CS 53000 - Introduction to Scientific Visualization

Data Representation and Basic Processing

August 29, 2011

Outline

- Grid Structure
- Interpolation
- Search

Input Data

- Discrete positions (vertices)
- *N* dimensions, *N*=*1*, *2*, *3*, ...
- With or without connectivity information
  - Structured
  - Unstructured
  - Scattered

grid topology

Grid Structure

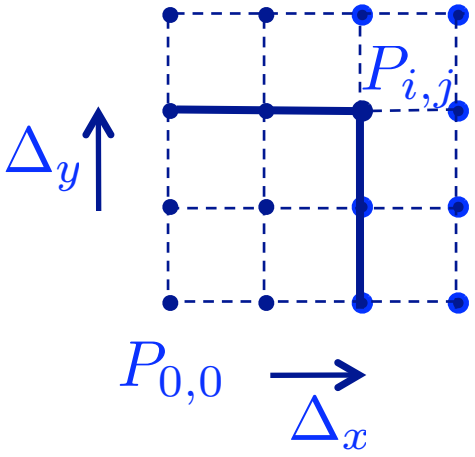
Classification

- Geometry
  - Position of vertices in Euclidean space
  - Structured / unstructured
- Topology
  - Cells
  - Connectivity information
  - Neighborhood definition
  - Structured / unstructured

Grid Geometry

Uniform

- implicit relationship between points
- positions can be computed (procedural)



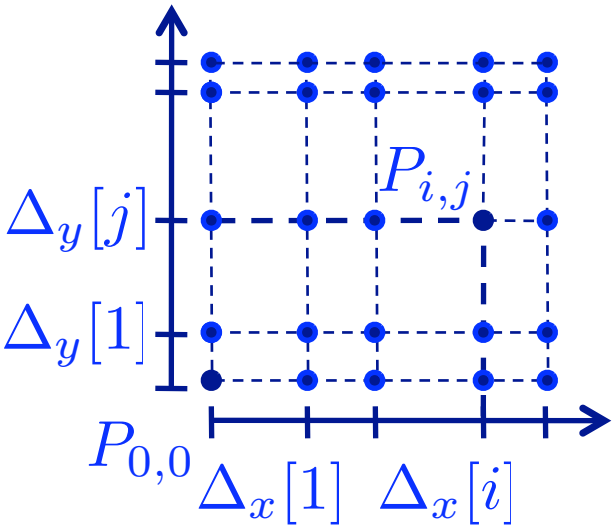
Uniform grid

$$P_{i,j,k} = P_{0,0} + i\Delta_x\vec{e}_x + j\Delta_y\vec{e}_y + k\Delta_z\vec{e}_z$$

Grid Geometry

Structured

- implicit relationship between points
- positions can be computed (procedural)



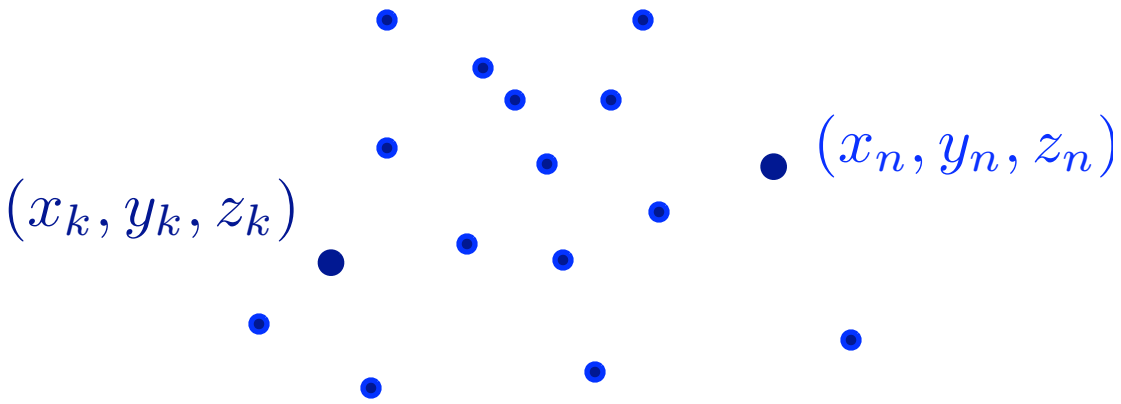
Rectilinear grid

$$P_{i,j,k} = P_{0,0} + \Delta_x[i]\vec{e}_x + \Delta_y[j]\vec{e}_y + \Delta_z[k]\vec{e}_z$$

Grid Geometry

Unstructured

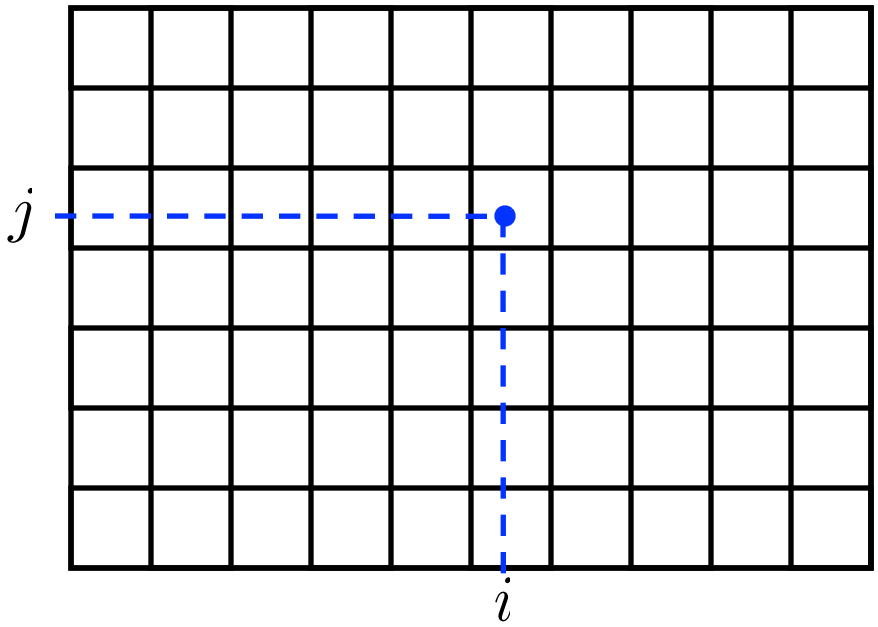
- No underlying structure
- Required explicit position of every vertex



Grid Topology

Structured *(quadrilateral / hexahedron)*

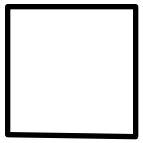
- Implicit connectivity between vertices
- Implicit cell definition



Associated with

structured geometry:

- very efficient position location

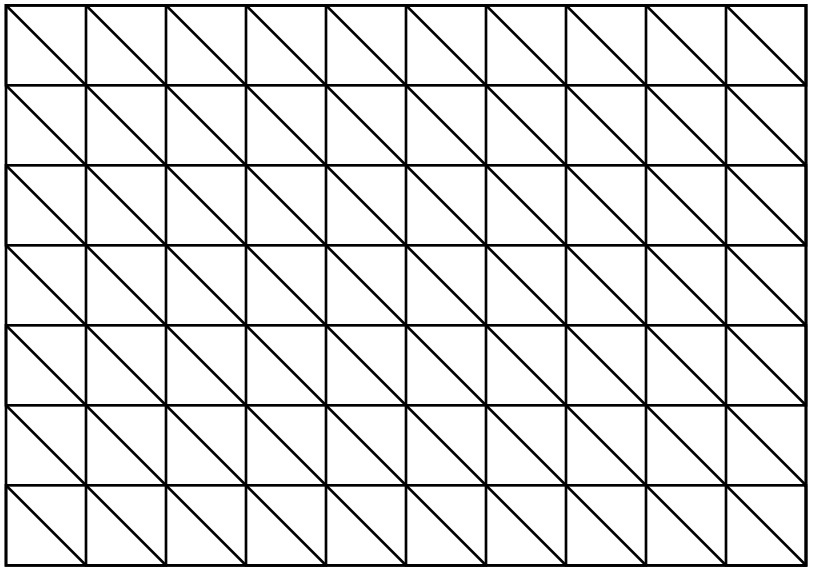


Pixel

Grid Topology

Structured *(quadrilateral / hexahedron)*

- Implicit connectivity between vertices
- Implicit cell definition



Associated with

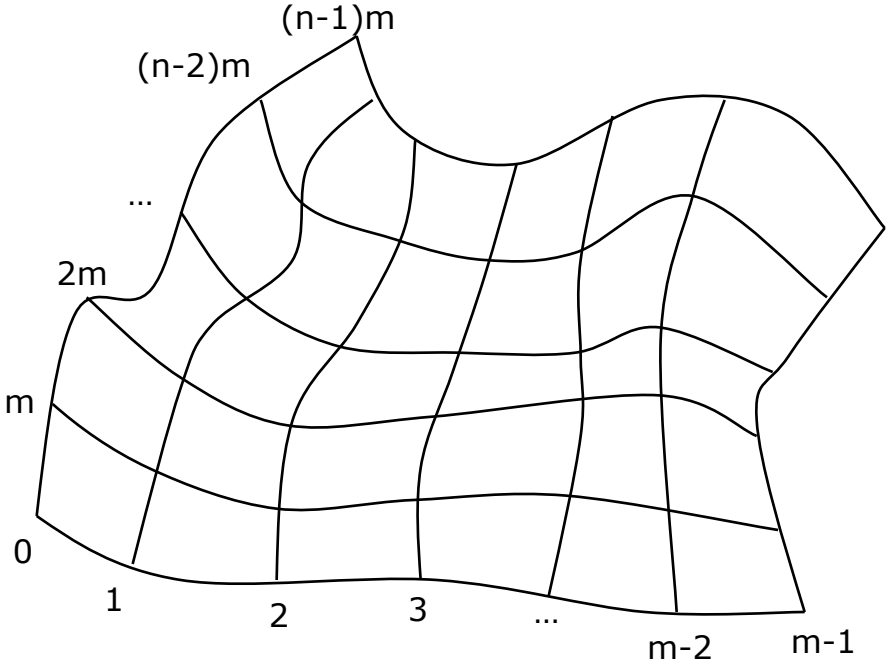
structured geometry:

- Implicit triangulation

Grid Topology

Structured *(quadrilateral / hexahedron)*

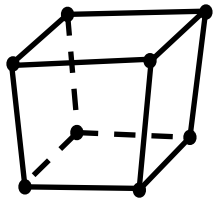
- Implicit connectivity between vertices
- Implicit cell definition



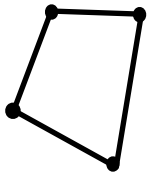
Associated with

unstructured geometry:

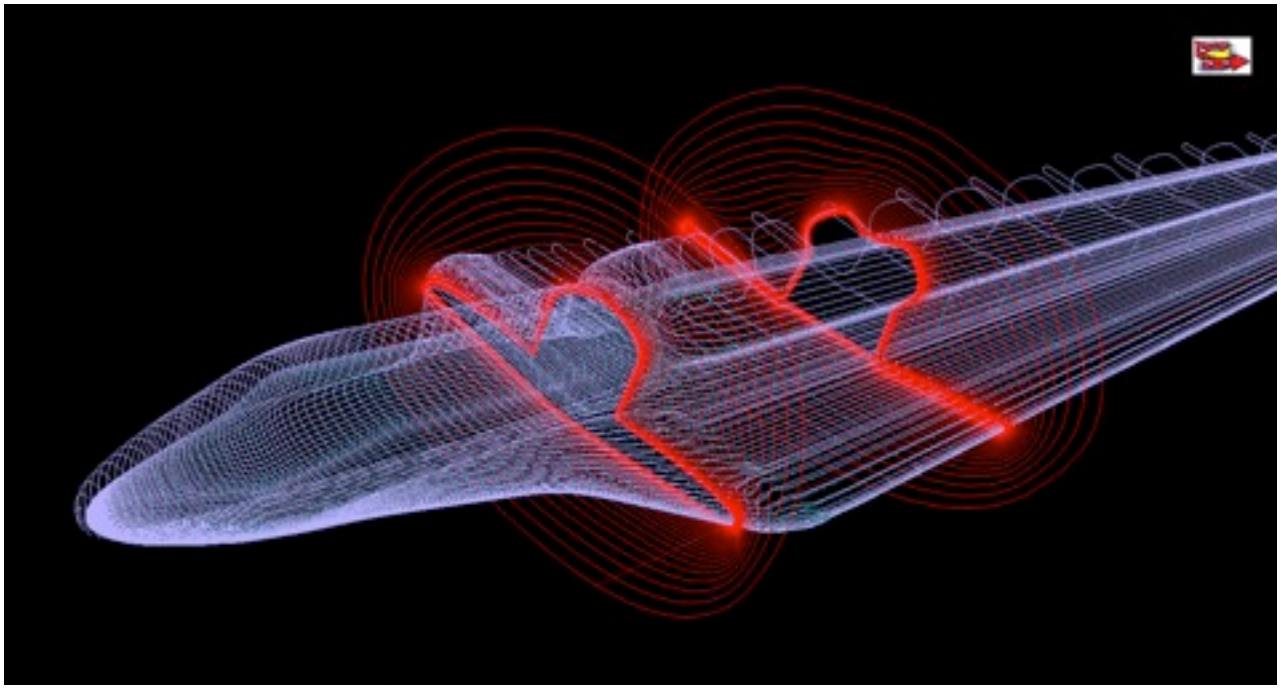
curvilinear grid

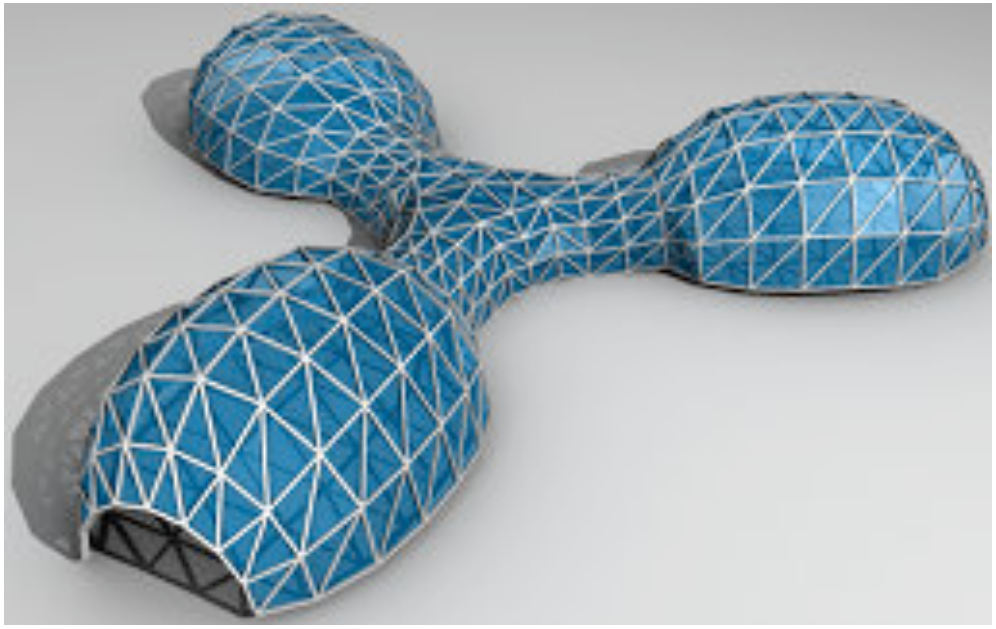


Hexahedron



Quadrilateral

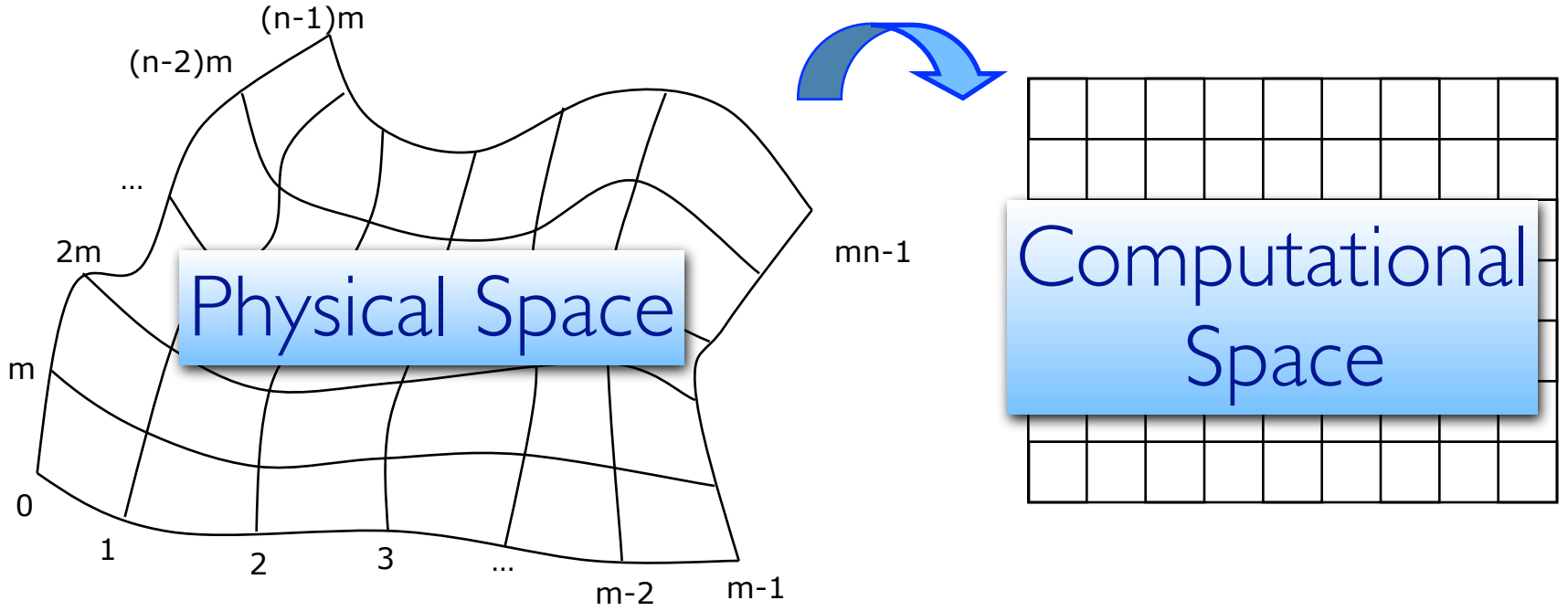




Grid Topology

Structured *(quadrilateral / hexahedron)*

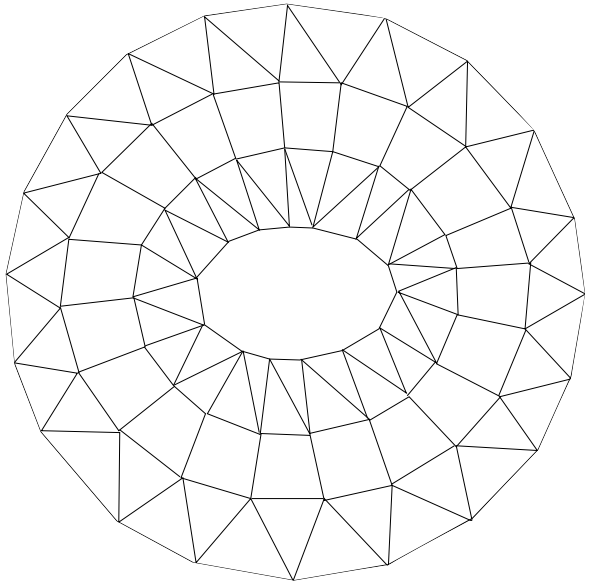
- Implicit connectivity between vertices
- Implicit cell definition

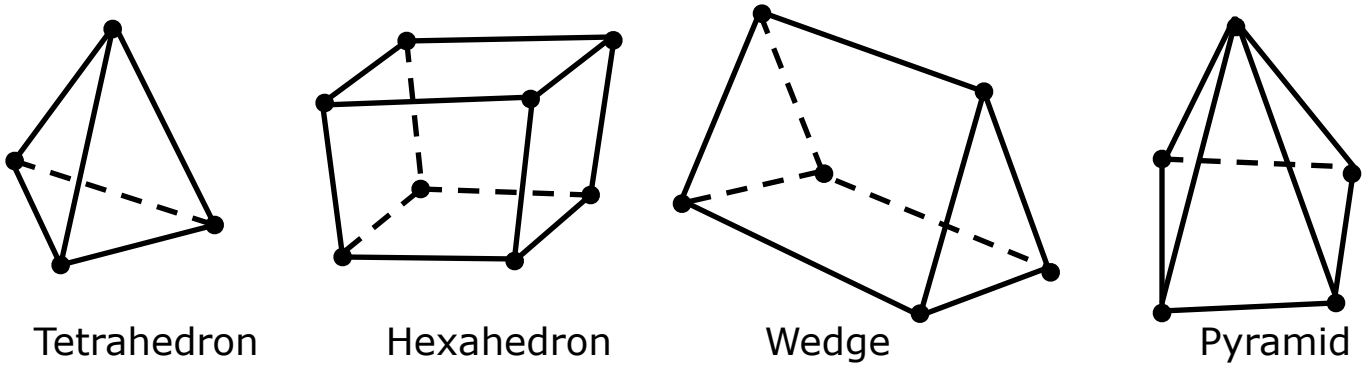


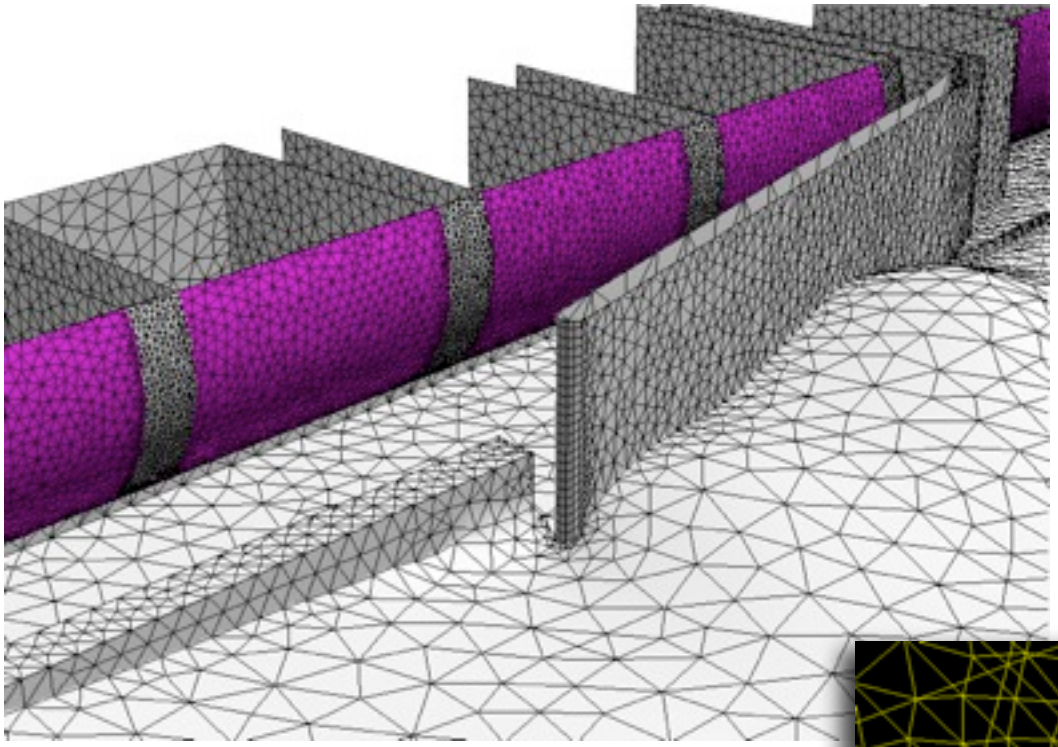
Grid Topology

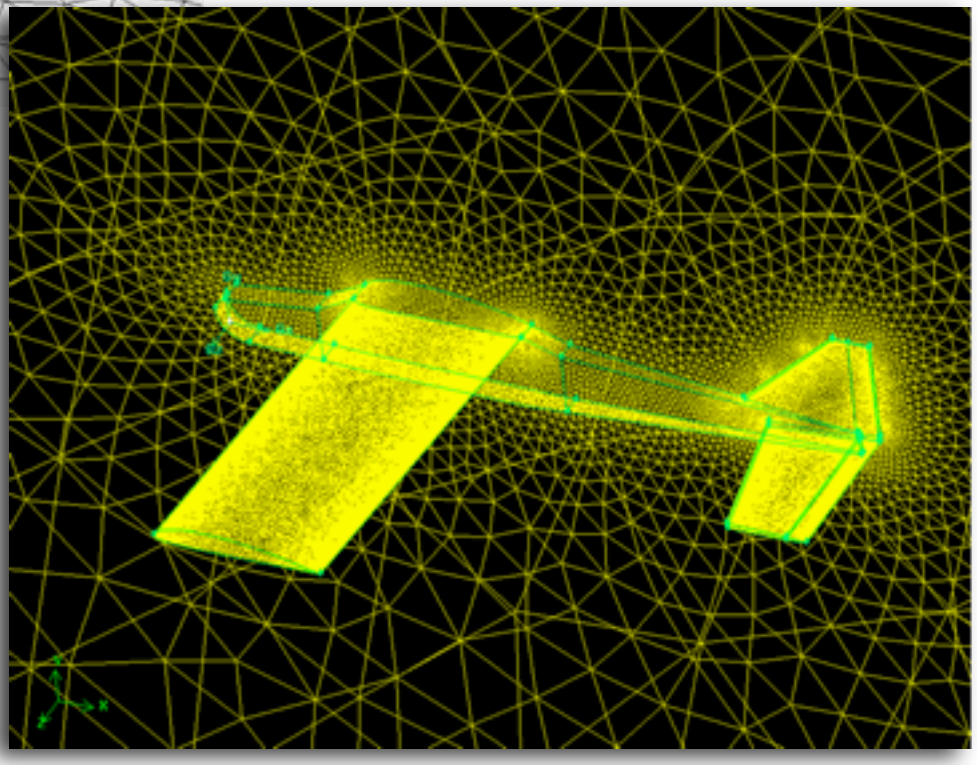
Unstructured *(any cell type)*

- Explicit cell definition
  - Types
  - Vertices



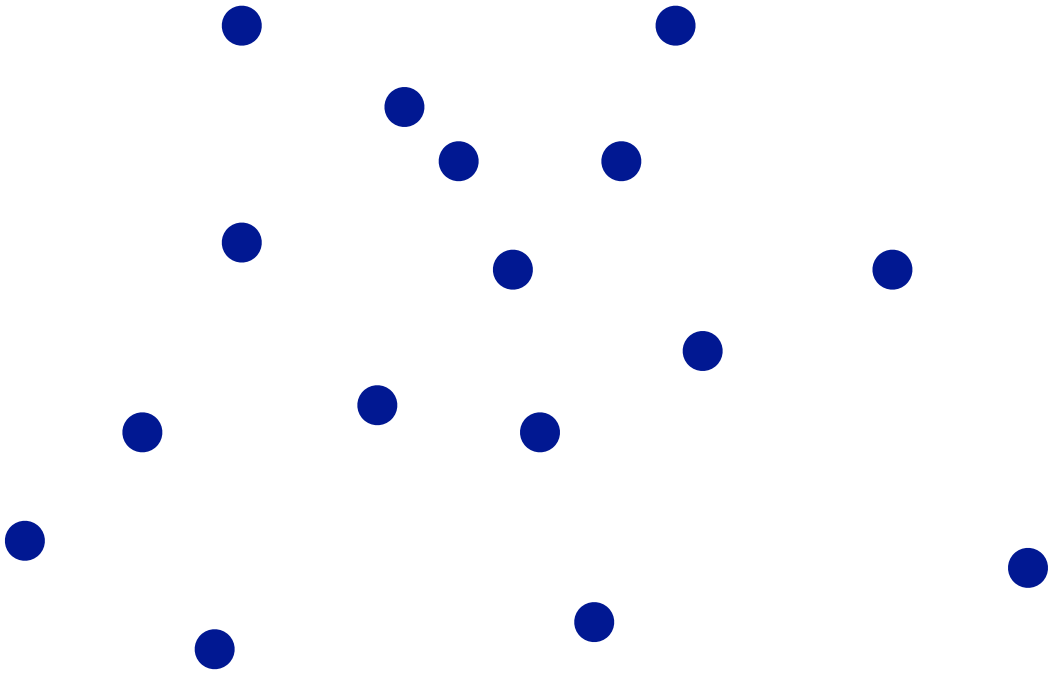


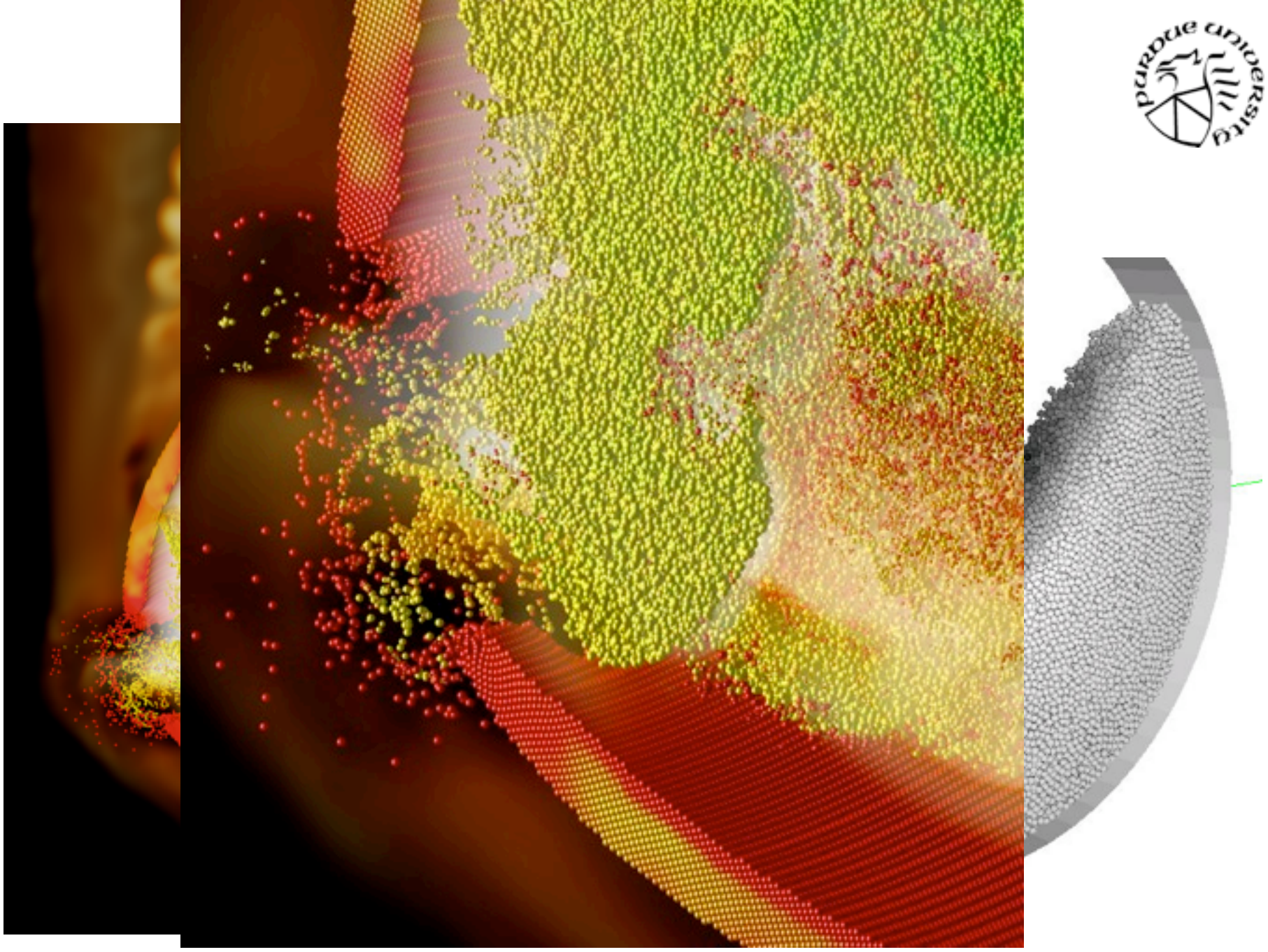




Grid Topology

- Mesh-free (no grid, no connectivity)



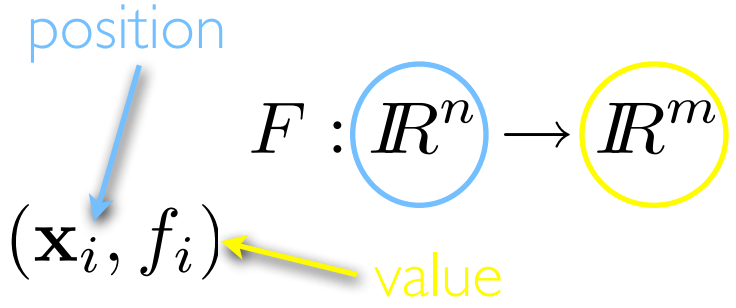


Outline

- Grid Structure
- Interpolation
- Search

Interpolation

- Continuous reconstruction of discrete input data



$$\forall i \in \{1, .., n\}, F(\mathbf{x}_i) = f_i$$

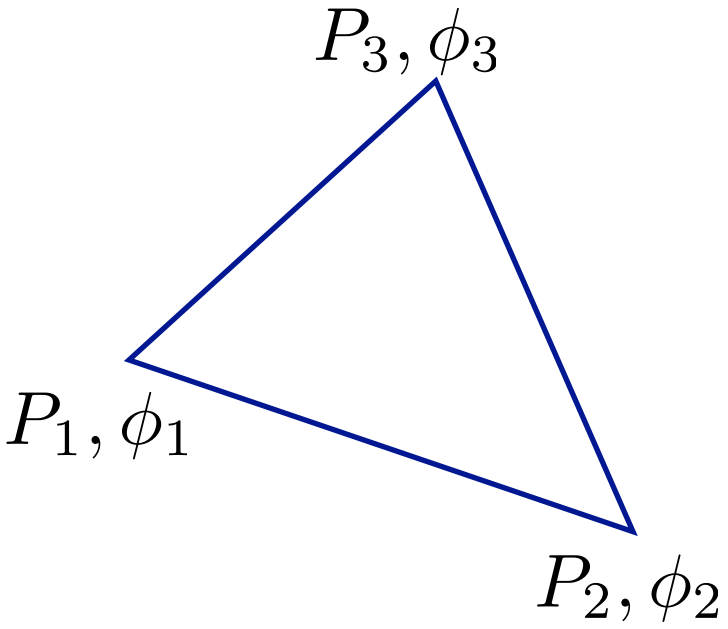
- Depends on grid structure (or lack thereof)
- Interpolation vs. approximation

Linear Interpolation

- In triangle (2D simplex)

$$\phi(x, y) = a + bx + cy$$

$$\forall i, \phi(P_i) = \phi_i$$

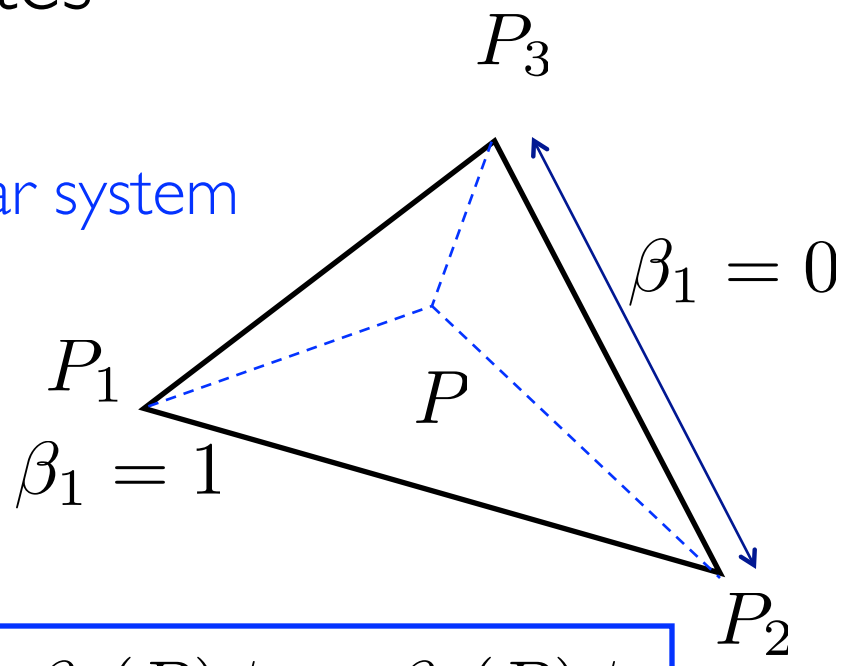


Linear Interpolation

- Barycentric coordinates

$$\begin{cases} P = \sum_{i=1}^{3} \beta_i P_i \\ \sum_{i=1}^{3} \beta_i = 1 \end{cases}$$

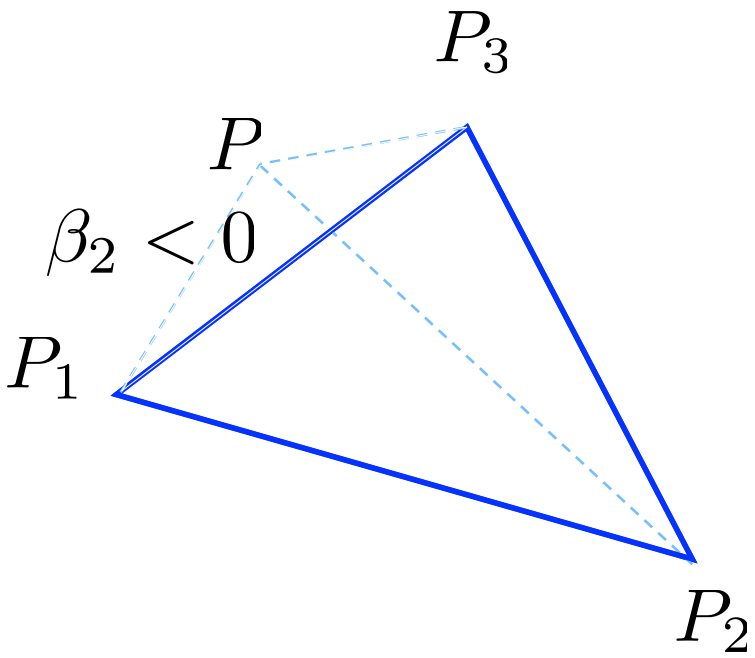
Linear system



$$\phi(P) = \beta_1(P)\phi_1 + \beta_2(P)\phi_2 + \beta_3(P)\phi_3$$

Linear Interpolation

- Barycentric coordinates
  - P lies outside triangle ⇔ at least one β_i is negative
  - Easy way to check if a position lies inside a triangle



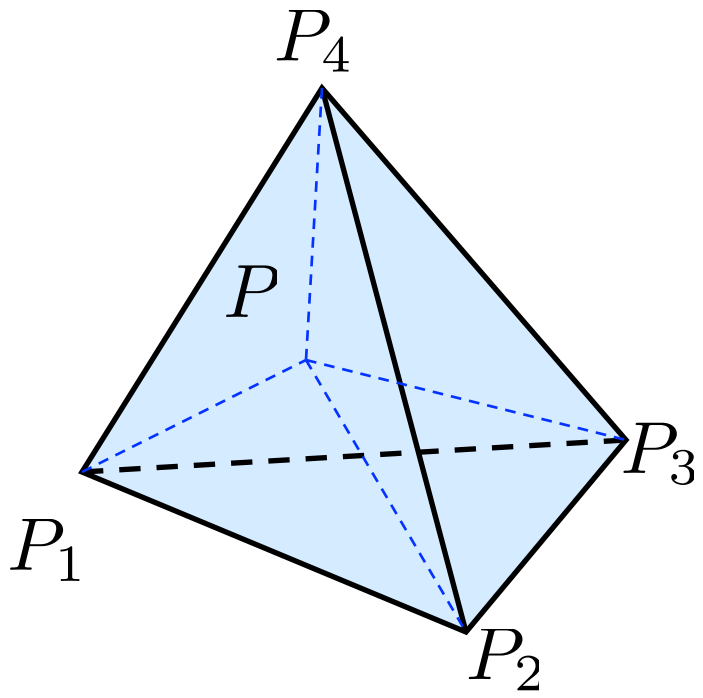
Linear Interpolation

- In tetrahedron (3D simplex)
  - Barycentric coordinates

$$P = \sum_{i=1}^{4} \beta_i P_i, \sum_{i=1}^{4} \beta_i = 1$$

$$\phi(P) = \sum_{i=1}^{4} \beta_i(P)\phi_i$$

  - Same inclusion test



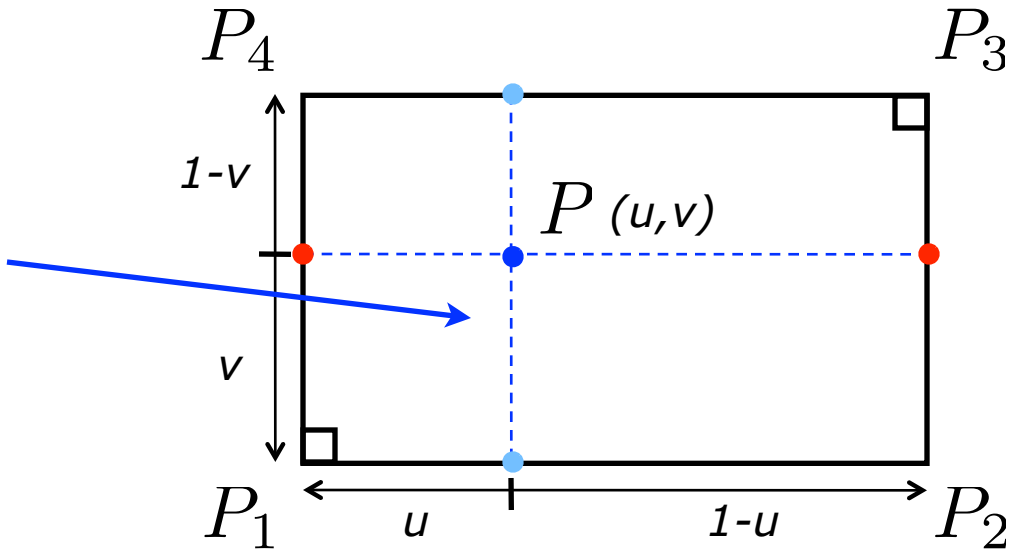
Bilinear Interpolation

- Bilinear interpolation

$$\phi(x, y) = axy + bx + cy + d$$

$$\forall i, \phi(P_i) = \phi_i$$

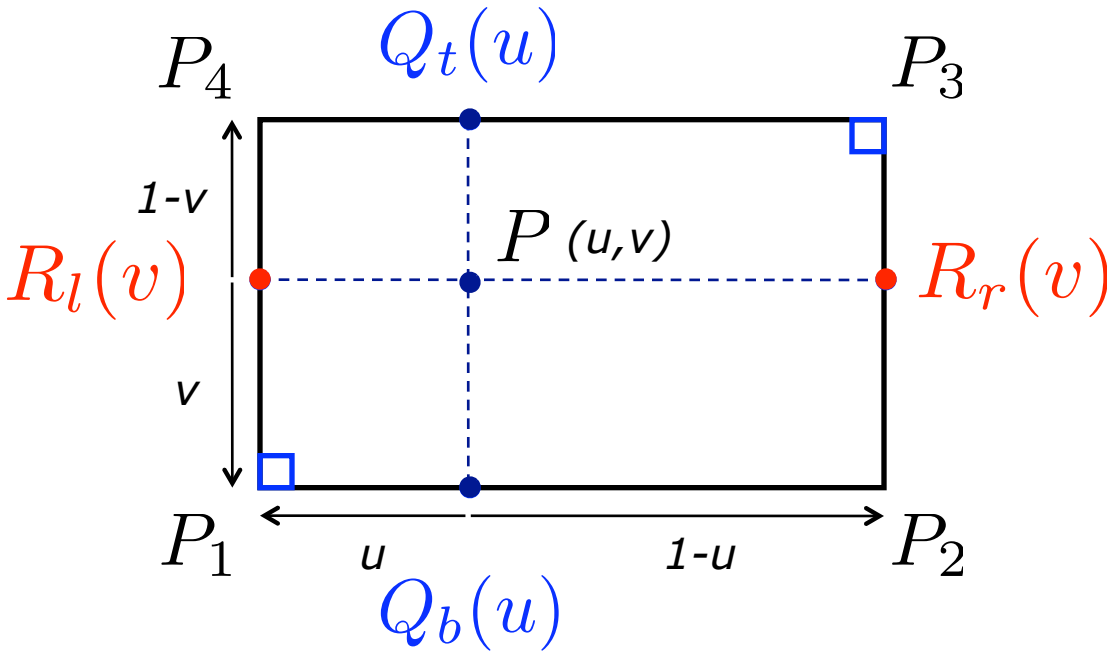
Combination of two consecutive linear interpolation



Bilinear Interpolation

- In rectangle

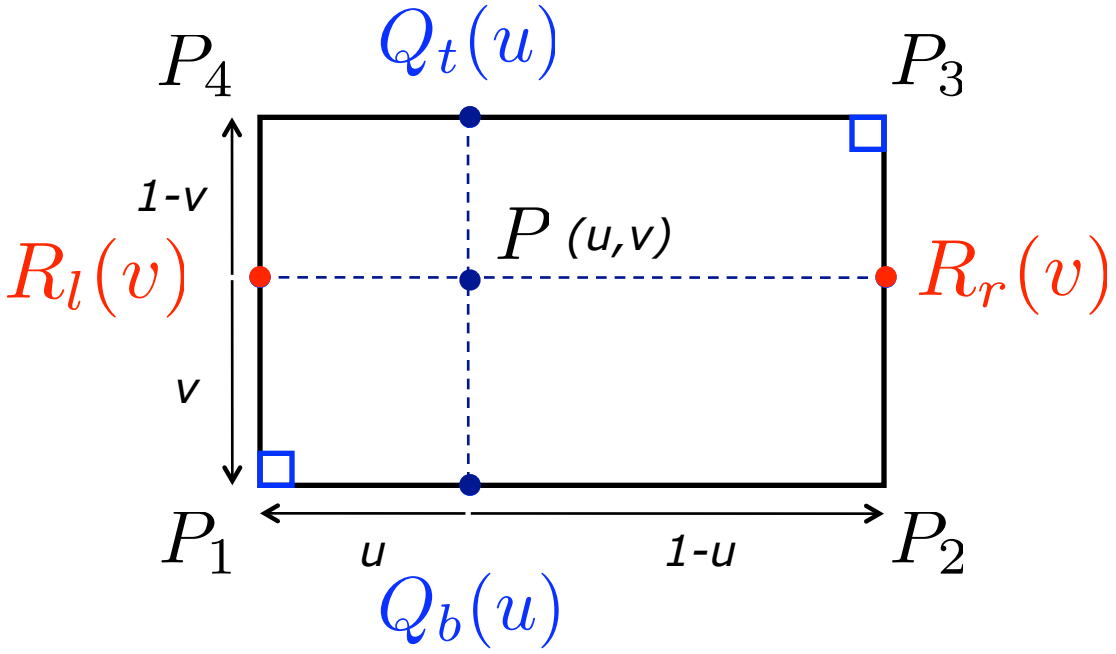
$$P = (1-v)Q_b(u) + vQ_t(u)$$
$$= (1-u)R_l(v) + uR_r(v)$$



Bilinear Interpolation

- In rectangle

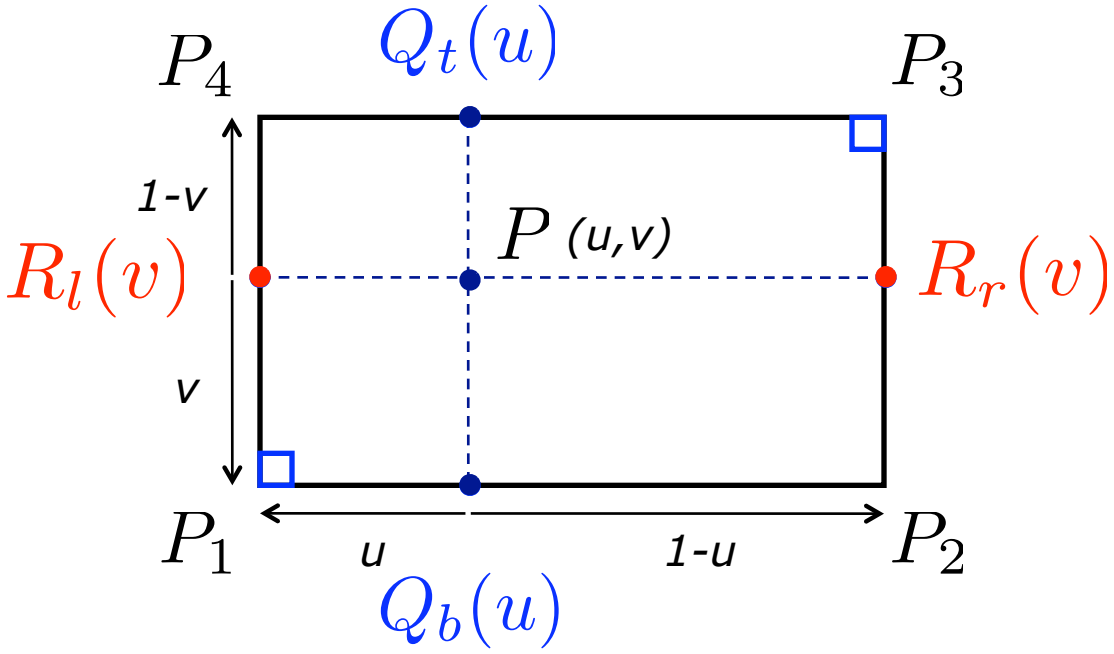
$$P = P_1 + u(P_2 - P_1) + v(P_4 - P_1) + uv(P_1 - P_2 + P_3 - P_4)$$



Bilinear Interpolation

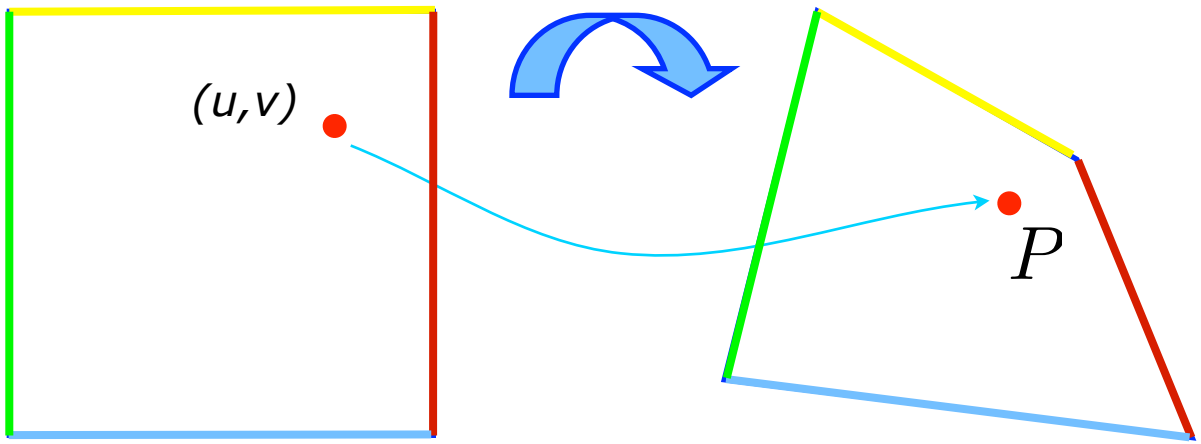
- In rectangle

$$\phi(P) = \phi_1 + u(\phi_2 - \phi_1) + v(\phi_4 - \phi_1) + uv(\phi_1 - \phi_2 + \phi_3 - \phi_4)$$



Bilinear Interpolation

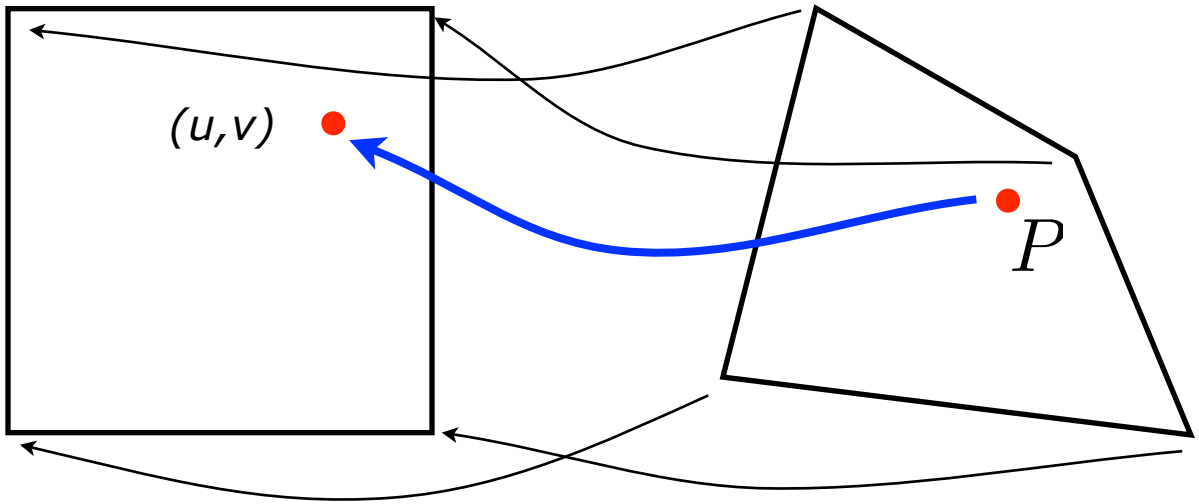
- In arbitrary quadrilateral
  i. Start from (u,v) coordinates (if available) in computational space and apply previous formula



Important: axis-parallel straight lines are mapped to straight lines

Bilinear Interpolation

- In arbitrary quadrilateral
  - ii. Otherwise
    1. Compute (inverse) mapping from physical space to computational space (unit square): quadratic eqns
    2. goto i.



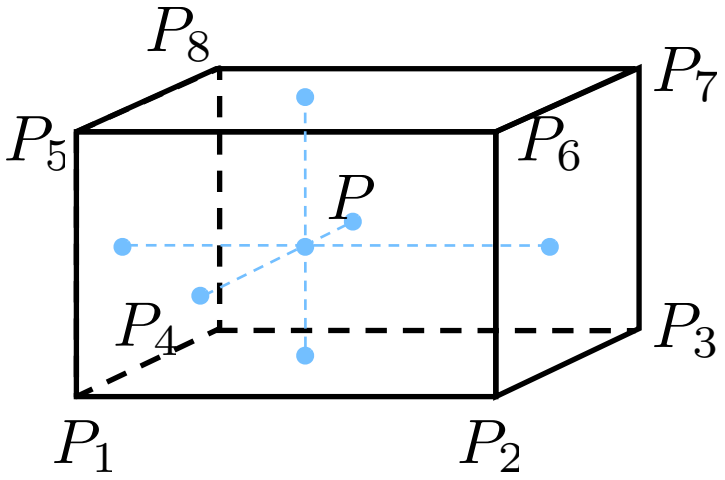
Trilinear Interpolation

- In a cuboid (axis parallel)
  - general formula

$$\phi(x,y,z) = axyz + bxy + cxz + dyz + ex + fy + gz + h$$

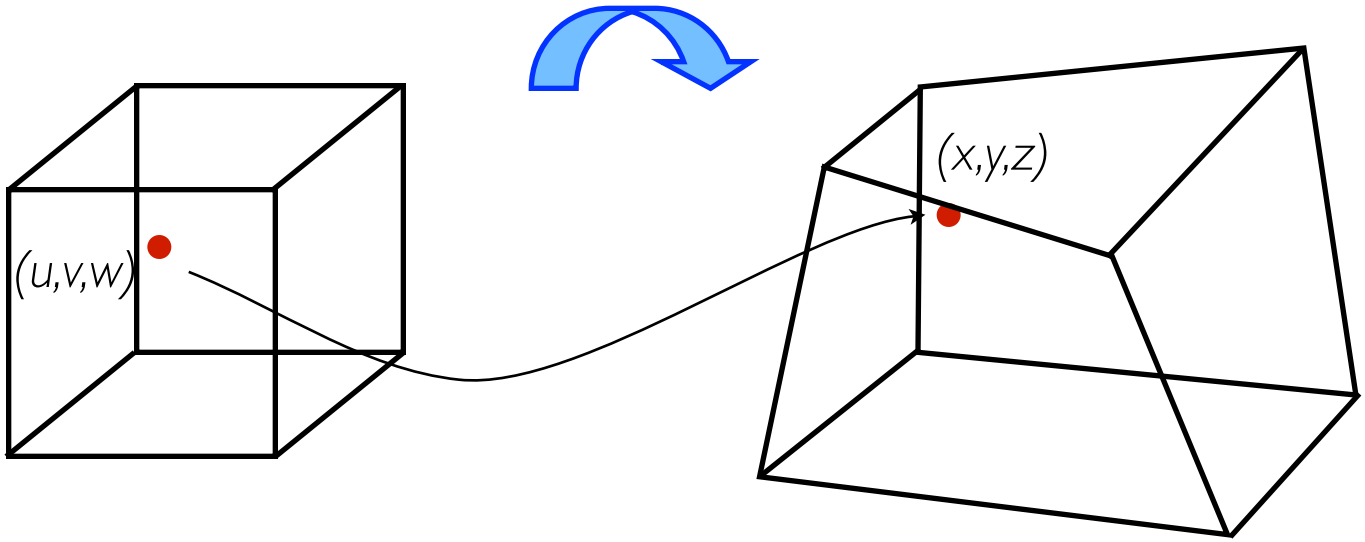
  - with local coordinates

$$\begin{aligned} P = P_1 \\ &+u(P_2 - P_1) \\ &+v(P_4 - P_1) \\ &+w(P_5 - P_1) \\ &+uv(P_1 - P_2 + P_3 - P_4) \\ &+uw(P_1 - P_2 + P_6 - P_5) \\ &+vw(P_1 - P_4 + P_8 - P_5) \\ &+uvw(P_1 - P_2 + P_3 - P_4 + P_5 - P_6 + P_7 - P_8) \end{aligned}$$



Trilinear Interpolation

- In arbitrary hexahedron
  i. Start from (u,v) coordinates (if available) in computational space and apply previous formula



Trilinear Interpolation

- In arbitrary hexahedron
  - ii. Otherwise
    1. compute (inverse) mapping from physical space to computational space (axis aligned unit cube)
       - I. nonlinear problem

$$(x, y, z)^T = F(u, v, w)^T$$

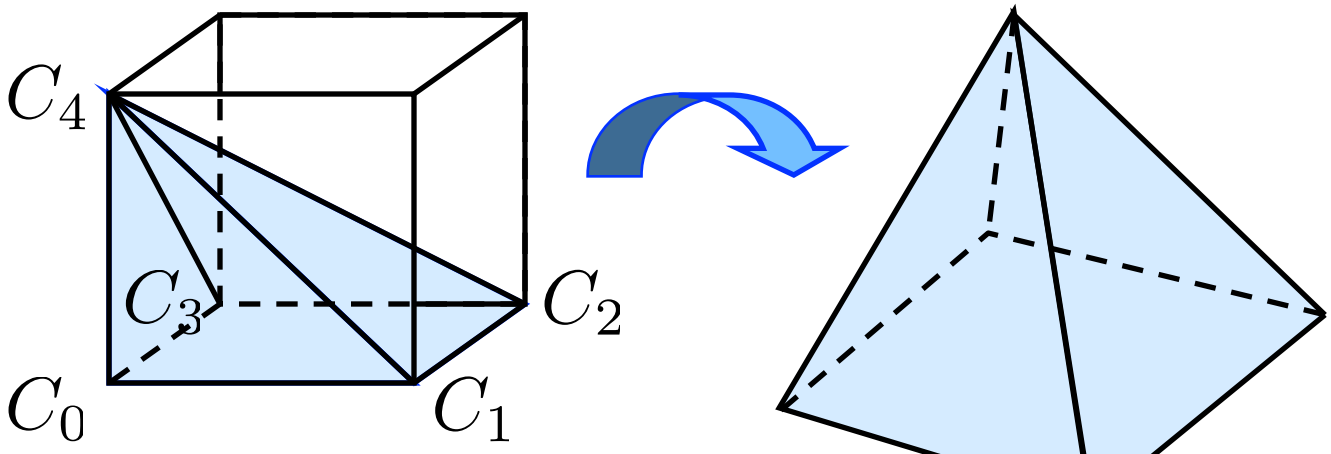
       - II. solved with iterative scheme

$$F(\mathbf{u} + \vec{\delta}) = F(\mathbf{u}) + J\vec{\delta} + ...$$
$$J\vec{\delta} = -F(\mathbf{u})$$

    2. goto i.

Trilinear Interpolation

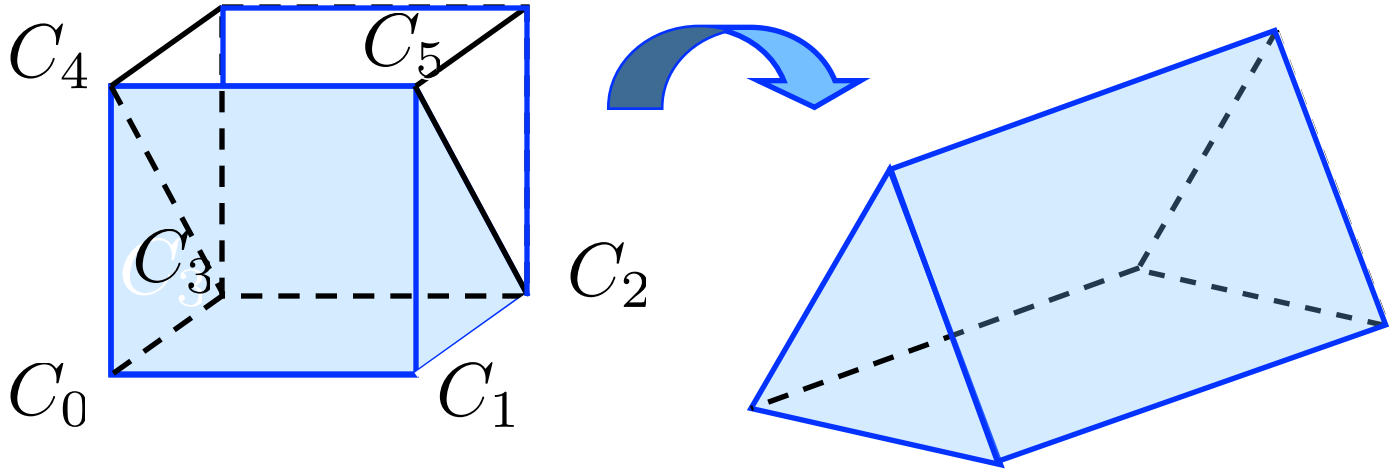
- In pyramids (special case of trilinear int.)



$$P(u,v,w) = (1-u)(1-v)(1-w)C_0$$
$$+u(1-v)(1-w)C_1$$
$$+uv(1-w)C_2$$
$$+(1-u)v(1-w)C3 + wC_4$$

Trilinear Interpolation

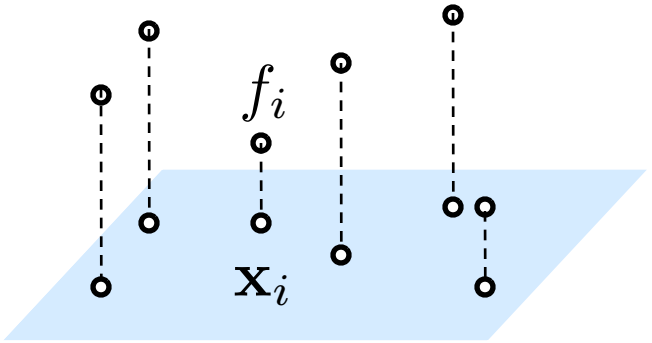
- In prisms (special case of trilinear int.)



$$P(u,v,w) = (1-u)(1-v)(1-w)C_0$$
$$+u(1-v)(1-w)C_1$$
$$+uv(1-w)C_2$$
$$+(1-u)v(1-w)C_3$$
$$+(1-u)wC_4 + uwC_5$$

Scattered Data Interpolation

- Shepard methods



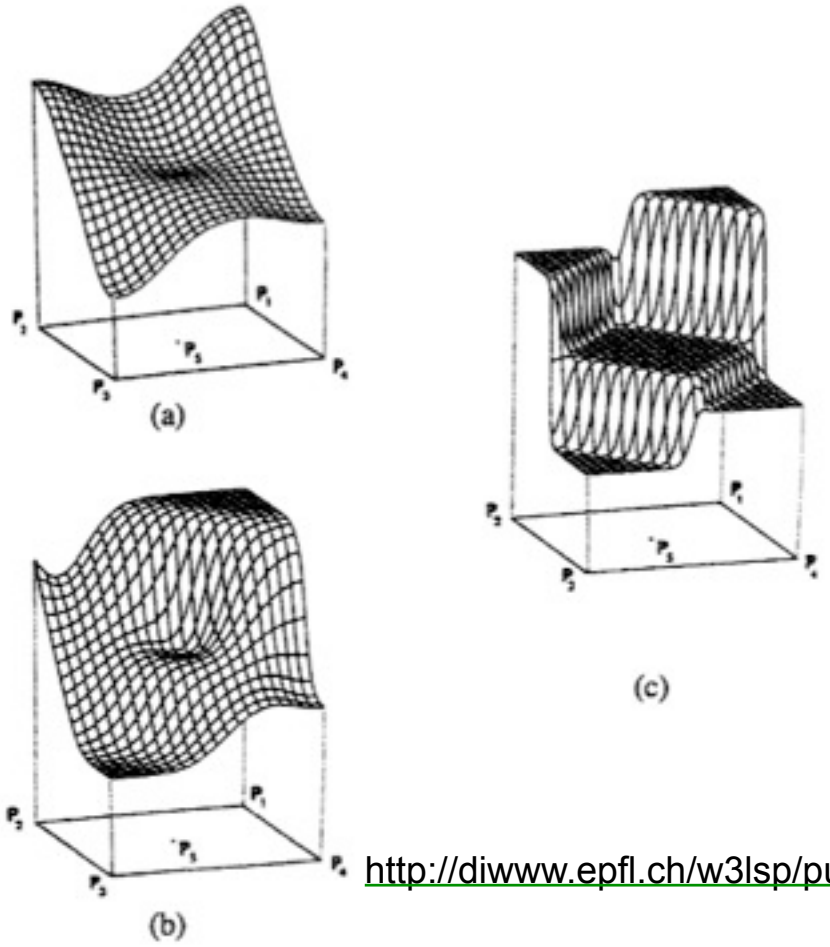
$$\sigma_i(\mathbf{x}) = \frac{1}{||\mathbf{x} - \mathbf{x}_i||^k}$$

Original Shepard

- flat spots
- global

Scattered Data Interpolation

- Shepard methods



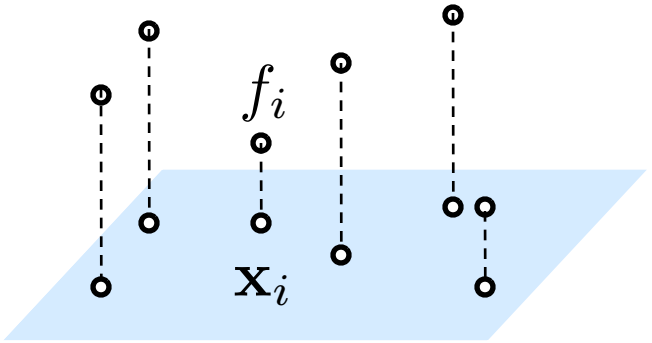
http://diwww.epfl.ch/w3lsp/publications/other/sdimfeisas.pdf

Scattered Data Interpolation

- Shepard methods

$$F(\mathbf{x}) = \sum_{i=1}^{N} \omega_i(\mathbf{x}) f_i$$

$$\omega_i(\mathbf{x}) = \frac{\sigma_i(\mathbf{x})}{\sum_{j=1}^{N} \sigma_j(\mathbf{x})}$$



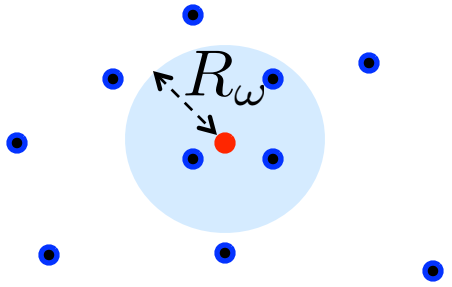
$$\sigma_i(\mathbf{x}) = \exp -(\alpha_i(x - x_i)^2 + \beta_i(y - y_i)^2 + \gamma_i(z - z_i)^2)$$

Scattered Data Interpolation

- Shepard methods

$$F(\mathbf{x}) = \sum_{i=1}^{N} \omega_i(\mathbf{x}) Q_i(\mathbf{x})$$

$$\omega_i(\mathbf{x}) = \frac{\sigma_i(\mathbf{x})}{\sum_{j=1}^{N} \sigma_j(\mathbf{x})}$$



$$\sigma_i(\mathbf{x}) = \frac{1}{||\mathbf{x} - \mathbf{x}_i||} \left(1 - \frac{||\mathbf{x} - \mathbf{x}_i||}{R_\omega}\right)^2$$

Franke, Nielson
quadratic fit
local

Radial Basis Functions

- Set of points: $X = (\mathbf{x}_i)_{i=1}^N \subset I\!R^d$
- Associated values: $(f_i)_{i=1}^N \subset I\!R$
- Find coefficients $(c_i)_{i=1}^N \subset I\!R$ satisfying

$$P_f(\mathbf{x}) = \sum_{i=1}^{N} c_i \phi(||\mathbf{x} - \mathbf{x}_i||)$$

- System of equations: $\mathbf{Ac} = \mathbf{f}$ with

$$A_{ij} = \phi(||\mathbf{x}_i - \mathbf{x}_j||)$$

Radial Basis Functions

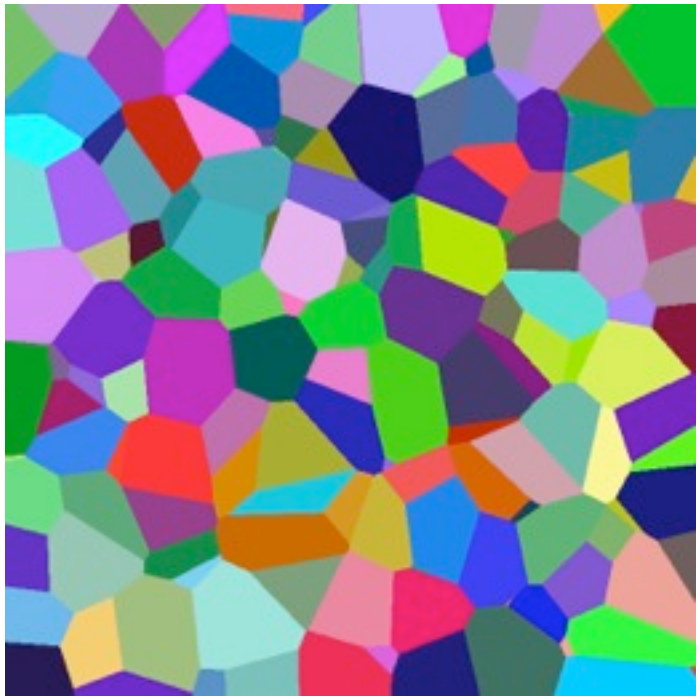
- Hardy's inverse multiquadrics

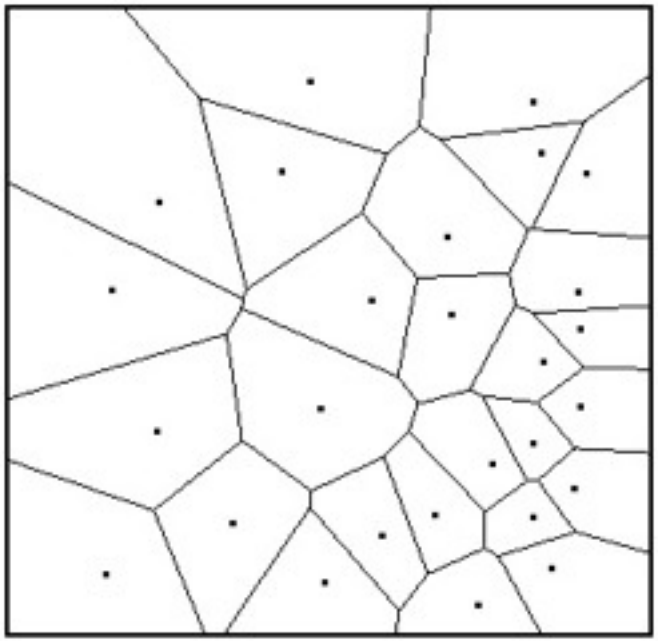
$$f(r) = \frac{1}{\sqrt{r^2 + c}}$$

- Other radial functions used in practice
  - Thin plate splines $f(r) = r^2 \ln r$
  - Truncated gaussians $f(r) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{r^2}{2\sigma^2}}$
  - Polyharmonic splines $f(r) = r^\beta$

But Also...

- Nearest Neighbor interpolation

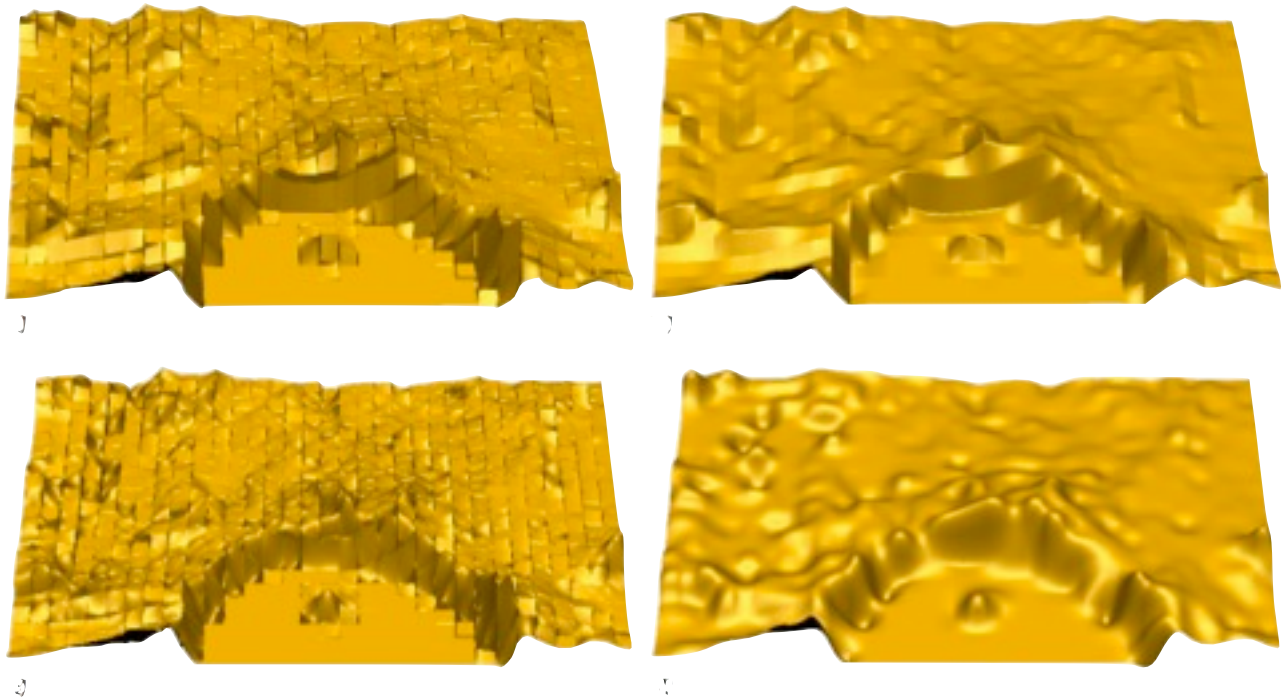




Voronoi diagram

But Also...

- Higher-order interpolation schemes
  - splines, local polynomial fit (interpolation, least sq., ...)
  - smooth reconstruction kernels (on uniform grids)



Outline

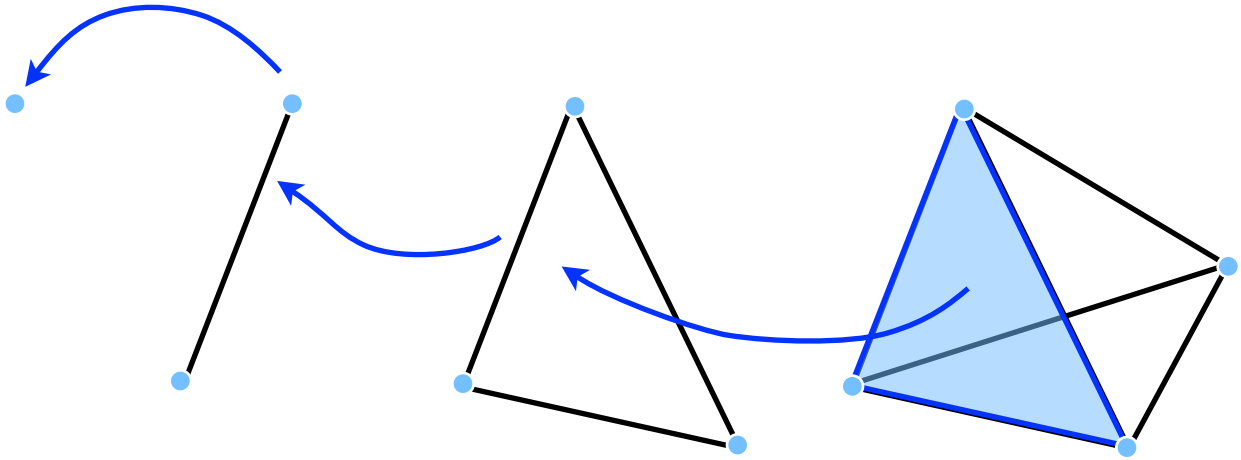
- Grid Structure
- Interpolation
- Search

Context

- Queries in large, unstructured grids
  - arbitrary probing and interpolation
  - resampling (e.g. on regular grid)
- Goal: speed up computation
- Expensive: avoid it when you can
  - iteration (loop over grid vertices)
  - use neighborhood information: leverage correlation between consecutive queries (e.g. streamline, isosurface)

Neighborhood Data

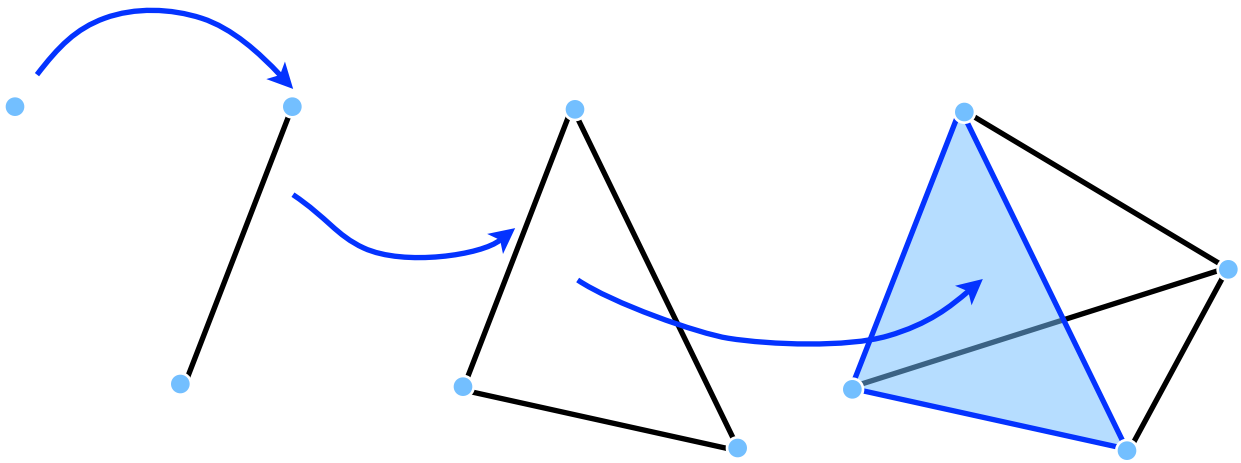
- Data structures store top-down relationships (cell to vertex)



- No direct support for neighborhood information (vertex to cell, cell to cell)

Neighborhood Data

- Bottom-up relationships obtained by computation



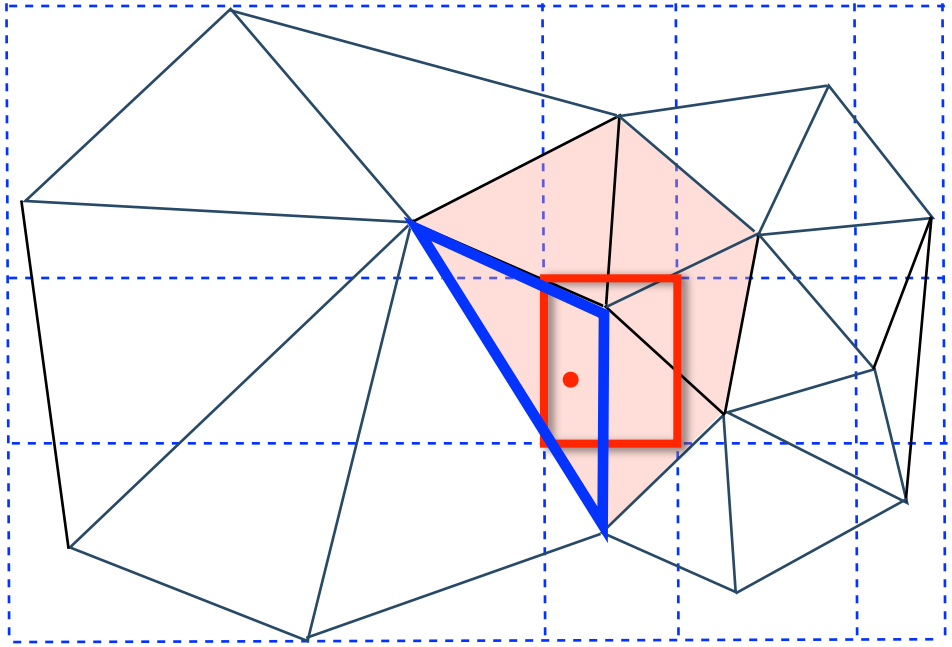
- Cell to cell relationship:
  cell → vertices → cells

Top-down

Bottom-up

Regular Space Subdivision

- Overlay uniform/rectilinear grid over unstructured domain
  - store in each bucket ID's of intersection cells (bbox test)

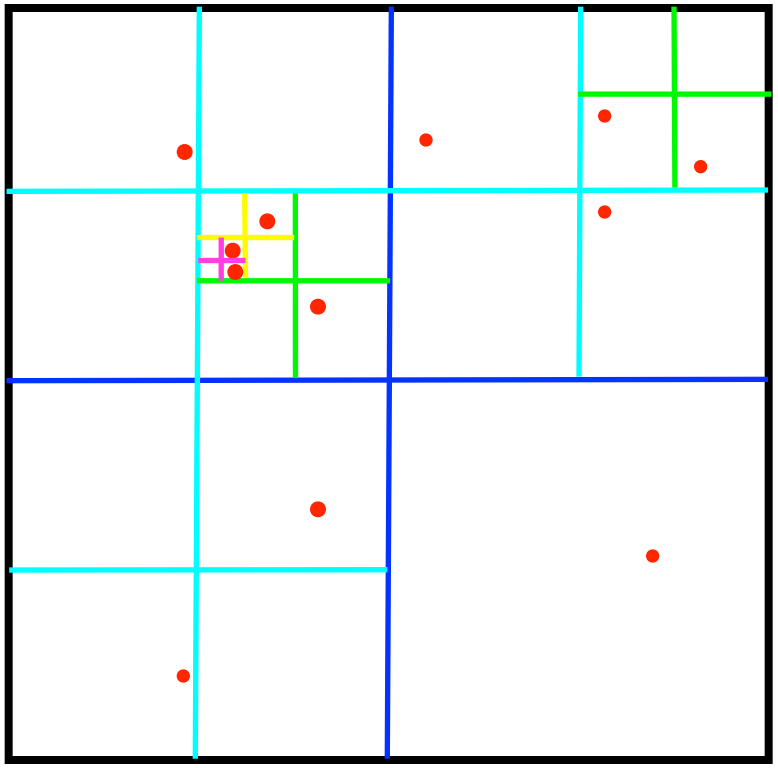


Regular Space Subdivision

- Easy to implement
- Satisfactory if cell distribution is fairly uniform
- Extremely slow for highly unstructured meshes
  - many cells per bucket
  - many buckets per cell

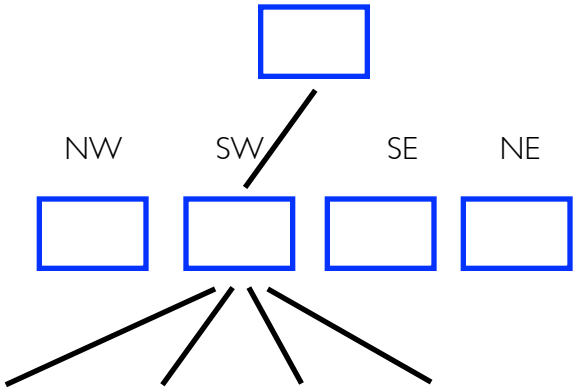
Octree

- Octrees (resp. quadtrees) regularly subdivide space along each coordinate axis at each level



Octree

- Octrees (resp. quadtrees) regularly subdivide space along each coordinate axis at each level



tree structure

Octree

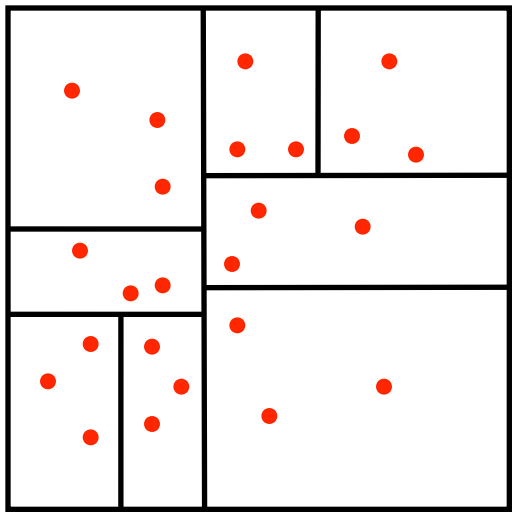
- Widely used in practice
- Easy to implement (if you know how to implement a tree...)
- Allows for local refinement
- Inefficient when cells are skinny
  - large depth value required
  - too many cells per bucket
  - many empty buckets

k-D Tree

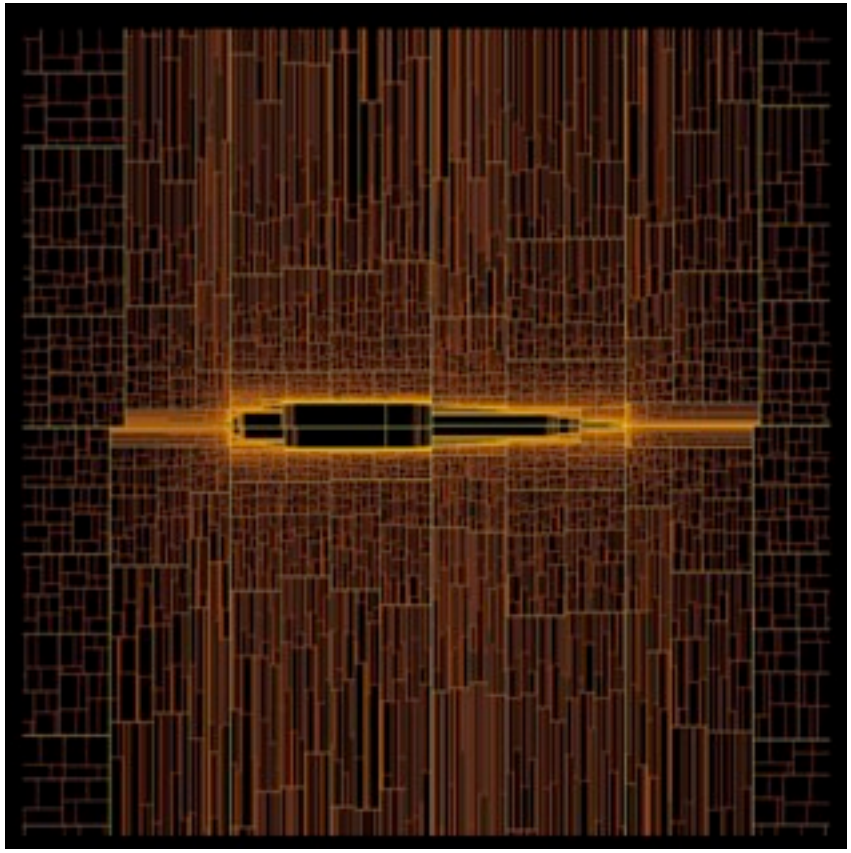
- Split spatial domain along successive axis-parallel hyperplanes at data point
  - Equal # of points in each half space
  - balanced tree (optimize overall height)

k-D Tree

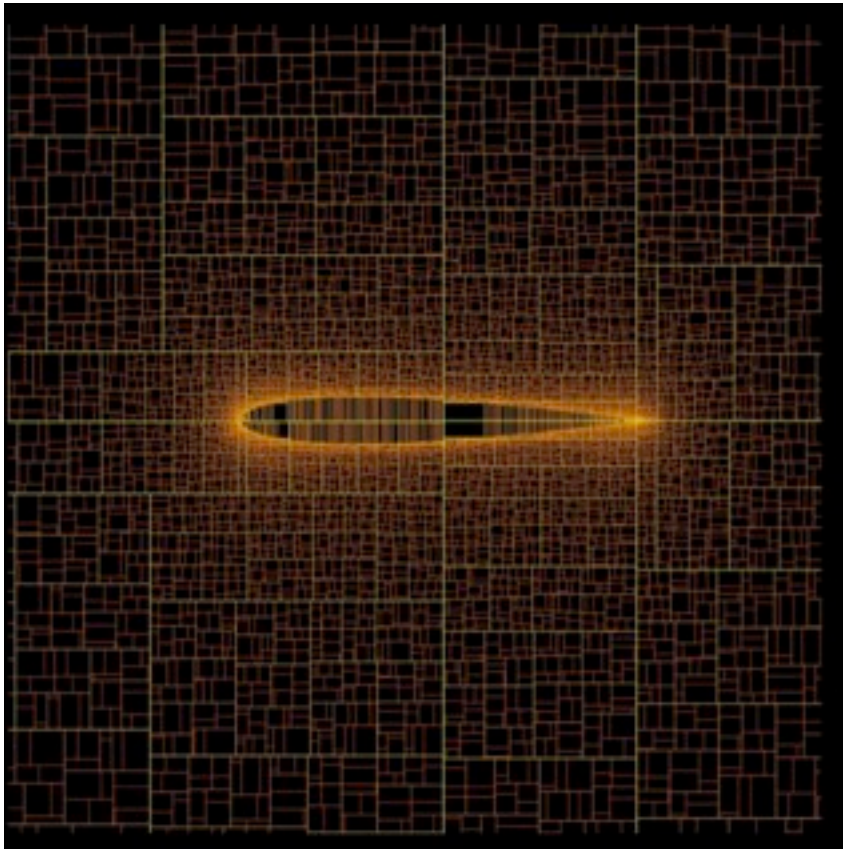
- Split spatial domain along successive axis-parallel hyperplanes at data point
  - Equal # of points in each half space
  - balanced tree (optimize overall height)



k-D Tree



Alternating split



Adaptive split