Algorithms for GIS

Spatial data:
Models and representation
(part I)

Laura Toma

Bowdoin College

Outline

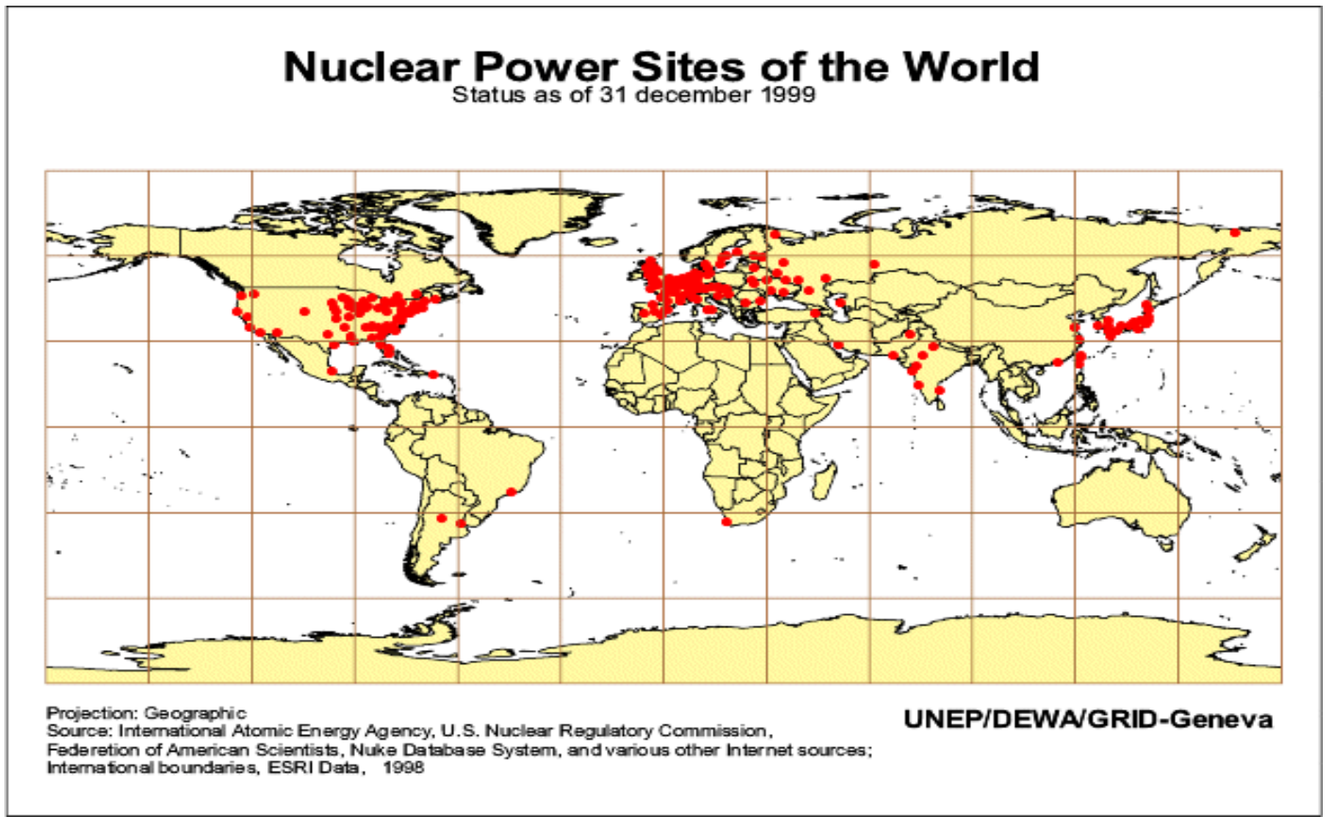
Spatial data in GIS applications

- Point data
- Networks
- Terrains
- Planar maps and meshes

- Data structures and representation
  - Networks
  - Terrains: raster and TIN
  - Next time: Planar maps and meshes: winged-edge, half-edge

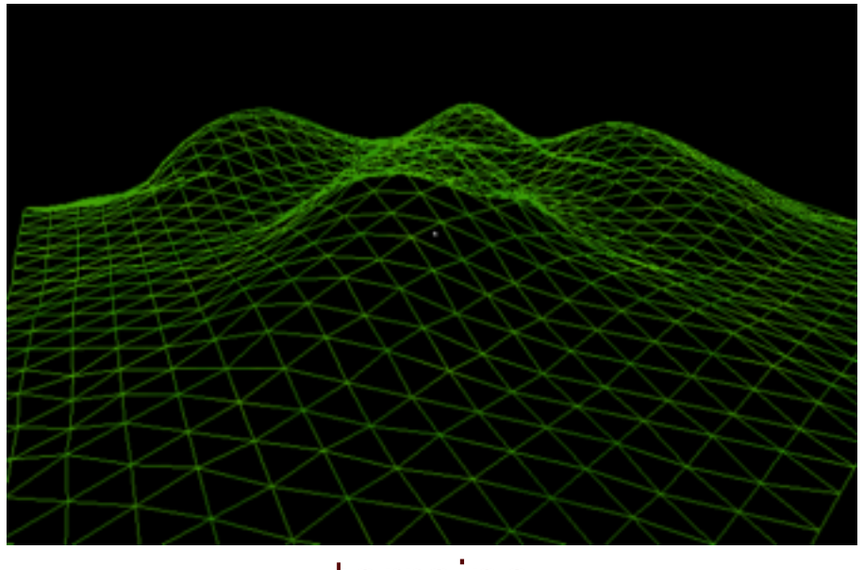
Spatial data in GIS

point data



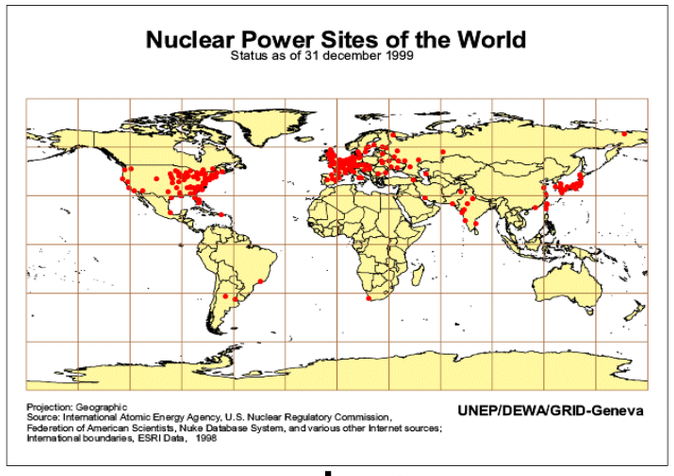
networks

planar maps



terrains

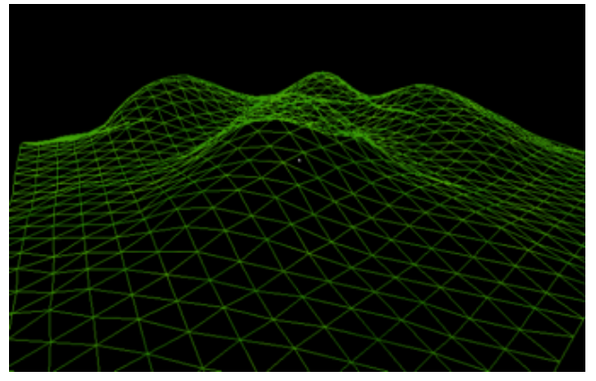
Spatial data in GIS



points

networks

planar maps



terrains

Suppose we have to solve a problem on networks/maps/terrains

- Find optimal routes
- Find all polygons neighboring the coast
- Model impact of a volcano explosion

What is a good representation?

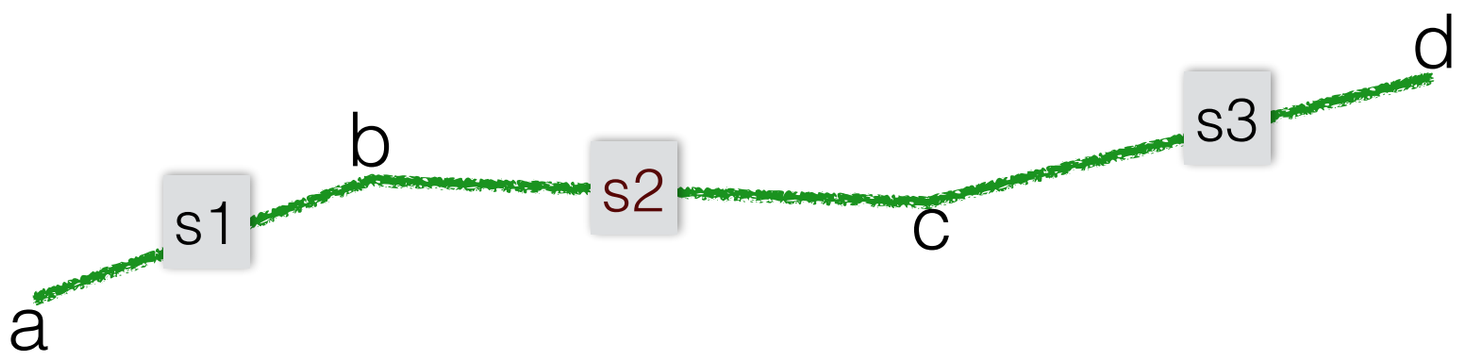
Networks

Data structures for networks

- What do we expect to do with a network?
  - traverse paths
  - find all vertices adjacent to a given vertex,…

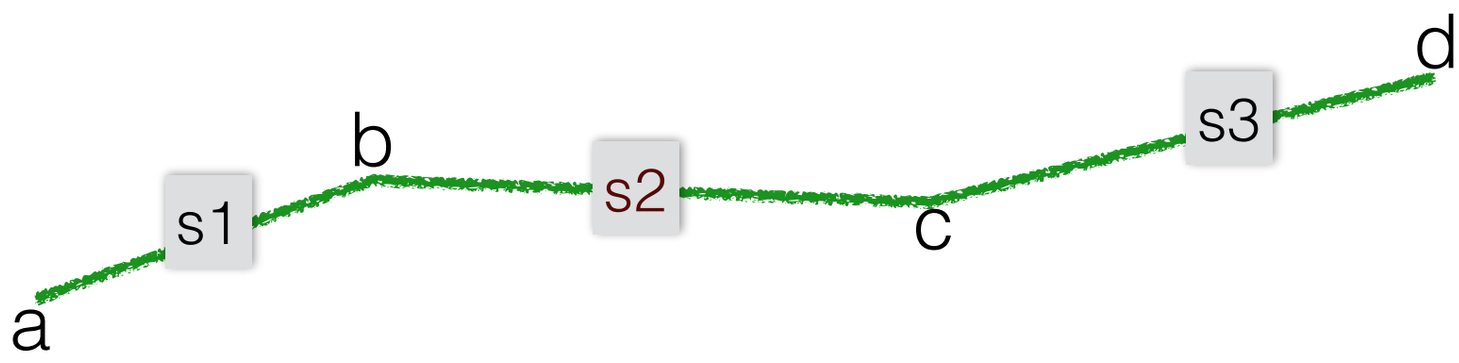
Data structures for networks

- First attempt
  - list of points, each point stores its coordinates
  - list of segments, each segment stores pointers to its vertices



Data structures for networks

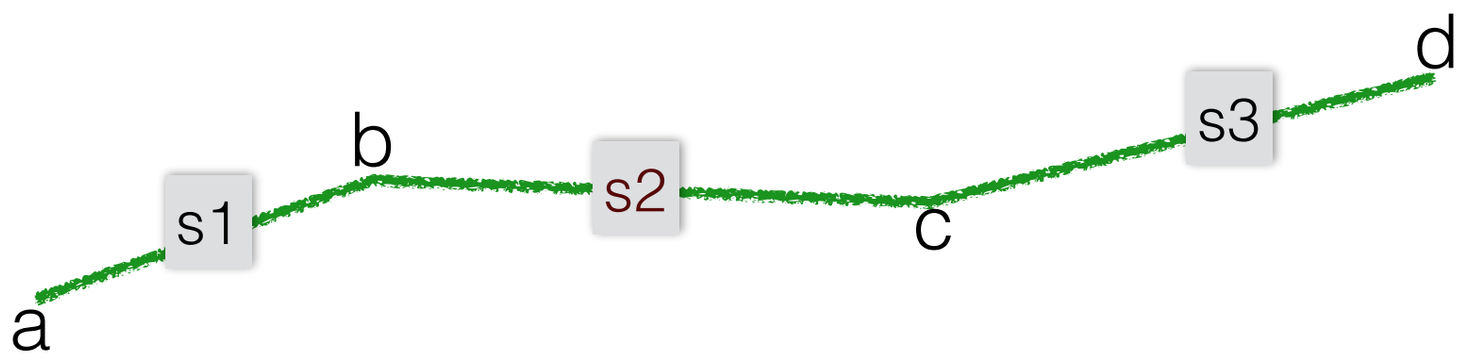
- First attempt
  - list of points, each point stores its coordinates
  - list of segments, each segment stores pointers to its vertices



- points = { a:(x_a, y_a), b:(x_b, y_b), c:(x_c, y_c), d:(x_d, y_d) }
- segments = { *(a, b), (b,c), (c,d)* }
- Assume you want to traverse a path starting at point *a*:
  - search through the segment list looking for a segment with startpoint *a*; you find *(a,b)*
  - search through the segment list looking for another segment with startpoint *b*; you find *(b,c)*

Data structures for networks

- First attempt
  - list of points, each point stores its coordinates
  - list of segments, each segment stores pointers to its vertices

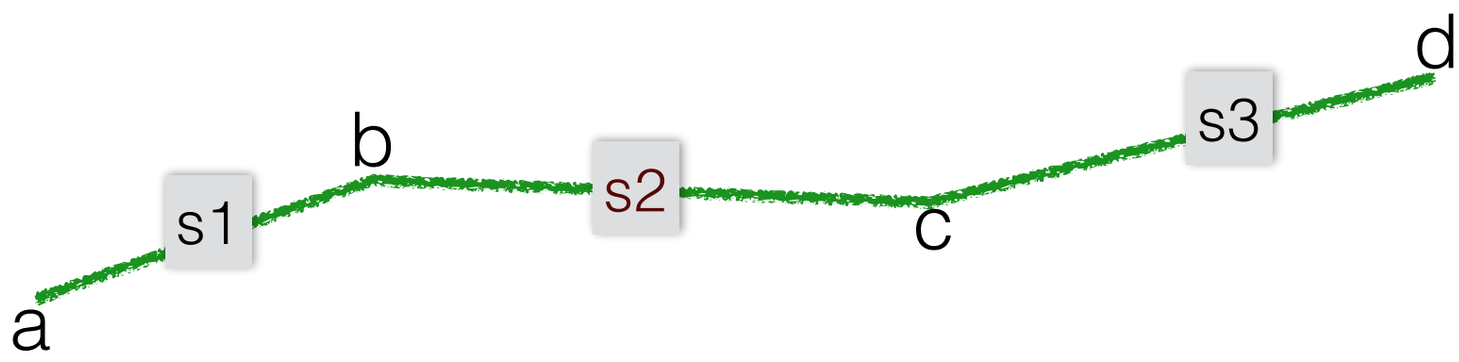


Spaghetti data structure (like spaghetti, no structure and they all mingle together)

Not efficient. Need a data structure that allows to traverse paths efficientl.

Data structures for networks

- Second attempt
  - a network is a graph!
  - use adjacency list (matrix, if dense)



- In practice, this graph needs to be built
  - You got raw data
  - Build the adjacency list corresponding to raw data

Exercise

Assume you download US road data. It comes as a file that might have the following format

- first all the vertices and their geometric coordinates
- then all edges, where an edge is given through the indices of its vertices.

Describe how you would build an adjacency list from it.

Estimate the amount of memory you need for your structure.

Analyze function of |V| vertices and |E| edges.

(1.1, 2.3)

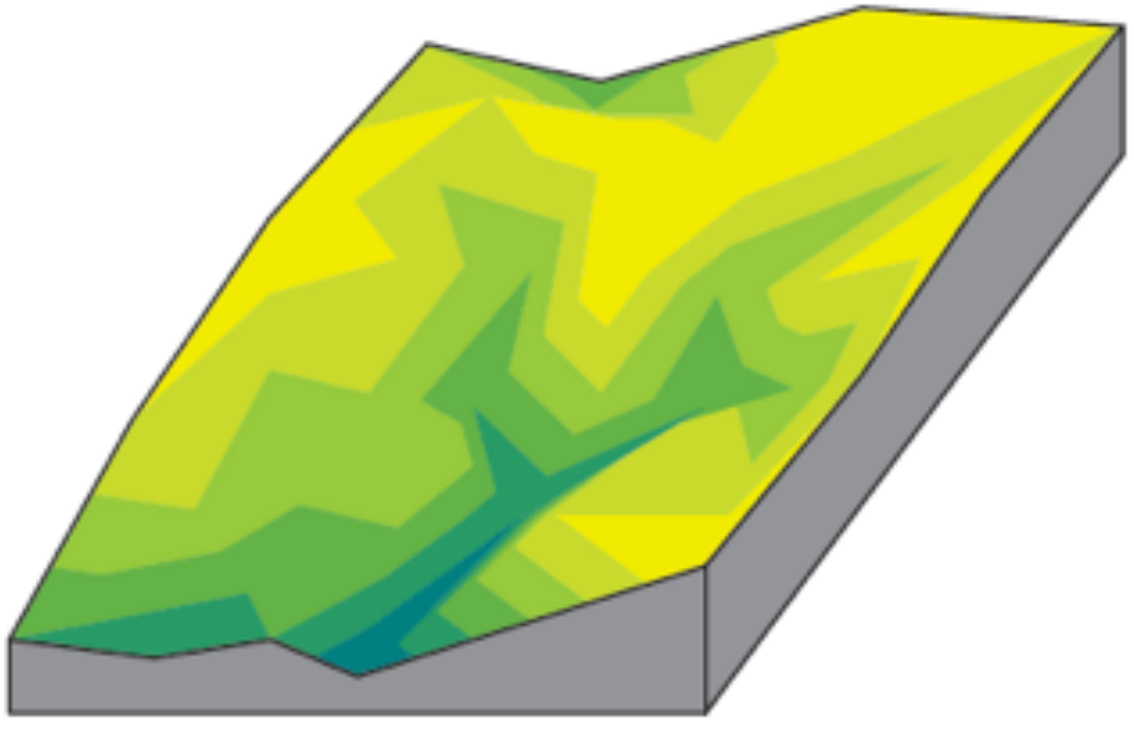
(3.4, 2.1)

(2.6, 1.8)

(1.4,8.2)
(0,1)
(1,2)
(2,3)

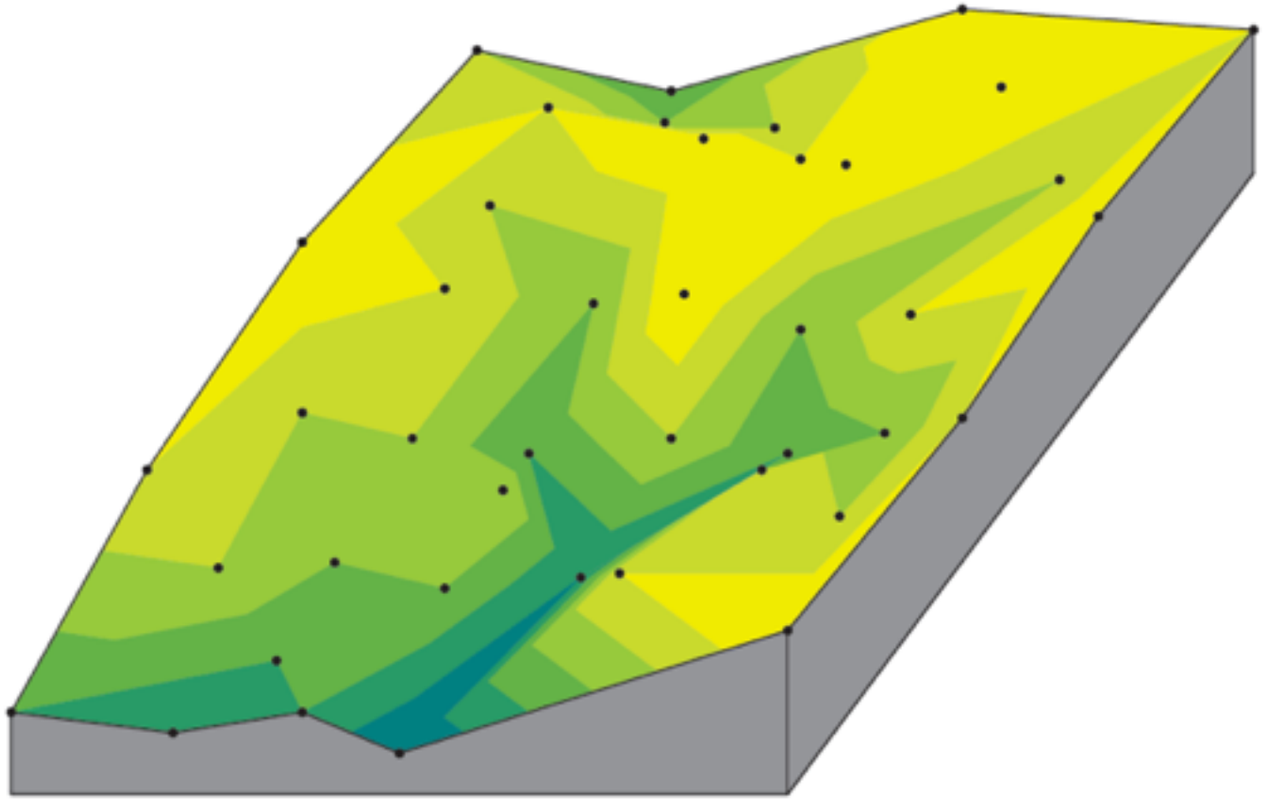
Terrains

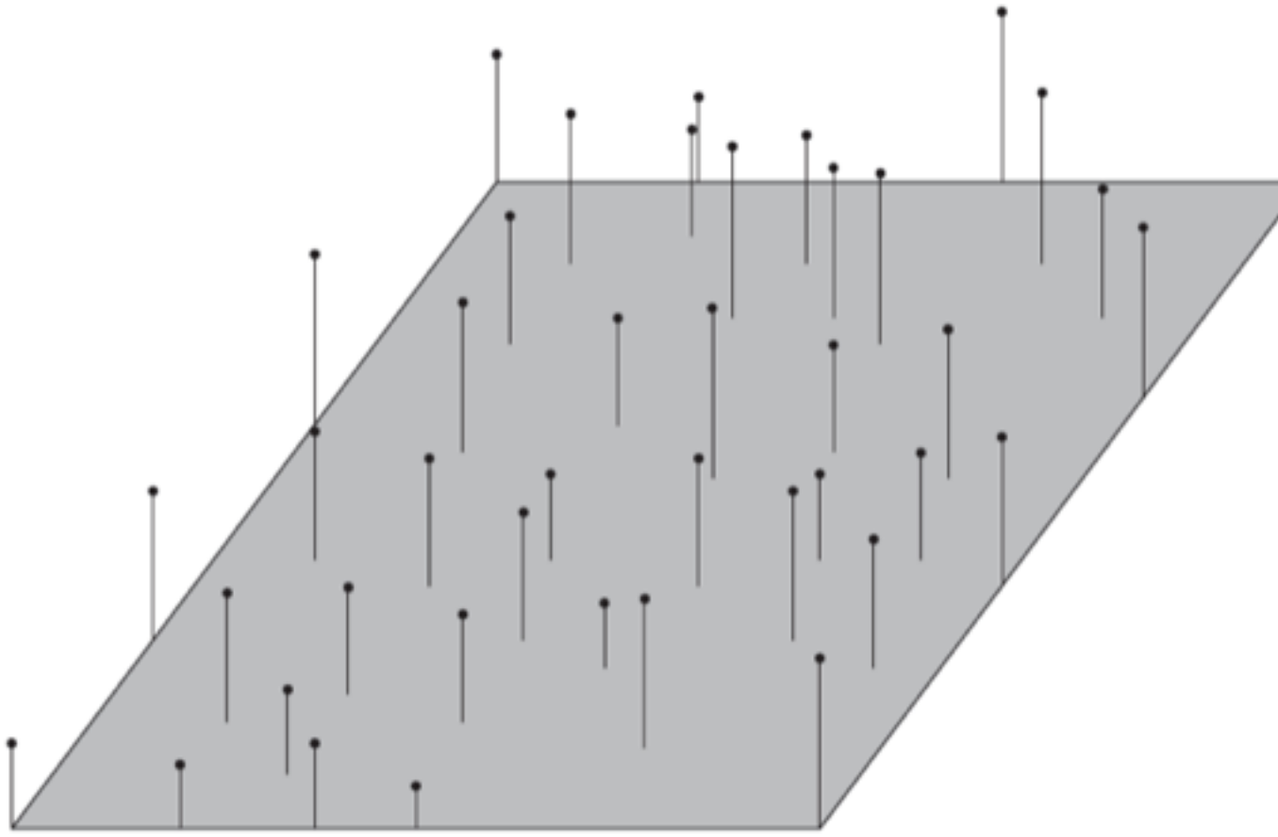
- A terrain is modeled as a function of two variables, z(x,y)
  - z can be elevation, rainfall, population, solar radiation, …
  - For any point (x,y) there is a unique value z(x,y). Put differently, any vertical line intersects it in at most one point



Digital terrain models

- In practice, terrain data comes as a set of sample points {(x,y)} and their sampled z value
- Want a digital terrain model
  - sample points + interpolation method + representation

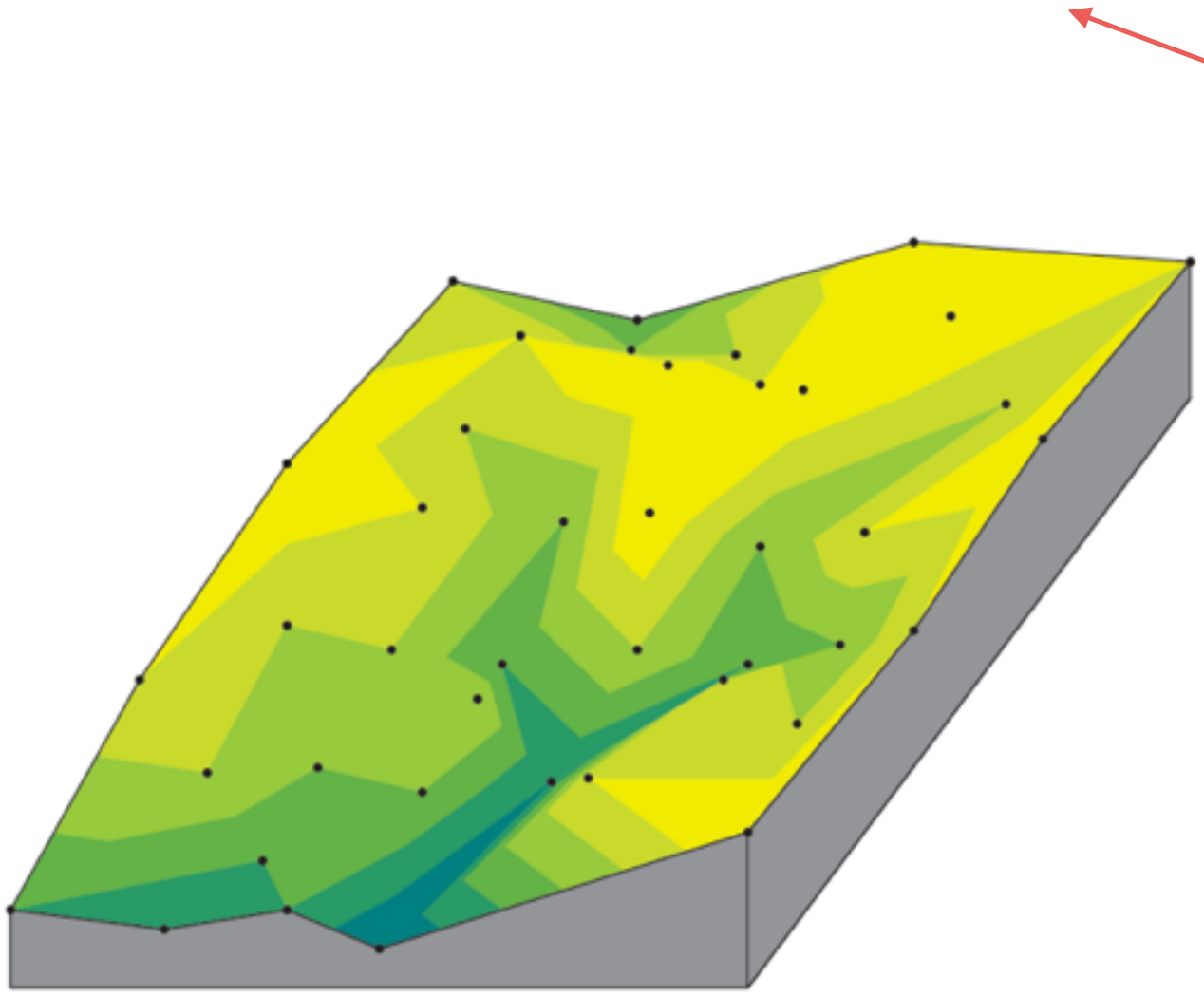


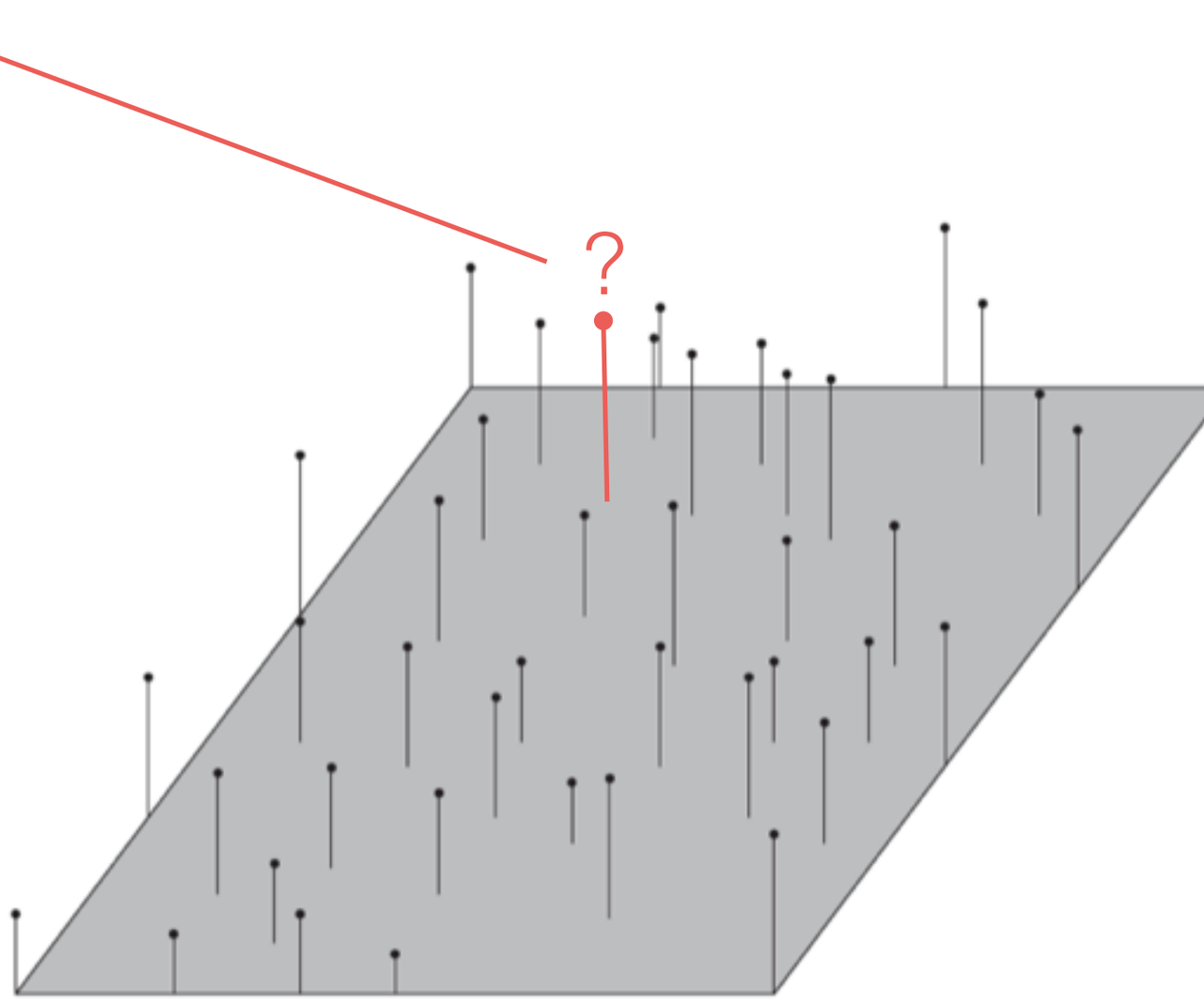


pictures from Herman Haverkort

Digital terrain models

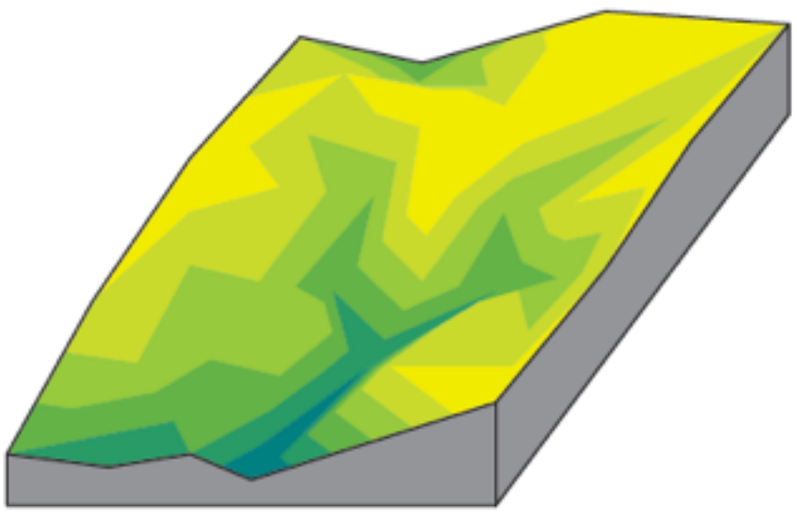
- In practice, terrain data comes as a set of sample points {(x,y)} and their sampled z value
- Want a digital terrain model
  - sample points + interpolation method + representation





pictures from Herman Haverkort

Digital terrain models



watershed maps

risks of
floods, landslides,
eruptions, erosion

visibility maps

costs/benefits of
roads, buildings, dikes,
hydropower plants,
solar power plants

least-cost paths

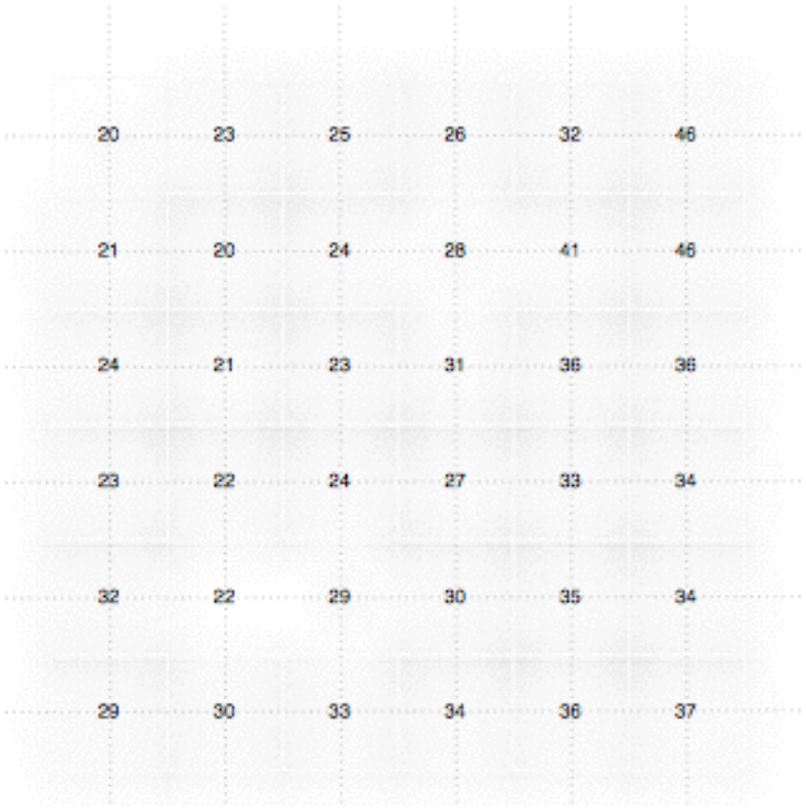
possible locations of
(pre-)historical roads
and settlements

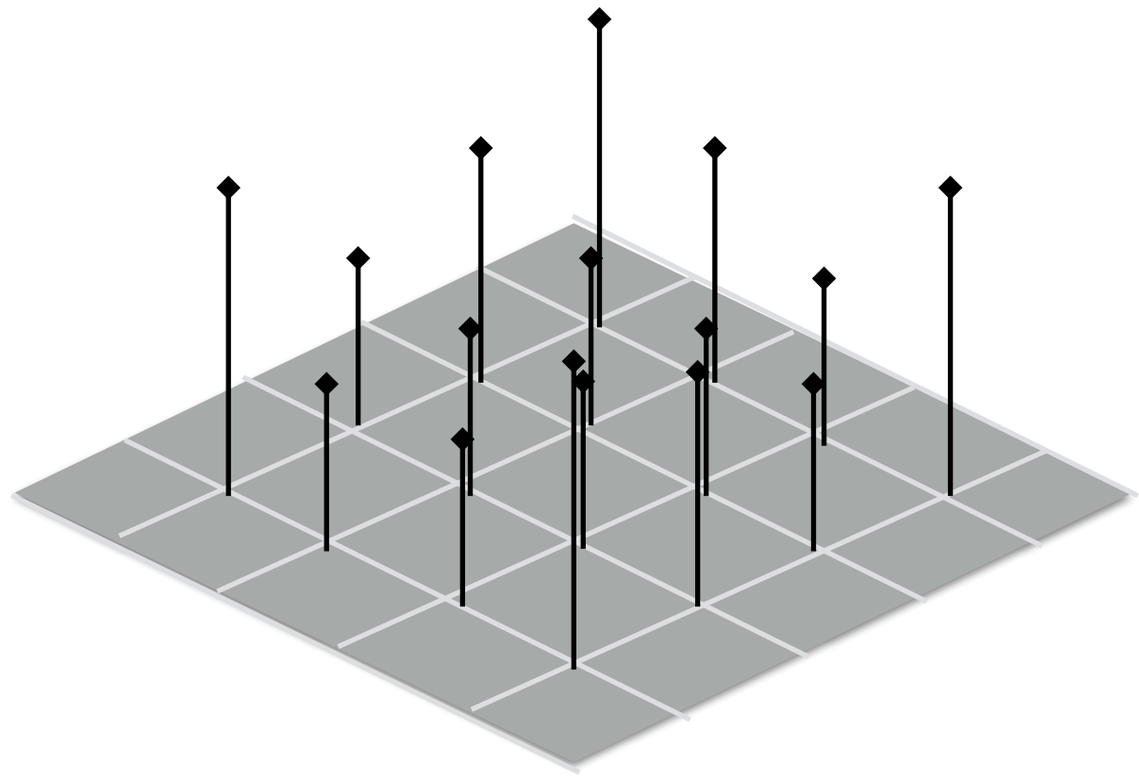
analysis of animal
behaviour and evolution

Digital Terrain Models:

Rasters and TINs

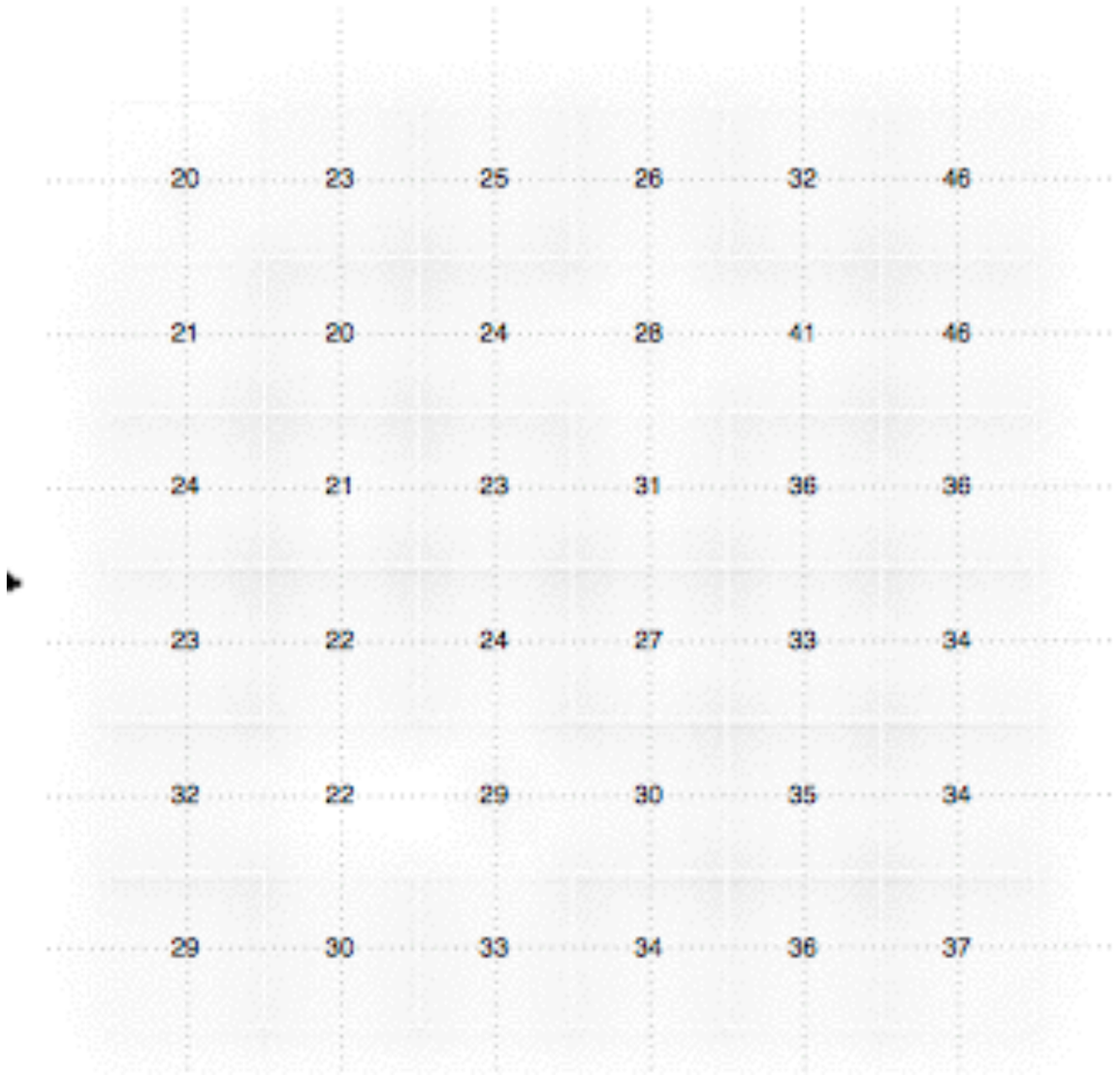
Terrain as a grid (raster)



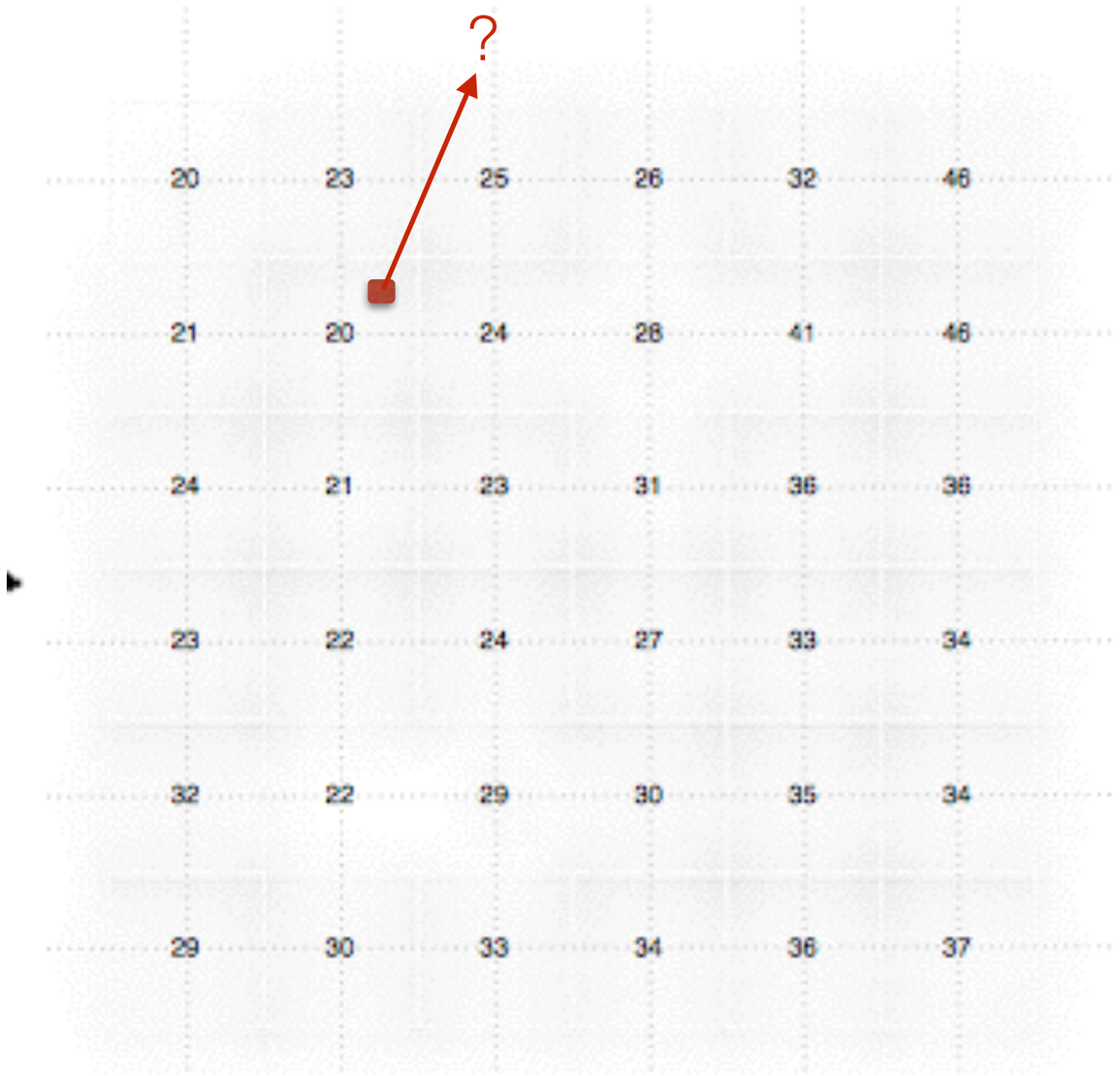


- Samples
  - uniform grid
- Representation
  - 2D array of values
- Interpolation method
  - nearest neighbor, linear, bilinear, splines, krigging, IDW, etc

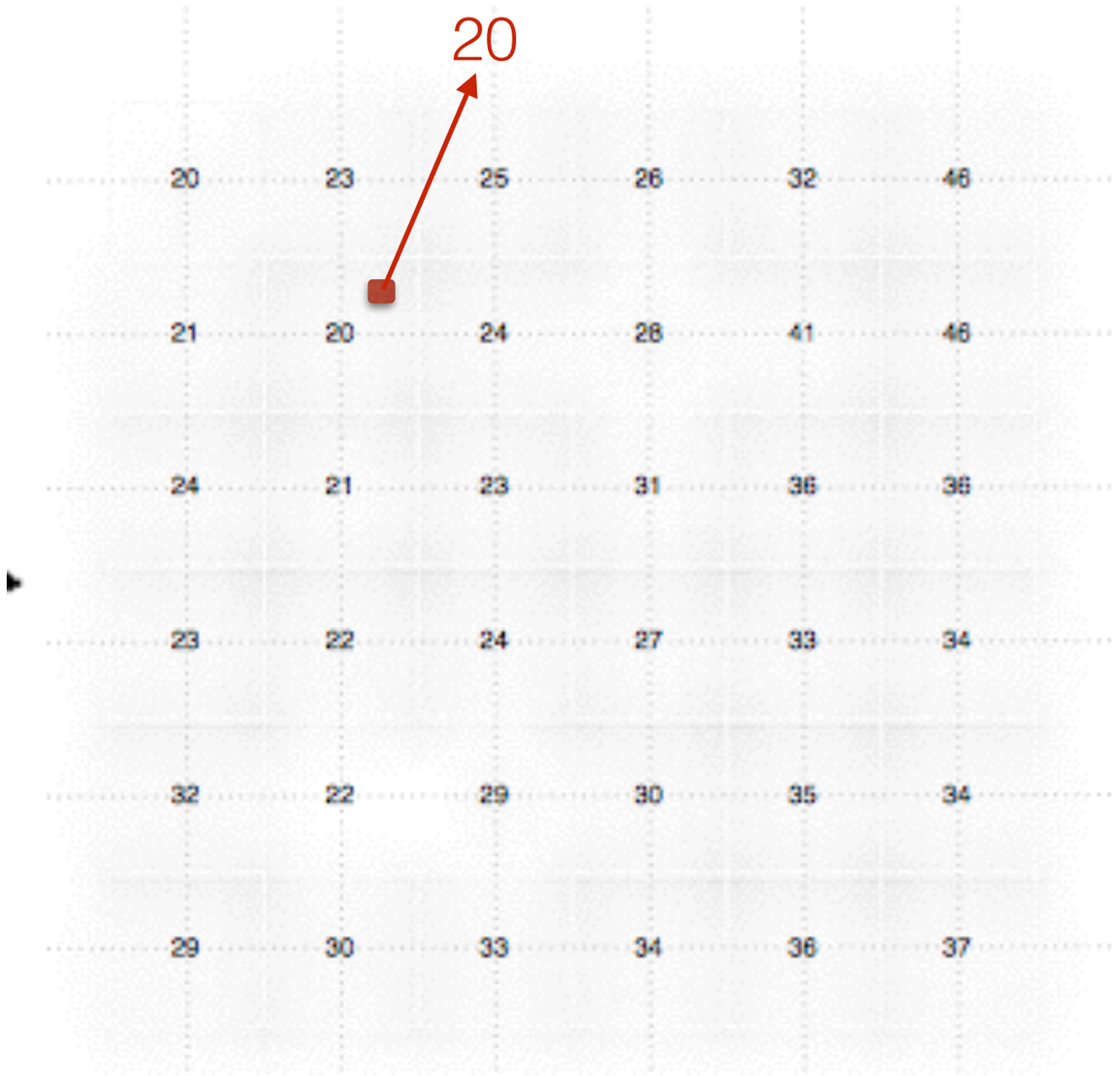
Interpolating grids



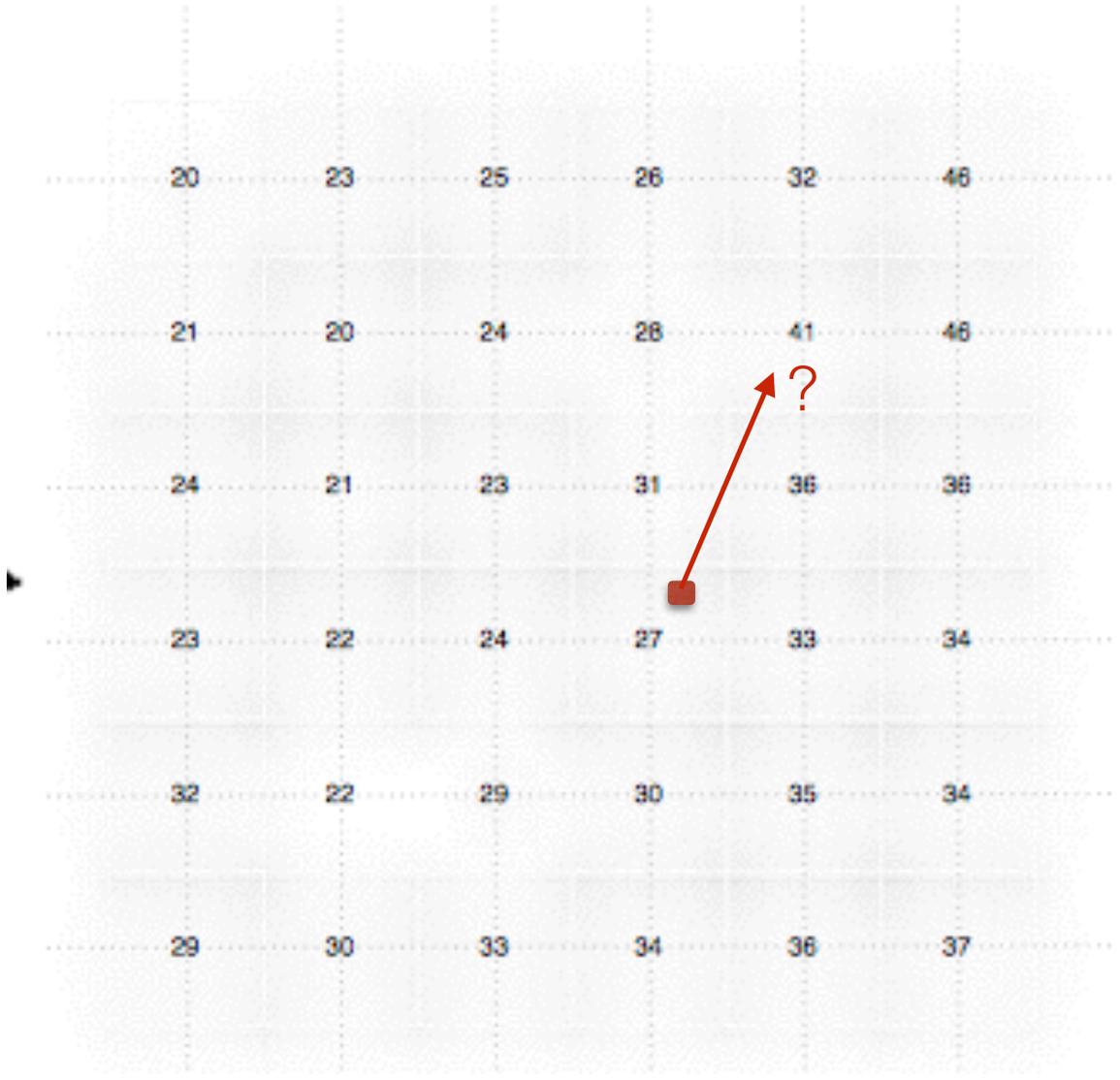
Interpolating grids



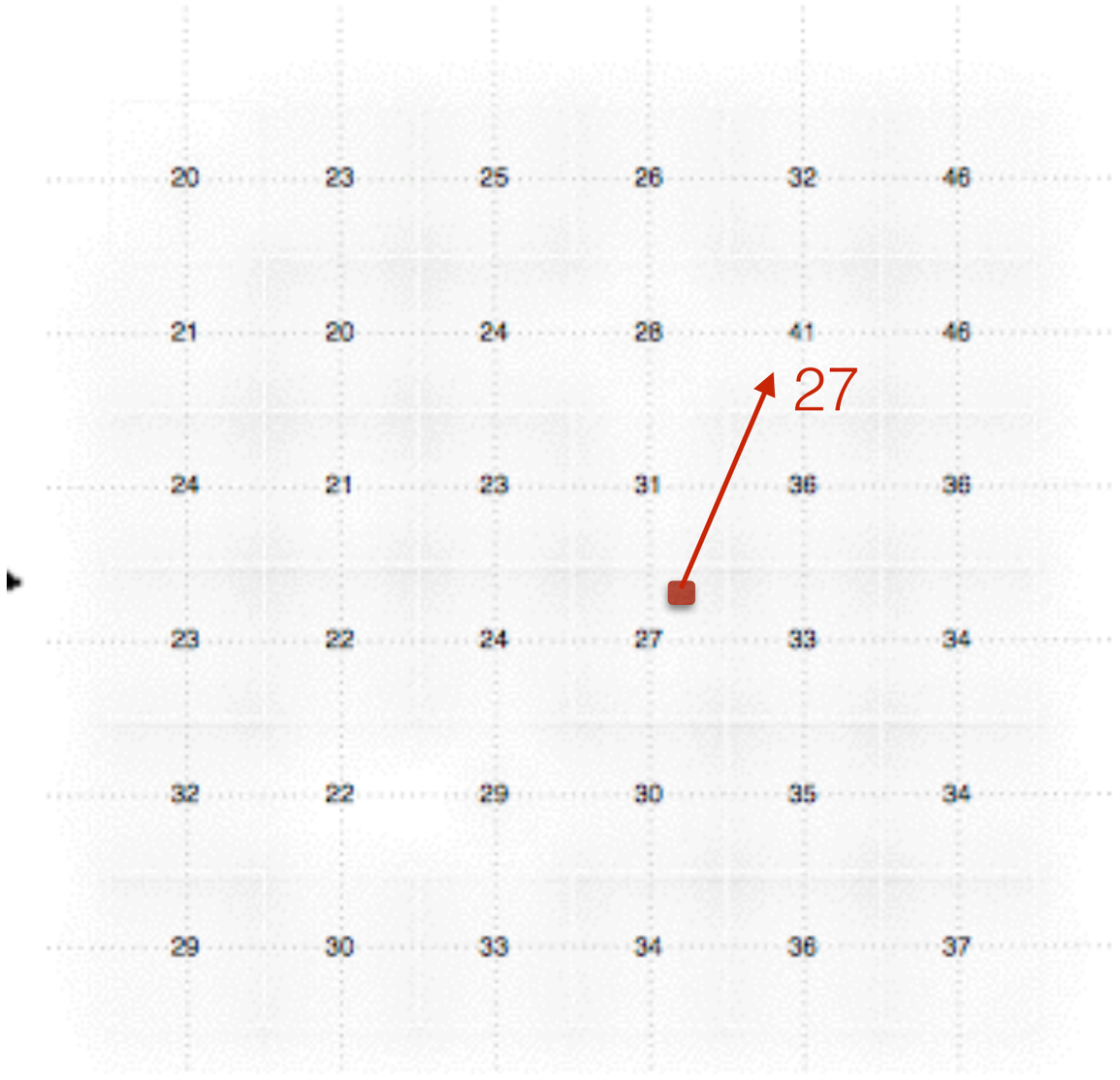
Nearest neighbor interpolation



Nearest neighbor interpolation



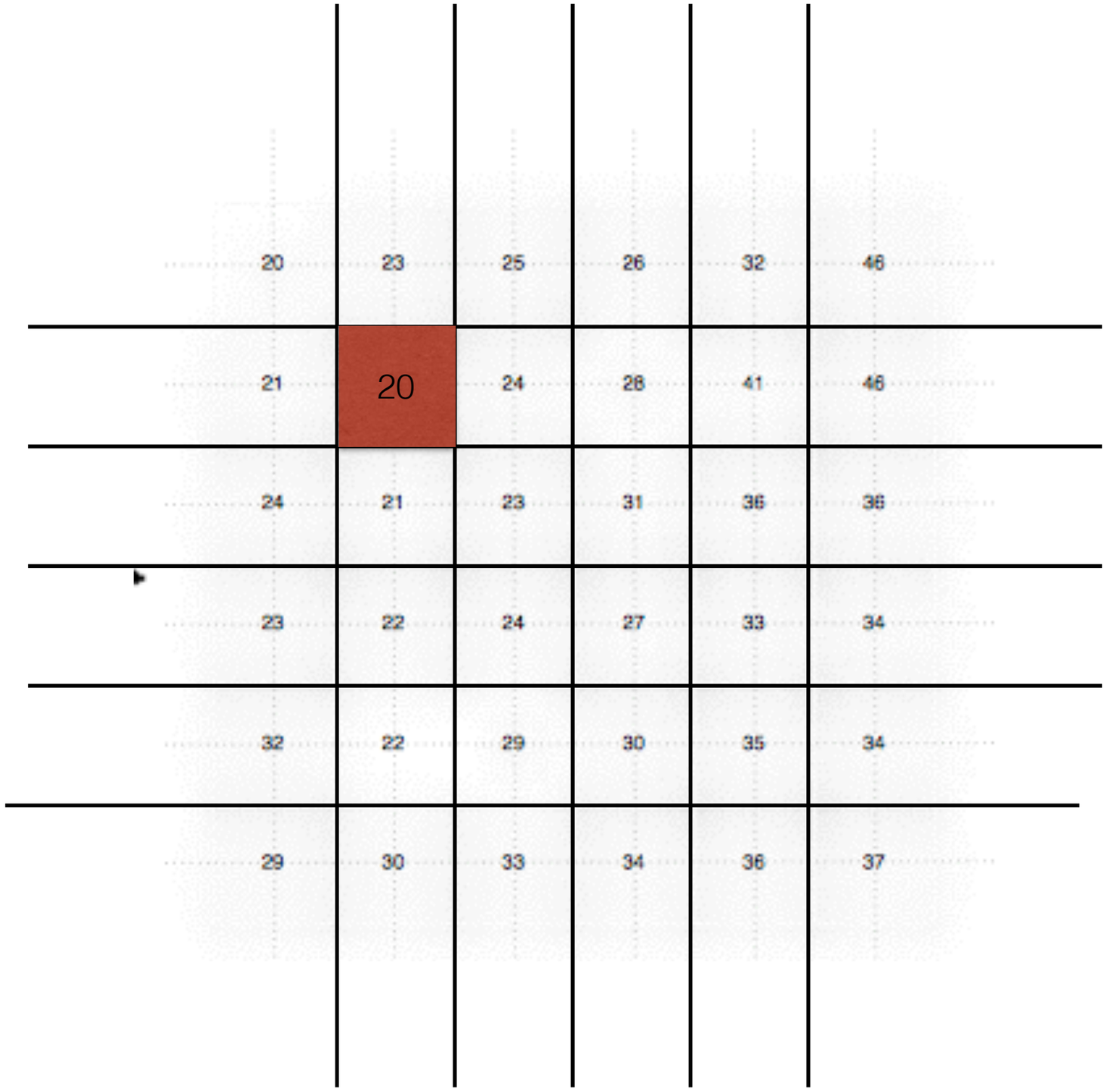
Nearest neighbor interpolation



Nearest neighbor interpolation

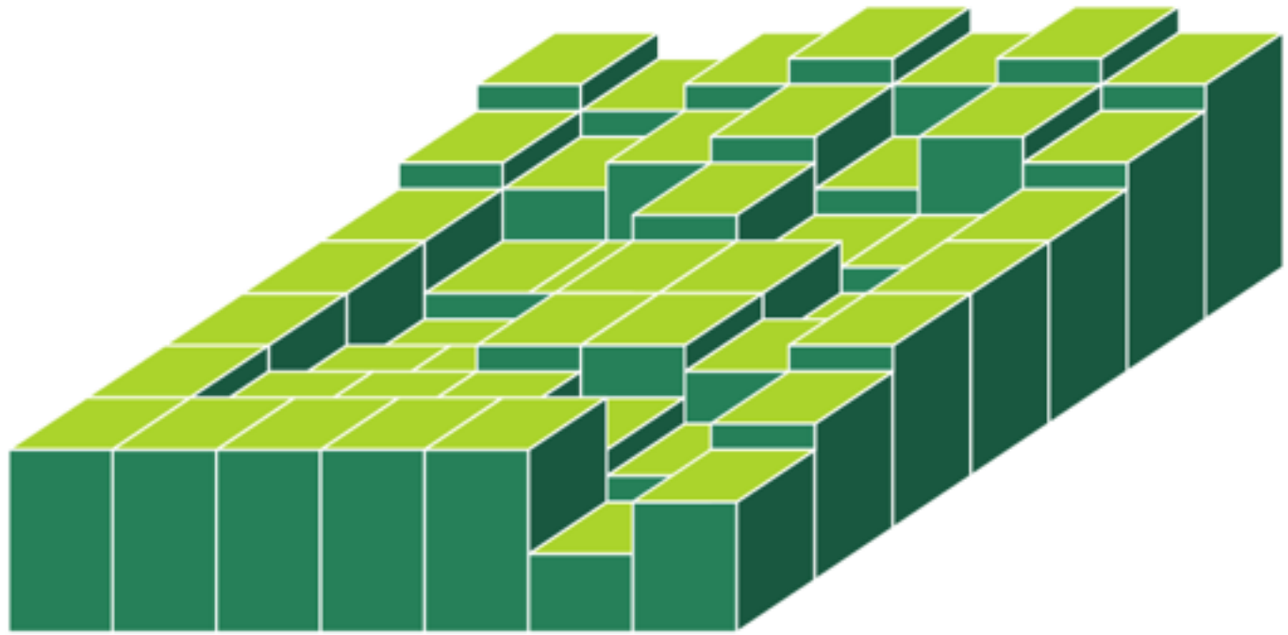
20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37

Nearest neighbor interpolation

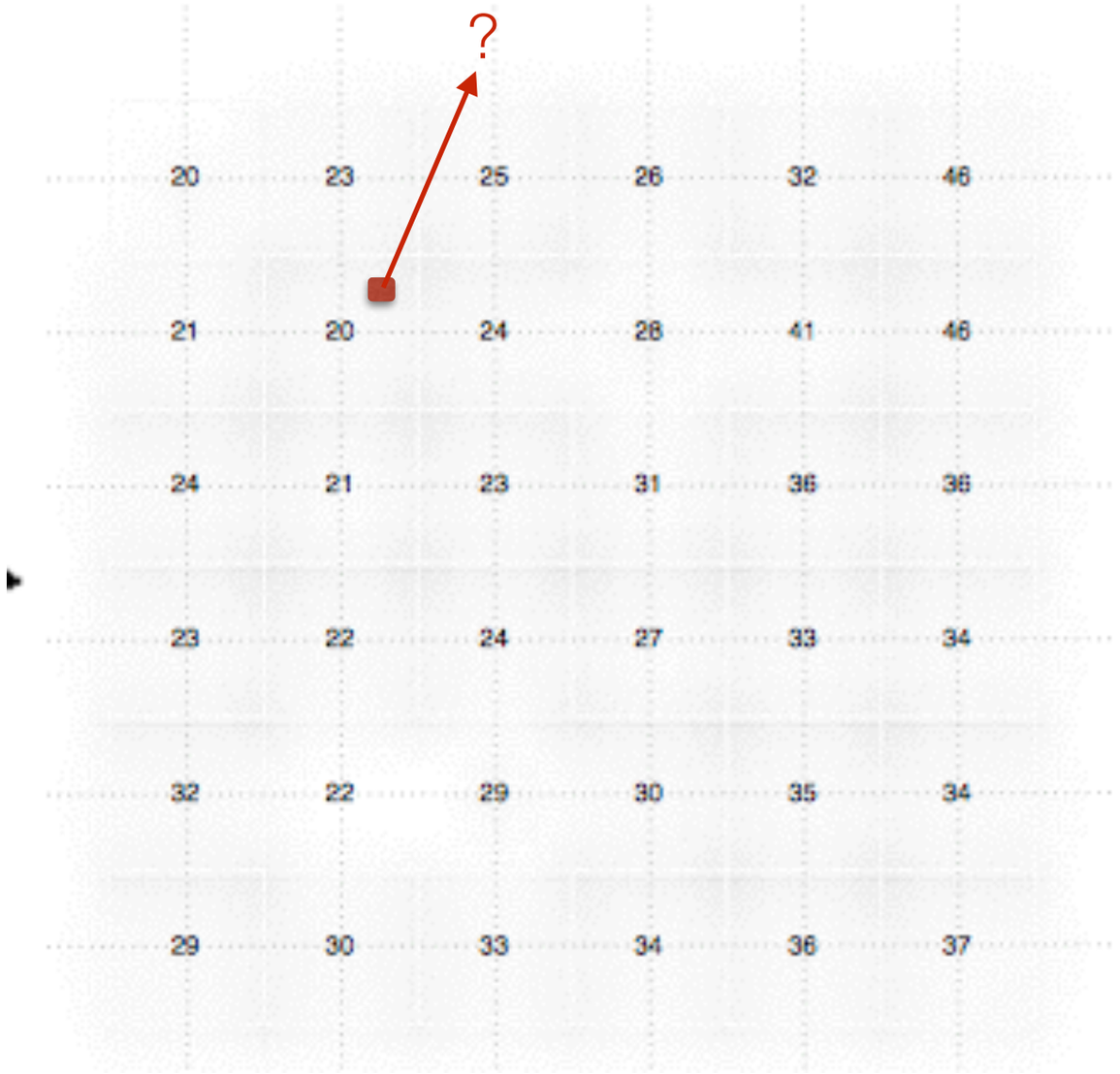


Grids with nearest neighbor interpolation

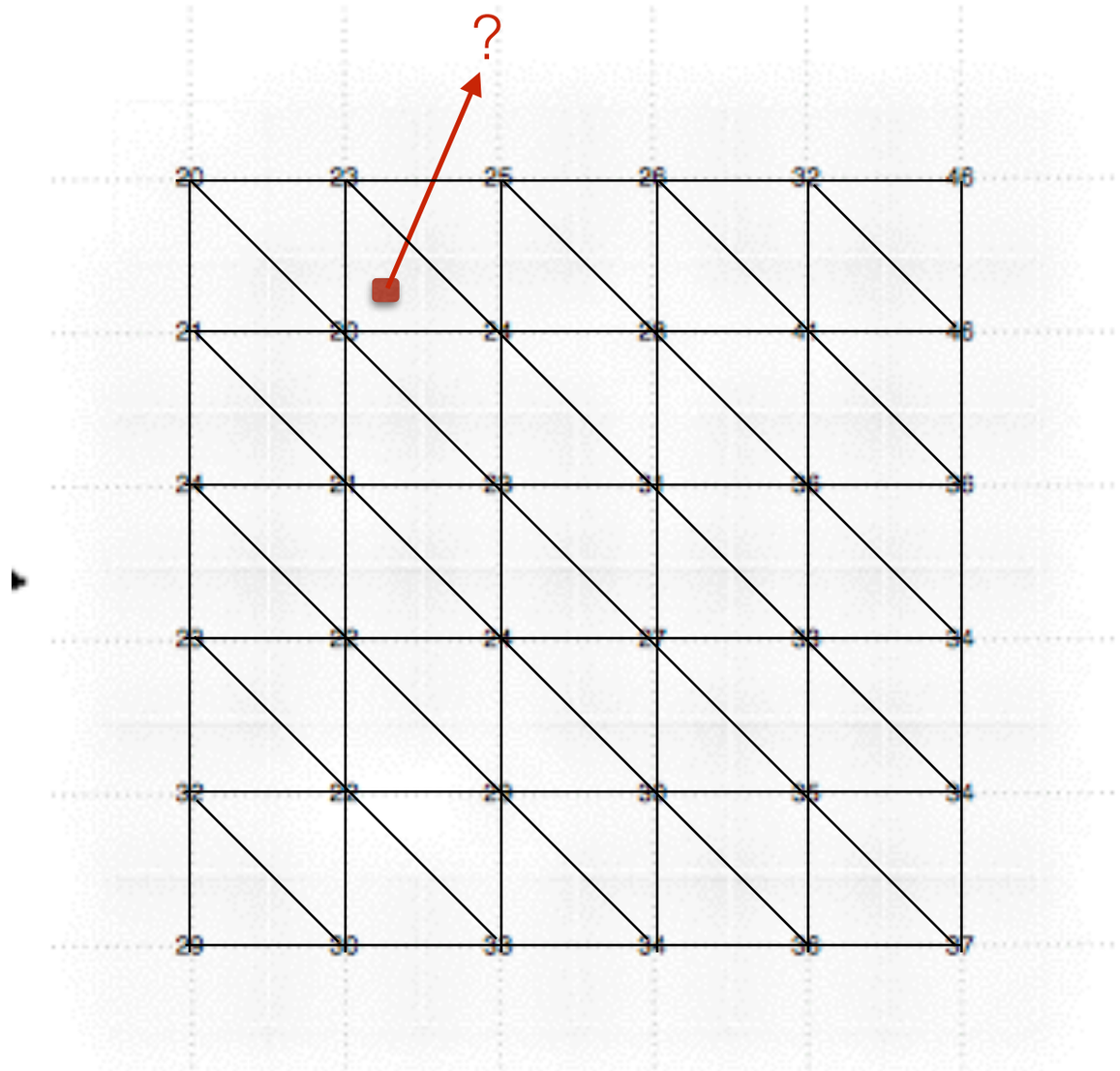
20	23	25	26	32	46
21	20	24	28	41	46
24	21	23	31	36	36
23	22	24	27	33	34
32	22	29	30	35	34
29	30	33	34	36	37



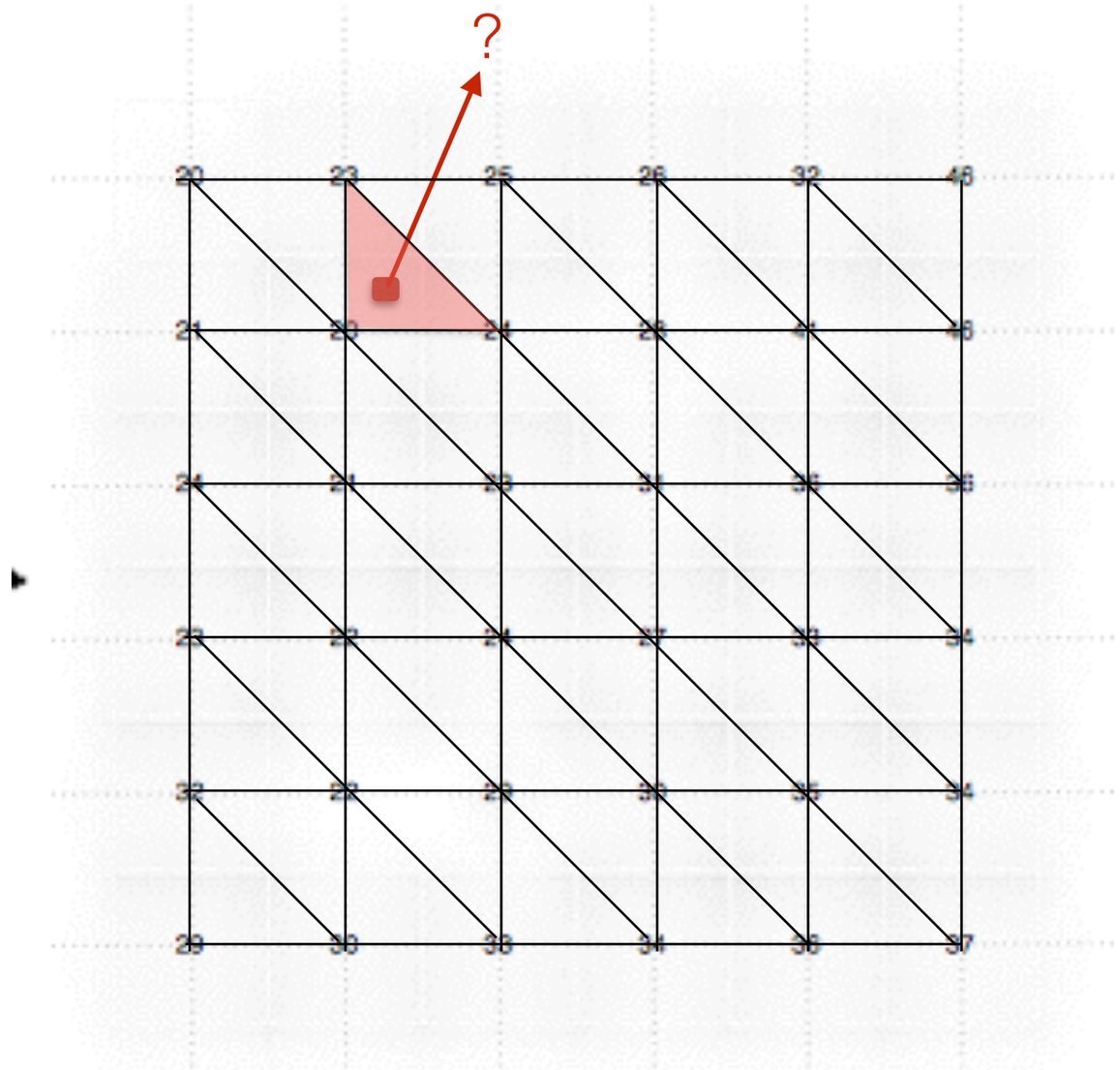
Linear interpolation



Linear interpolation

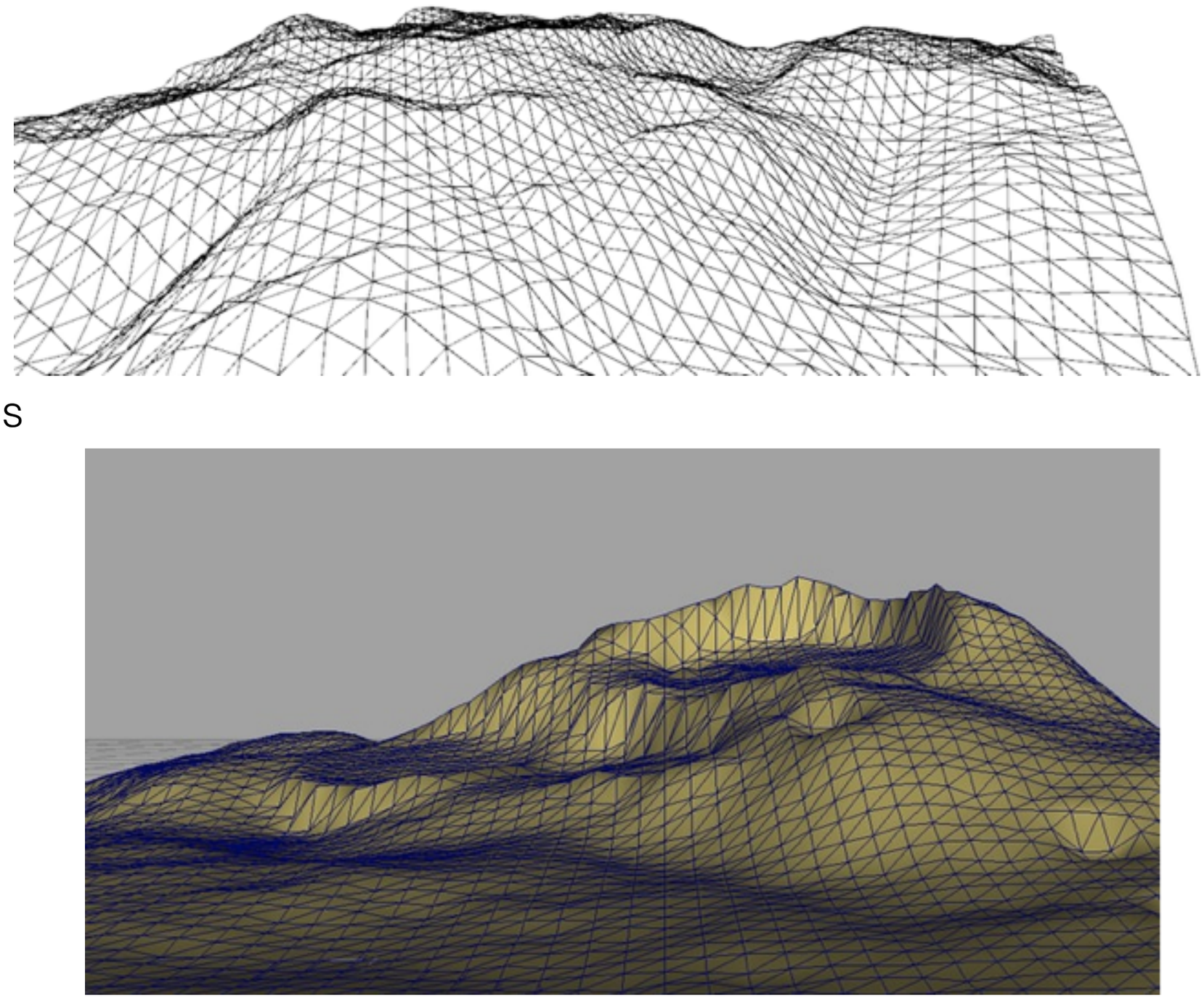


Linear interpolation



Grids with linear interpolation

Terrain: mesh of triangles on the grid points



Grids

- Grid data easy to obtain from aerial imagery
  - image: grid of color pixels
  - SAR interferometry: by combining Synthetic Aperture Radar (SAR) images of the same area it is possible to generate elevation maps
  - massive amounts of aerial imagery available

Grid elevation data sources

- USGS
  - http://earthexplorer.usgs.gov

- SRTM 90m elevation data for entire world
  - http://srtm.csi.cgiar.org/SELECTION/inputCoord.asp
  - can download tiles anywhere in the world
  - SRTM 30m data available for the entire USA (50+GB)

- Grid elevation from LiDAR
  - below 2m resolution
  - Huge!

- Grids available in arc-ascii format

Grid arc-ascii format

```
ncols         391
nrows         472
xllcorner     271845
yllcorner     3875415
cellsize      30
NODATA_value  -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 781 773 766 761 756 751 745 738 730 723 716 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 775 767 759 754 750 746 741 735 728 721 714 709 705 700 696 694 693 692
691 690 691 692 696 701 707 714 722 728 732 733 731 726 721 718 718 722 729 737 746 755 761 764 762 760 760 759 754 748 741 733 725 718
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999
-9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999 770. . . . . . .
```

Exercise

Consider an area of 300km by 300km to be represented as a grid.

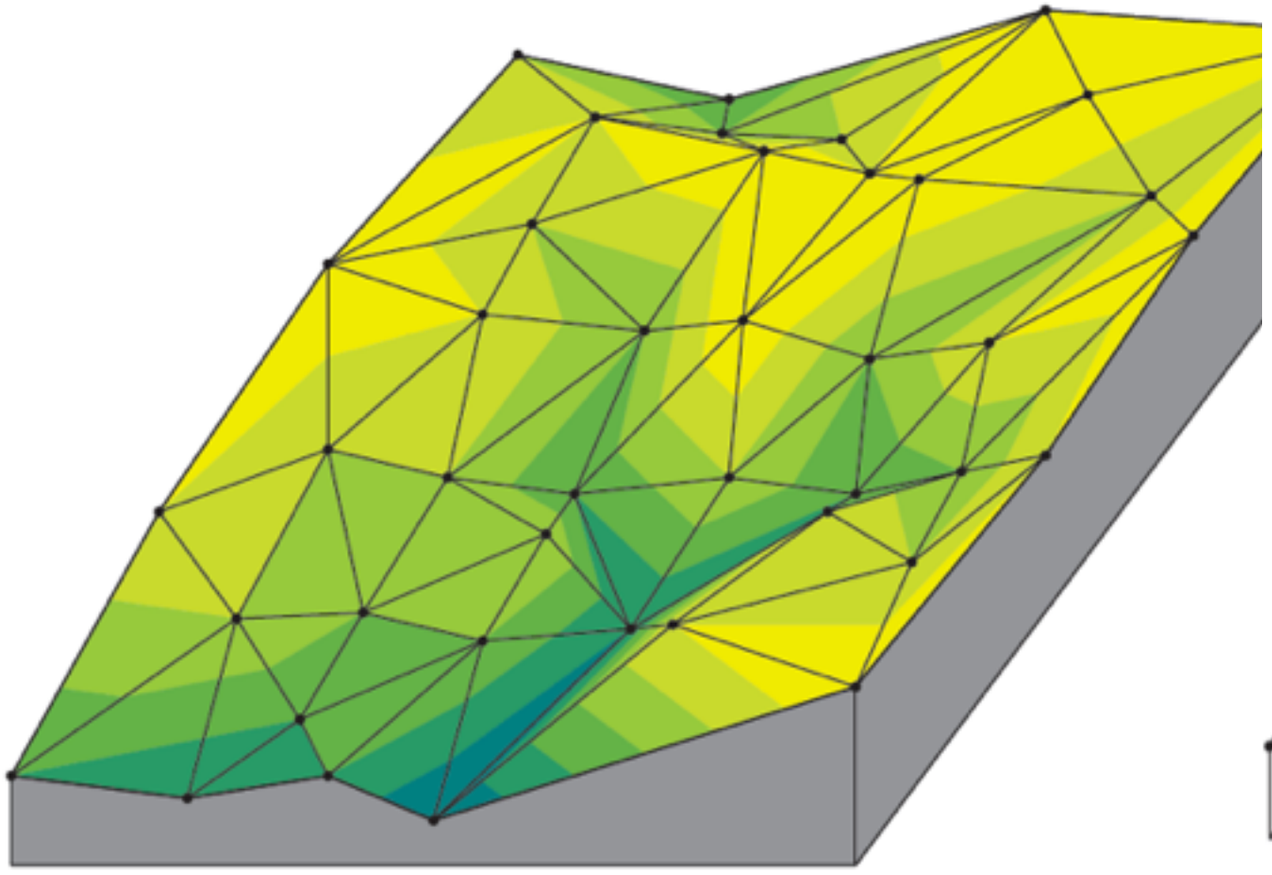
How big (how many points) is the grid at:

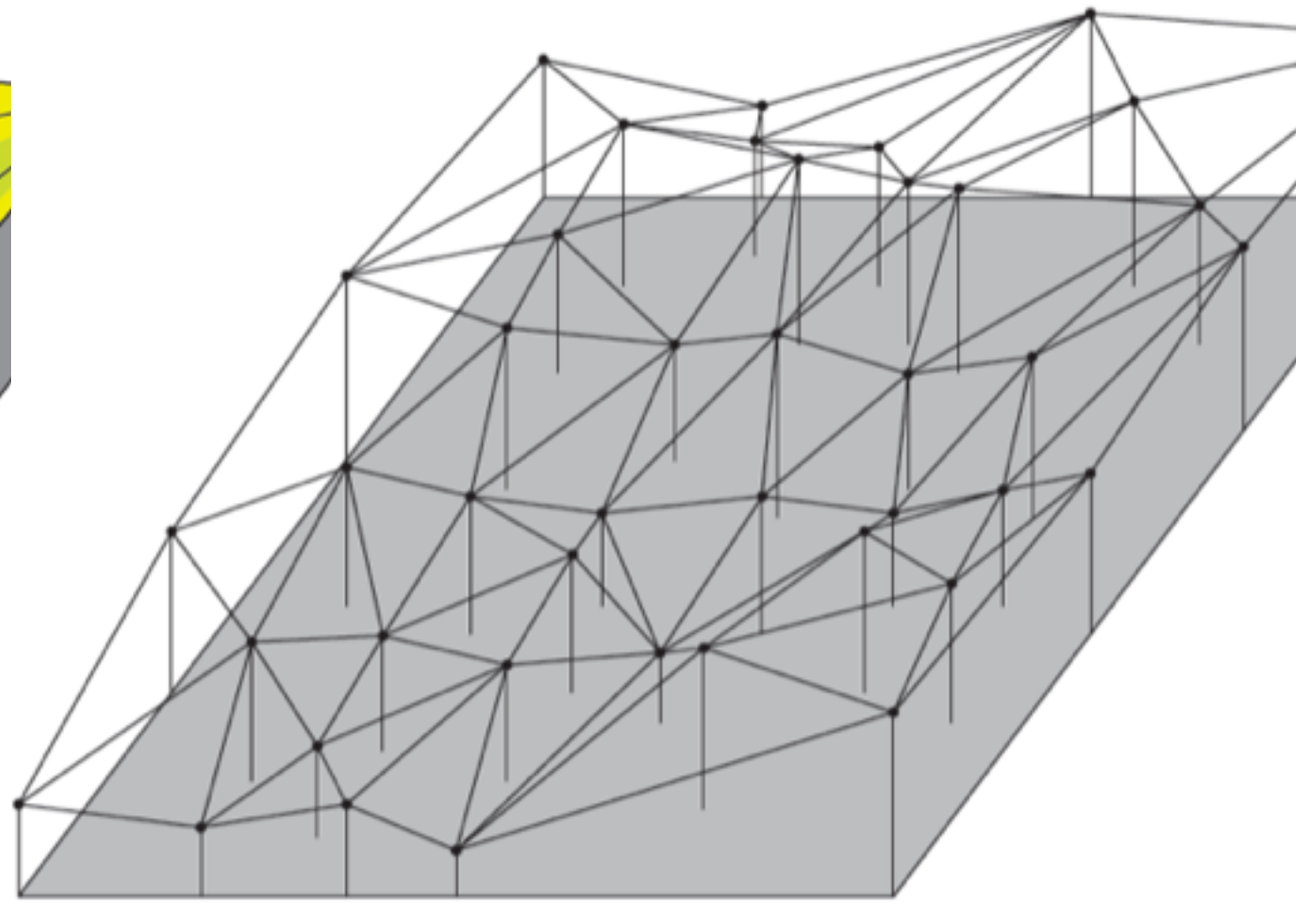
A. 100m resolution

B. 10m resolution

C. 1m resolution

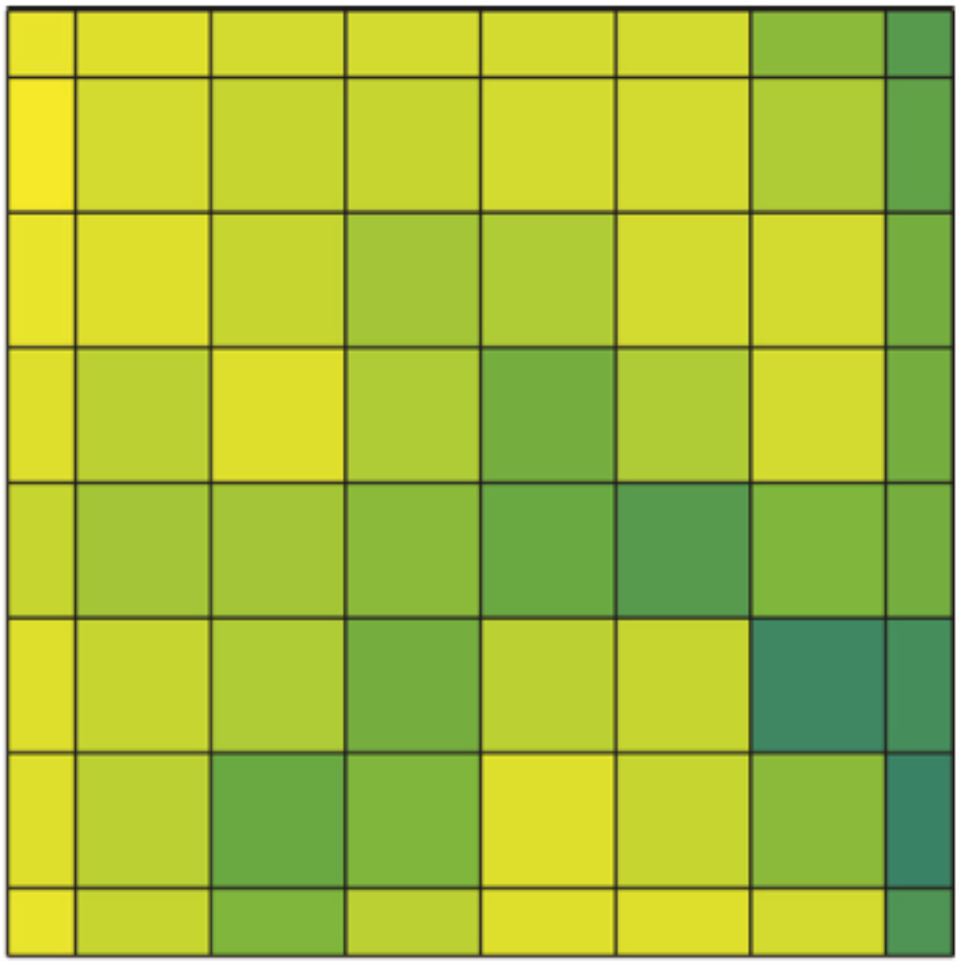
Terrain as a TIN (**t**riangulated **i**rregular **n**etwork)

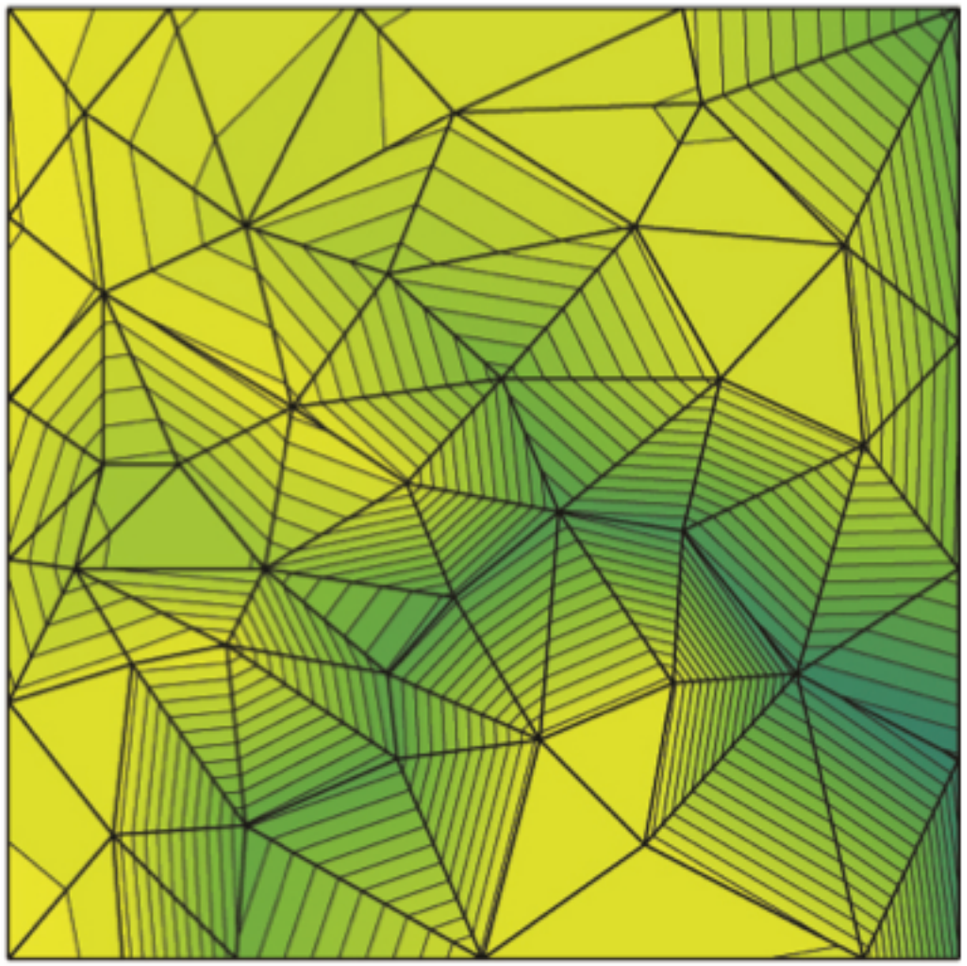




- Samples
  - points arbitrarily distributed, variable resolution
- Interpolation method
  - linear
- Representation
  - triangular mesh ← Data structure for triangular meshes?

Grid or TIN?





Grid

- Pros:
  - implicit topology
  - implicit geometry
  - simple algorithms
  - readily available in this form
- Cons:
  - uniform resolution ==> space waste
  - space becomes prohibitive as resolution increases

TIN

- Pros:
  - variable resolution
  - potentially space efficient
- Cons:
  - representation ???

    How do we store a TIN ???
  - More complicated algorithms

Digital terrain models: Grid or TIN ?

Assume we have a grid for an area of 300km by 300km at 1m resolution. The elevation values are represented as floating point numbers.

A. How much space does the grid use, in GB?

B. Assume the grid undergoes a process of simplification, so that 90% of the grid points are eliminated, leaving 10% of the points. These points are represented as a TIN. How much space (in GB) does the TIN need?

Digital terrain models: Grid or TIN ?

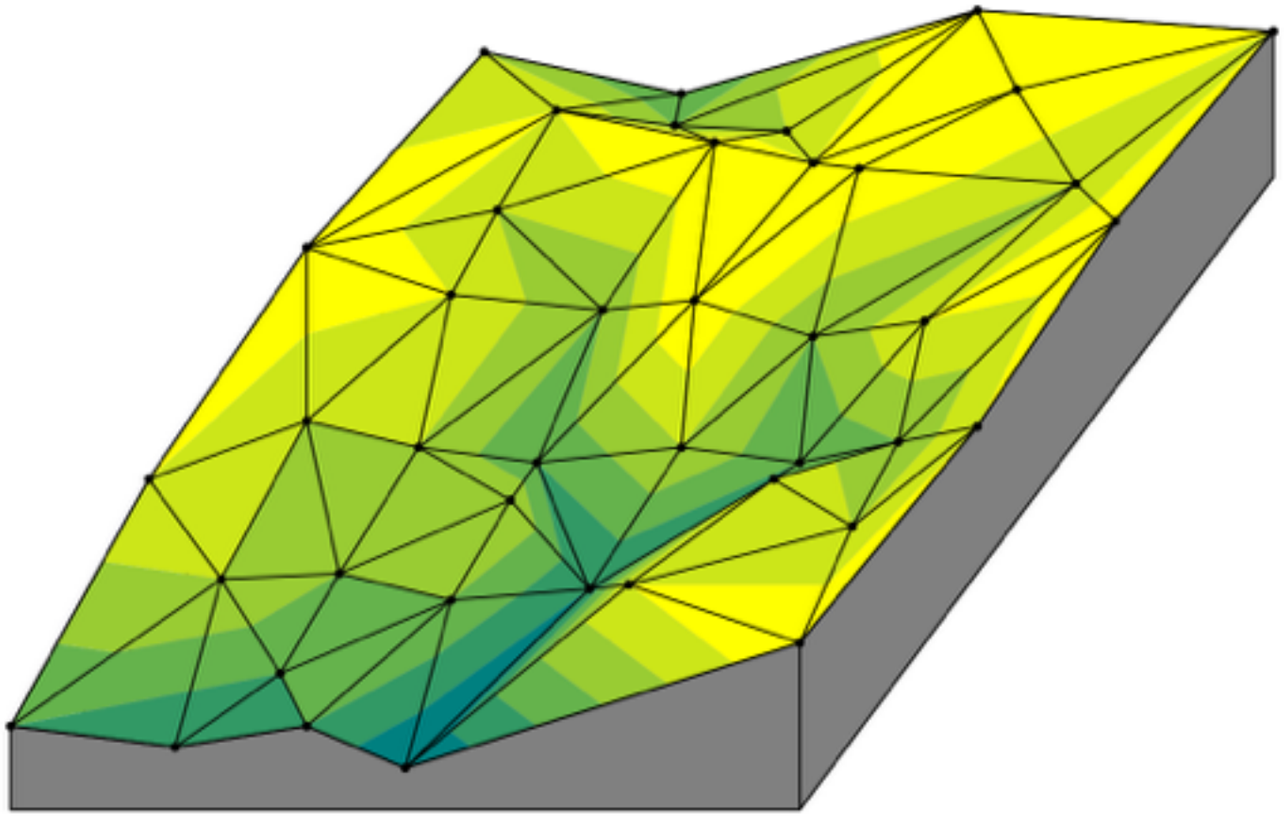
Assume we have a grid for an area of 300km by 300km at 1m resolution. The elevation values are represented as floating point numbers.

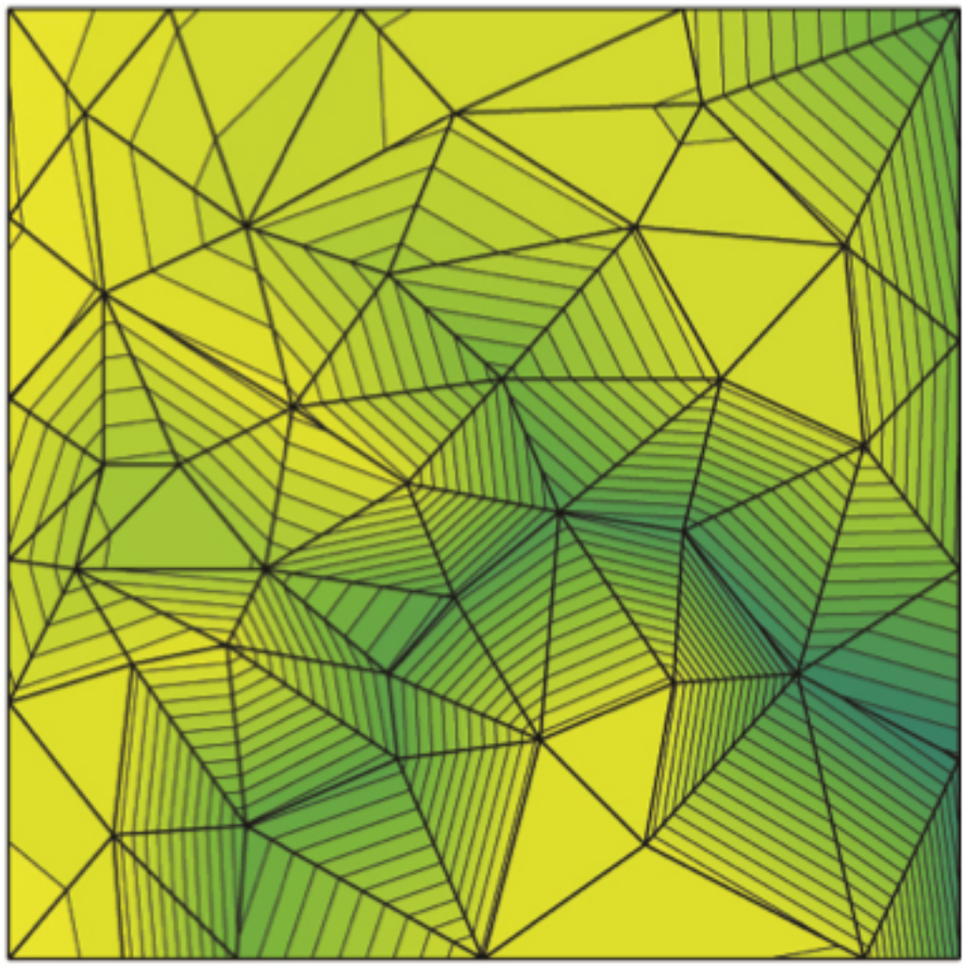
A. How much space does the grid use, in GB?

B. Assume the grid undergoes a process of simplification, so that 90% of the grid points are eliminated, leaving 10% of the points. These points are represented as a TIN. How much space (in GB) does the TIN need?

- We need to know
  - How do we store a TIN
  - How many triangles and edges as function of number of points

Data structures for TINs





What do we expect to do on a TIN?

- walk along an edge/triangle path
- given an edge, find the two faces that are adjacent to this edge
- walk along the boundary of a face (triangle)
- find all edges and all triangles incident to a point

A good data structure for TINs should do all these fast

(Topological) data structures for TINs

Edge-based

- arrays of vertices, edges and triangles
- every vertex stores:
  - its coordinates
- every edge stores:
  - 2 references to its adjacent vertices
  - 2 references to its adjacent triangles
- every triangle stores:
  - 3 references to its 3 edge

Triangle-based

- arrays of vertices and triangles (edges are not stored explicitly)
- every vertex stores:
  - its coordinates
- every triangles stores:
  - 3 references to its incident vertices
  - 3 references to its adjacent triangles

- Note: CGAL uses triangle-based

- These are simplified versions of more general structures

(Topological) data structures for TINs

Edge-based

- arrays of vertices, edges and triangles
- every vertex stores:
  - its coordinates
- every edge stores:
  - 2 references to its adjacent vertices
  - 2 references to its adjacent triangles
- every triangle stores:
  - 3 references to its 3 edge

Triangle-based

- arrays of vertices and triangles (edges are not stored explicitly)
- every vertex stores:
  - its coordinates
- every triangles stores:
  - 3 references to its incident vertices
  - 3 references to its adjacent triangles

geometry

- Note: CGAL uses triangle-based

- These are simplified versions of more general structures

(Topological) data structures for TINs

Edge-based

- arrays of vertices, edges and triangles
- every vertex stores:
  - its coordinates
- every edge stores:
  - 2 references to its adjacent vertices
  - 2 references to its adjacent triangles
- every triangle stores:
  - 3 references to its 3 edge

topology

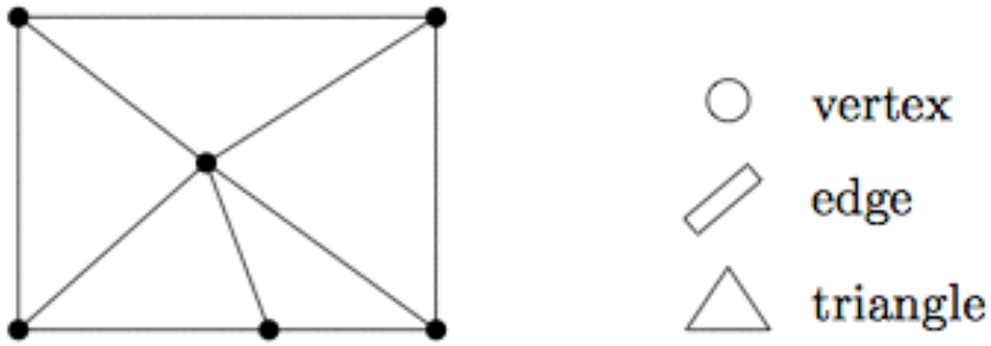
Triangle-based

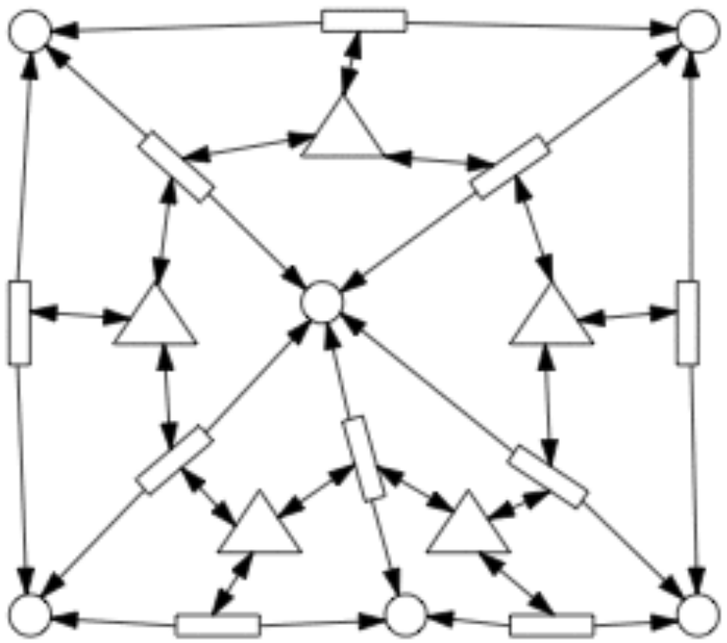
- arrays of vertices and triangles (edges are not stored explicitly)
- every vertex stores:
  - its coordinates
- every triangles stores:
  - 3 references to its incident vertices
  - 3 references to its adjacent triangles

- Note: CGAL uses triangle-based

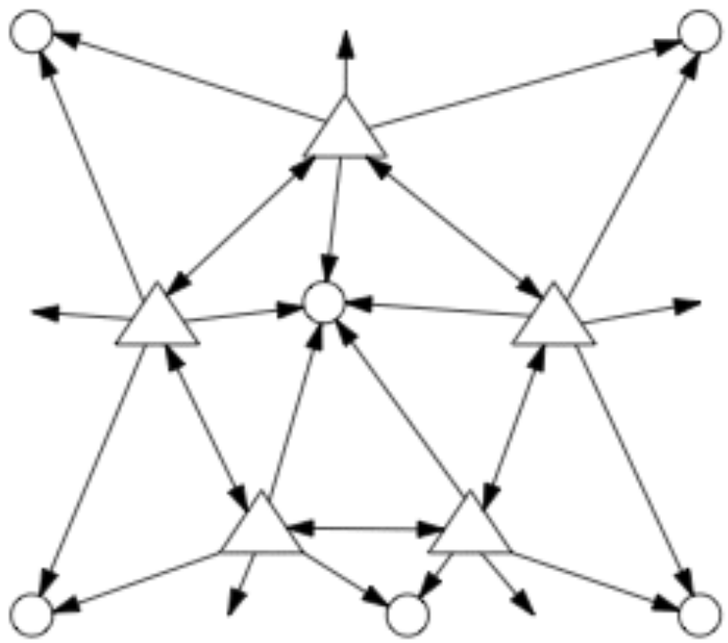
- These are simplified versions of more general structures

Data structures for TINs





edge-based



triangle-based

Assume we have a triangulation with n points.

How much memory do we need to store it in a topological structure?

- edge-based
- triangle-based

Denote

- e = number of edges
- f = number of triangles (faces)

Assume we have a triangulation with n points.

How much memory do we need to store it in a topological structure?

- edge-based
- triangle-based

Denote

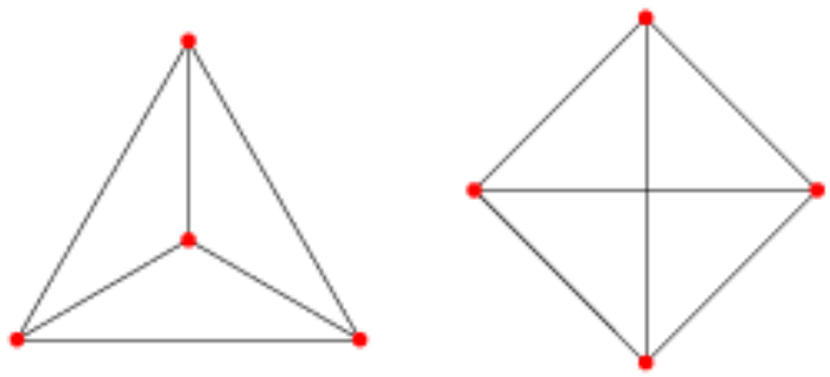
- e = number of edges
- f = number of triangles (faces)

Turns out there is a formula that gives e and f as function of n!

Detour

Definition:

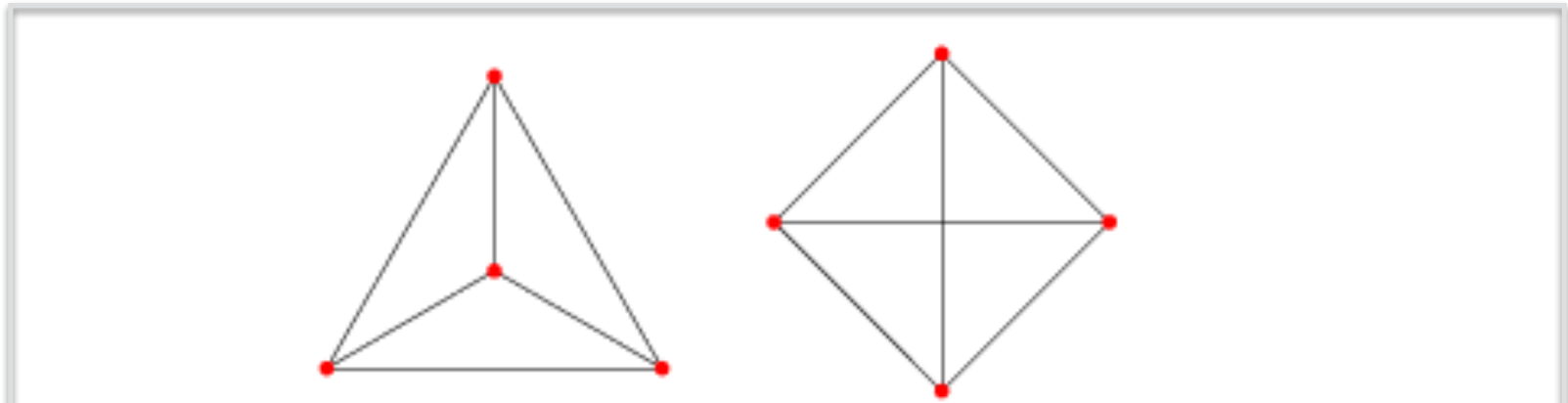
A graph is called **planar** if it can be drawn in the plane such that no two edges intersect except at their endpoints. Such a drawing is called a **planar embedding** of the graph.



Detour

Definition:

A graph is called **planar** if it can be drawn in the plane such that no two edges intersect except at their endpoints. Such a drawing is called a **planar embedding** of the graph.



Two drawings of the same graph.

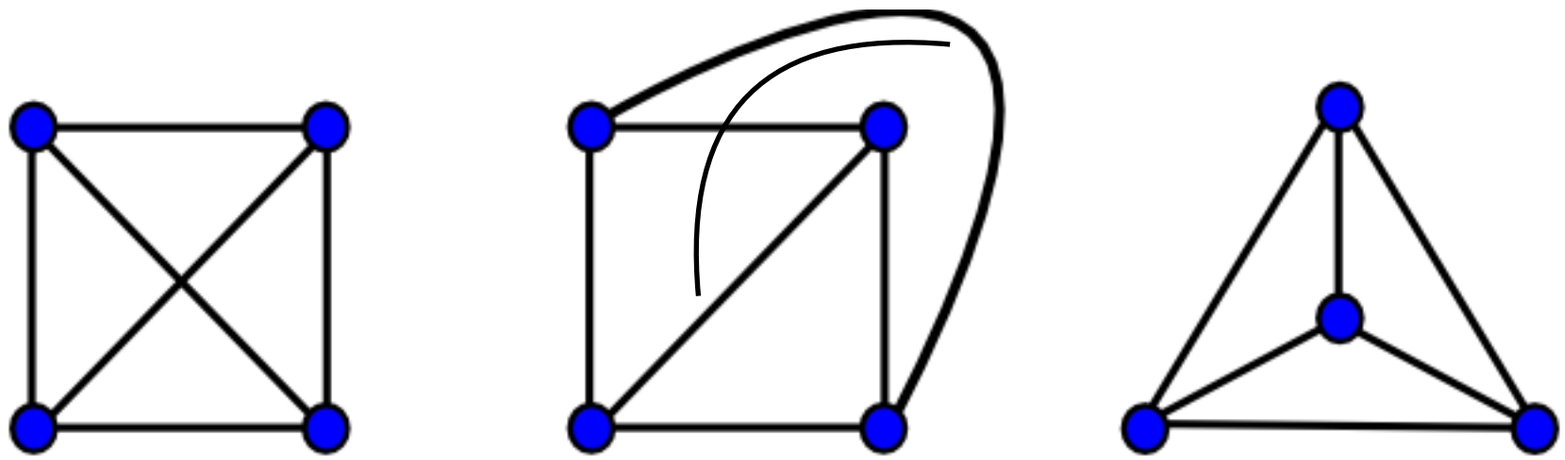
Since there exists a planar embedding, the graph is planar.

Note that it is possible to draw non-planar embeddings of planar graphs.

Detour

Definition:

A graph is called **planar** if it can be drawn in the plane such that no two edges intersect except at their endpoints. Such a drawing is called a **planar embedding** of the graph.

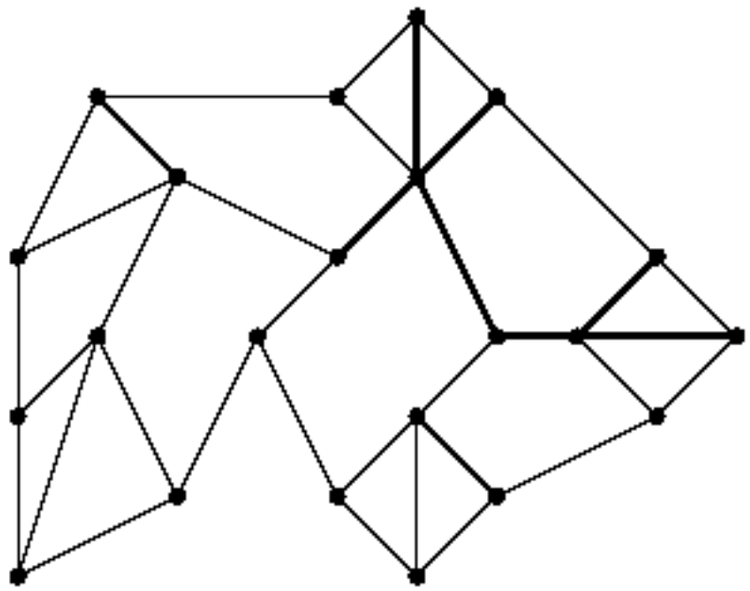


http://people.hofstra.edu/geotrans/eng/methods/img/planarnonplanar.png

Note: Edges can be represented as simple curves in the drawing

Detour

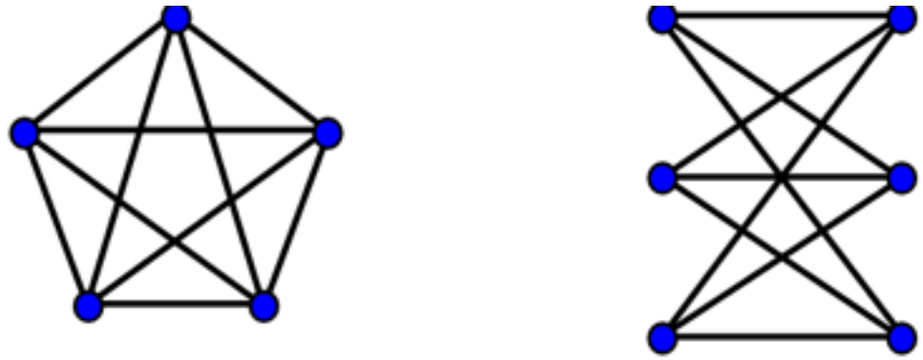
A planar graph introduces a subdivision of the plane into regions called faces, which are polygons bounded by the graph's edges.



Detour

Some results:

- Any planar graph has a planar straight-line drawing where edges do not intersect [Fary's theorem].
- A graph is planar iff it has no subgraphs isomorphic with K5 or K3,3 [Kuratowski's theorem].



- A graph is planar iff it has a dual graph.
- Any planar graph has at least one vertex of degree <= 5.
- There are a number of efficient algorithms for planarity testing that run in $o(n^3)$, but are difficult to implement.

Detour

Euler formula:

The following relation exists between the number of edges, vertices and faces in a connected planar graph: $v - e + f = 2$.

- Notes:
  - For c connected components: $v - e + f - c = 1$
  - Also true for any convex polyhedral surface in 3D
  - The Euler characteristic $X = v - e + f$ is an invariant that describes the shape of space; it is $X=2$ for planar graphs, convex polyhedra,etc; can be extended to topological spaces.

n = nb of points in the TIN

Detour

- From Euler formula we know n-e+f = 2
- Furthermore, each triangle has 3 edges and each edge is in precisely 2 triangles (assuming the outside face is a triangle). This means 3f = 2e.
- We get:
  - the number of faces in a triangulation with n vertices is f = 2n-4
  - the number of edges in a triangulation with n vertices is e = 3n-6
- If the outside face is not triangulated it can be shown that
  - $e < 3n-6$, $f < 2n-4$
- Intuition: Given n points, the planar graph with largest number of edges and faces is a complete triangulation.

Theorem:

A triangulation with n vertices has at most 3n-6 edges and at most 2n-4 faces.

Assume we have a triangulation with n points.

How much memory do we need to store it in a topological structure?

- edge-based
- triangle-based

Digital terrain models: Grid or TIN ?

Assume we have a grid for an area of 300km by 300km at 1m resolution. The elevation values are represented as floating point numbers.

A. How much space does the grid use, in GB?

B. Assume the grid undergoes a process of simplification, so that 90% of the grid points are eliminated, leaving 10% of the points. These points are represented as a TIN. How much space (in GB) does the TIN need?

- We need to know
  - How do we store a TIN
  - How many triangles and edges as function of number of points

Grid or TIN?

- Pros:
  - implicit topology
  - implicit geometry
  - simple algorithms
  - readily available in this form
- Cons:
  - uniform resolution ==> space waste
  - space becomes prohibitive as resolution increases

Grid

- Pros:
  - variable resolution
  - potentially space efficient
- Cons:
  - need to built and store topology
  - stored topology takes space
  - more complex programming (pointers..);

TIN