EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Interpolation & Filtering

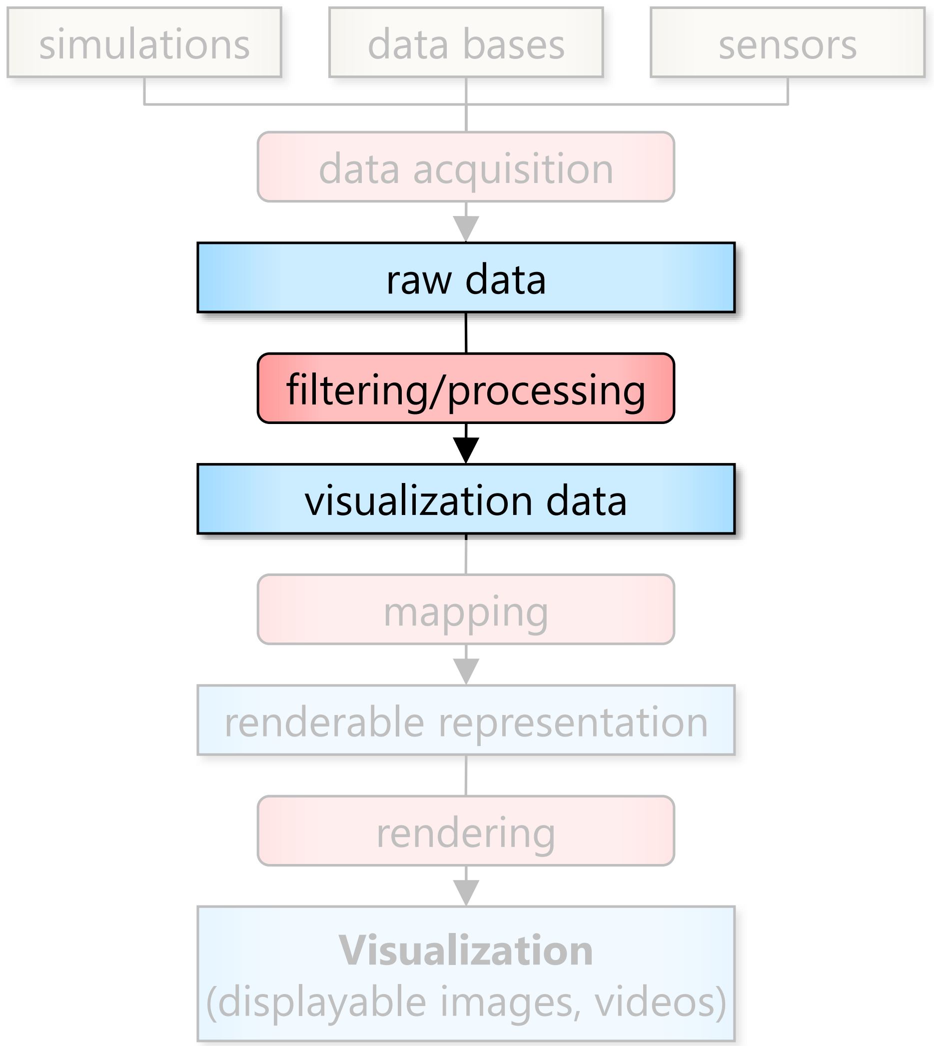
Scientific Visualization – Summer Semester 2021

Jun.-Prof. Dr. **Michael Krone**

Contents

- Voronoi diagrams, Delaunay triangulation
- Univariate interpolation
- Differentiation on grids
- Interpolation on grids
- Interpolation without grids
- Filtering by projection or selection

Focus:
Second step of visualization pipeline



Motivation

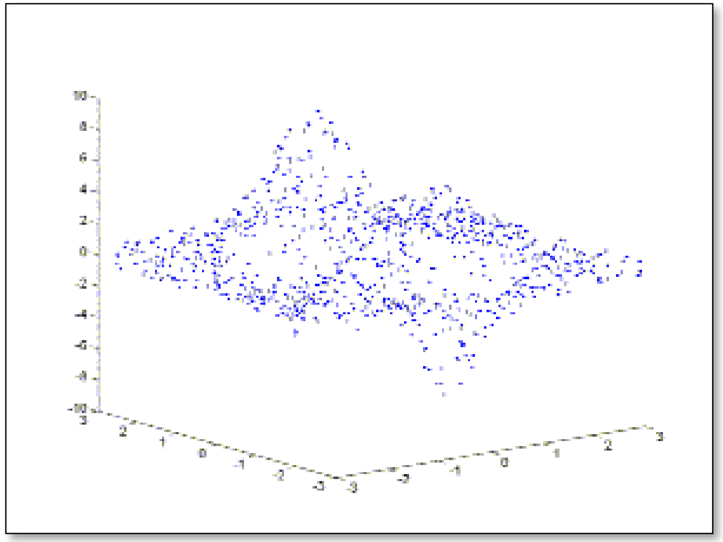
- Data is often **discretized** in space and/or time
- Finite number of **samples**
    - The continuous signal is usually known only at a few points (data points)
    - In general, data is needed in between these points
- By **interpolation** we obtain a representation that matches the function at the data points
    - Evaluation at any other possible point

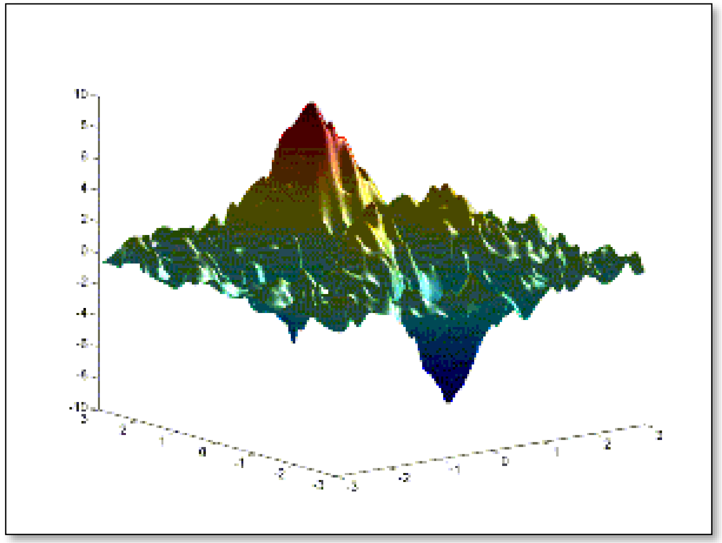
→ **Reconstruction of signal** at points that are not sampled

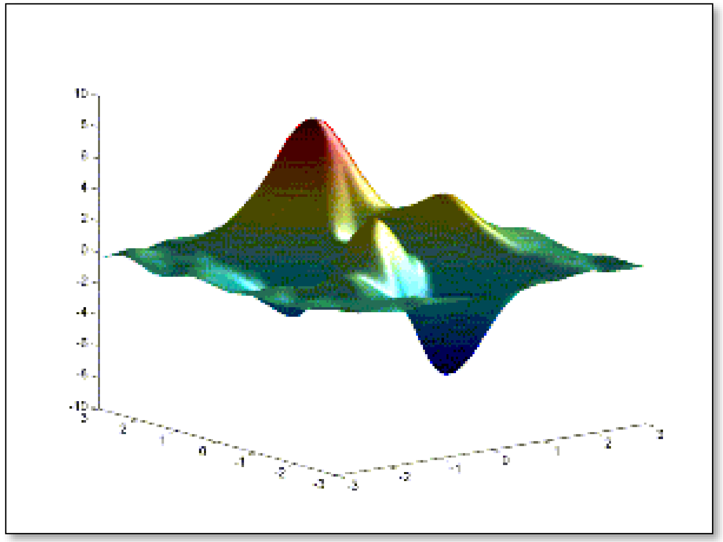
→ Assumptions needed for reconstruction

- Often smooth functions

Motivation



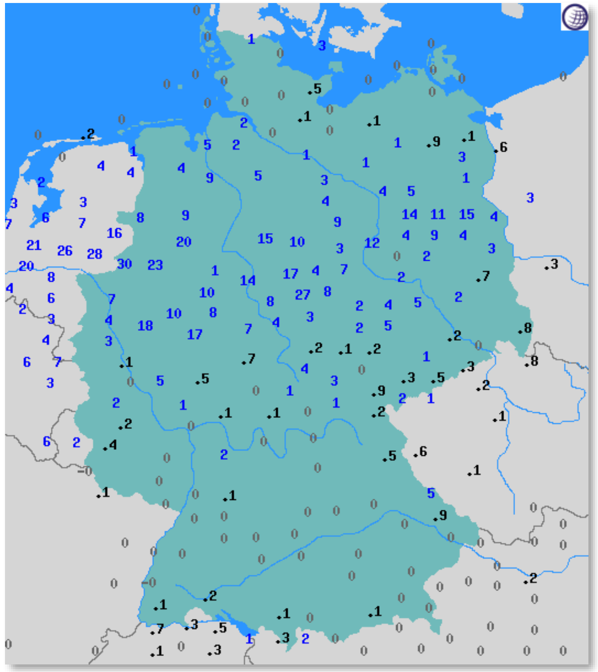


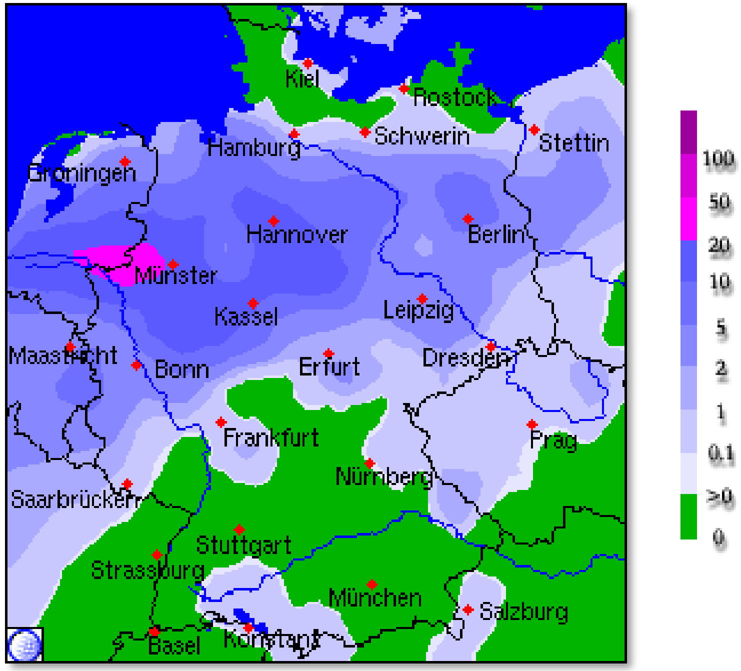


Motivation

- **Example:** precipitation of 6 hours

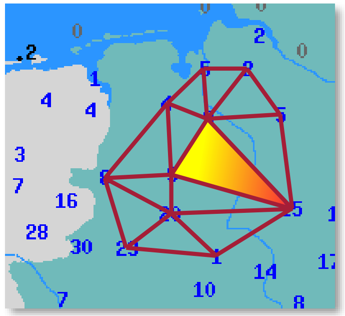
Aachen (205 m)	6.5
Ahaus (46 m)	30.0
Angermünde (55 m)	1.0
Artern (166 m)	2.5
Aue (397 m)	0.3
Augsburg/Mühlhausen (477 m)	0.0
Bad Hersfeld (273 m)	0.7
[illegible]ssingen (266 m)	0.1
[illegible] (158 m)	10.0
Wittenberg (106 [illegible]	[illegible]
Würzburg (272 m)	[illegible]
Zinnwald (882 m)	0.3
Zugspitze (2962 m)	1.9
Zwiesel (613 m)	0.9
Öhringen (277 m)	0.1





(Source: www.wetteronline.de, 07.05.2007 20:00)

Voronoi Diagrams and Delaunay Triangulation

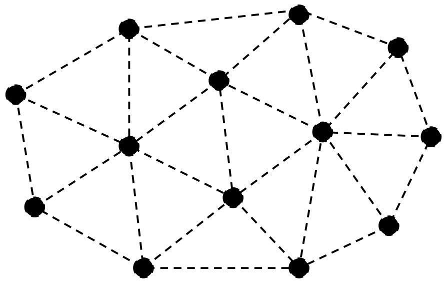


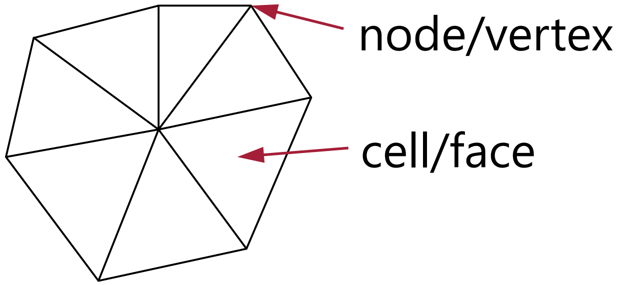
- **Given:** irregularly distributed positions without connectivity information
- **Problem:** obtain connectivity to find a "good" triangulation
  - Interpolate within resulting triangles (see later)

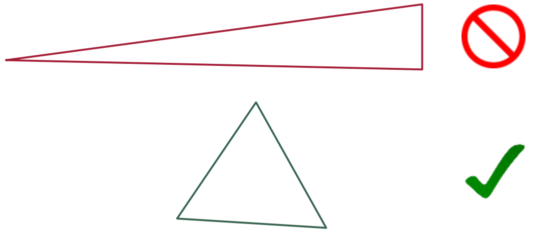
→ For a set of points there are many possible triangulations

- A measure for the quality of a triangulation is the aspect ratio of its triangles

→ *Avoid long, thin triangles!*

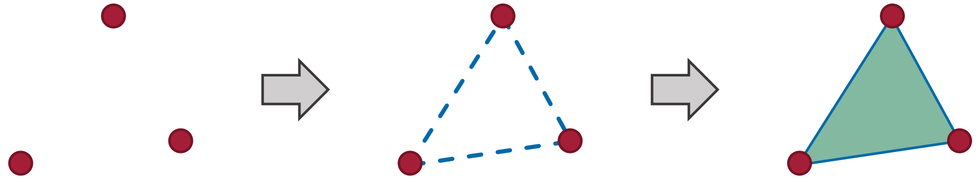






Voronoi Diagrams and Delaunay Triangulation

- Scattered data triangulation
  - A triangulation of data points $S = s_0, s_1, \dots, s_m \in \mathbb{R}^2$ consists of
    - **Vertices** (0D) = S
    - **Edges** (1D) connecting two vertices
    - **Faces/cells** (2D) connecting three vertices

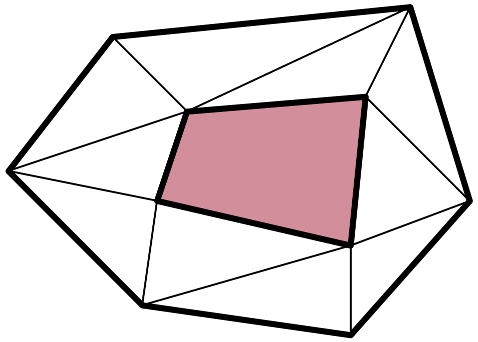


- A triangulation of scattered data must satisfy the following requirements:
  - $\bigcup \text{faces} = conv(S)$, i.e. the union of all faces including the boundary is the convex hull of all vertices
  - The intersection of two triangles is either empty, or a common vertex, or a common edge, or a common face (tetrahedra)
  - Partitioning of the space

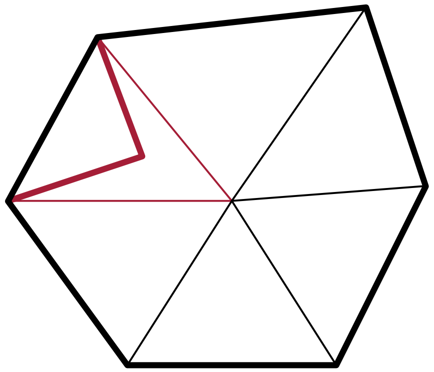
Voronoi Diagrams and Delaunay Triangulation

- Triangulations with

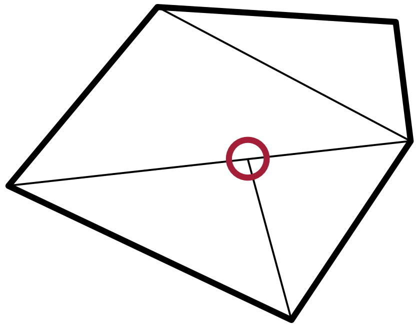
holes,



overlapping faces,



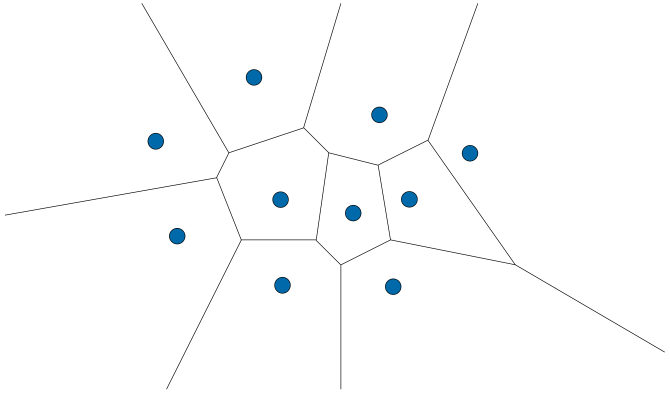
T-nodes



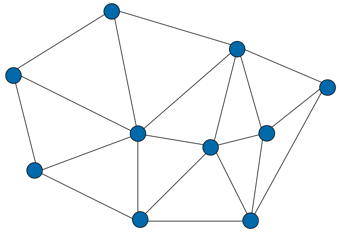
are **not** valid!

Voronoi Diagrams and Delaunay Triangulation

- How to get connectivity/triangulation from scattered data ?
  - Voronoi diagram
  - Delaunay triangulation



Voronoi diagram

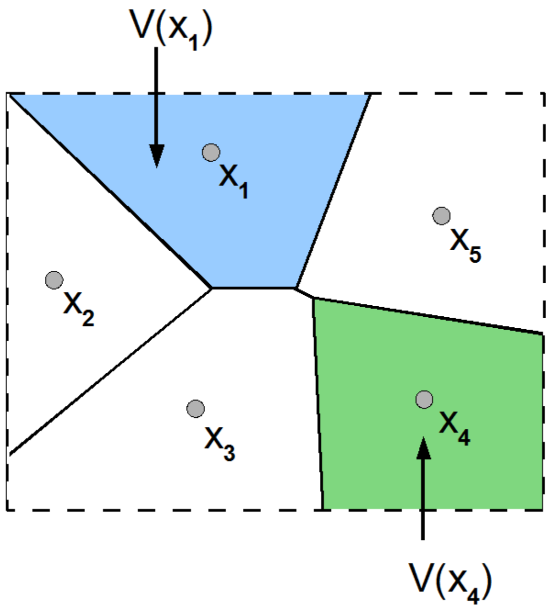


Delaunay triangulation

Voronoi Diagrams and Delaunay Triangulation

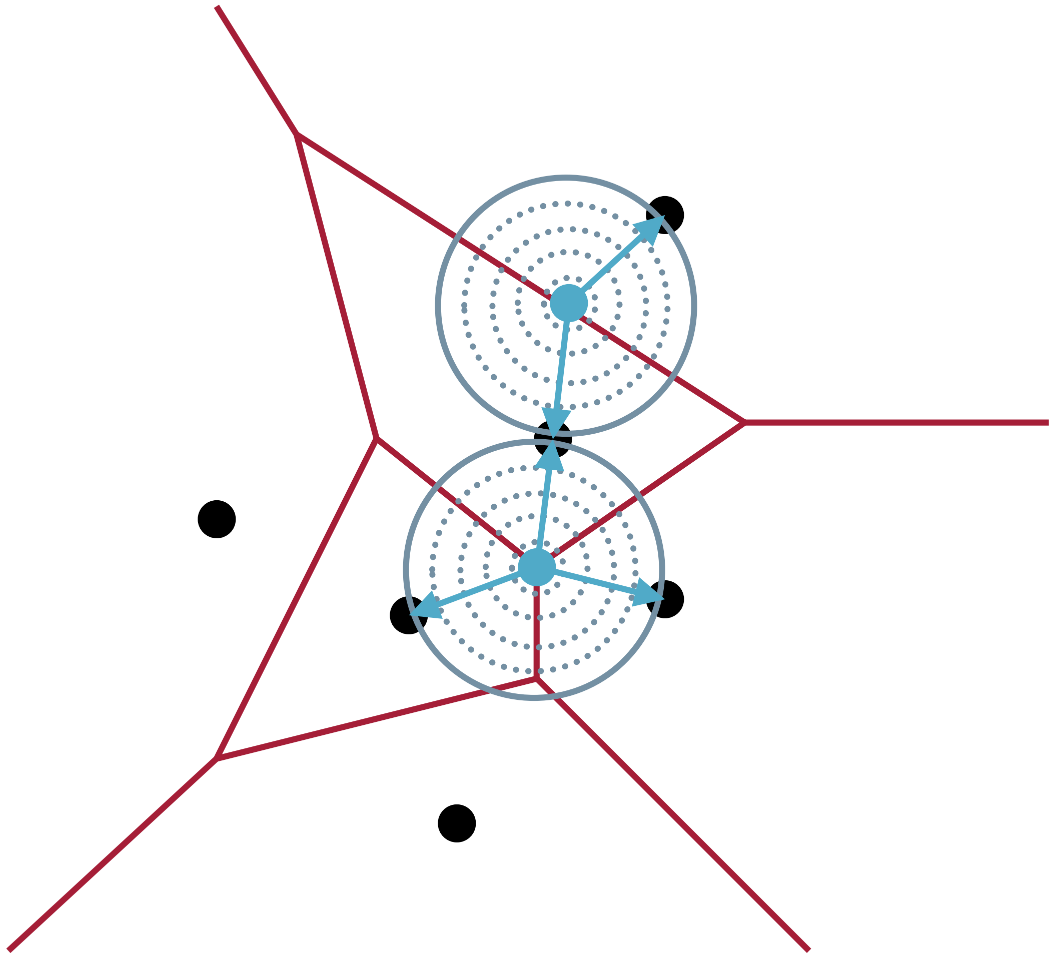
- Voronoi diagram
  - For each sample, every point within a Voronoi region is closer to it than to every other sample
  - **Given:** a set of points $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$ a distance function $dist(x, y)$
  - The Voronoi diagram $Vor(X)$ contains for each point x_i a cell $V(x_i)$ with

$$V(x_i) = \{\, x \mid dist(x, x_i) \; < \; dist(x, x_j) \; \forall j \neq i \,\}$$



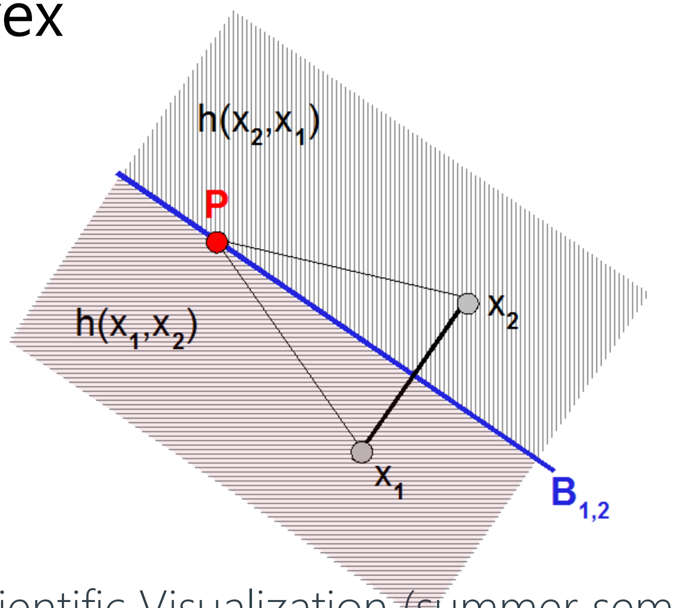
Voronoi Diagrams and Delaunay Triangulation

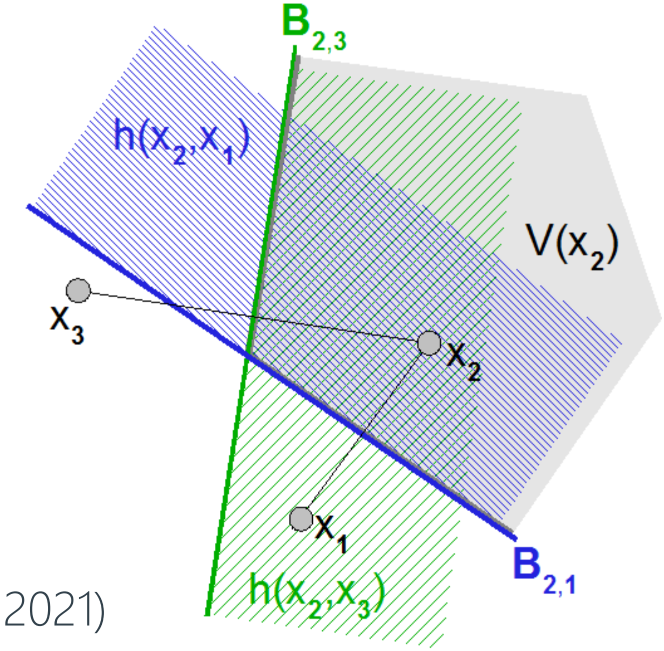
- Voronoi diagram
  - Cases where a point has equal distance to >3 samples are degenerate
  - Degenerate cases can be avoided by adding a perturbation, i.e., randomly moving the samples by $\epsilon \ll 1$



Voronoi Diagrams and Delaunay Triangulation

- Voronoi cells
  - The half space $h(x_i, x_j)$ is separated by the perpendicular bisector $B_{i,j}$ between the points x_i and x_j
  - $h(x_i, x_j)$ contains x_i
  - Voronoi cell = intersection of all half spaces: $V(x_i) = \bigcap_{j \neq i} h(x_i, x_j)$
  - Voronoi cells are convex

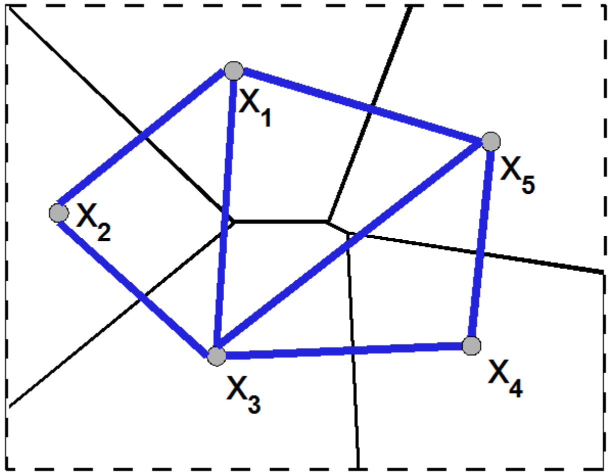




Voronoi Diagrams and Delaunay Triangulation

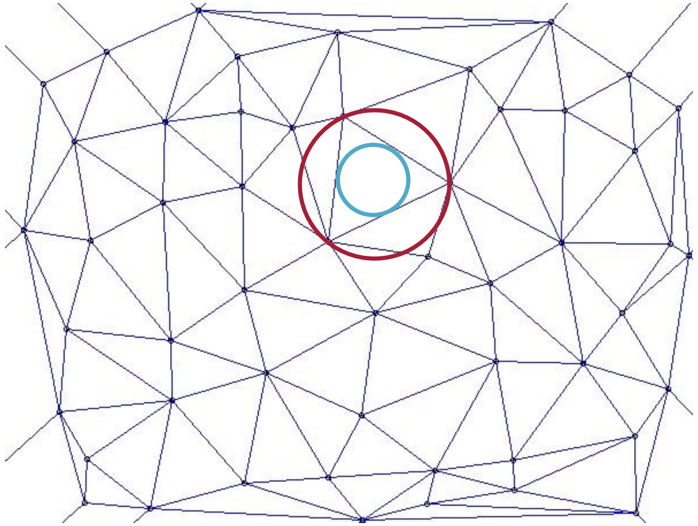
- Delaunay graph $Del(X)$
  - The geometric dual (topologically equal) of the Voronoi diagram $Vor(X)$
  - Points in X are nodes
  - Two nodes x_i and x_j are connected if the Voronoi cells $V(x_i)$ and $V(x_j)$ share an edge

- Delaunay cells are convex
- Delaunay triangulation = triangulation of the Delaunay graph



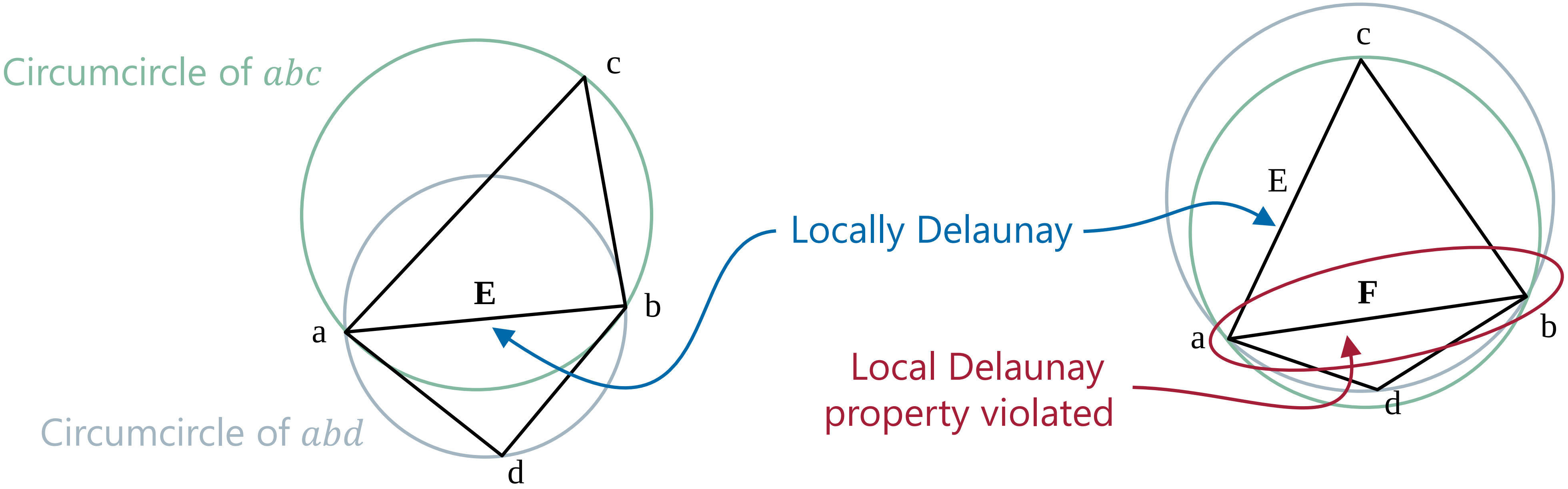
Voronoi Diagrams and Delaunay Triangulation

- Delaunay triangulation in 2D
  - Three points $x_i, x_j, x_k \in X$ belong to a face from $Del(X)$ if no further point lies inside the circle around x_i, x_j, x_k
    - For each triangle, the circumcircle does not contain any other sample
  - Two points x_i, x_j form an edge if there is a circle around x_i, x_j that does not contain a third point from X (*global Delaunay property*)
  - Maximizes the minimum angle of the triangulation
  - Maximizes the ratio of $\frac{(radius\ of\ incircle)}{(radius\ of\ circumcircle)}$
  - It is unique (independent of the order of samples) for all but some very specific cases



Voronoi Diagrams and Delaunay Triangulation

- Local Delaunay property:
- An edge between two points a and b is locally Delaunay
  - if it belongs to only one triangle (edge of the convex boundary) **–or–**
  - if it belongs to two triangles abc and abd and d lies outside the circumcircle of abc (and c lies outside the circumcircle of abd)



Voronoi Diagrams and Delaunay Triangulation

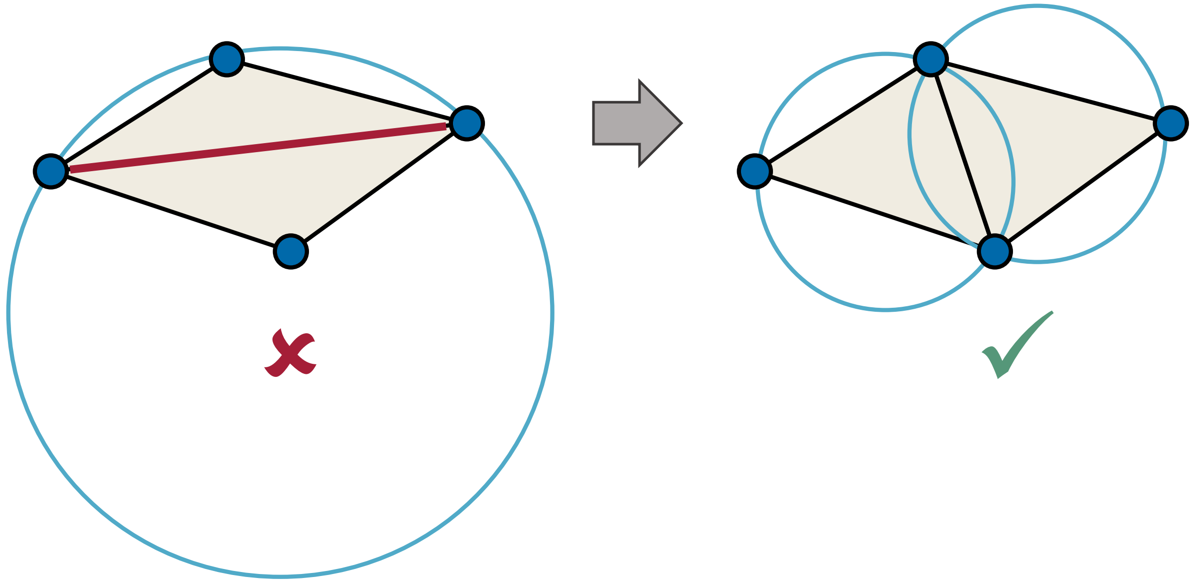
- Algorithms for Delaunay triangulations
  - Edge flip algorithm:

Remember: no holes, no overlapping faces, no T-nodes

```
find an initial (valid) triangulation
find all edges where local Delaunay property is violated
push these edges onto the stack
while (stack not empty) {
    pop edge from stack
    flip this edge
    push all adjacent edges for which the Delaunay property is violated due to flip
}
```

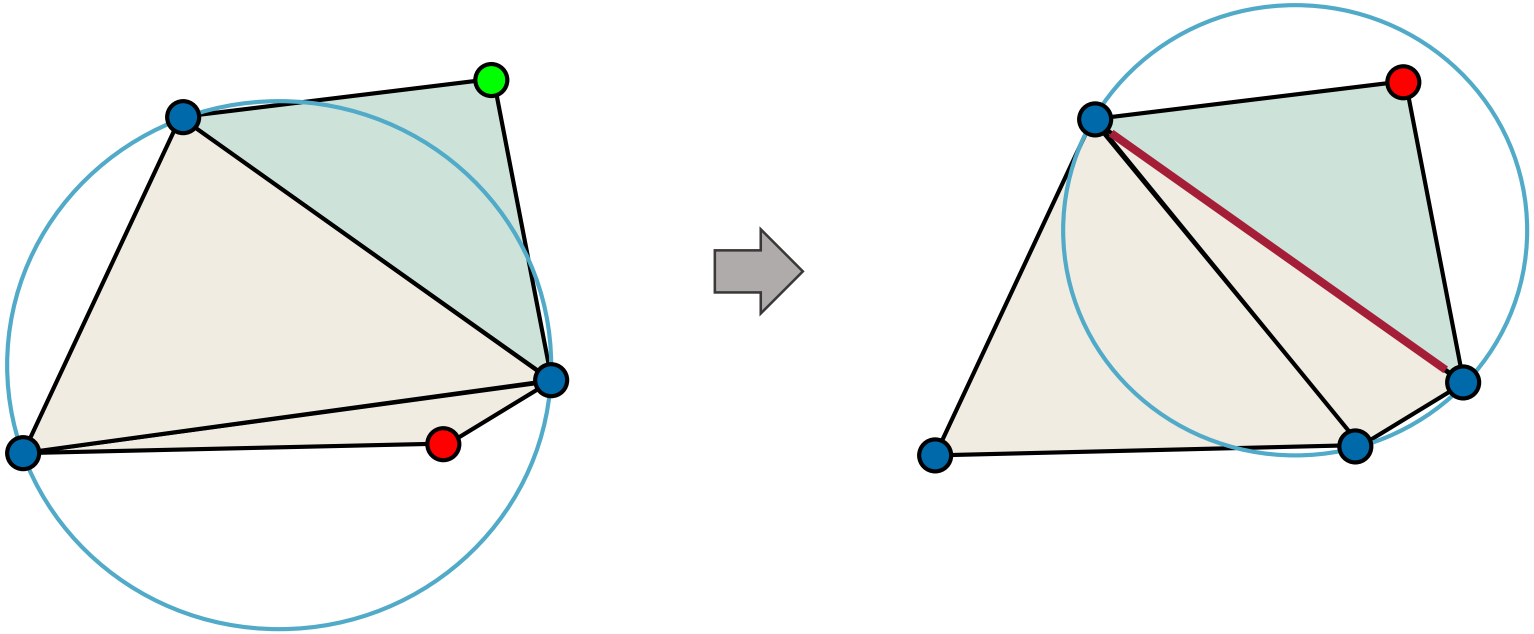
Voronoi Diagrams and Delaunay Triangulation

- Edge flip algorithm
  - For arbitrary (valid) triangulation:
    fix all parts that do not meet the local Delaunay property



Voronoi Diagrams and Delaunay Triangulation

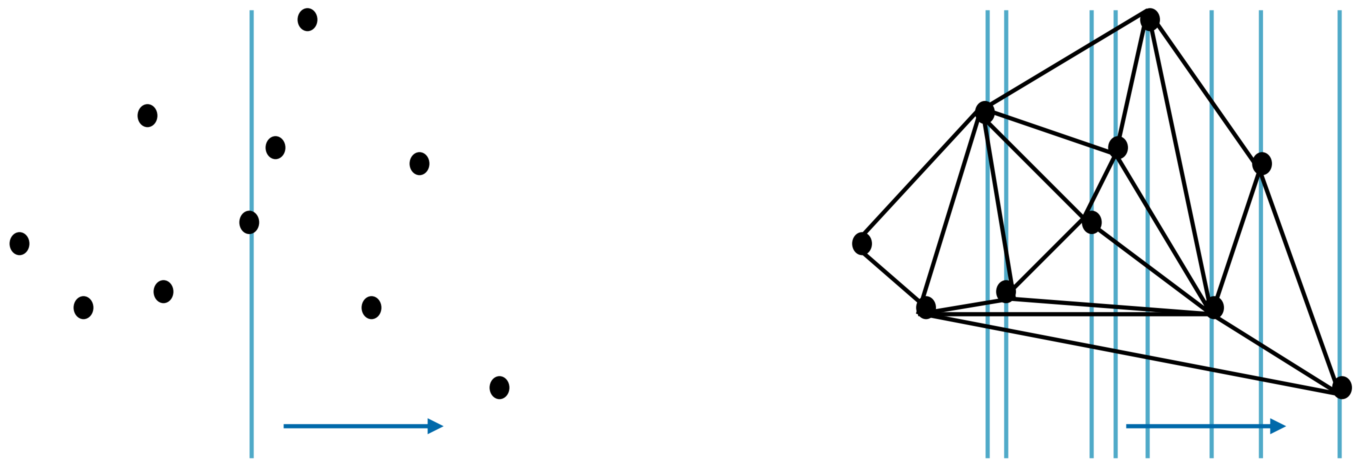
- Possible side effects when flipping one edge



→ `push all adjacent edges for which the Delaunay property is violated due to flip`

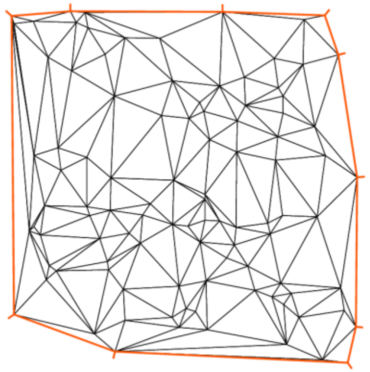
Voronoi Diagrams and Delaunay Triangulation

- Plane-sweep algorithm for finding an initial triangulation
  - Imaginary vertical sweepline that passes from left to right
  - As the sweepline moves:
    - Problem has been solved for the data to the left of the sweepline
    - Is currently being solved for the data at or near the sweepline and
    - Is going to be solved sometime later for the data to right of the sweepline
  - Reduces a problem in 2D space to a series of problems in 1D space



Voronoi Diagrams and Delaunay Triangulation

- Plane-sweep algorithm for finding an initial triangulation

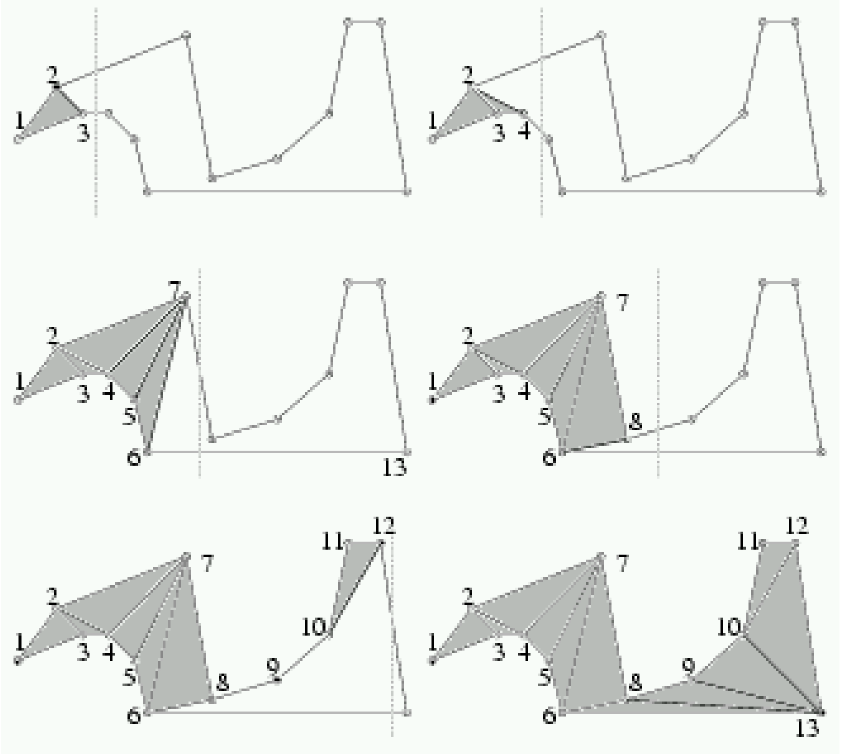


```
sort points from left to right;
construct initial triangle using first three vertices (i=1...3);
for (i=4...n) {
   use last inserted point p_{i-1} as starting point;
   walk counterclockwise along convex polygon (hull) of triangulation until the
          tangent points, inserting edges between p_i and polygon points;
   walk clockwise along convex polygon of triangulation until the tangent points,
          inserting edges between p_i and polygon points;
   update convex hull;
}
```

→ Connect the new point p_i to all other points of the convex hull if the new edge does not intersect the hull

Voronoi Diagrams and Delaunay Triangulation

- Plane-sweep algorithm
  - Also for triangulating polygons



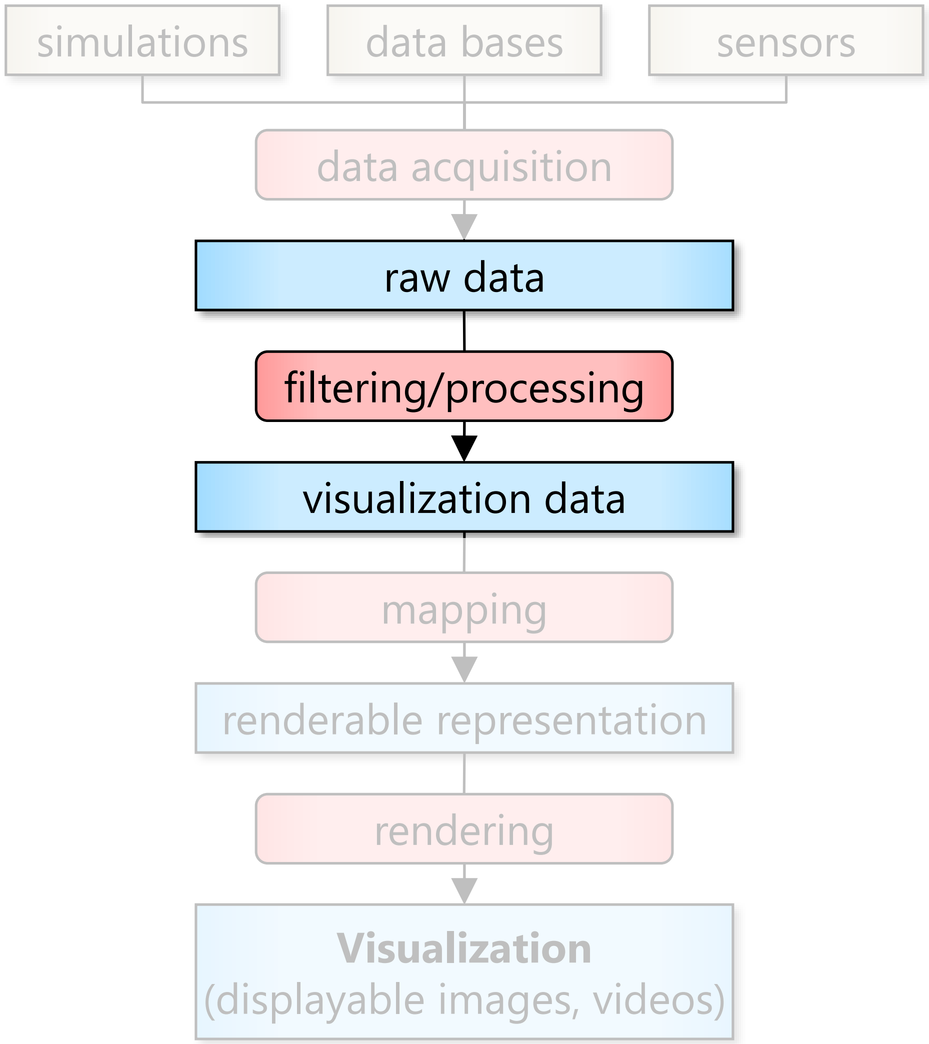
- Other techniques exist, e.g.
  - Radial sweep
  - Intersecting half spaces
  - Divide and conquer (merge-based or split-based)

  →*More efficient than edge flip and plane-sweep*

Contents

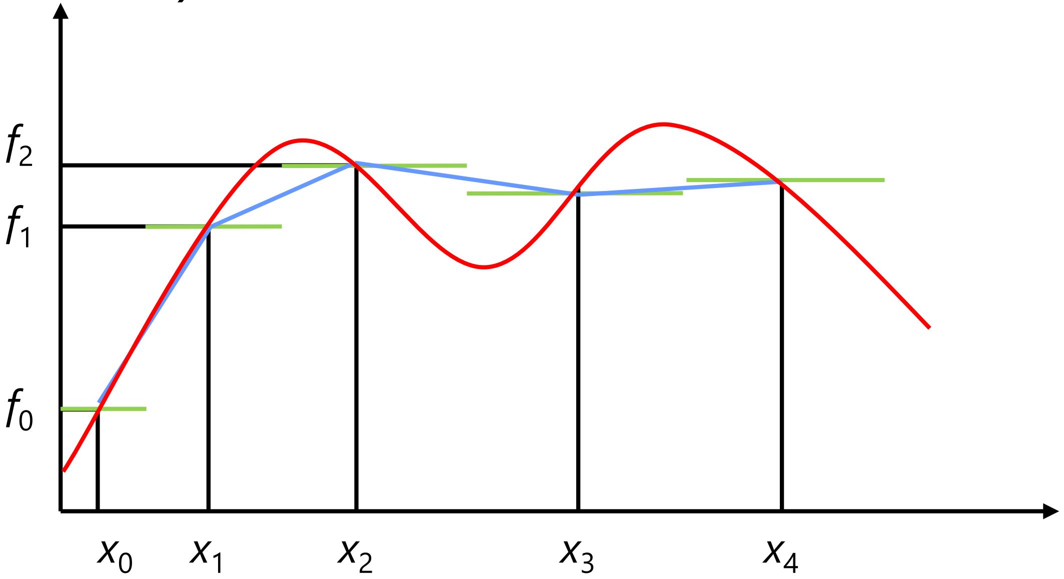
- Voronoi diagrams, Delaunay triangulation
- **Univariate interpolation**
- Differentiation on grids
- Interpolation on grids
- Interpolation without grids
- Filtering by projection or selection

Focus:
Second step of visualization pipeline



Univariate Interpolation

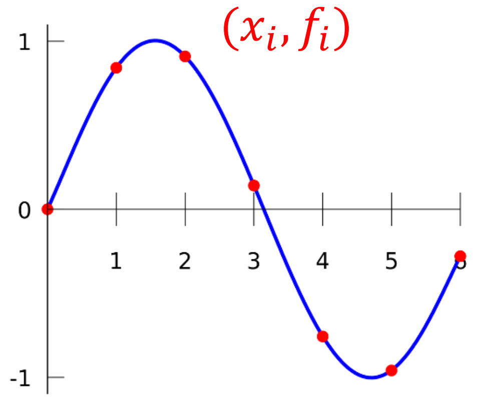
- Univariate interpolation: interpolation for one variable
  - Nearest neighbor (0-order)
  - Linear (first-order)
  - Smooth (higher-order)



Univariate Interpolation

- Taylor interpolation
- Basis functions: Taylor (monom) basis (polynomials) $m_i = x^i$ with $i \in \mathbb{N}_0$
- $P_m = \{1, x, x^2, \dots, x^m\}$ is $(m+1)$-dimensional vector space of all polynomials with maximum degree m
- Coefficients c_i with $f = \Sigma_i \, c_i \cdot x^i$
- Representation of samples:

$$f(x_j) = f_j \quad \forall j = 0 \dots n-1$$



- Interpolation problem

$$\boldsymbol{V} \cdot \boldsymbol{c} = \boldsymbol{f}$$

samples

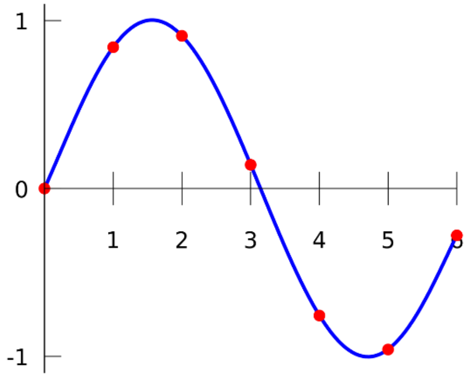
coefficients (to be solved)

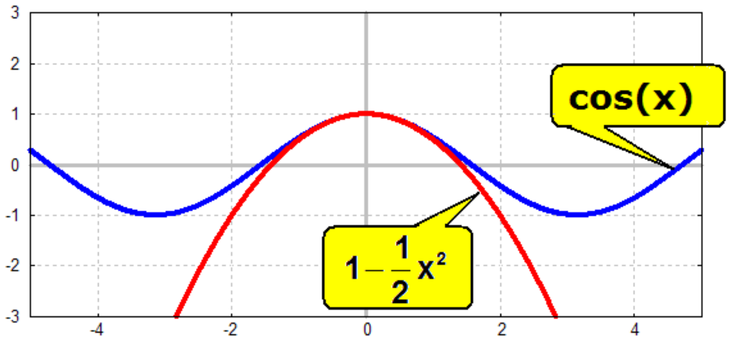
with the Vandermonde matrix $V_{ji} = x_j^i$

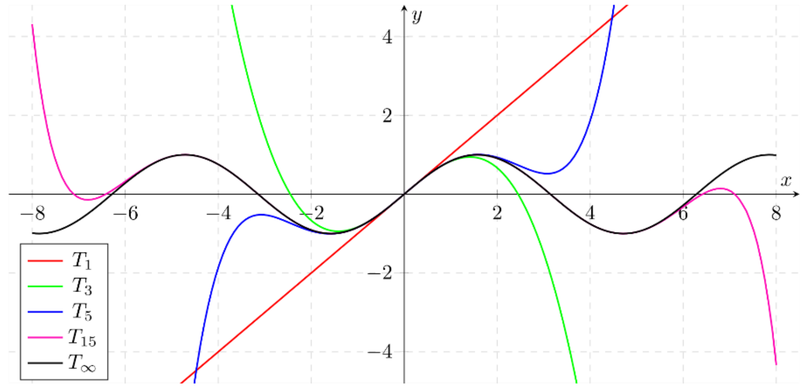
$$V(x_1, x_2, \dots, x_n) = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix}$$

Univariate Interpolation

- Properties of Taylor interpolation
  - Unique solution
  - Numerical problems / inaccuracies
  - Complete system has to be solved again if a single value is changed
  - Not intuitive





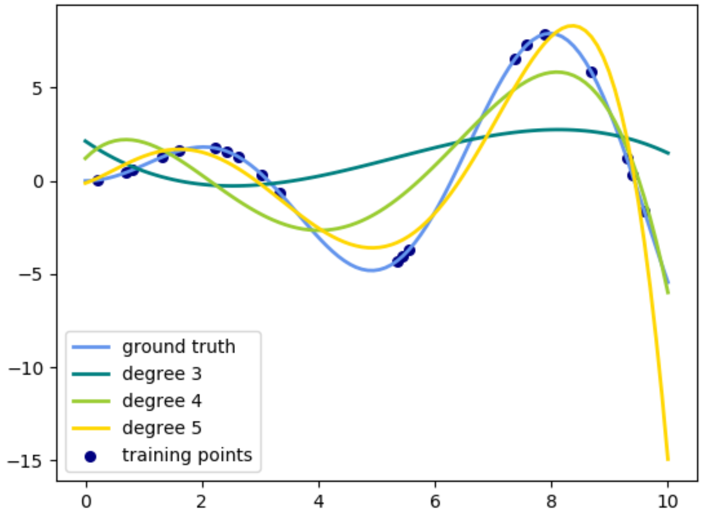


Univariate Interpolation

- Generic interpolation problem:
  - Given are n sample points $X = \{x_i\} \subseteq \Omega \subseteq \mathbb{R}$ with function values f_i
  - n-dimensional function space $\Phi_n(\Omega)$ with basis $\{\phi_{i=1\ldots n}\}$
  - Coefficients c_i with $f = \Sigma_i c_i \cdot \phi_i$
  - Representation of samples: $f(x_j) = f_j \; \forall j = 1 \ldots n$
  - Solving the linear system of equations

$$\boldsymbol{M} \cdot \boldsymbol{c} = \boldsymbol{f}$$

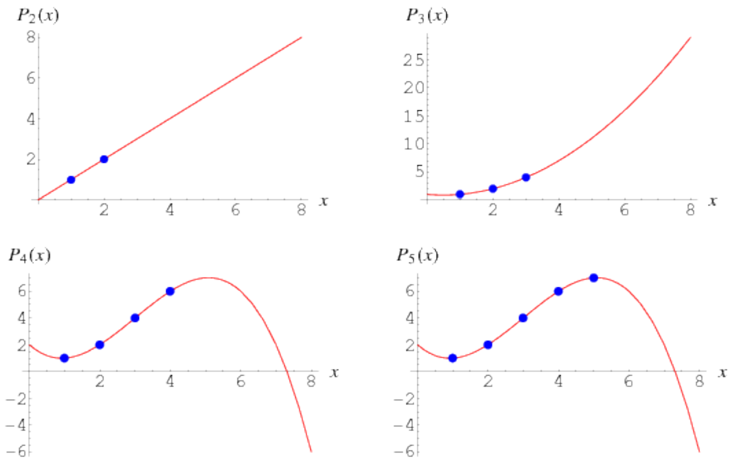
with $\boldsymbol{M_{ji}} = \phi_i(x_j)$, $\boldsymbol{c_i} = c_i$, and $\boldsymbol{f_j} = f_j$



  - ***Note:*** number of points n determines dimension of vector space (= maximum degree of polynomials + 1)

Univariate Interpolation

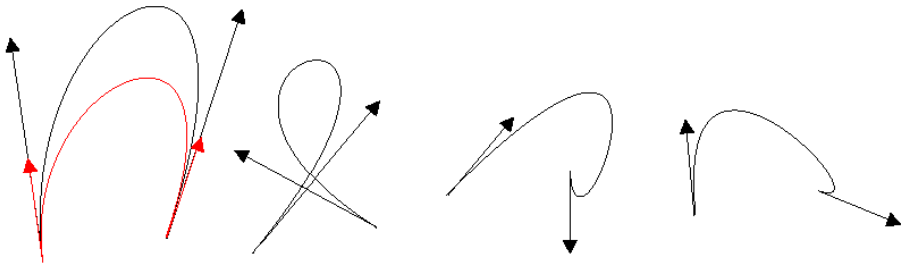
- Other basis functions result in other interpolation schemes
- Polynomial basis functions
  - Taylor basis $\phi_i(x) = x^i$
  - Lagrange basis $\phi_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$
  - Newton basis $\phi_i(x) = \prod_{j=1}^{i} (x - x_j)$
  - Bernstein basis (Bezier) $\phi_i^k(x) = \binom{k}{i} (1-x)^{k-i} x^i$
- Spline basis
  - Bezier splines
  - B-splines $\phi_i^k(x) = \frac{x - t_i}{t_{i+k} - t_i} \phi_i^{k\text{-}1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} \phi_{i+1}^{k\text{-}1}(x)$

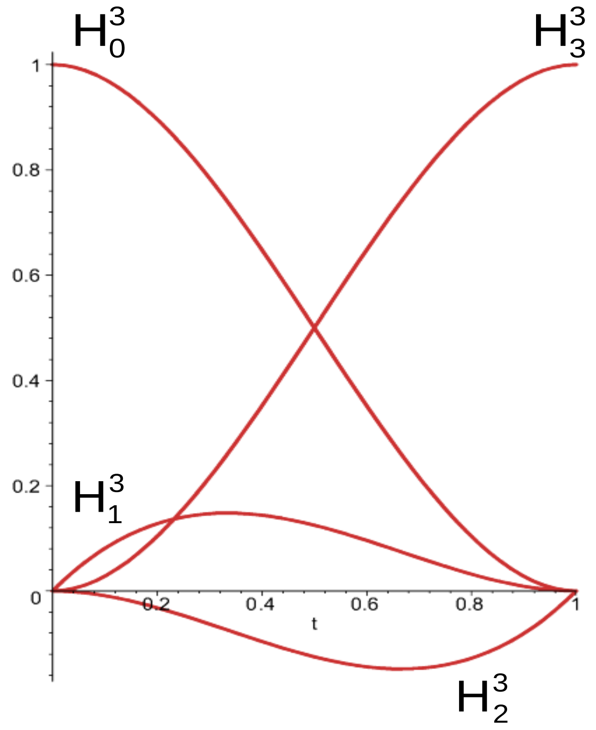


Example: Lagrange

Univariate Interpolation

- Cubic Hermite polynomials **H**
- Linear combination of basis functions
- Coefficients describe:
  - End points $\boldsymbol{p_k}, \boldsymbol{p_{k+1}}$
  - Tangent vectors $\boldsymbol{m_k}, \boldsymbol{m_{k+1}}$ at end points





$$f(x) = H_0^3(t)\boldsymbol{p_k} + H_1^3(t)\boldsymbol{m_k} + H_2^3(t)\boldsymbol{m_{k+1}} + H_3^3(t)\boldsymbol{p_{k+1}}$$

whit $t = (x - x_k)/(x_{k+1} - x_k)$

Basis functions:

$$H_0^3(t) = (1+2t)(1-t)^2$$

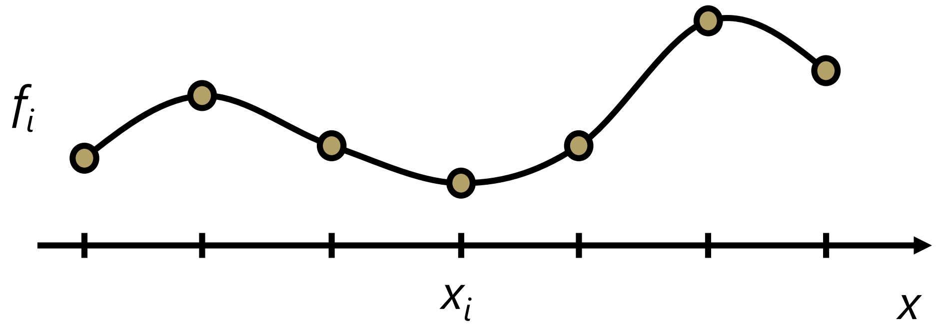
$$H_1^3(t) = t(1-t)^2$$

$$H_2^3(t) = t^2(1-t)$$

$$H_3^3(t) = t^2(3-2t)$$

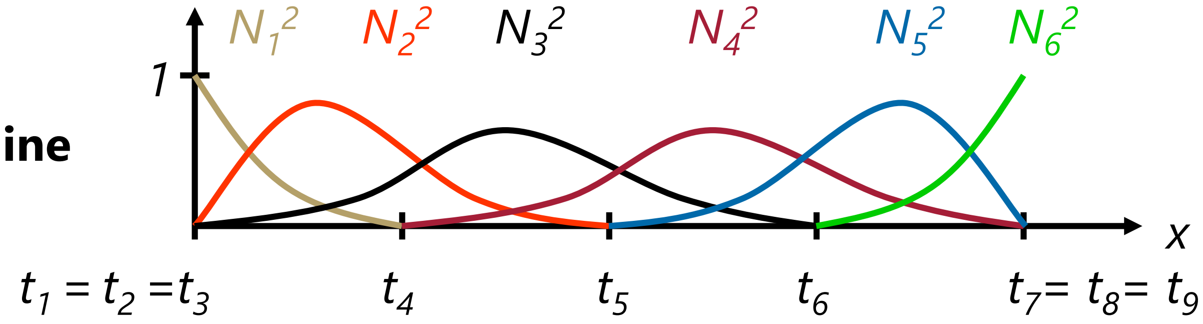
Univariate Interpolation

- **Problem:** coupling of number of samples n and degree of polynomials



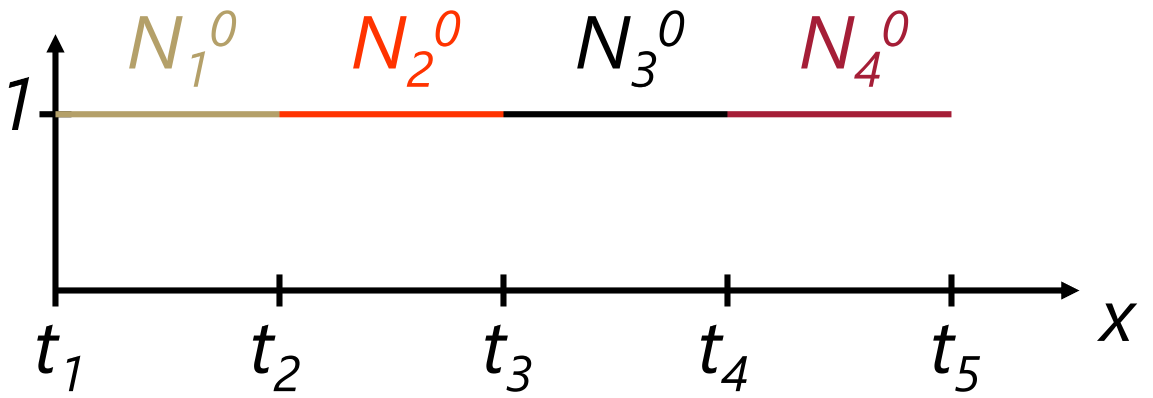
- **Solution:** Spline interpolation
  - Smooth piecewise polynomial function
  - Continuity / smoothness at segment boundaries
  - Avoid oscillation
  - Basis functions N

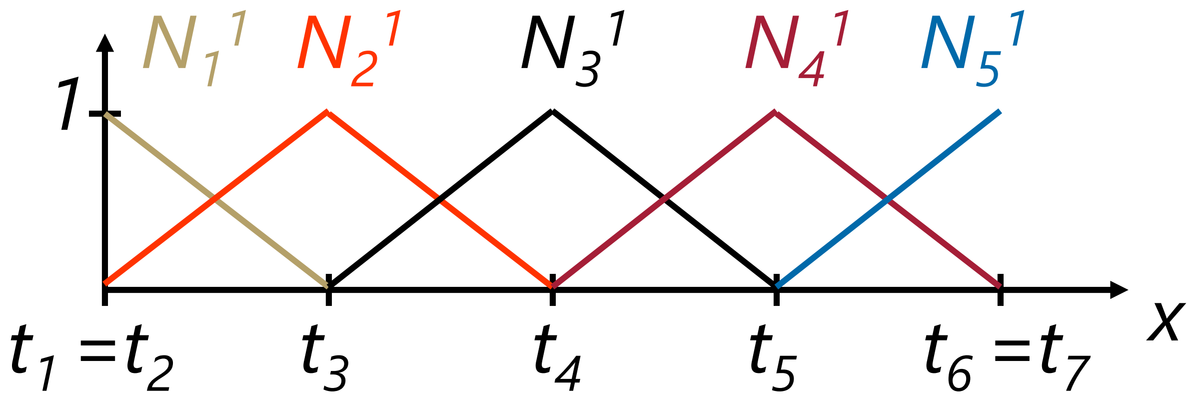
B-Spline



Univariate Interpolation

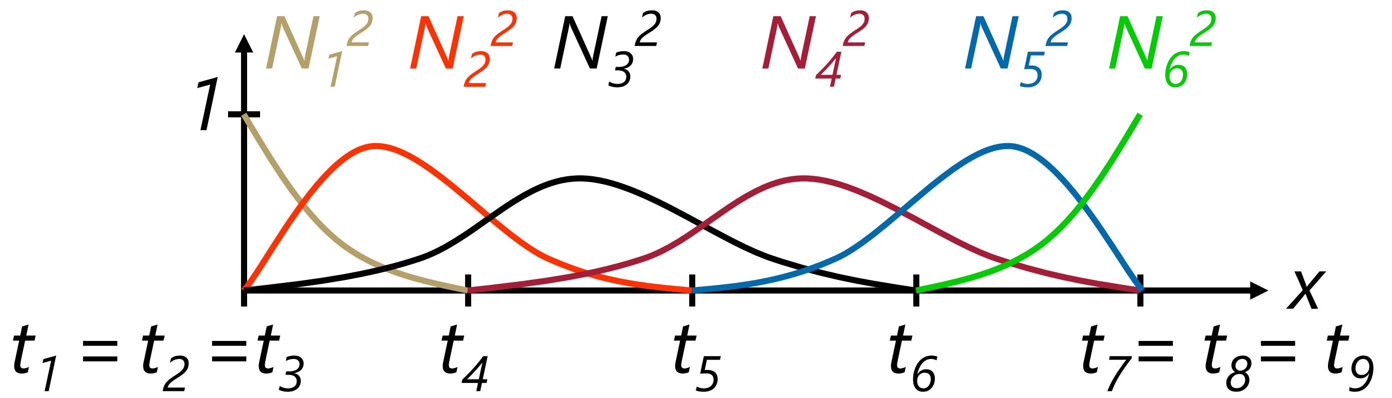
- B-Splines

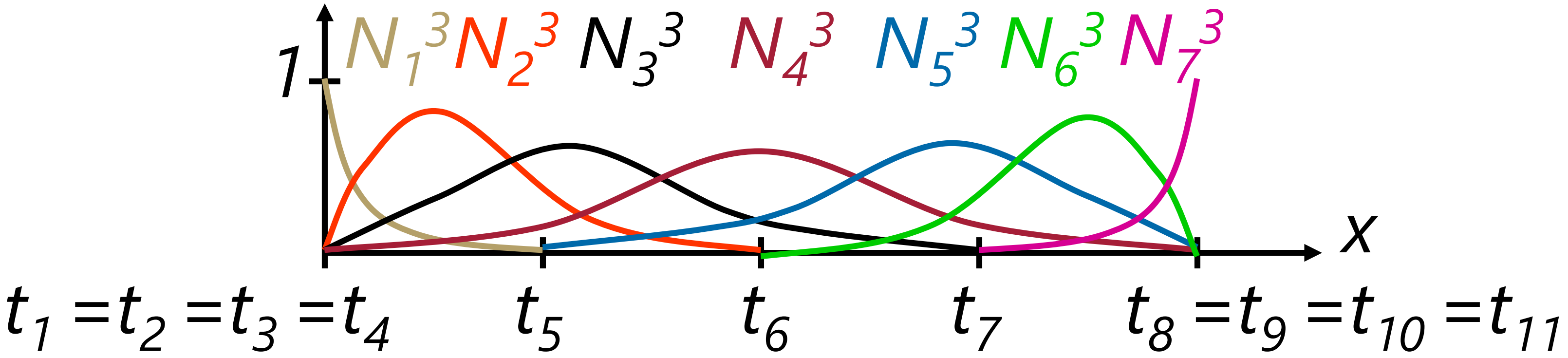




Univariate Interpolation

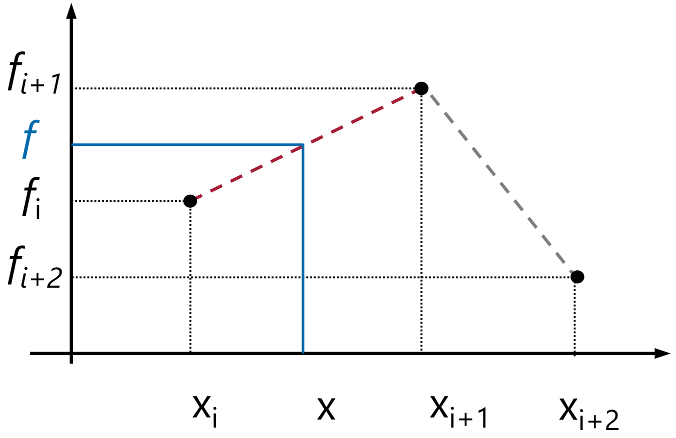
- B-Splines





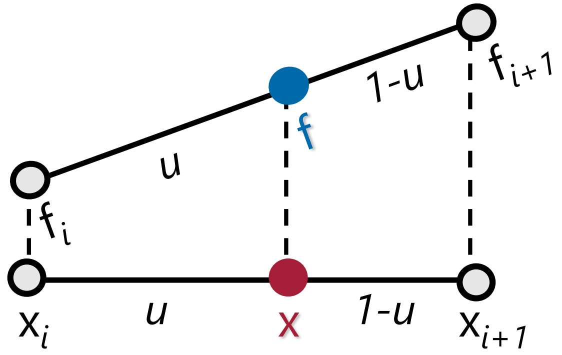
Univariate Interpolation

- Piecewise linear interpolation
  - Simplest approach (except for nearest-neighbor sampling)
  - Fast to compute
  - Often used in visualization applications
  - Only C^0 continuity at cell boundaries
  - Data points: $(x_0, f_0), \ldots, (x_n, f_n)$
  - For any point x with $x_i \leq x \leq x_{i+1}$ described by local coordinate $u = (x - x_i)/(x_{i+1} - x_i) \in [0,1]$



that is $$x = x_i + u(x_{i+1} - x_i) = (1-u)x_i + ux_{i+1}$$

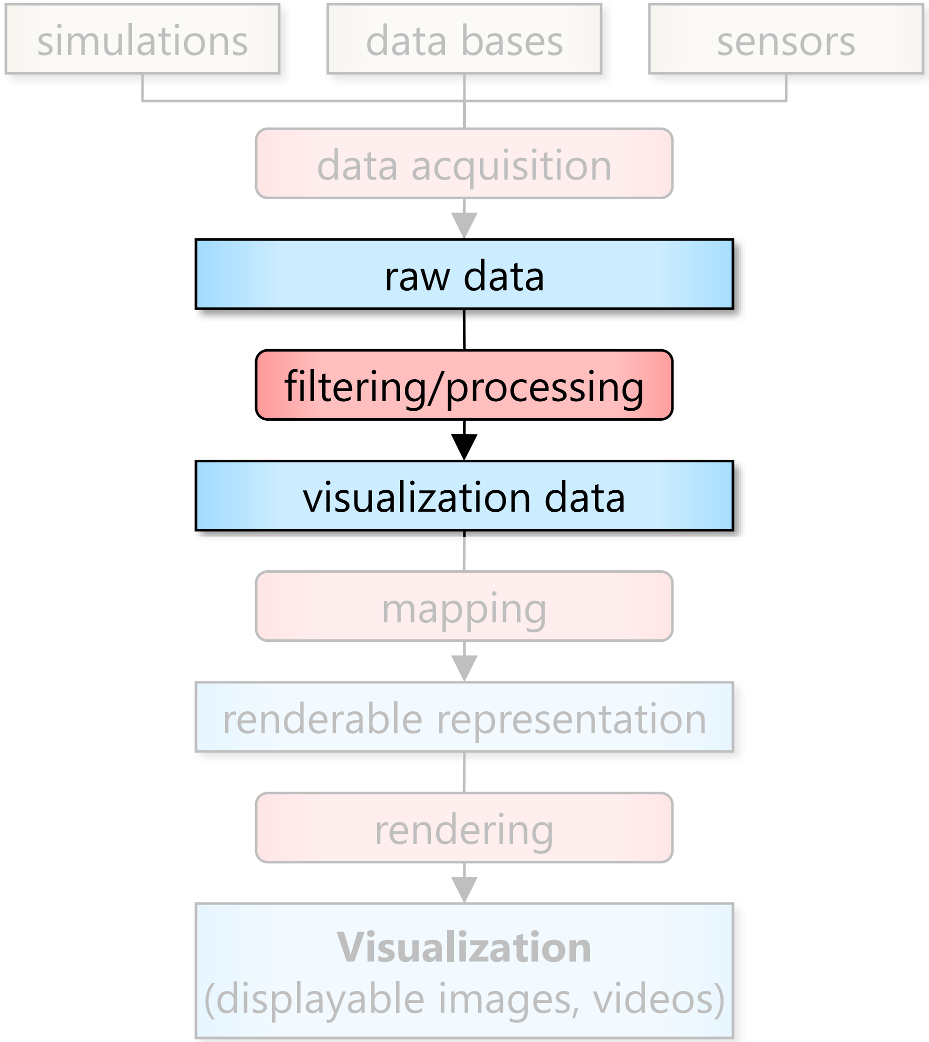
evaluate $$f(x) = (1-u)f_i + uf_{i+1}$$



Contents

- Voronoi diagrams, Delaunay triangulation
- Univariate interpolation
- **Differentiation on grids**
- **Interpolation on grids**
- **Interpolation without grids**
- Filtering by projection or selection

Focus:
Second step of visualization pipeline



Differentiation on Grids

- First approach
  - Replace differential by “finite differences”
  - Note that approximating the derivative by

$$f'(x) = \frac{df}{dx} \to \frac{\Delta f}{\Delta x}$$

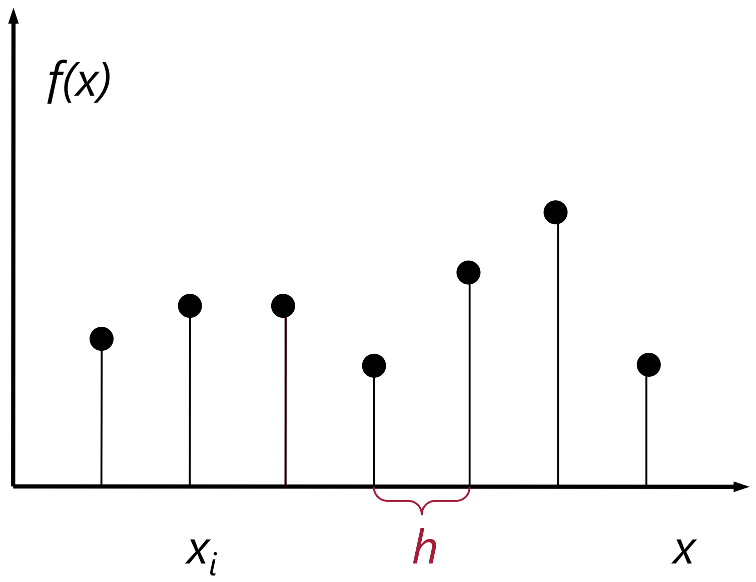
  causes subtractive cancellation and large rounding errors for small h

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

- Second approach
  - Approximate/interpolate (locally) by differentiable function and differentiate this function

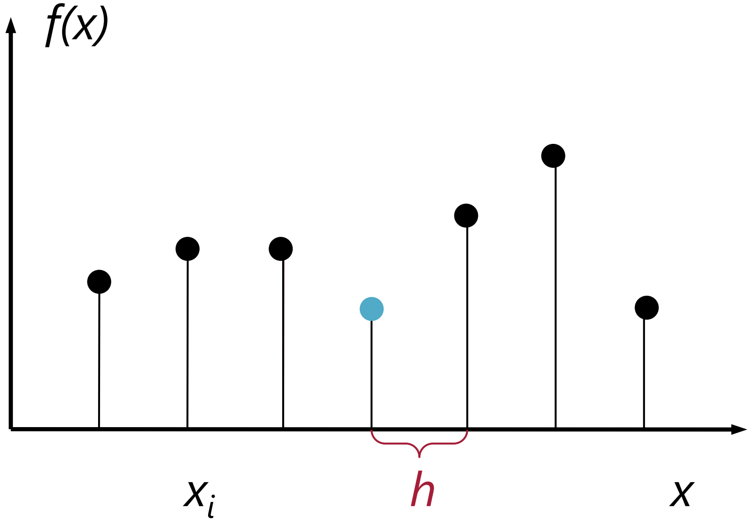
Differentiation on Grids

- Finite differences on uniform grids with grid size *h* (1D case)



Differentiation on Grids

- Finite differences on uniform grids with grid size *h* (1D case)
  - Forward differences $$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$$
  - Backward differences $$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$$
  - Central differences $$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$



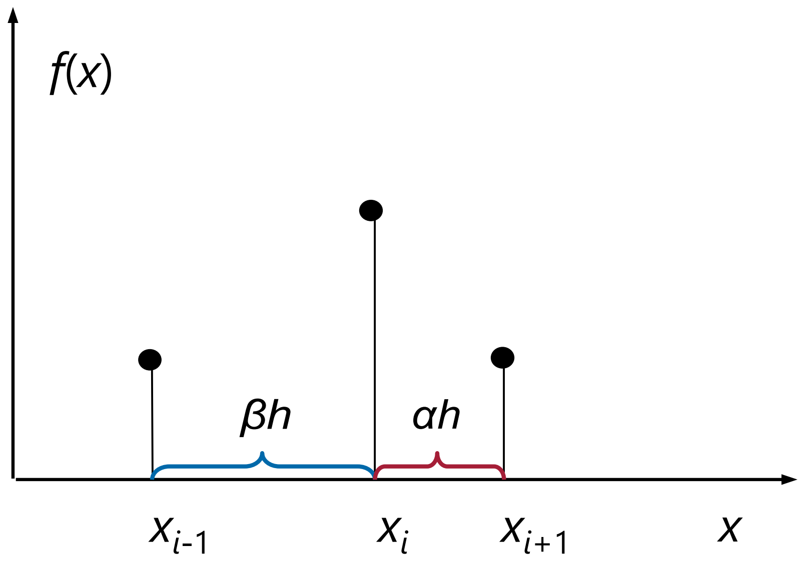
- Error estimation:
  - Forward/backward differences are first-order
  - Central differences are second-order

Differentiation on Grids

- Finite differences on non-uniform grids
  - Forward and backward differences as for uniform grids with

$$x_{i+1} - x_i = \alpha h$$

$$x_i - x_{i-1} = \beta h$$

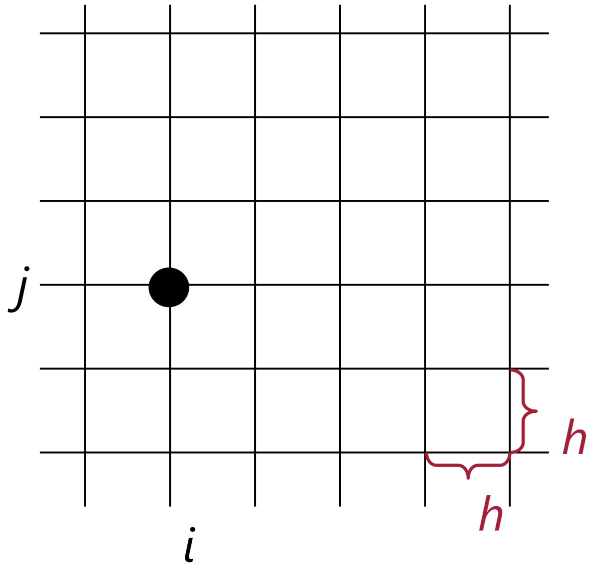


Differentiation on Grids

- 2D or 3D uniform or rectangular grids
  - Partial derivatives

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$$

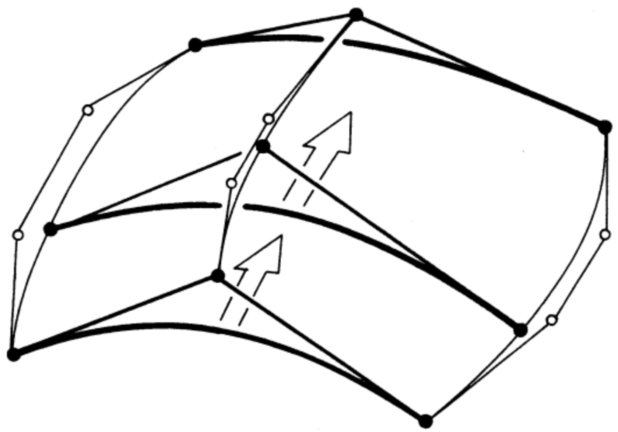
  - Same as in univariate case along each coordinate axis
  - **Example:** gradient in a 3D uniform grid computed via central differences
    → *Normals for lighting!*



$$\text{grad } f = \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{f_{i+1,j,k} - f_{i-1,j,k}}{2h} \\ \frac{f_{i,j+1,k} - f_{i,j-1,k}}{2h} \\ \frac{f_{i,j,k+1} - f_{i,j,k-1}}{2h} \end{pmatrix}$$

Interpolation on Grids

- Manifolds with more than 1D
- Tensor product
- Combination of several univariate interpolations
- **Example** for 2D surface:
  - $n \cdot m$ values f_{ij} with $i = 1 \dots n$ and $j = 1 \dots m$ given at points $X \times Y = (x_1, \dots, x_n) \times (y_1, \dots, y_m)$
  - n univariate basis functions $\xi_i(x)$ on X
  - m univariate basis functions $\psi_j(y)$ on Y
  - $n \cdot m$ basis functions on $X \times Y$: $\phi_{ij}(x, y) = \xi_i(x) \cdot \psi_j(y)$
  - Tensor product: $f(x,y) = \sum_{i=1, j=1}^{n,m} \phi_{ij}(x,y)\, c_{ij}$



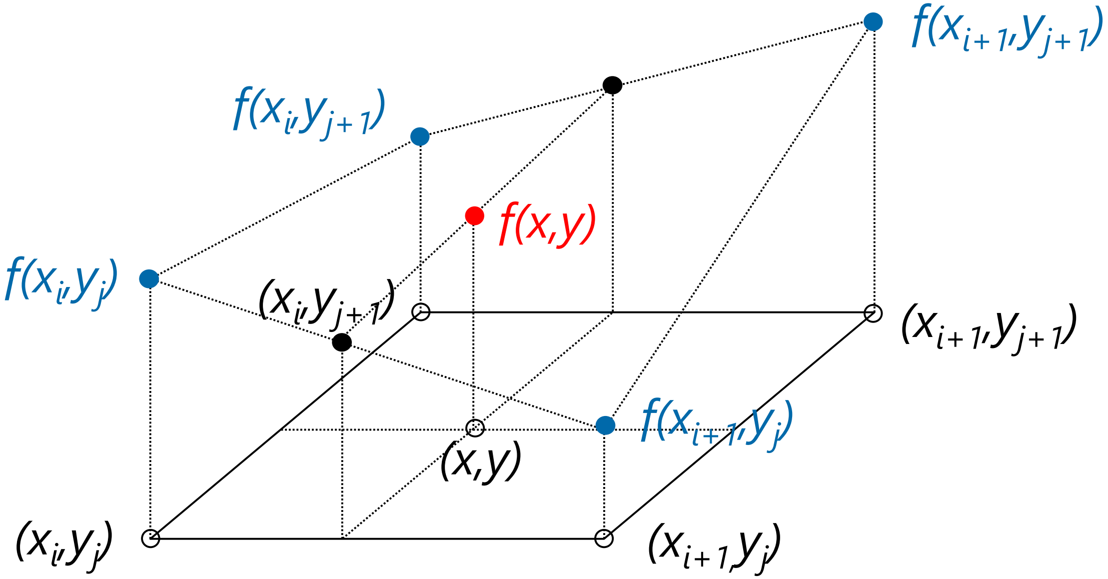
Interpolation on Grids

- Tensor product $f(x,y) = \sum_{i=1,\,j=1}^{n,m} \phi_{ij}(x,y)\, c_{ij}$

- Solve a linear system of equations for the unknown coefficients c_{ij}
- Extension to k dimensions in the same way

Interpolation on Grids

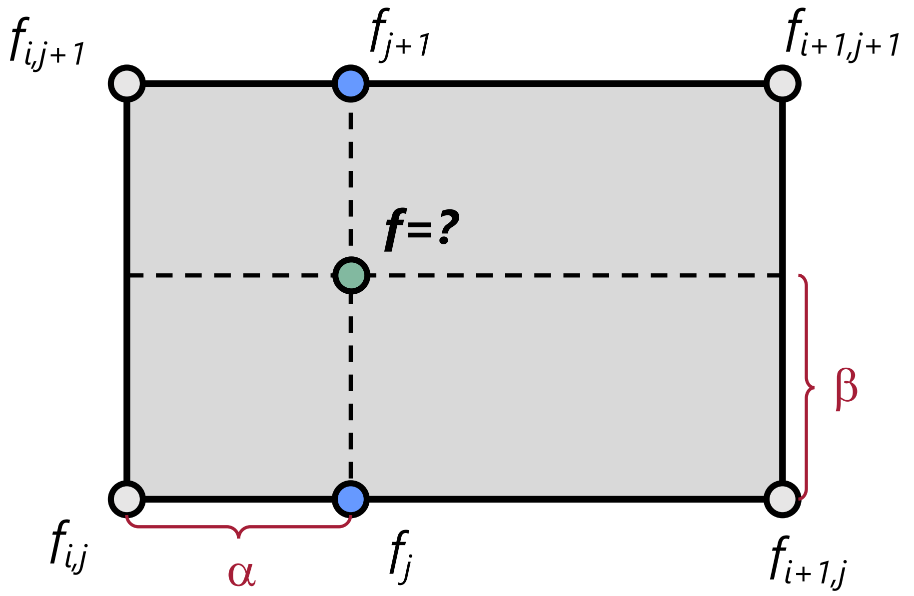
- Bilinear interpolation on a rectangle
  - Tensor product for two linear interpolations
  - 2D local interpolation in a cell
  - Known solution of the linear system of equations for the coefficients c_{ij}
  - Four data points $(x_i, y_j), \dots, (x_{i+1}, y_{j+1})$ with scalar values $f_{i,j} = f(x_i, y_j), \dots$
  - Bilinear interpolation of points (x, y) with $x_i \le x < x_{i+1}$ *and* $y_j \le y < y_{j+1}$
  - "Two times linear"



Interpolation on Grids

- Bilinear interpolation on a rectangle

$$f(x,y) = (1-\beta)\big((1-\alpha)f_{i,j} + \alpha f_{i+1,j}\big) + \beta\big((1-\alpha)f_{i,j+1} + \alpha f_{i+1,j+1}\big)$$
$$= (1-\beta)\, f_j + \beta\, f_{j+1}$$



with
$$f_j = (1-\alpha)\, f_{i,j} + \alpha\, f_{i+1,j}$$
$$f_{j+1} = (1-\alpha)\, f_{i,j+1} + \alpha\, f_{i+1,j+1}$$

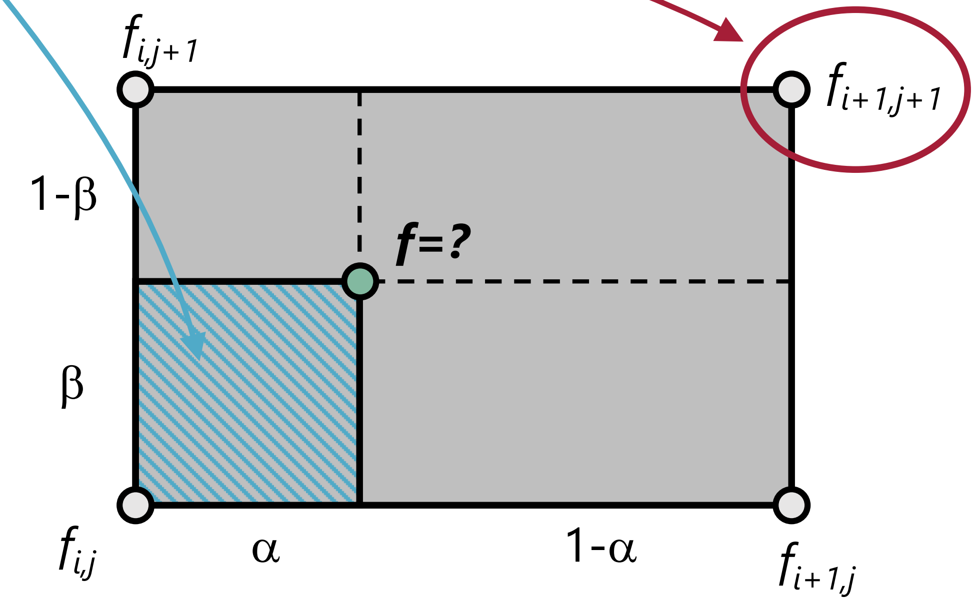
and local coordinates $\alpha = \dfrac{x - x_i}{x_{i+1} - x_i}$, $\beta = \dfrac{y - y_i}{y_{i+1} - y_i}$, $\alpha, \beta \in [0,1]$

Interpolation on Grids

- Bilinear interpolation on a rectangle

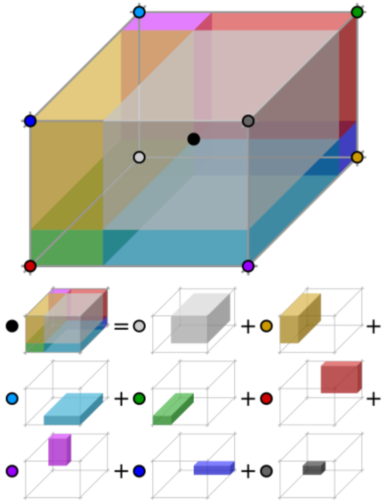
$$f(x,y) = (1-\alpha)(1-\beta)\, f_{i,j} + \alpha(1-\beta)\, f_{i+1,j} + (1-\alpha)\beta\, f_{i,j+1} + \alpha\beta\, f_{i+1,j+1}$$

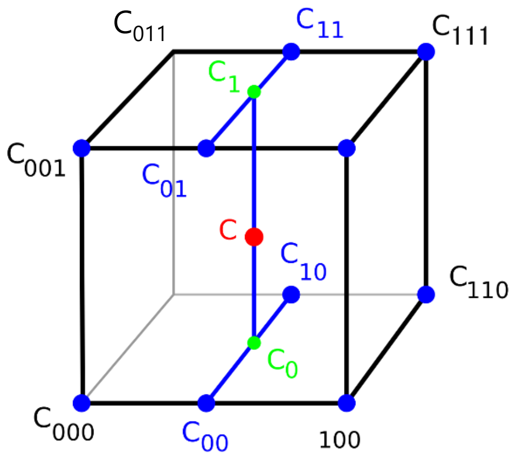
  - Weighted by local area of the opposite node
  - Bilinear interpolation is not linear (but quadratic)
  - Cannot be inverted easily



Interpolation on Grids

- Trilinear interpolation on a 3D uniform grid
  - Straightforward extension of bilinear interpolation (3 local coordinates α,β,γ)
  - Known solution of the linear system of equations for the coefficients c_{ijk}
  - Trilinear interpolation is not linear (but cubic)
  - Efficient evaluation: $f(\alpha,\beta,\gamma) = a + \alpha(b + \beta(e + h\gamma)) + \beta(c + f\gamma) + \gamma(d + g\alpha)$ with coefficients a, b, c, d, e, f, g, h from data at the corner nodes
- Extension to higher order of continuity
  - Piecewise cubic interpolation in 1D
  - Piecewise bicubic interpolation in 2D
  - Piecewise tricubic interpolation in 3D
  - Based on Hermite polynomials



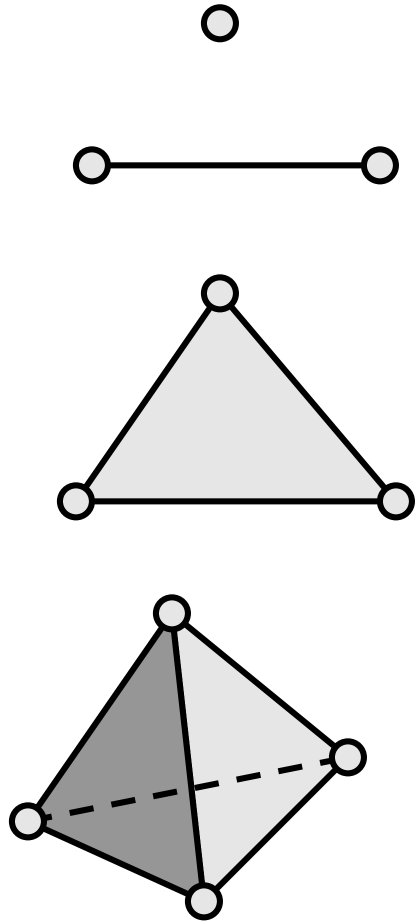


Interpolation on Grids

- Interpolation on un/structured grids (triangle meshes etc.)?
- Affine combination of points $\boldsymbol{x}$ (in Euclidean space):
  - Linear combination $\Sigma_i \, \alpha_i \cdot \boldsymbol{x_i}$
  - $0 \leq \alpha_i \leq 1, \forall \, i$
  - $\Sigma_i \, \alpha_i = 1$
  - α_i are Barycentric coordinates
- Affinely independent set of points:
  - No point can be expressed as affine combination of the other points
  - Maximum number of points is $d + 1$ in $\mathbb{R}^d$
  - Barycentric interpolation is linear

Interpolation on Grids

- d-Simplex in $\mathbb{R}^d$
  - $d + 1$ affinely independent points
  - Span of these points
  - 0D: point
  - 1D: line
  - 2D: triangle
  - 3D: tetrahedron



Interpolation on Grids

- Barycentric interpolation on a simplex
  - $d + 1$ points $\boldsymbol{x_i}$ with function values f_i
  - Point $\boldsymbol{x}$ within the simplex described as affine combination of $\boldsymbol{x_i}$
  - Possible approach: solve for coefficients α_i based on $\boldsymbol{x} = \Sigma_i \alpha_i \cdot \boldsymbol{x_i}$ and $\Sigma_i \alpha_i = 1$
  - Function value at $\boldsymbol{x}$: $f = \Sigma_{i\alpha_i} \cdot f_i$ is affine combination of f_i
- Barycentric coordinates from area/volume considerations:

$$\alpha_i = \frac{\mathbf{Vol}\left(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, \mathbf{x}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{d+1}\right)}{\mathbf{Vol}\left(\mathbf{x}_1, \ldots, \mathbf{x}_{d+1}\right)}$$

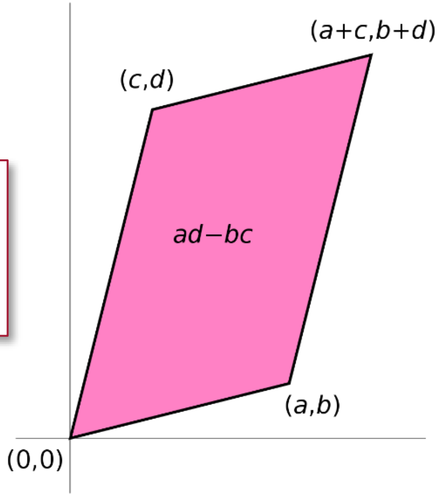
replace $\boldsymbol{x_i}$ by $\boldsymbol{x}$

$$\mathbf{Vol}\left(\mathbf{x}_1, \ldots, \mathbf{x}_{d+1}\right) = \mathbf{det}\begin{pmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_{d+1} \\ 1 & \cdots & 1 \end{pmatrix}$$

→ generalized measure for area/volume

Determinant (2×2):

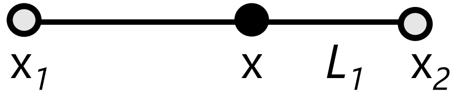
$$\mathbf{det}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$



Interpolation on Grids

- Barycentric coordinates from area/volume considerations

$d = 1$

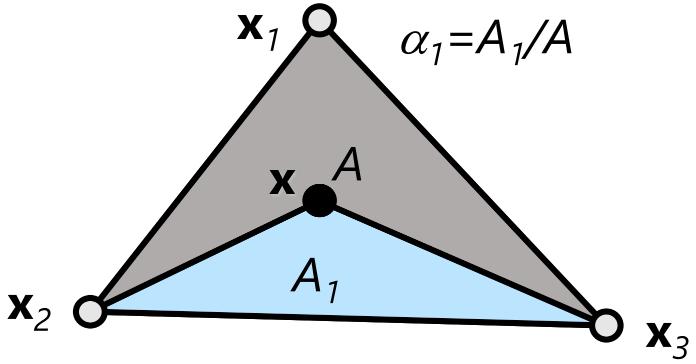


$\alpha_1 = L_1/L$

Opposite local length

$$x = \alpha_1 x_1 + \alpha_2 x_2 \qquad \alpha_1 + \alpha_2 = 1 \quad \rightarrow \quad x = (1-u)x_1 + u x_2 \qquad u = \alpha_2$$

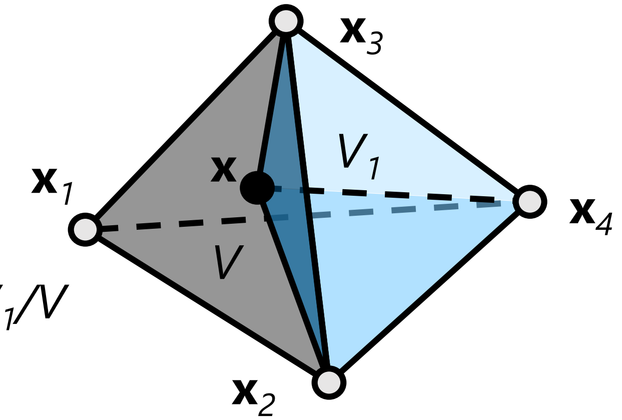
$d = 2$



$\alpha_1 = A_1/A$

Opposite local area

$d = 3$

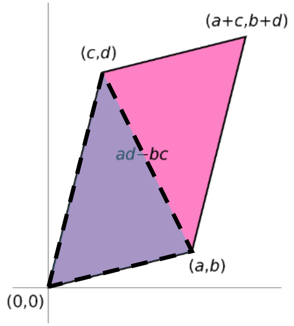


$\alpha_1 = V_1/V$

Opposite local volume

Interpolation on Grids

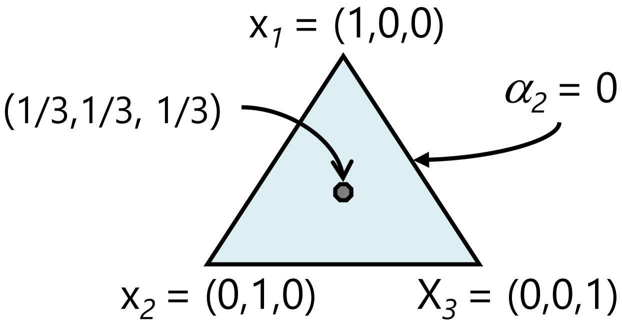
Determinant (2 × 2):
$$\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$



- Barycentric interpolation in a triangle
  - Geometrically, Barycentric coordinates are given by the ratios of the area of the whole triangle and the subtriangles defined by x and any two points of x_1, x_2, x_3.

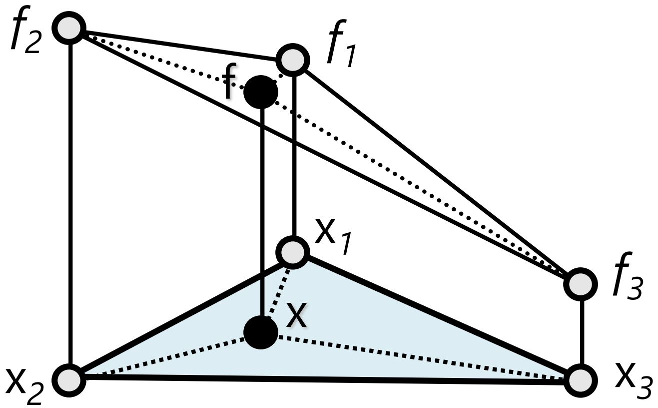
$$\textbf{Vol}(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3) = \textbf{det}\begin{pmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{pmatrix} = \textbf{det}\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \pm 2\,\mathrm{Area}(\Delta(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3))$$



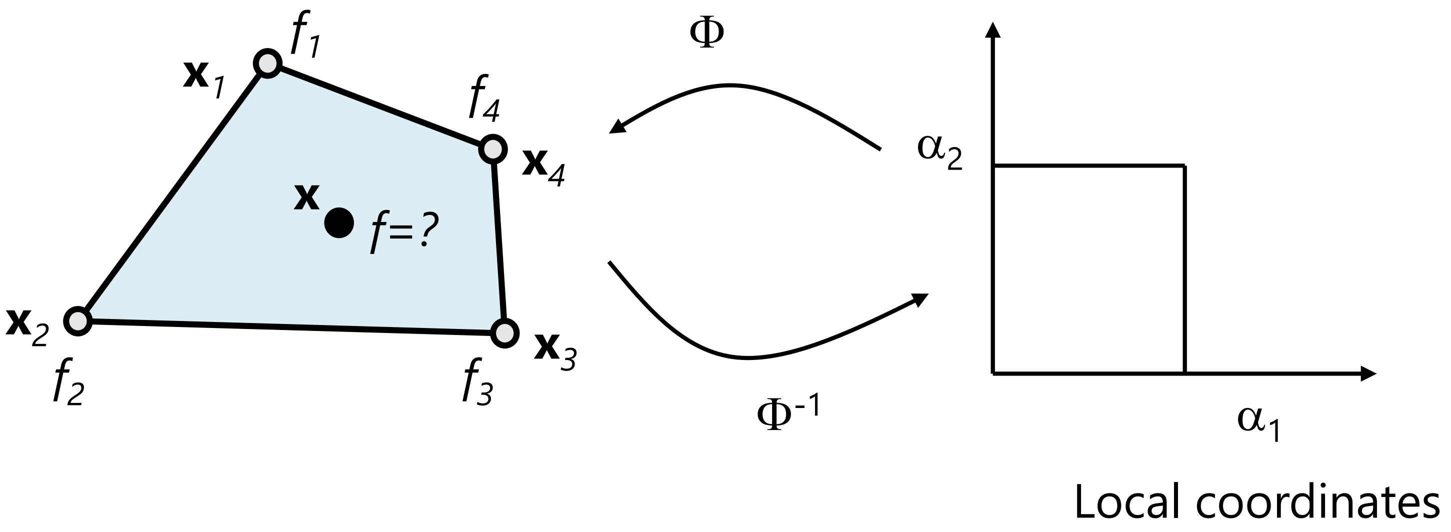
$$\alpha_1 = \frac{\textbf{Vol}(\mathbf{x},\mathbf{x}_2,\mathbf{x}_3)}{\textbf{Vol}(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)} \qquad \alpha_1 + \alpha_2 + \alpha_3 = 1$$

$$\mathbf{x} = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \alpha_3\mathbf{x}_3 \qquad f(\mathbf{x}) = \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3$$



Interpolation on Grids

- Interpolation in a generic quadrilateral
  - Main application: curvilinear grids
  - **Problem:** find a parameterization for arbitrary quadrilaterals

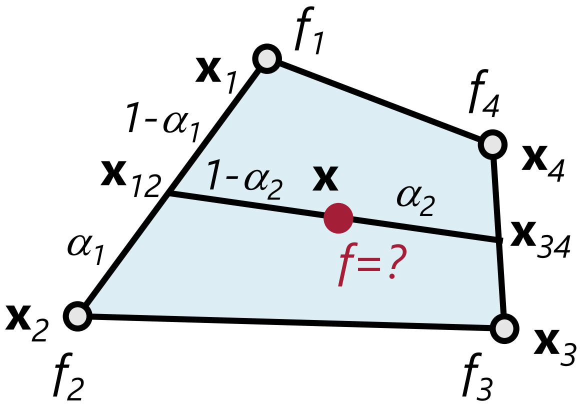


Physical space *Computational space*

Interpolation on Grids

- Mapping ϕ from quadratic domain to quadrangular domains is known: **Bilinear interpolation**

$$x_{12} = \alpha_1 \cdot \boldsymbol{x_1} + (1 - \alpha_1) \cdot \boldsymbol{x_2} \qquad \alpha_1 \in [0,1]$$
$$x_{34} = \alpha_1 \cdot \boldsymbol{x_4} + (1 - \alpha_1) \cdot \boldsymbol{x_3}$$
$$x = \alpha_2 \cdot \boldsymbol{x_{12}} + (1 - \alpha_2) \cdot \boldsymbol{x_{34}} \qquad \alpha_2 \in [0,1]$$

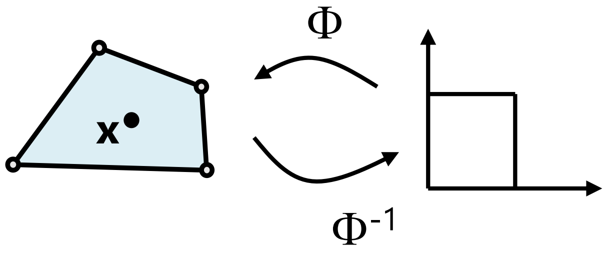


- Computing the inverse of ϕ is more complicated:
  - Analytically solve quadratic system for α_1, α_2
  - Or: numerical solution by Newton iteration

- Final value:

$$f = \alpha_2 \cdot (\alpha_1 \cdot f_1 + (1 - \alpha_1) \cdot f_2) + (1 - \alpha_2) \cdot (\alpha_1 \cdot f_4 + (1 - \alpha_1) \cdot f_3)$$

Interpolation on Grids



Newton's method:
$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

- Jacobi matrix $J(\Phi)$
  - $J(\Phi)_{ij} = {}^{\partial\Phi_i}/_{\partial\alpha_j}$
  - $J(\Phi)_{.j}$ describes direction and speed of position changes of Φ when α_j are varied
- **Newton iteration**

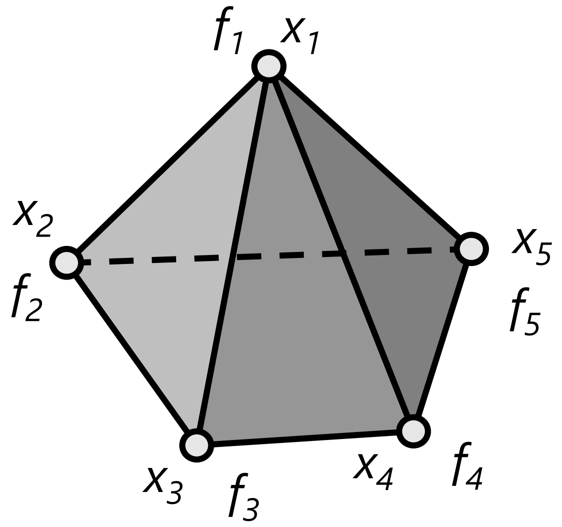
```
start with seed points as start configuration, e.g., αi = 1/2
while ( ||x - Φ(α1,α2,α3)|| > ε )
    compute J(Φ(α1,α2,α3))
    transform Δx to local coordinates (comp. space):
          Δxα = ( x - Φ(α1,α2,α3)) / J(Φ(α1,α2,α3))
    update αi = αi + Δxα,i
```

Maximum error ε

- Stencil-walk algorithm [Bunnig '89] (in context of flow visualization)

Interpolation on Grids

- Other primitive cell types possible
- **Example:** Pyramid

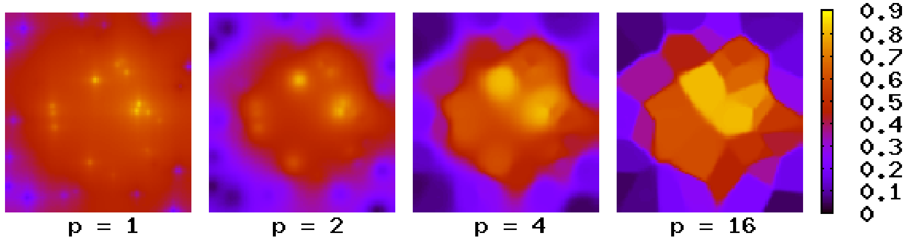


→ bilinear on base face
→ then linear

Interpolation on Grids

- Inverse Distance Weighting
  - Shepard interpolation [D. Shepard, A two-dimensional interpolating function for irregularly spaced data. Proc. ACM. Nat. Conf., 517--524, 1968]
  - Originally developed for scattered data
  - Interpolated values: $f(\boldsymbol{x}) = \sum_i \phi_i(x) f_i$
  - Sample points are vertices of the cell
  - Basis functions: $\phi_i(\boldsymbol{x}) = \frac{\|\boldsymbol{x}-\boldsymbol{x_i}\|^{-p}}{\sum_j \|\boldsymbol{x}-\boldsymbol{x_j}\|^{-p}} \quad \Longrightarrow \quad f(\boldsymbol{x}) = \frac{\sum_i(\|\boldsymbol{x}-\boldsymbol{x_i}\|^{-p} \cdot f_i)}{\sum_i \|\boldsymbol{x}-\boldsymbol{x_i}\|^{-p}}$
  - Define values at sample points $f(\boldsymbol{x_i}) := f_i = \lim_{x \to x_i} f(x)$

Interpolation without Grids



- Shepard interpolation
  - Different exponents for inner and outer neighborhood (default: 2 in the inner neighborhood and 4 in the outer neighborhood)
  - Neighborhood sizes determine how many points are included in inverse distance weighting
  - The neighborhood size can be specified in terms of
    - Radius or
    - Number of points or
    - Combination of the two
  - Neighborhood is not given explicitly (as opposed to inverse distance weighting on grids)

Interpolation without Grids

- Radial basis functions (RBF)
  - n function values f_i given at n points x_i
  - Interpolant $$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|)$$
  - Univariate radial basis $\phi(r)$
  - Examples:
    - Polynomials r^v
    - Gaussians $\exp(-ar^2)$

Interpolation without Grids

- Radial basis functions (RBF)
  - n equations for n unknowns
  - Well-defined system of linear equations (vector / matrix notation):

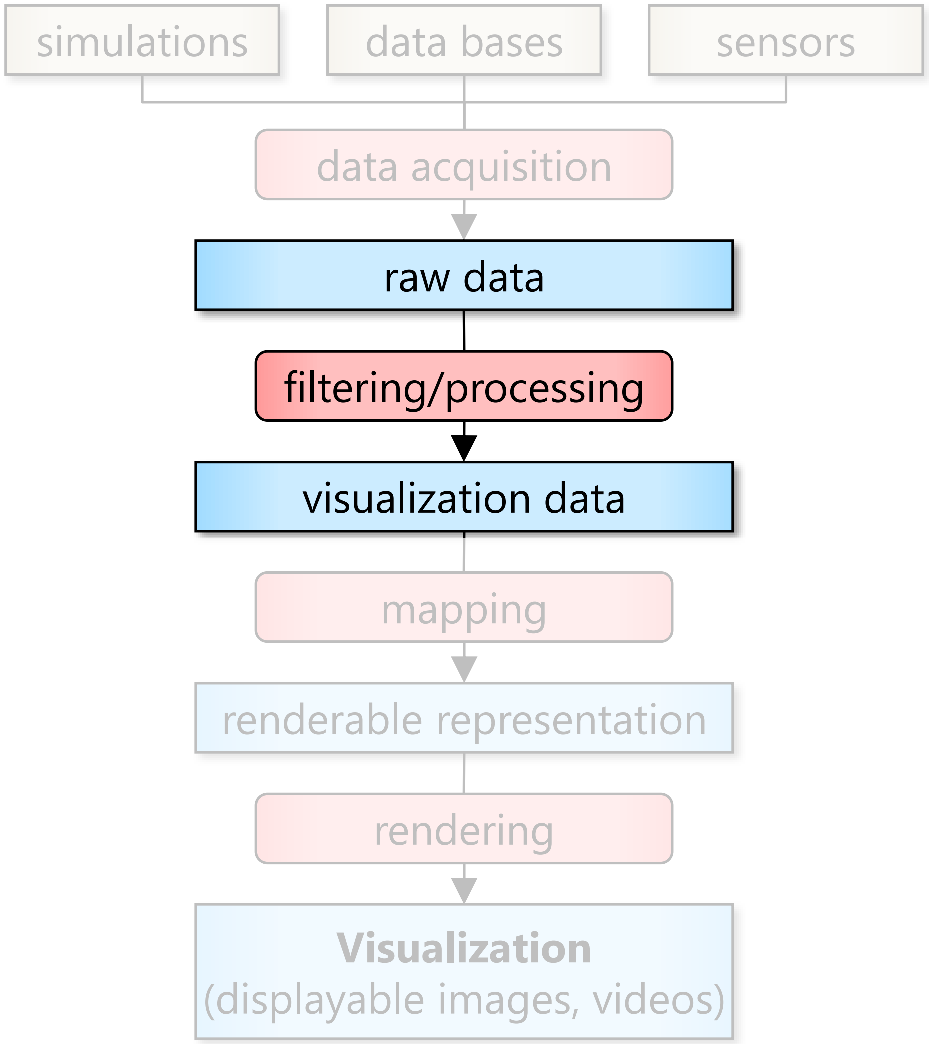
$$\begin{bmatrix} \phi(\|x_1 - x_1\|) & \cdots & \phi(\|x_1 - x_n\|) \\ \vdots & \ddots & \vdots \\ \phi(\|x_n - x_1\|) & \cdots & \phi(\|x_n - x_n\|) \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

→ Compute λ_i for the chosen basis function $\phi(r)$ by solving equation system

Contents

- Voronoi diagrams, Delaunay triangulation
- Univariate interpolation
- Differentiation on grids
- Interpolation on grids
- Interpolation without grids
- Filtering by projection or selection

Focus:
Second step of visualization pipeline



Filtering by Projection or Selection

- **Very often:** too much information to be visualized at once
- Strategy is to reduce the displayed information by filtering

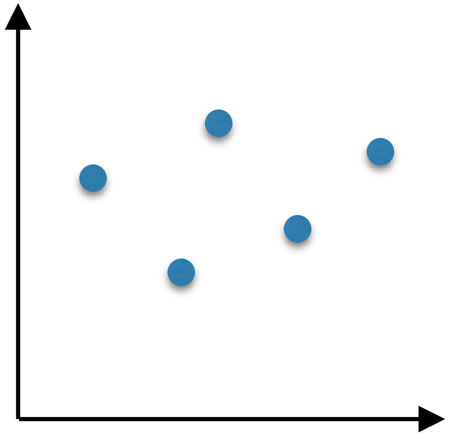
- **Popular approach:**
  Reduce from $\boldsymbol{ndmv}$ to $n'\boldsymbol{d}m'\boldsymbol{v}$, with $n' < n$ and/or $m' < m$ [Wong]
- Techniques:
  - Projection
  - Selection
  - Slicing
- User input needed

Filtering by Projection or Selection

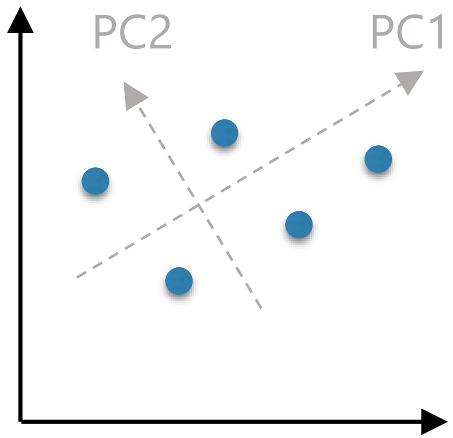
- Projection π
  - Can be applied to both the
    - *domain* and *data values*
  - Projection into subspaces
  - Often a mapping to a subset of the original values is chosen

- Selection σ
  - Selection of data according to logical conditions (predicates)
  - **Example:**
    - Height field 2**d**1**v** represented by (x,y,h)
    - $D_\sigma = \{ (x, y, h) \mid (x^2 + y^2 < (5km)^2) \land (h > 1km) \}$

Axis-Aligned Projection



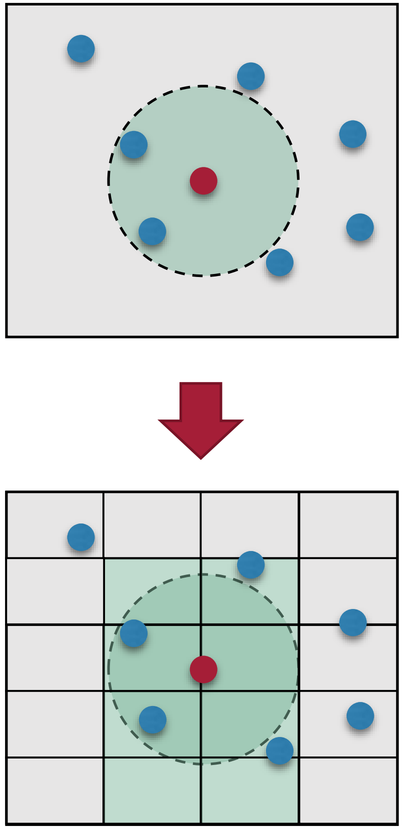
Principle Components



Filtering by Projection or Selection

- Selection σ
  - Selection of data according to logical conditions (predicates)
  - **Example:**
    - Particles P in $\mathbb{R}^3$
    - Often, the efficient selection of neighboring for filtering is required
      → Select all neighboring particles within a fixed radius r
      → Select the n closest particles (neighbors) to a certain particle p_i
    - **Problem:** Brute-force algorithm: $O(n^2)$ → slow for large $|P|$
    - **Solution:** Use data structures for efficient neighbor search
      - Often hierarchical/tree-based (kd-Tree, Quadtree/Octree)
      - *Simple solution:* uniform grid data structure for fixed-radius neighbor search
        - Divide bounding box into uniform grid → $O(1)$
        - Sort all Particles into grid cell that contains their center point → $O(n)$
        - Neighbor search: only test particles in grid cells within search radius r → worst case: $O(n^2)$

Fixed-radius neighbor search



Filtering by Projection or Selection

- Slicing
- **Example:** 2D cutting surface (slice) through a 3D volume