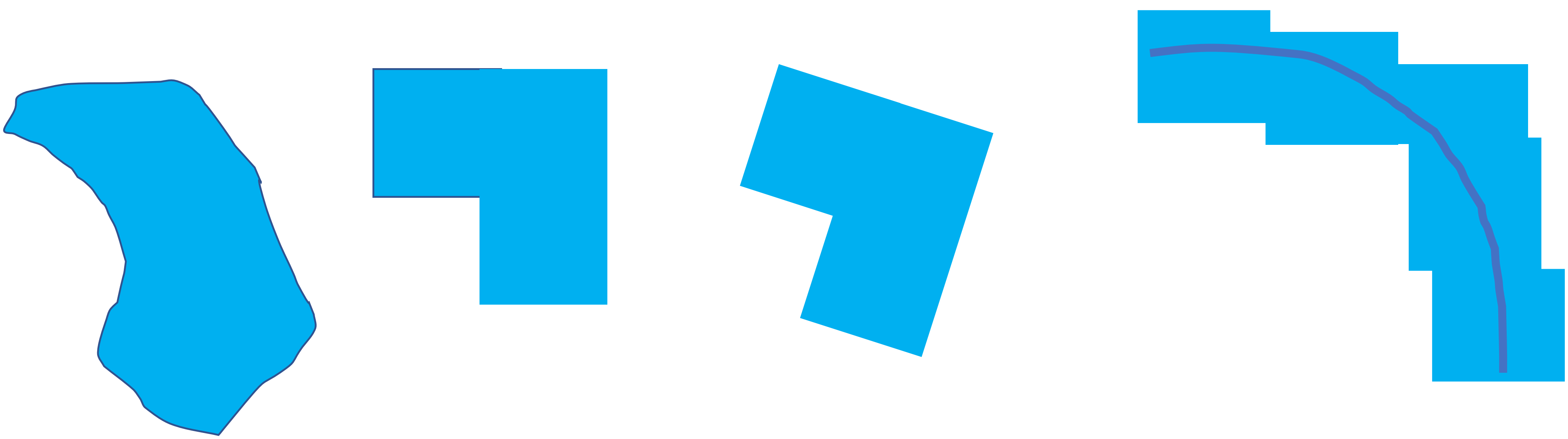
nD Convex Polytopes for Point Cloud Query.

Rod Thompson

23/Jan/2023

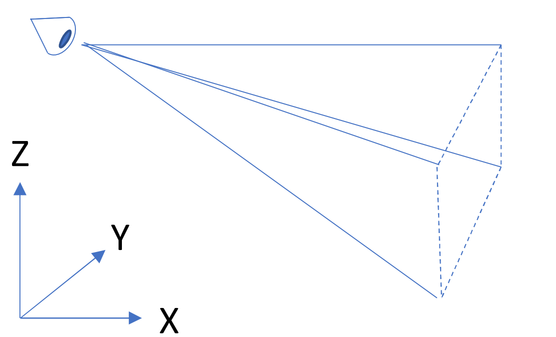
Why Convex Polytope

- The Convex region is more selective than a minimum bounding hyper-rectangular window
- It is not a fully generic region (which can have concavities)

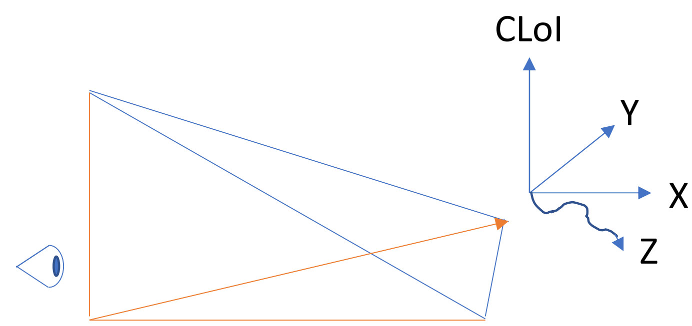


Why Convex Polytope

- But – it is useful (in multi-dimensional point clouds)
- It can be significantly faster in nD – especially with n > 3.

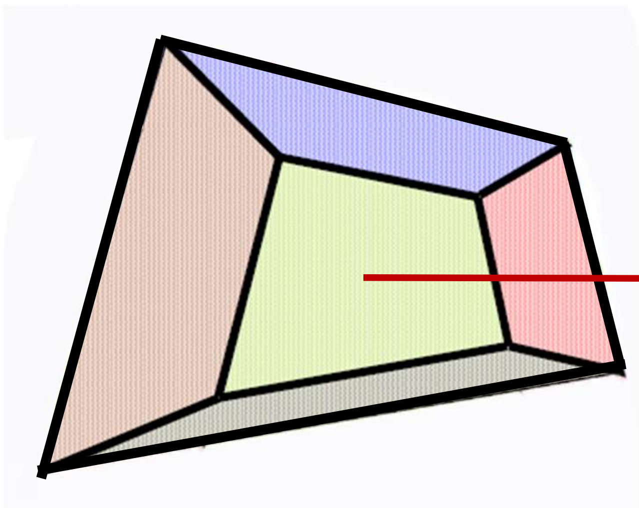


The classic 3D View Frustum

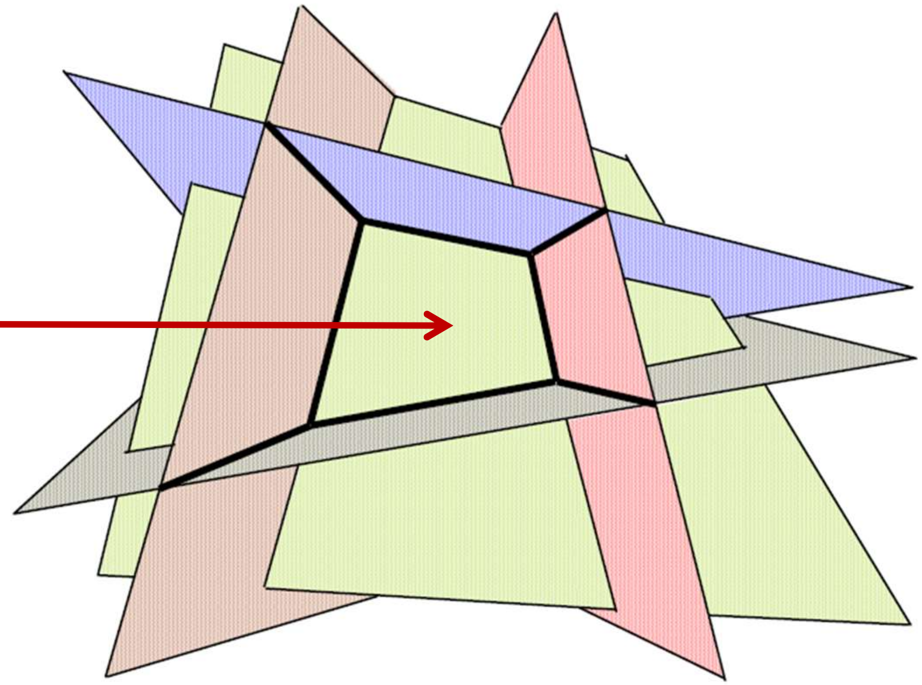


4D View Frustum with CLoI
(only drawn in 3D)

Why is it Fast?



A convex region (in this case 3D)

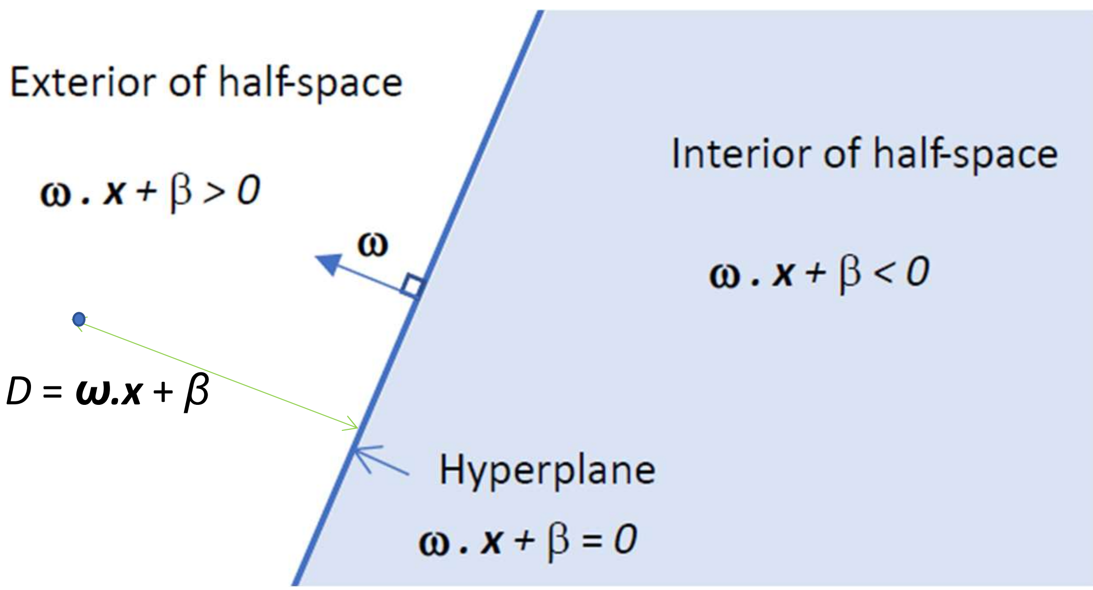


Can be represented by a set of half spaces
(These should extend to infinity, but impossible to visualise)

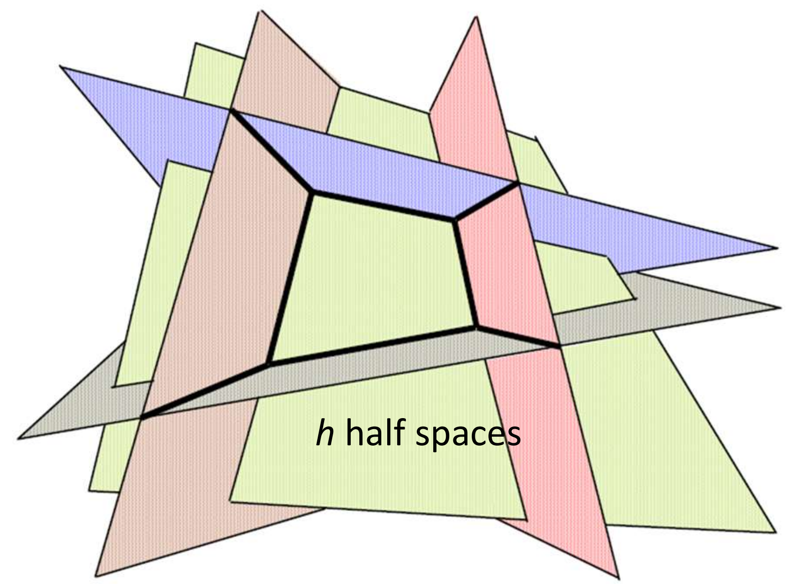
A half space is defined as ($\boldsymbol{\omega}$, β) *where* $\boldsymbol{\omega}$ is a unit vector ($\boldsymbol{\omega}.\boldsymbol{\omega} = 1$) and β *is a scalar.*

Is a Point Within a Half Space?

The half space is defined as ($\boldsymbol{\omega}$*, β) where* $\boldsymbol{\omega}$ is a unit vector ($\boldsymbol{\omega} . \boldsymbol{\omega} = 1$) and *β is a scalar.*



This question "Is a point within a halfspace" is of $O(n)$ in nD space



So the question of "Is a point within a region defined by *h* half spaces" is $O(nh)$.

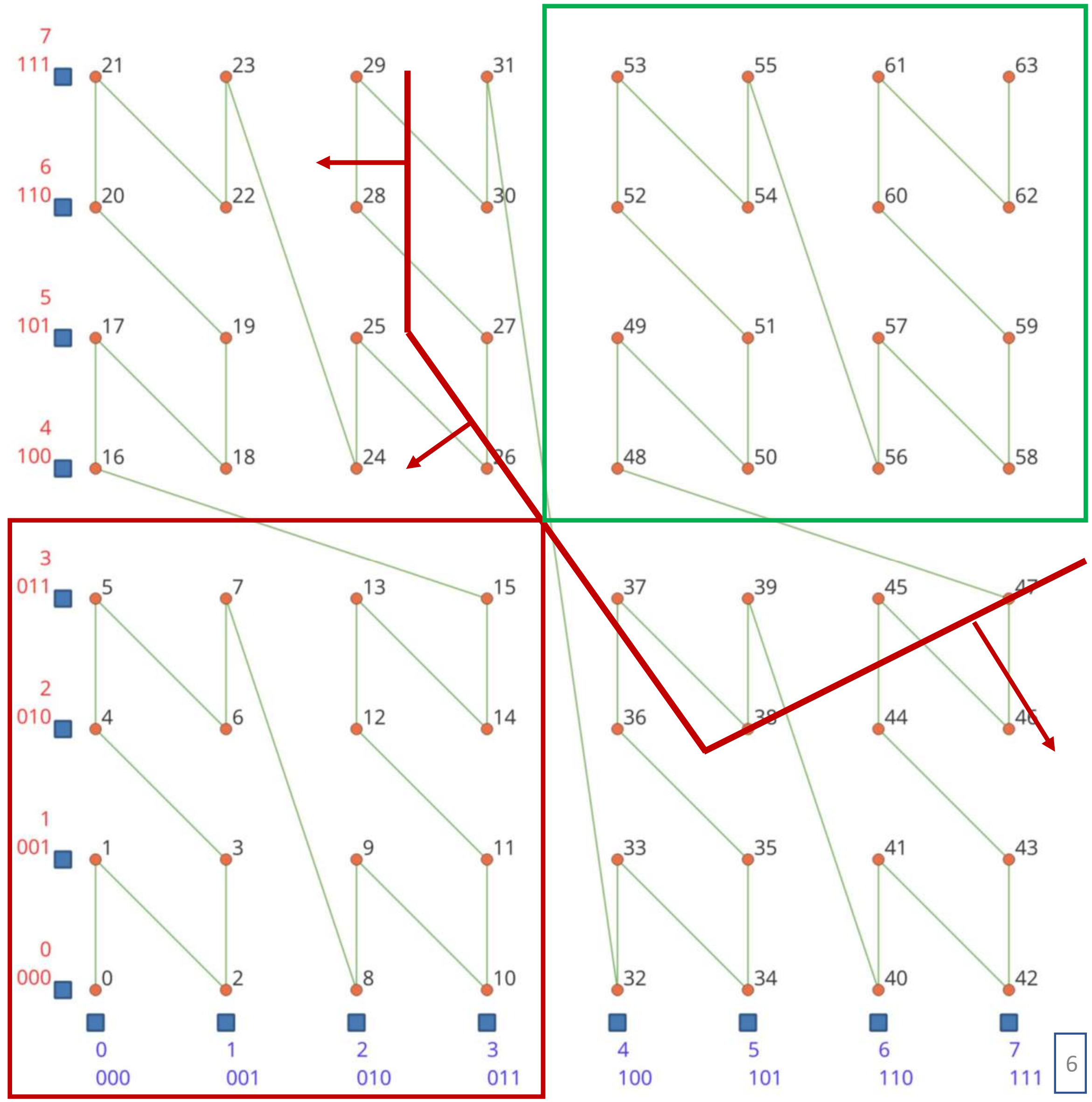
Searching an Index

If all the corners of an index hypercube are outside any halfspace, there is no need to process this index node

If all the corners of an index hypercube are within all halfspaces, this index node can be used to generate a b-tree index range.

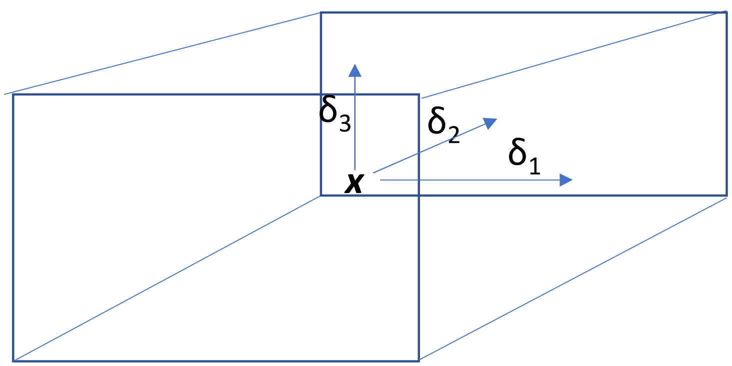
So we only need to test 2^n points to do the first tests on an index node

BUT we can do better than that!



Testing Only 2 Corners

- If an index is centred on point $\boldsymbol{x} = (x_1, x_2, x_3, \ldots)$, then we can describe the corners of the index hypercube as $(x_1 \pm \delta_1, x_2 \pm \delta_2, x_3 \pm \delta_3, \ldots)$ with $(\delta_i > 0)$ $(2^n$ corners)



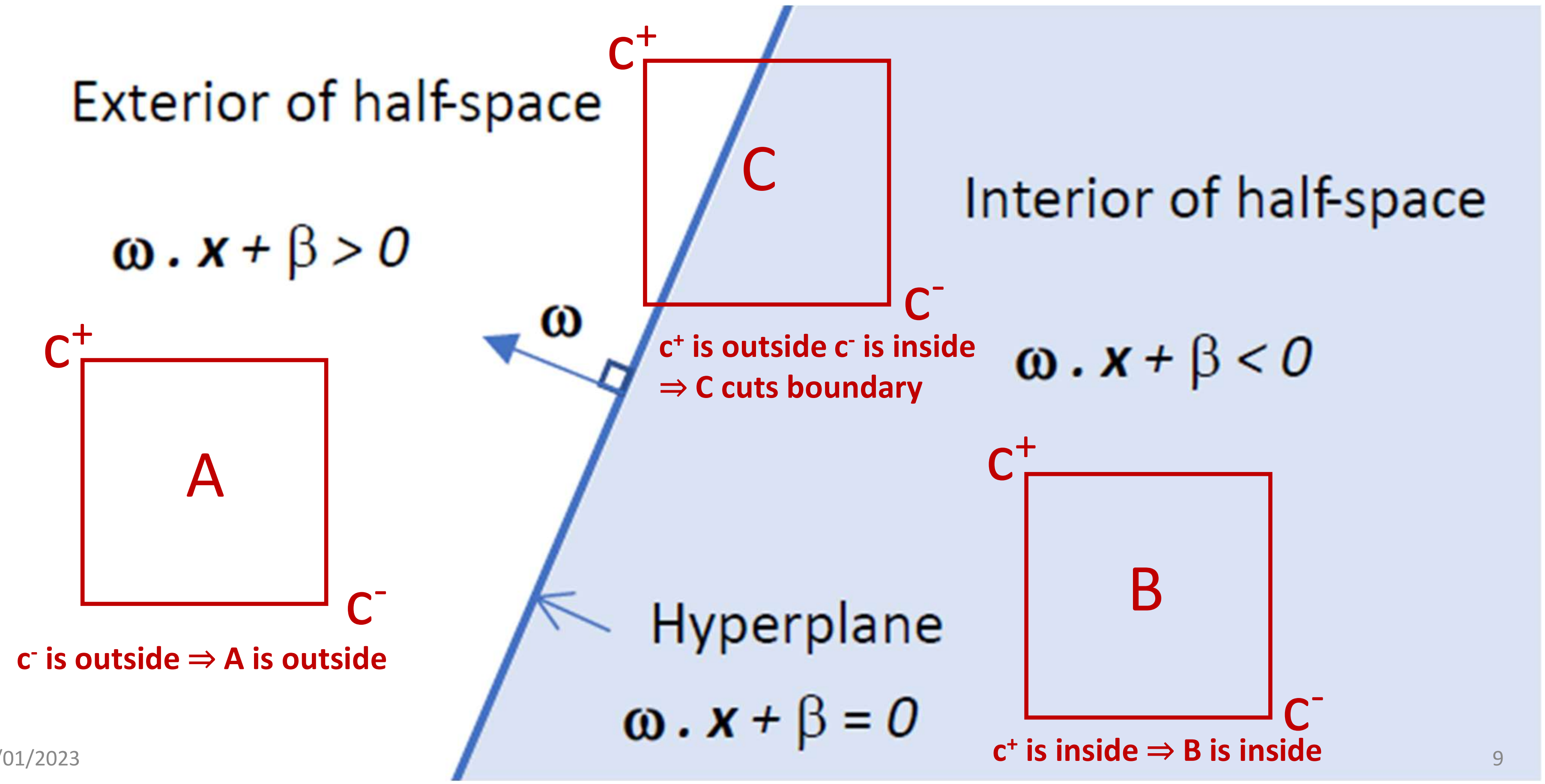
Note that the index boxes will probably be square in the x/y plane, but may have a different δ value in the z direction (and CLoI) – i.e. in many cases $\delta_1 = \delta_2 \neq \delta_3 \neq \delta_4 \ldots$

Min and Max Corners (in relation to a halfspace)

- For halfspace $\boldsymbol{H} = (\boldsymbol{\omega}, \beta)$, and index box centred on $\boldsymbol{x}$, we can define two points c^+ and c^- where
- $\mathbf{c}^+ = (c^+_i : i = 1 \dots n)$ where $c^+_i = x_i + \delta_i \,\mathrm{sign}(\omega_i) : i = 1 \dots n$
- $\mathbf{c}^- = (c^-_i : i = 1 \dots n)$ where $c^-_i = x_i - \delta_i \,\mathrm{sign}(\omega_i) : i = 1 \dots n$
- Let $D^+ = (\boldsymbol{\omega}.\boldsymbol{c}^+ + \beta)$, $D^- = (\boldsymbol{\omega}.\boldsymbol{c}^- + \beta)$,
- Clearly $D^+ > D^-$ and for any other corner c' with $D' = (\boldsymbol{\omega}.\boldsymbol{c}' + \beta)$,

  $D^+ \geq D' \geq D^-$
- So the corner $\mathbf{c}^+$ is the one most likely to be outside $\boldsymbol{H}$, while $\mathbf{c}^-$ is most likely to be inside.

Using c^+ and c^- to select index boxes

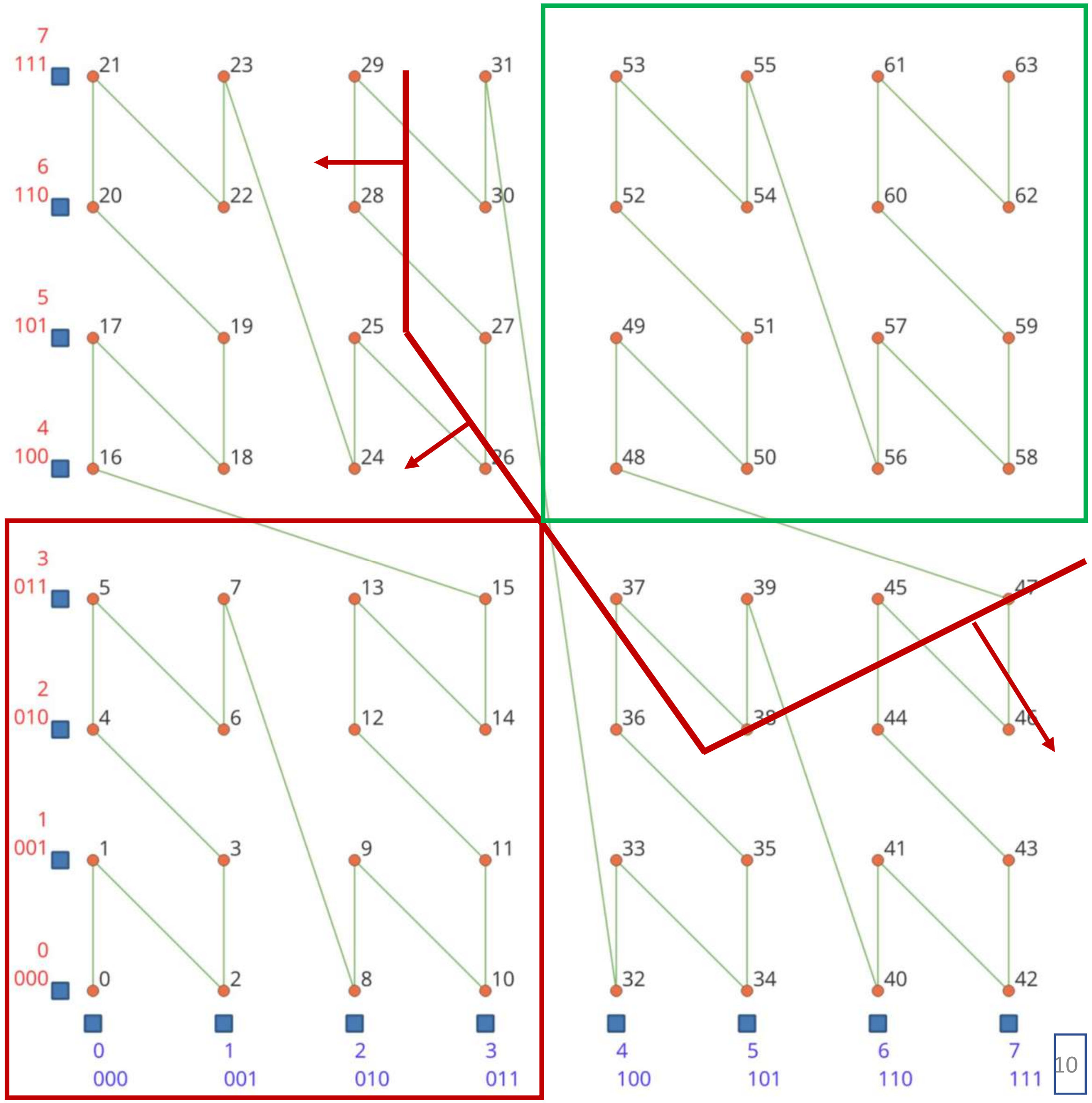


Searching x nodes of the index

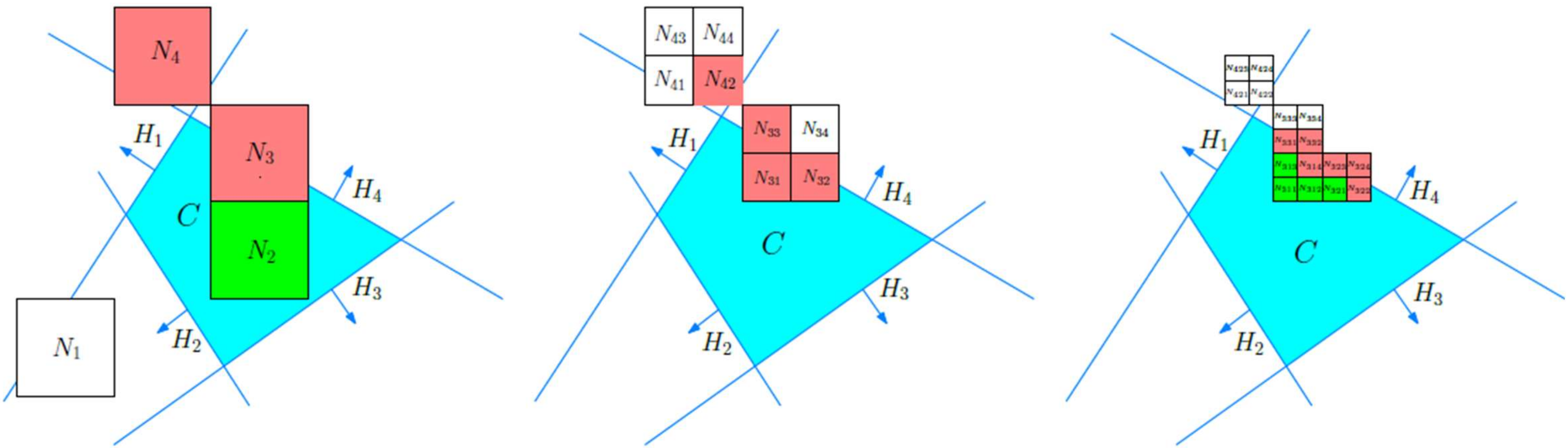
For each halfspace: O(h):

- For each index node O(x):
  - Calculate $\mathbf{c}^+$ and $\mathbf{c}^-$ O(n)
  - Calculate D^+ and D^- O(n)
  - If $D^- > 0$ discard node
  - If $D^+ < 0$ publish node
  - Otherwise pass node on for further processing

So complexity is O(hnx)



Applying this to a Convex Polytope



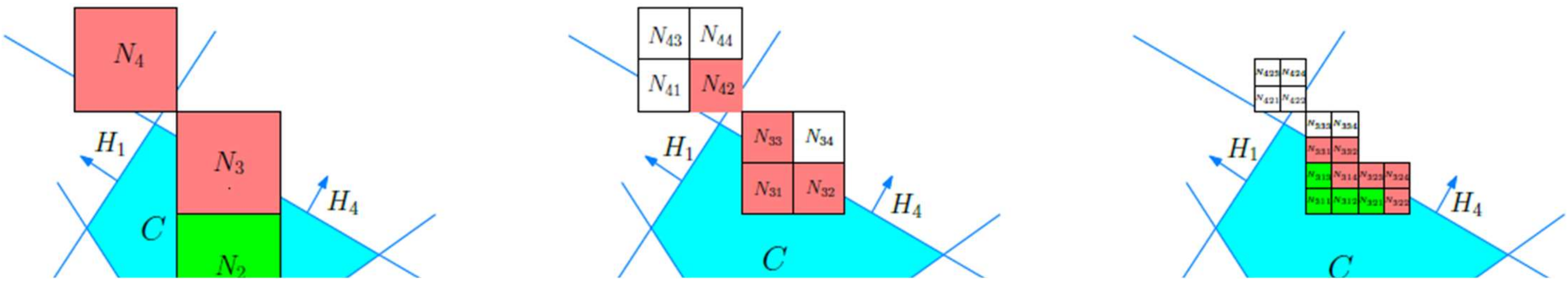
(a) Representative nodes (b) Refining once (c) Refining twice

White index hypercubes are eliminated by being outside at least one halfspace.

Each green index hypercubes is converted to a single key ranges by being within all halfspaces.

Red index hypercubes need to be split into their 2^n finer nodes which are then similarly processed.

Guard Halfspaces



Looking at N_4 in the above case study, it is totally outside the query region, but not outside any one halfspace.

This means it is traced to the next level, where all sub-nodes except N_{42} are eliminated

When N_{42} is traced to the next level, all subnodes N_{421} to N_{424} are eliminated.

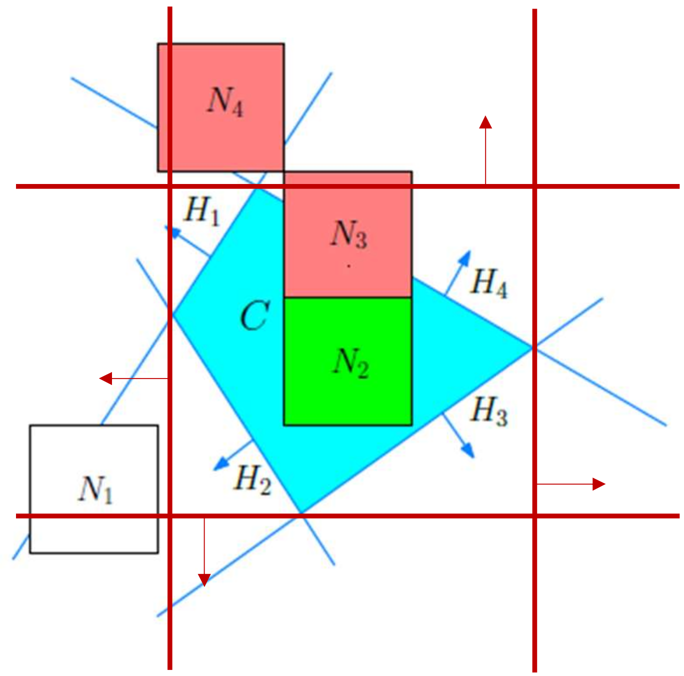
So no harm is done – the answer is correct.
But it may be time consuming – especially if the sub-nodes are not in-memory.

It happens more often at acute dihedral angle edges.

Guard Halfspaces

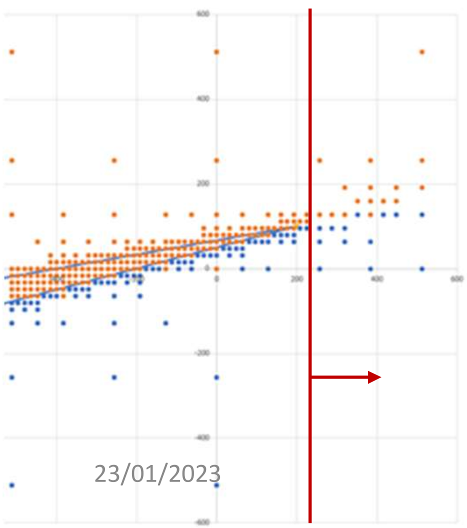
Add half-spaces at any corners/edges that may need to be guarded.

e.g. in this case N_4 is eliminated in the first pass.



Hypothesis – no more than 2^n guards are needed for the whole convex region.

Hypothesis – up to n-1 guards are needed for an individual edge.



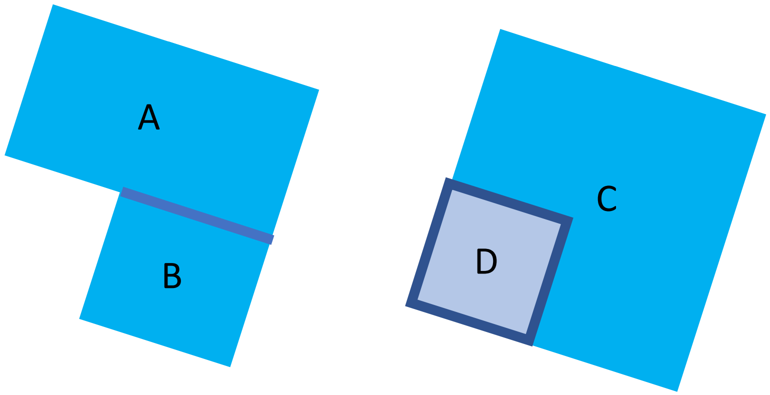
Issues: It gets complicated to understand for $n > 3$.

It adds half spaces to the definition, which may slow the extraction more than is gained by eliminating the “false positive nodes”.

Research is needed.

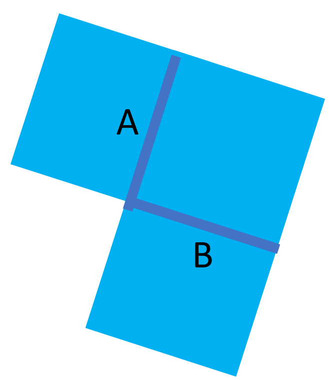
Non-convex regions

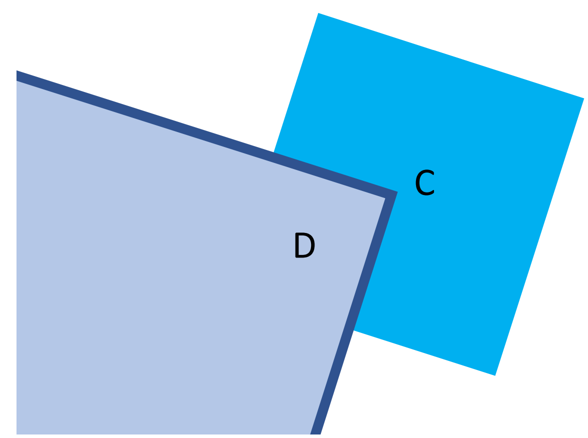
- It may still be useful using these methods for non-convex regions (because fully general regions in nD can be very slow and hard to process)
- 2 basic approaches (non-exclusive)
  - Additive - A or B
  - Subtractive – C and not D
- The complexity now becomes O(*nch*)
  - In *n*D, *c* convex regions, *h* number of half spaces in most complex convex region.



Non-convex regions – unusual features

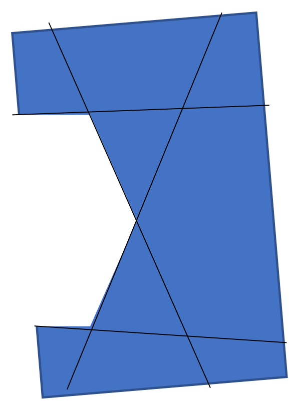
- Unlike most cases where regions are divided into convex subregions – the regions shouldn't be made non-overlapping
- Better results if A and B are fully overlapping
- Best result if D extends to infinity.



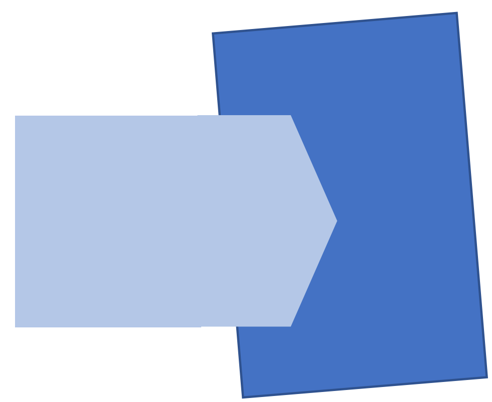


Open Questions

- How effective are guard halfspaces?
- How effective is splitting non-convex regions into convex subregions?
- How to split non-convex regions?
- How to use additive vs. subtractive splitting most effectively?



additive – 4 convex regions



subtractive – 2 convex regions

nD Convex Polytopes for Point Cloud Query.

Questions?

Rod Thompson

23/Jan/2023