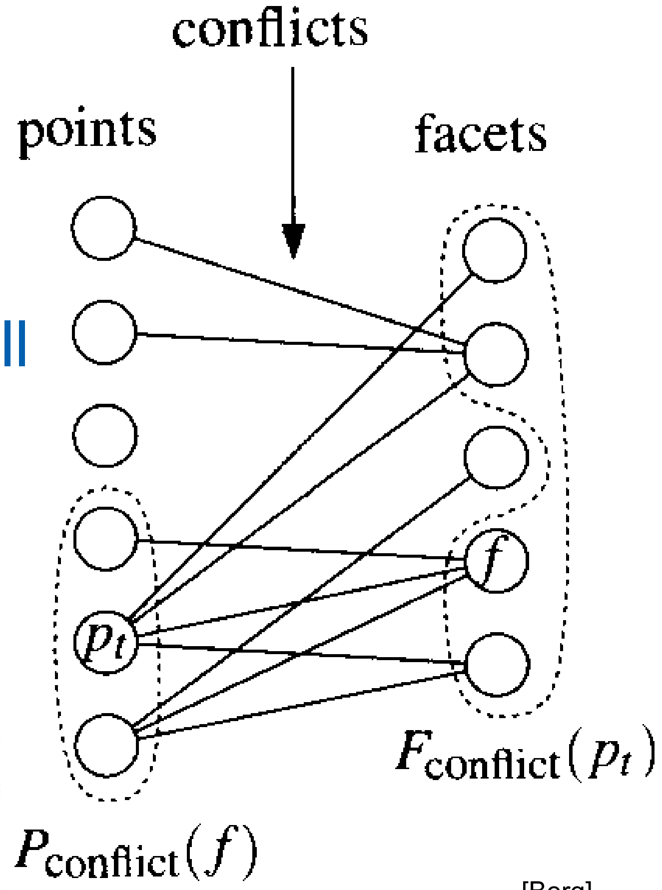
  facets of CH$(P_r)$… facets of convex hull

  conflict arcs … conflict, as visible facets cannot be in CH

- Maintains sets:

  **$P_{conflict}(f)$** … points, that see $f$

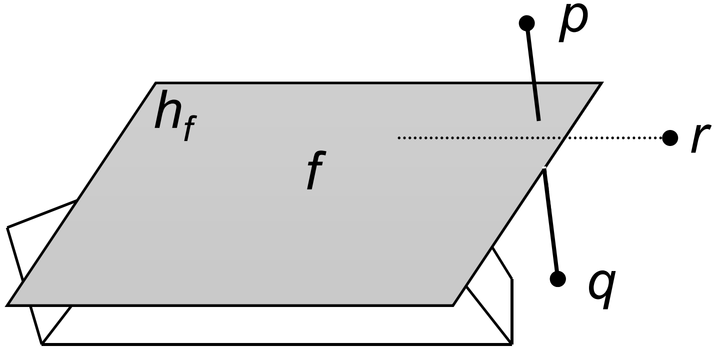
  **$F_{conflict}(p_r)$**… facets visible from $p_r$ (visible region – deleted after insertion of $p_r$)




[Berg]

# Visibility between point and face

- Face $f$ is visible from a point $p$ if that point lies in the open half-space on the other side of $h_f$ than the polytope




$f$ is visible from $p$ ($p$ is *above* the plane)

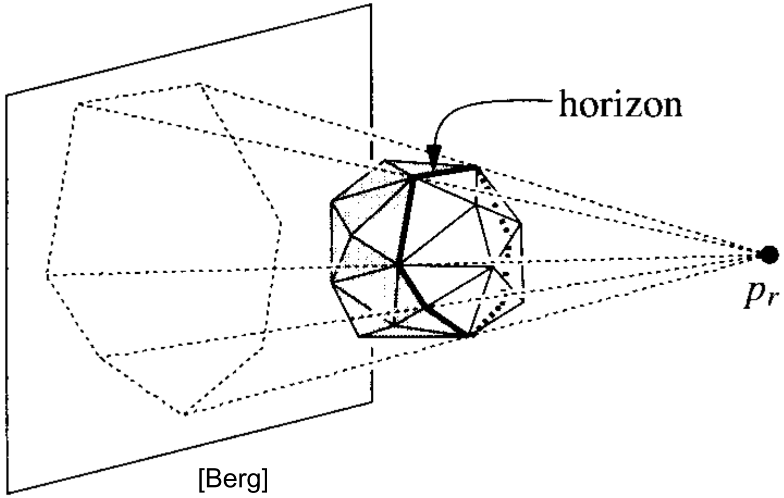
$f$ is not visible from $r$ lying *in the plane* of $f$ (this case will be discussed next)

$f$ is not visible from $q$

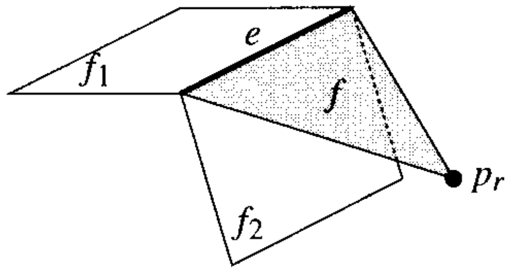
$p \in P_{conflict}(f)$, p is among the points that see the face f
$f \in F_{conflict}(p)$ f is among the faces that visible from point p

# New triangles to horizon

- Horizon = edges *e* incident to visible and invisible facets




[Berg]




[Berg]

- New triangle *f* connects edge *e* on horizon and point $p_r$ and
  – creates new node for facet f updates the conflict graph
  – add arcs to points visible f (subset from $P_{coflict}(f_1) \cup P_{coflict}(f_2)$ )
- Coplanar triangles on the plane $ep_r$ are merged with new triangle.
  Conflicts are copied from the deleted triangle (same plane)

# Incremental Convex hull algorithm

**IncrementalConvexHull(P)**

*Input:* Set of $n$ points in general position in 3D space

*Output:* The convex hull C=CH(P) of P

1. Find four points that form an initial tetrahedron, C = CH($\{p_1, p_2, p_3, p_4\}$)
2. Compute random permutation $\{p_5, p_6, \ldots, p_n\}$ of the remaining points
3. Initialize the conflict graph with all visible pairs $(p_t, f)$, where $f$ is facet of C and $p_t$, $t > 4$, are non-processed points
4. **for** $r$ = 5 to $n$ **do** …insert $p_r$, into $C$
5. **if**( $F_{conflict}(p_r)$ is not empty) **then** …$p_r$ is outside, any facet is visible
6. Delete all facets $F_{conflict}(p_r)$ from C … only from hull C, not from G
7. Walk around visible region boundary, create list $L$ of horizon edges
8. **for** all $e \in L$ **do**
9. connect $e$ to $p_r$ by a new triangular facet $f$
10. **if** $f$ is coplanar with its neighbor facet $f'$ along $e$
11. **then** merge $f$ and $f'$, take conflict list from $f'$
12. **else** … determine conflicts for new face $f$

… [continue on the next slide]

# Incremental Convex hull algorithm (cont…)

```
        else … not coplanar => determine conflicts for new face f
            Create node for f in G        //… new face in conflict graph G
            Let f_1 and f_2 be the facets incident to e in the old CH(P_{r-1})
            P(e) = P_coflict(f_1) ∪ P_coflict(f_2)
            for all points p ∈ P(e) do
                if f is visible from p, then add(p, f) to G   … new edges
    Delete the node corresponding to p_r and the nodes corresponding
    to facets in F_coflict(p_r) from G, together with their incident arcs
return C
```

Complexity: Convex hull of a set of points in $E^3$ can be computed in $O(n \log n)$ randomized expected time

For proof see: [Berg, Section11.3]

# Conclusion

- Recapitulation of 2D algorithms
- 3D algorithms
  - Gift wrapping
  - D&C
  - Randomized incremental

# References

[Berg] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars: Computational Geometry: *Algorithms and Applications*, Springer-Verlag, 3rd rev. ed. 2008. 386 pages, 370 fig. ISBN: 978-3-540-77973-5, Chapter 11, http://www.cs.uu.nl/geobook/

[Boissonnat] J.-D. Boissonnat and M. Yvinec, *Algorithmic Geometry*, Cambridge University Press, UK, 1998. Chapter 9 – Convex hulls

[Preparata] Preperata, F.P., Shamos, M.I.: *Computational Geometry. An Introduction.* Berlin, Springer-Verlag,1985.

[Mount] David Mount, - *CMSC 754: Computational Geometry, Lecture Notes for Spring 2007*, University of Maryland, Lecture 3. http://www.cs.umd.edu/class/spring2007/cmsc754/lectures.shtml

[Chan] Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions., *Discrete and Computational Geometry*, 16, 1996, 361-368. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.389

**OI-OPPA. European Social Fund**
**Prague & EU: We invest in your future.**