OPPA

PRAHA
PRAGUE
PRAGA
PRAG

EVROPSKÁ UNIE

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

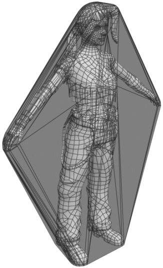
A4M39VG Beneath - beyond method

Tomáš Dřínovský

Czech Technical University in Prague
Faculty of Electrical Engineering

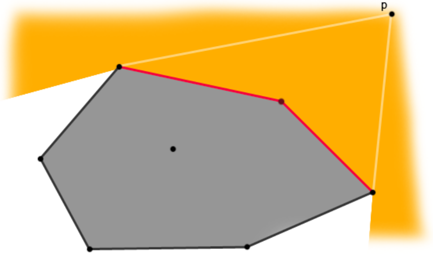
18.10.2012

Beneath - beyond method



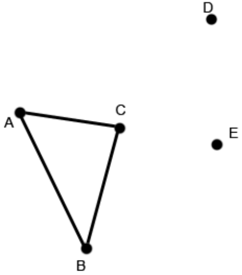
- Computes convex hull of point set
- Works in d-dimensions
- Online property
- Comparable performance to that of gift wrapping method

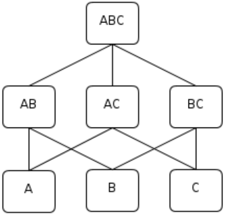
Idea



1. New point represents source of a light.
2. Keep faces of the previous convex hull if they lie in the shadow.
3. Delete enlightened faces.
4. Construct supporting faces of the light cone.

Convex hull incidence graph

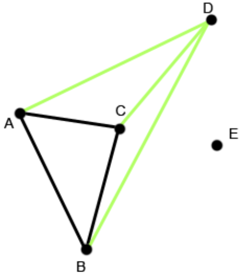


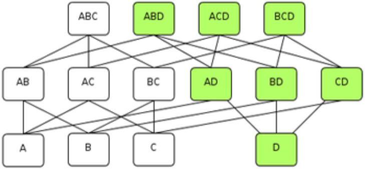


Algorithm

Initialization:
Sort n points along arbitrary axis.
Take first d points, which define facet as initial CH.
Iteration i=1 to n:
If point p_i doesn't lie in affine hull CH and dim(CH) $<$ d perform *pyramidal update*
else perform *non-pyramidal update*.

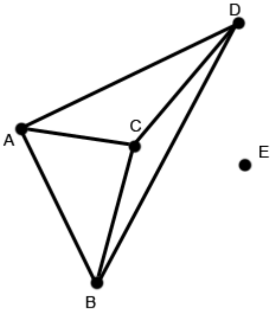
Pyramidal update

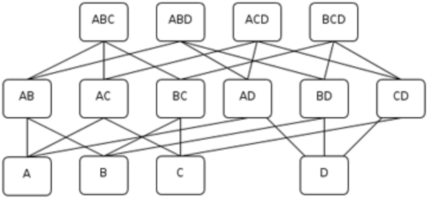




New point connects with all points in the current hull and new faces are created.

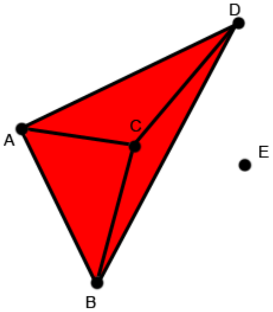
Non-pyramidal update

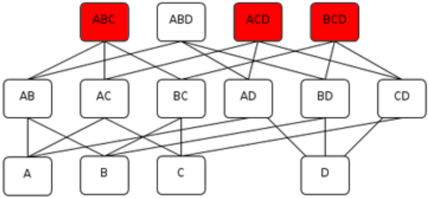




- Facets are colored red if they separates the point p from CH and blue if they don’t (red-light, blue-shadow).
- D-1 faces are colored red if it’s subface of only red faces, blue if it’s subface of only blue faces and purple otherwise.

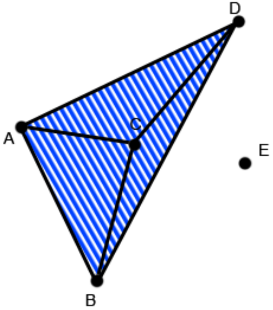
Non-pyramidal update

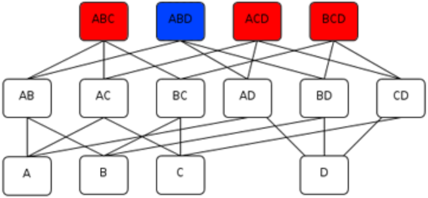




- Facets are colored red if they separates the point p from CH and blue if they don't (red-light, blue-shadow).
- D-1 faces are colored red if it's subface of only red faces, blue if it's subface of only blue faces and purple otherwise.

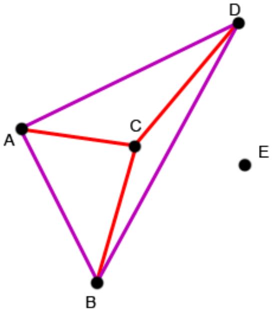
Non-pyramidal update

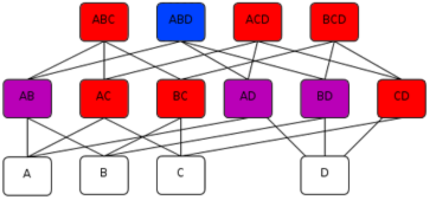




- Facets are colored red if they separates the point p from CH and blue if they don’t (red-light, blue-shadow).
- D-1 faces are colored red if it’s subface of only red faces, blue if it’s subface of only blue faces and purple otherwise.

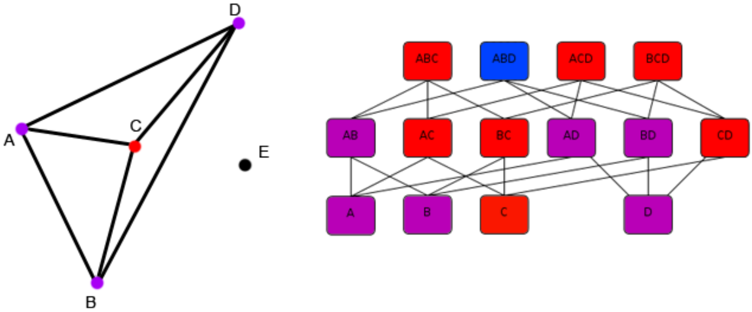
Non-pyramidal update





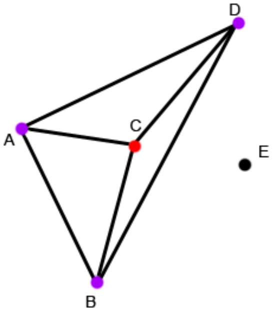
- Facets are colored red if they separates the point p from CH and blue if they don’t (red-light, blue-shadow).
- D-1 faces are colored red if it’s subface of only red faces, blue if it’s subface of only blue faces and purple otherwise.

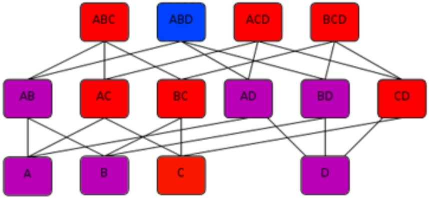
Non-pyramidal update



- Facets are colored red if they separates the point p from CH and blue if they don’t (red-light, blue-shadow).
- D-1 faces are colored red if it’s subface of only red faces, blue if it’s subface of only blue faces and purple otherwise.

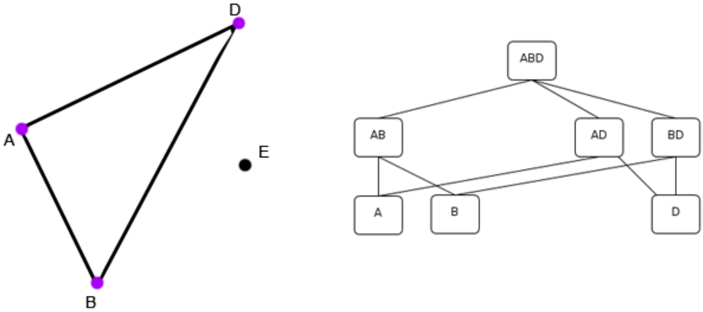
Non-pyramidal update





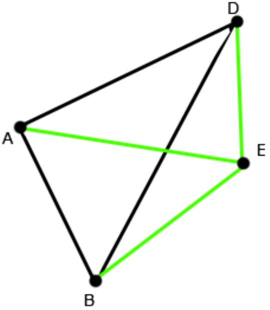
- Faces colored blue remain in the convex hull, faces colored red are deleted.
- Purple faces are connected with the added point, so they form new edges, facets etc..

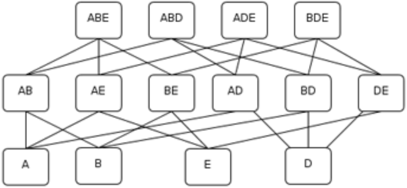
Non-pyramidal update



- Faces colored blue remain in the convex hull, faces colored red are deleted.
- Purple faces are connected with the added point, so they form new edges, facets etc..

Non-pyramidal update





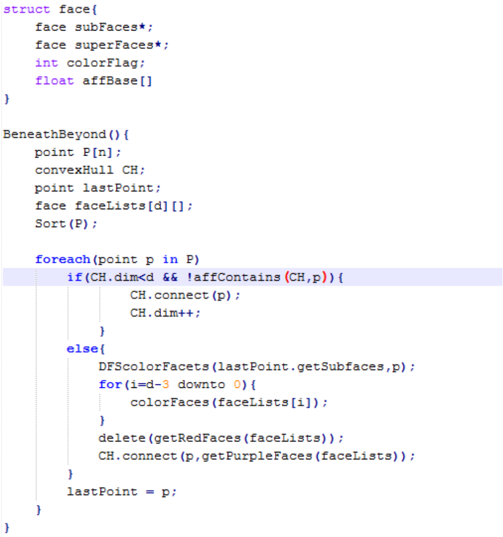
- Faces colored blue remain in the convex hull, faces colored red are deleted.
- Purple faces are connected with the added point, so they form new edges, facets etc..

More details on coloring

We don’t want to iterate through all the faces

- Sorting ensures that each new point lies outside or on the convex hull.
- Algorithm finds first red facet. Since the points are presorted along same axis, each point p_i cast light at least on one facet created from $p_i - 1$.
- Since red facets forms connected set, we can use DFS to obtain all the red facets.

Pseudocode



```
struct face{
    face subFaces*;
    face superFaces*;
    int colorFlag;
    float affBase[]
}

BeneathBeyond(){
    point P[n];
    convexHull CH;
    point lastPoint;
    face faceLists[d][];
    Sort(P);

    foreach(point p in P)
        if(CH.dim<d && !affContains(CH,p)){
                CH.connect(p);
                CH.dim++;
            }
        else{
            DFScolorFacets(lastPoint.getSubfaces,p);
            for(i=d-3 downto 0){
                colorFaces(faceLists[i]);
            }
            delete(getRedFaces(faceLists));
            CH.connect(p,getPurpleFaces(faceLists));
        }
        lastPoint = p;
    }
}
```

Complexity

Time complexity

$O\left(n \log n + n^{[(d+1)/2]}\right)$

Space complexity

$O\left(n^{(d/2)}\right)$

For proof see [Edelsbrunner Herbert, 2004] Section 8.4.5, Chapter 6

Reference

- [Edelsbrunner Herbert, 2004]
  Algorithms in Combinatorial Geometry
- [Preparata F.P., Shamos M.I. 1985]
  Computational Geometry. An Introduction

Questions

Thank you for attention. Any questions?

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