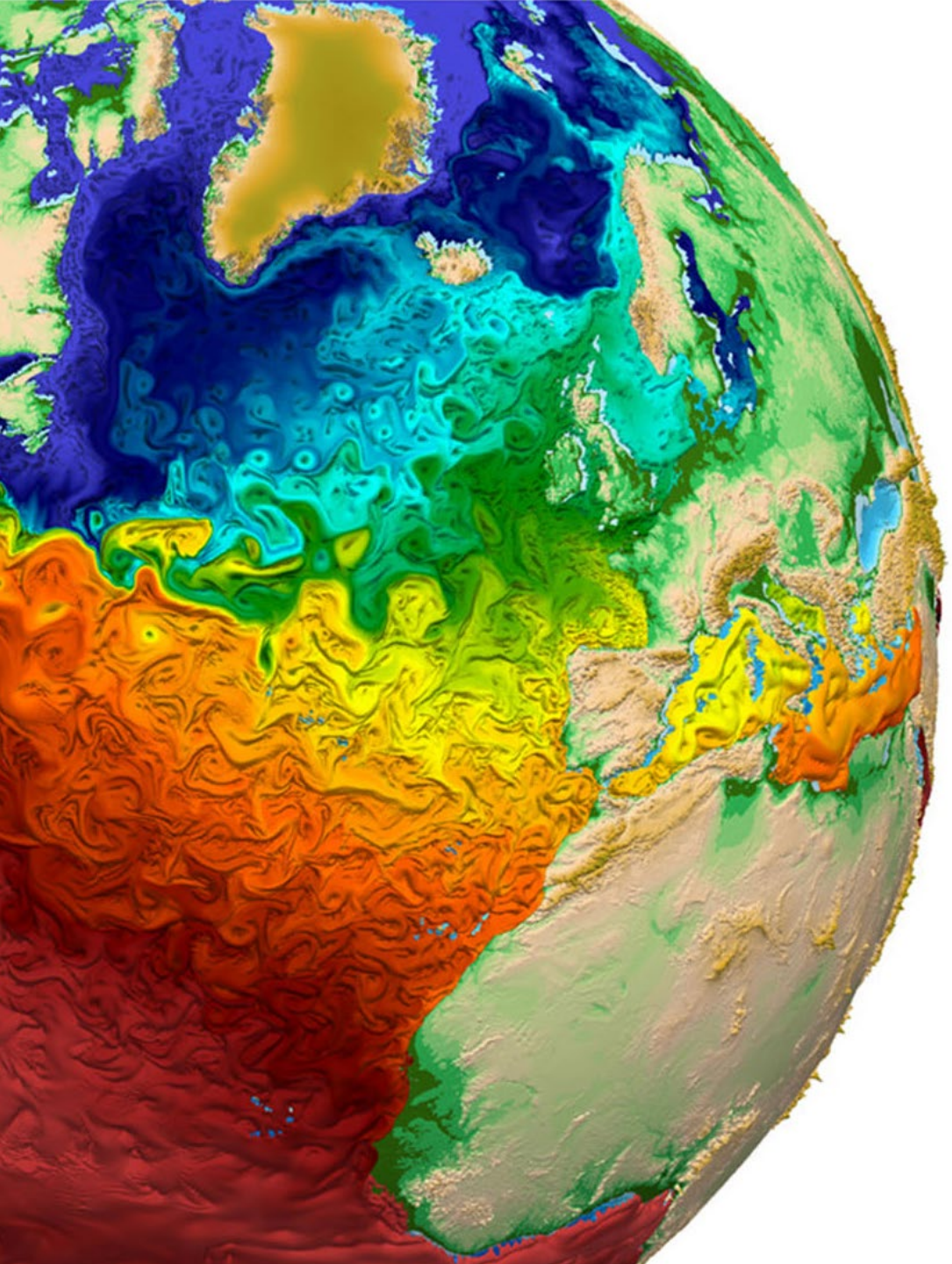


Domain Modeling

Terrain Generation for GIS

Scientific Visualization
Professor Eric Shaffer

Acknowledgment

We will look at a proposed method for generating

- Raster DEMs (Digital Elevation Models)
- TINs (Triangulated Irregular Networks)

From LiDAR (laser imaging, detection, and ranging) data

The proposed method is from

Generating Raster DEM from Mass Points Via TIN Streaming. Isenburg M., Liu Y., Shewchuk J., Snoeyink J., Thirion T. (2006) In: Raubal M., Miller H.J., Frank A.U., Goodchild M.F. (eds) Geographic Information Science. GIScience 2006. Lecture Notes in Computer Science, vol 4197. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11863939_13

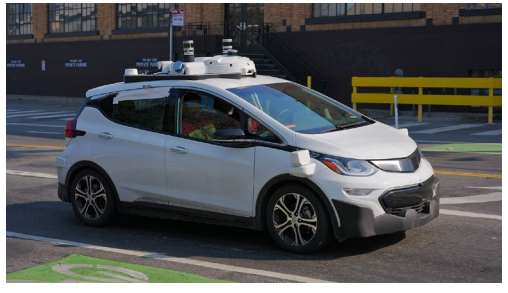
Content in this presentation comes courtesy of Martin Isenburg's publicly available slides

ILLINOIS

LiDAR

Lidar is a method for measuring distances (ranging) by illuminating the target with laser light and measuring the reflection with a sensor. Differences in laser return times and wavelengths can then be used to make digital 3-D representations of the target. It has terrestrial, airborne, and mobile applications.

- Wikipedia

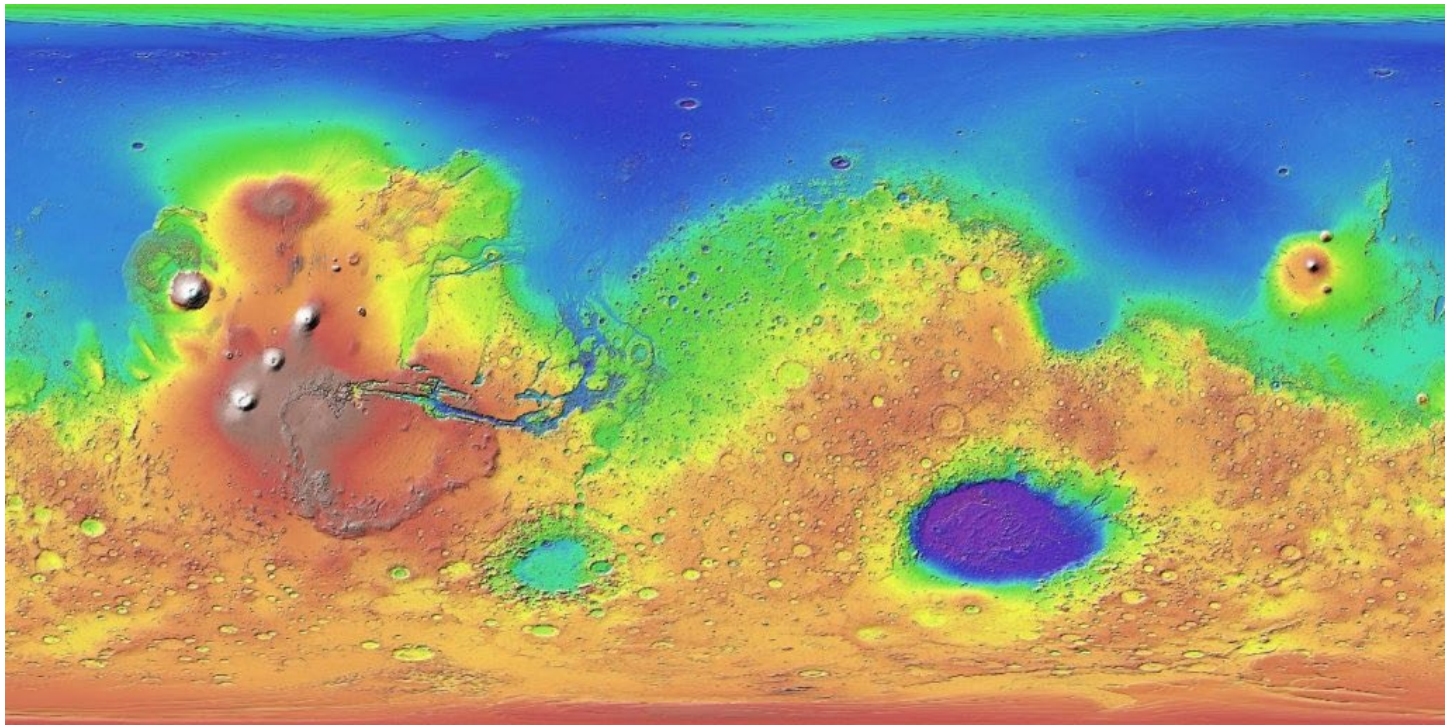


ILLINOIS

DEMs

A digital elevation model (DEM) is a raster data set

A regular grid of elevations arranged by column and row



Mars MGS MOLA Global Color Shaded Relief 463m v1

Visualization and DEMs:

- estimate normal vectors
  - generate shading
- visualize slope
- contour lines

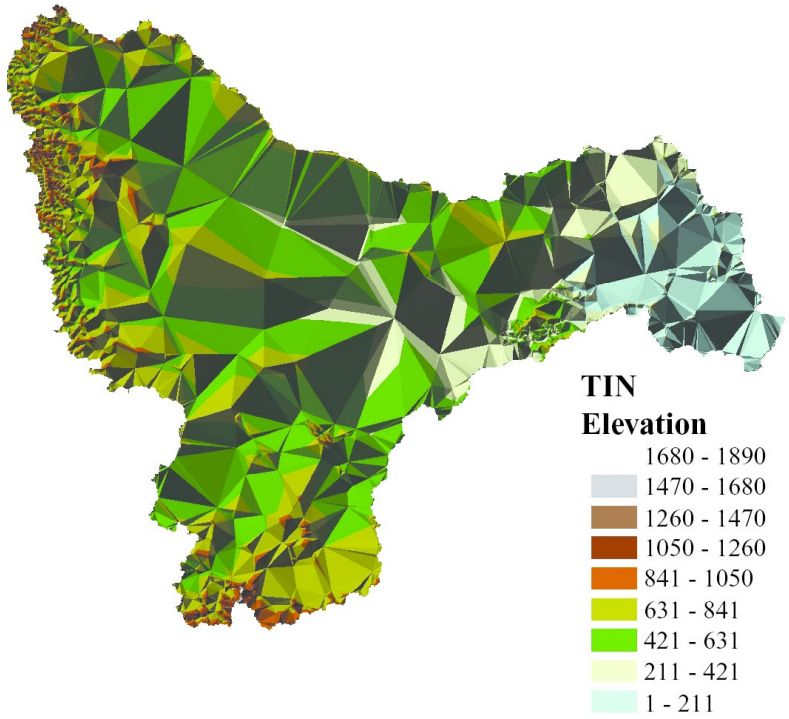
and more…

ILLINOIS

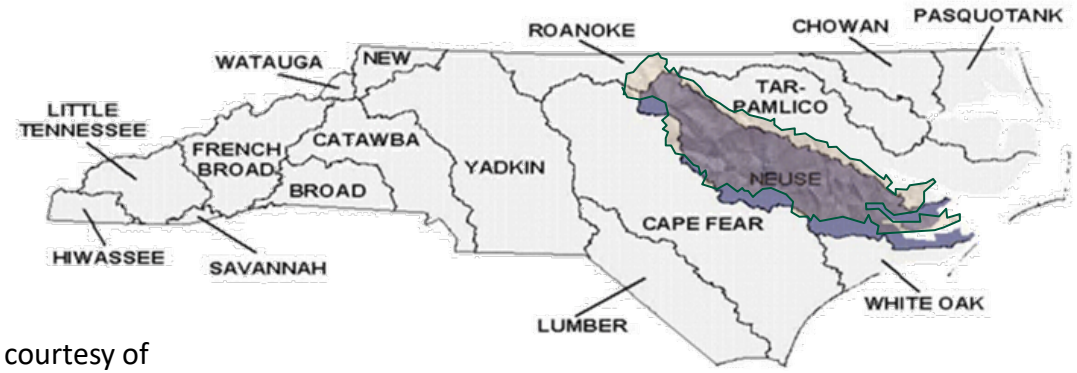
TINs

Triangulated Irregular Networks

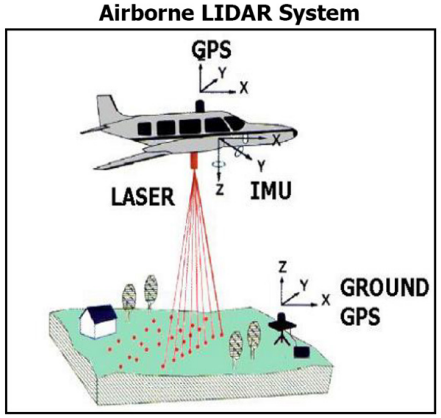
- Triangulated surface mesh of a terrain
- Irregular = unstructured, allows adaptive sampling of areas with rapidly changing elevations



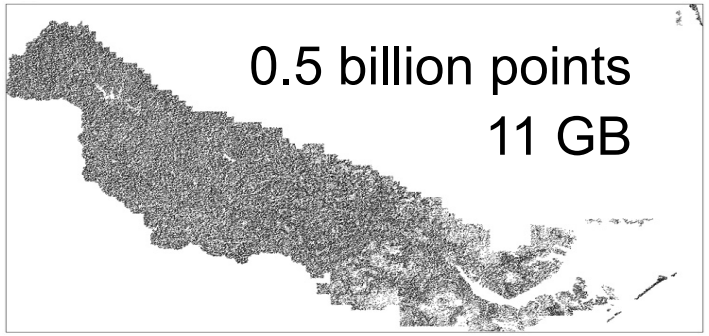
Aerial LiDAR Acquisition (~2005)



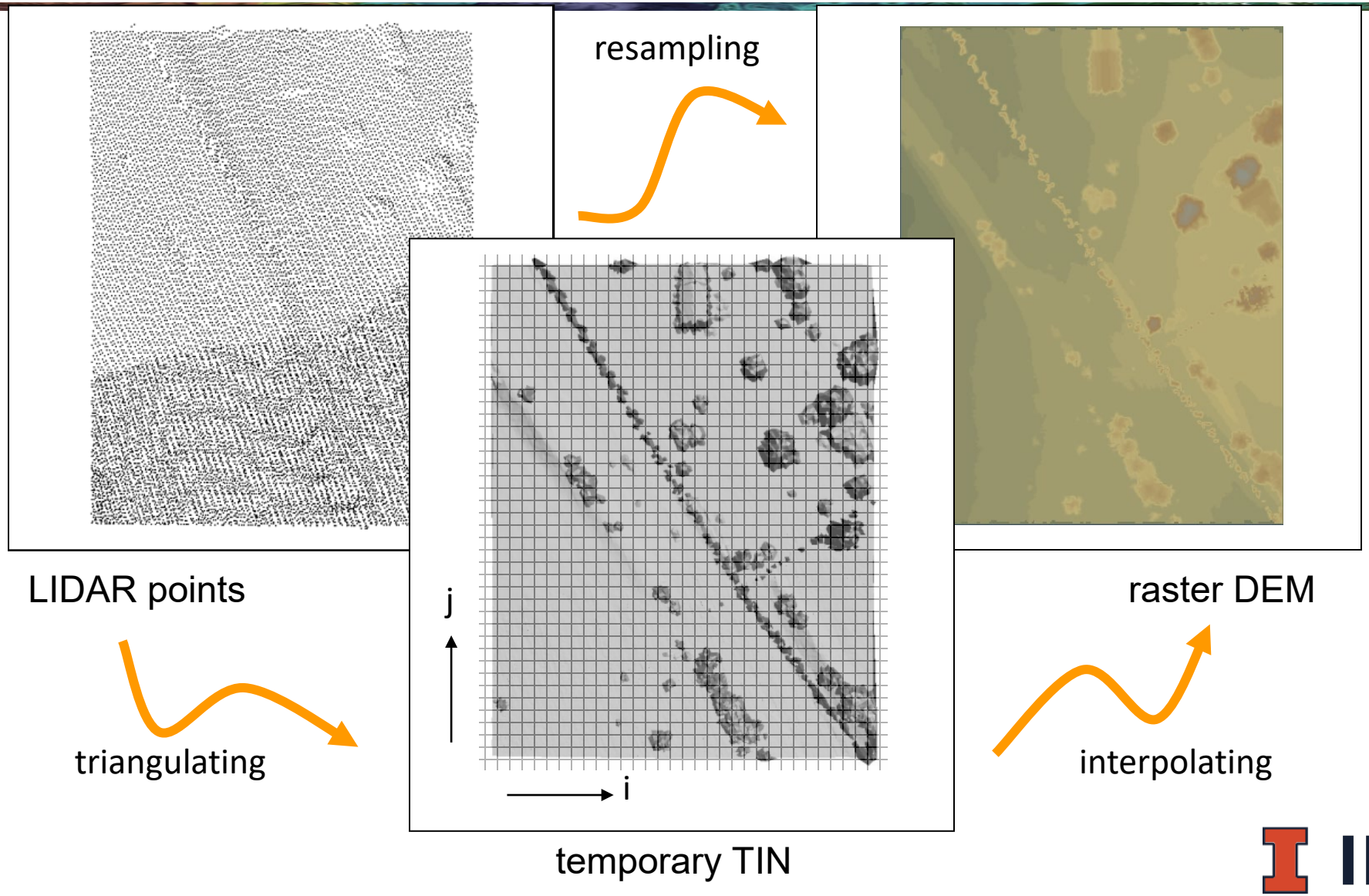
courtesy of
www.ncfloodmaps.com



Neuse-River Basin



Generating DEMs from LIDAR

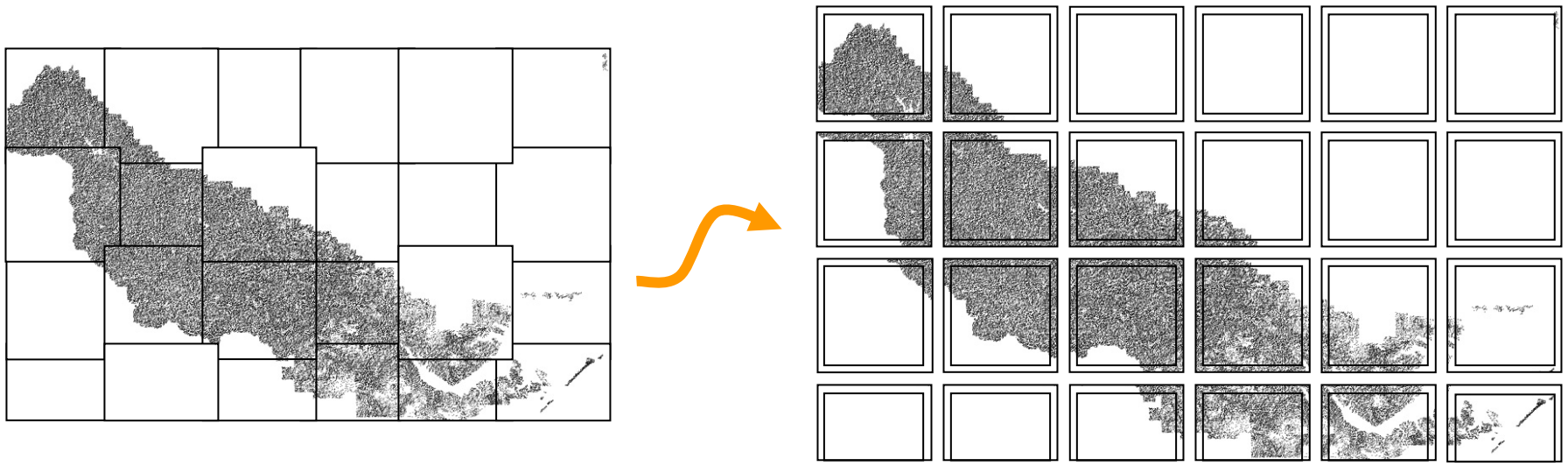


Goal: Make DEM Production from LiDAR Scalable

In 2005 popular GIS packages could process

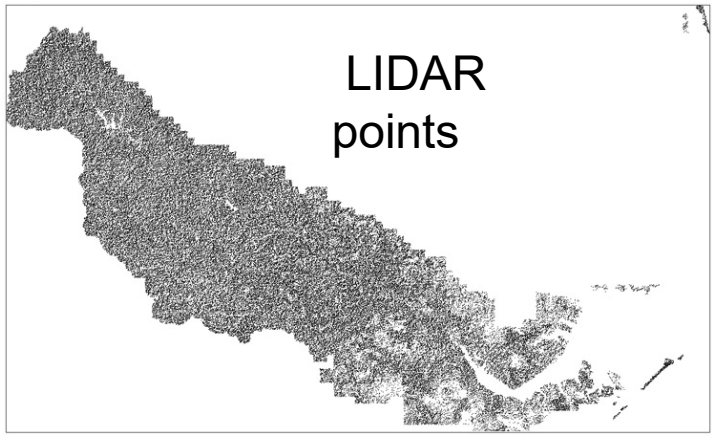
- ArcGIS 9.1 (limit: ~ 25 million points)
- QTModeller 4 (limit: ~ 50 million points)
- GRASS (limit: ~ 25 million points)

Popular strategy: break data into (overlapping) tiles

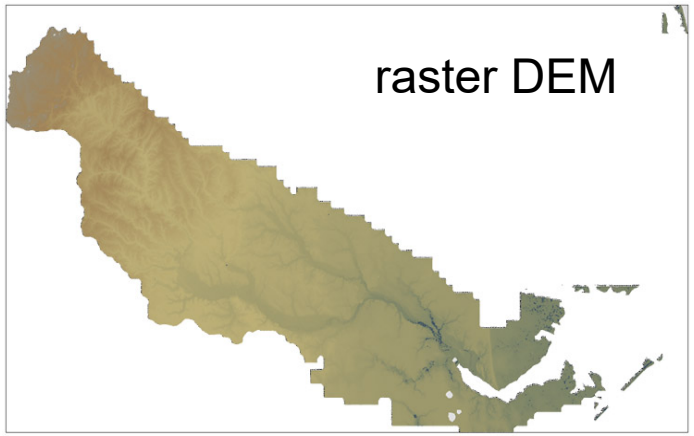


Some Results

500,141,313 Points
11 GB
(binary, xyz, doubles)



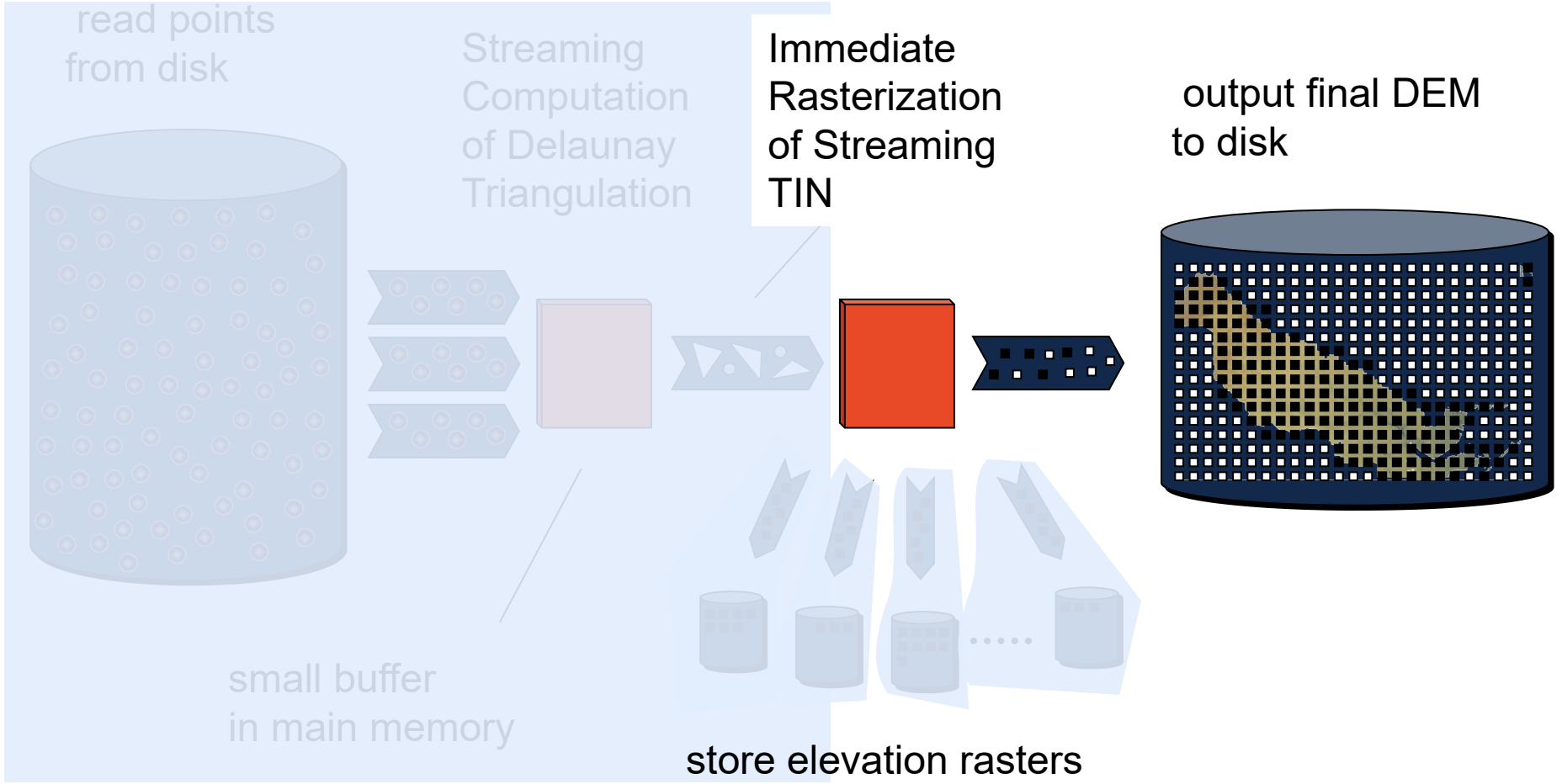
50,394×30,500 DEM
3 GB
(binary, BIL, 16 bit, 20 ft)



on a household laptop with two harddisks
in 67 minutes
64 MB of main memory

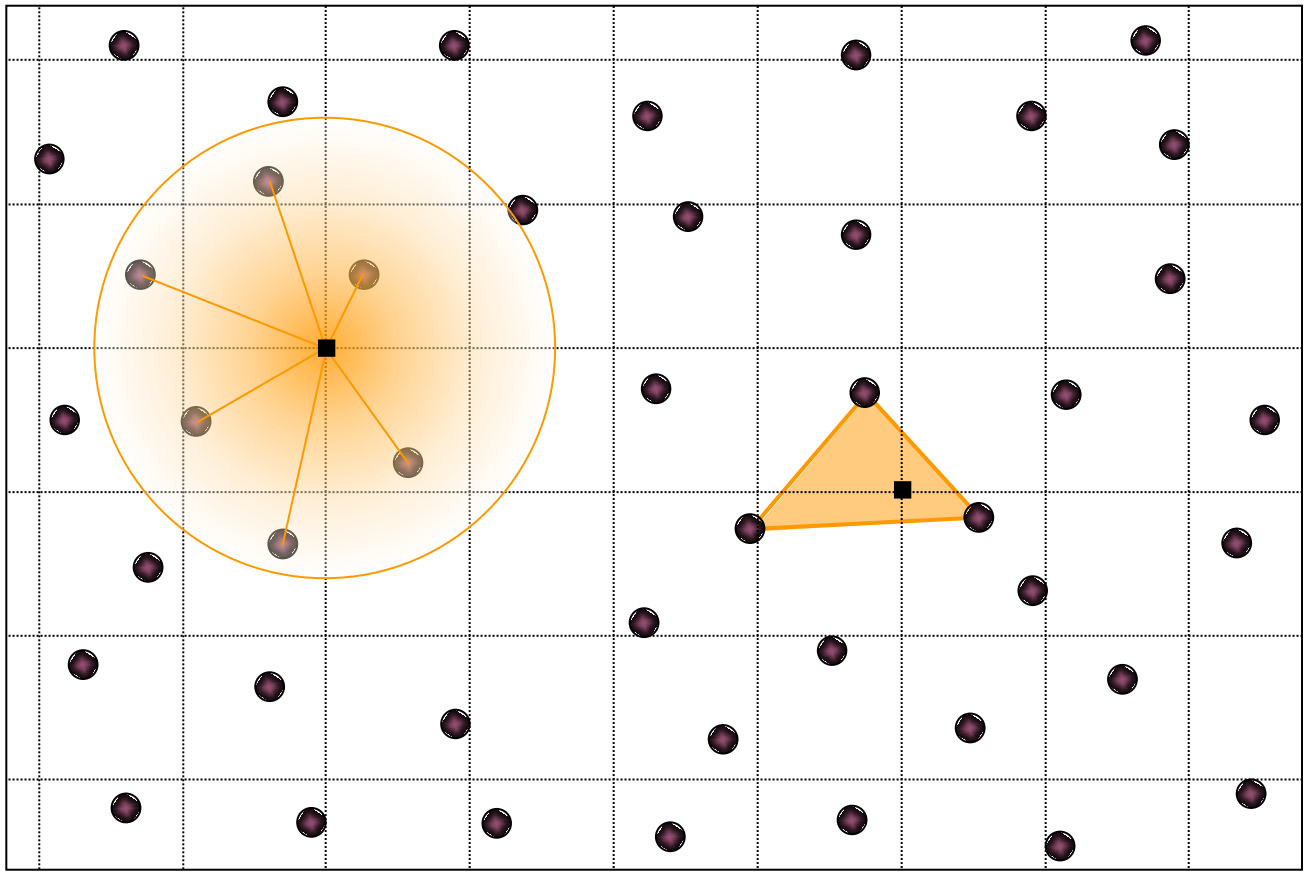
ILLINOIS

DEM Generation via TIN Streaming



TIN to DEM: Interpolation Methods

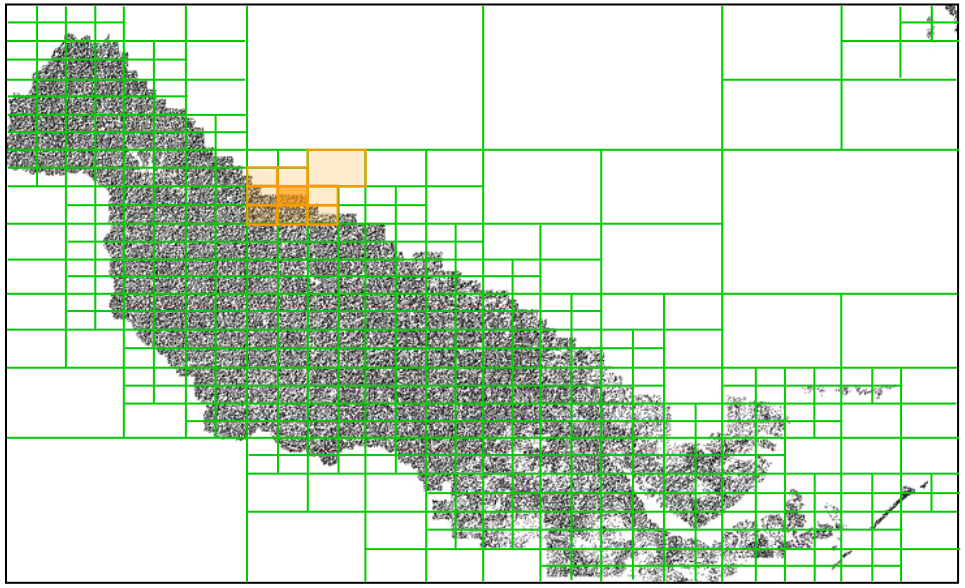
- Shepard
- RBFs
- Kriging
- TIN
  - linear
  - higher order



Previous Work: External Memory Quadtree

From Point Cloud to Grid DEM: A Scalable Approach
[Agarwal et al. ‘06]

- rearrange points on disk into quadtree
  - process cell
    - find its neighbors
    - interpolate all (i,j)
    - write rasters to temporary file



  - sort temporary file on (i,j) & write DEM

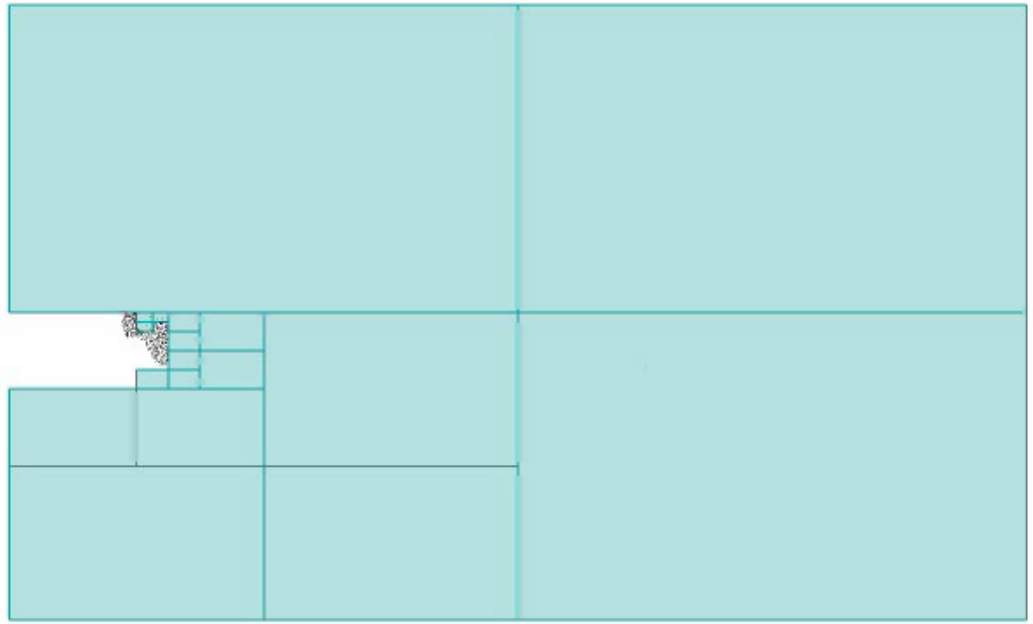
Our Approach: TIN Streaming

- reorganize computations … not points
  - compute TIN in places where all points have already arrived
  - raster, output & deallocate
  - keep parts that miss neighbors

- enhance points with spatial finalization

Our Approach: TIN Streaming

- reorganize computations … not points

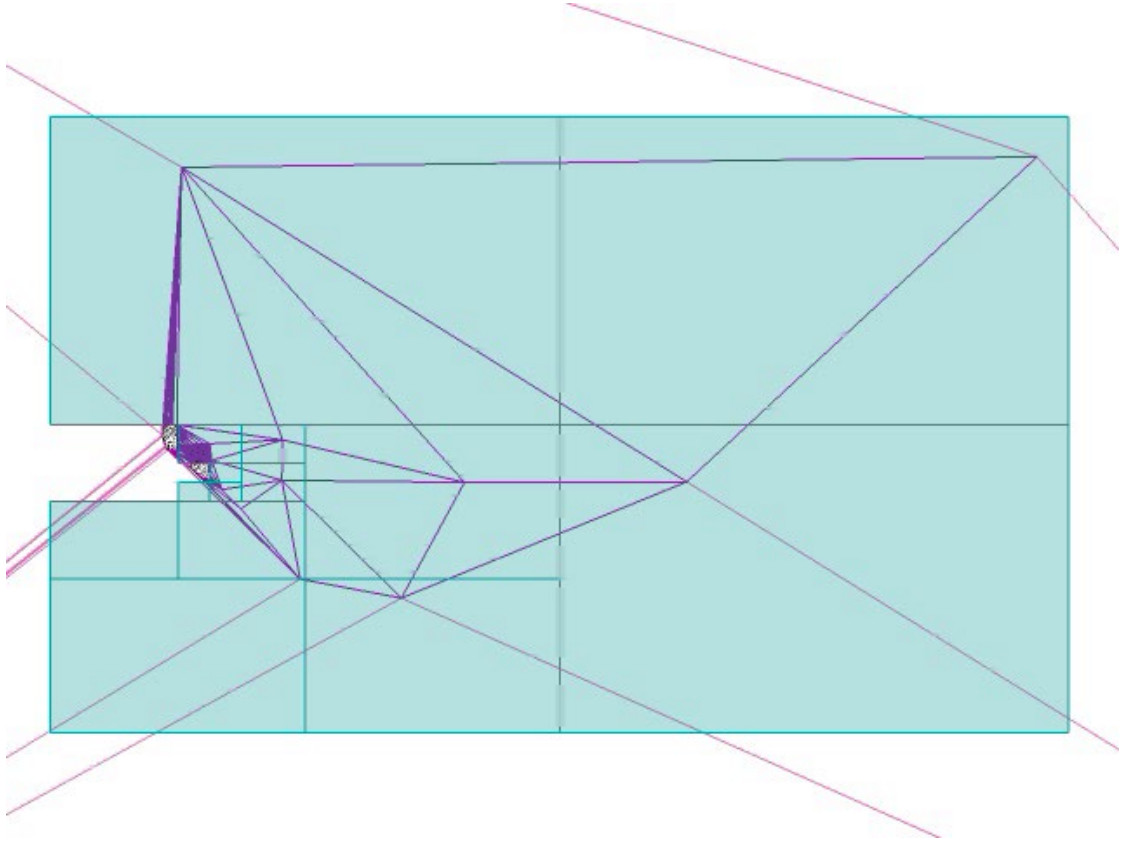


  - compute TIN in places where all points have already arrived
  - raster, output & deallocate
  - keep parts that miss neighbors

- enhance points with spatial finalization

Our Approach: TIN Streaming

- reorganize computations … not points

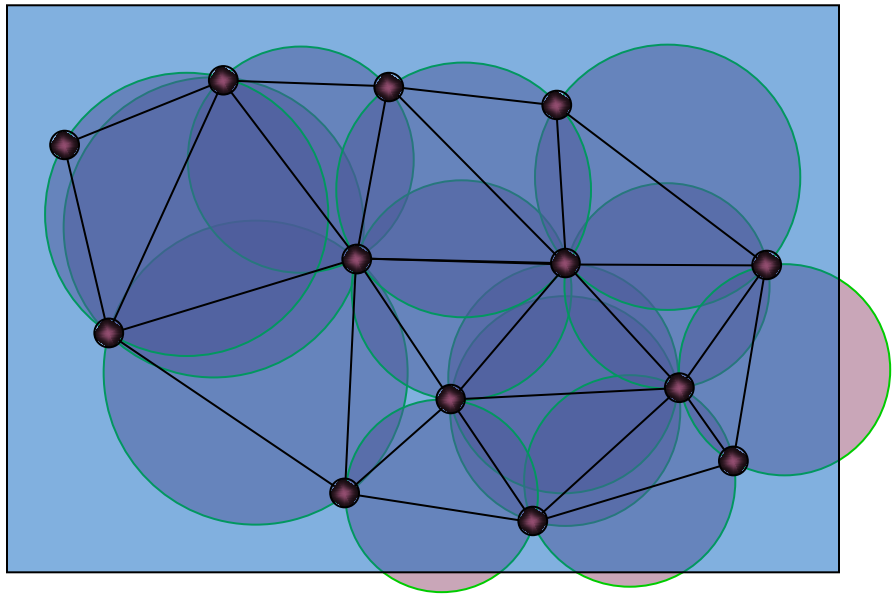


  - compute TIN in places where all points have already arrived
  - raster, output & deallocate
  - keep parts that miss neighbors

- enhance points with spatial finalization

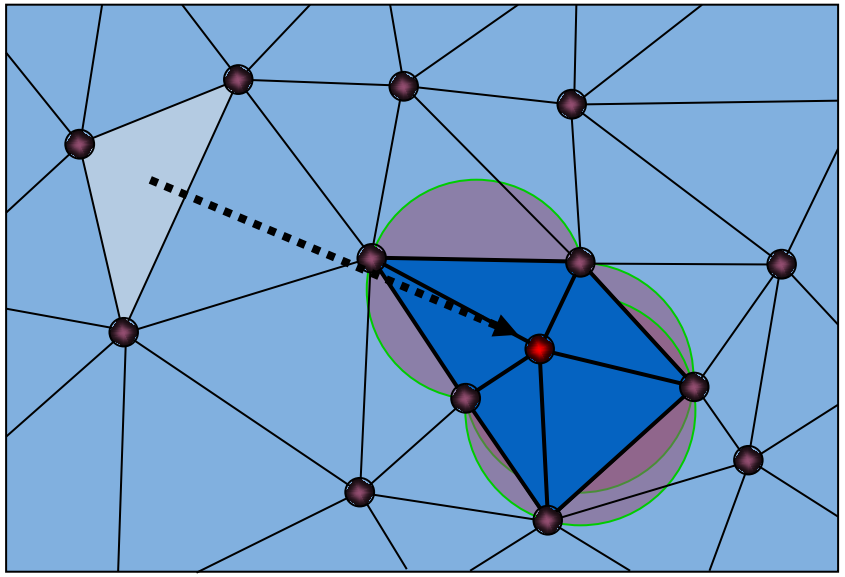
Delaunay Triangulation

- standard algorithm for TIN construction
- good properties for interpolating points



- a triangulation in which every triangle has an empty circumscribing circle

Incremental Point Insertion



[Lawson '77]
[Bowyer '81]
[Watson '81]

- locate triangle enclosing the point

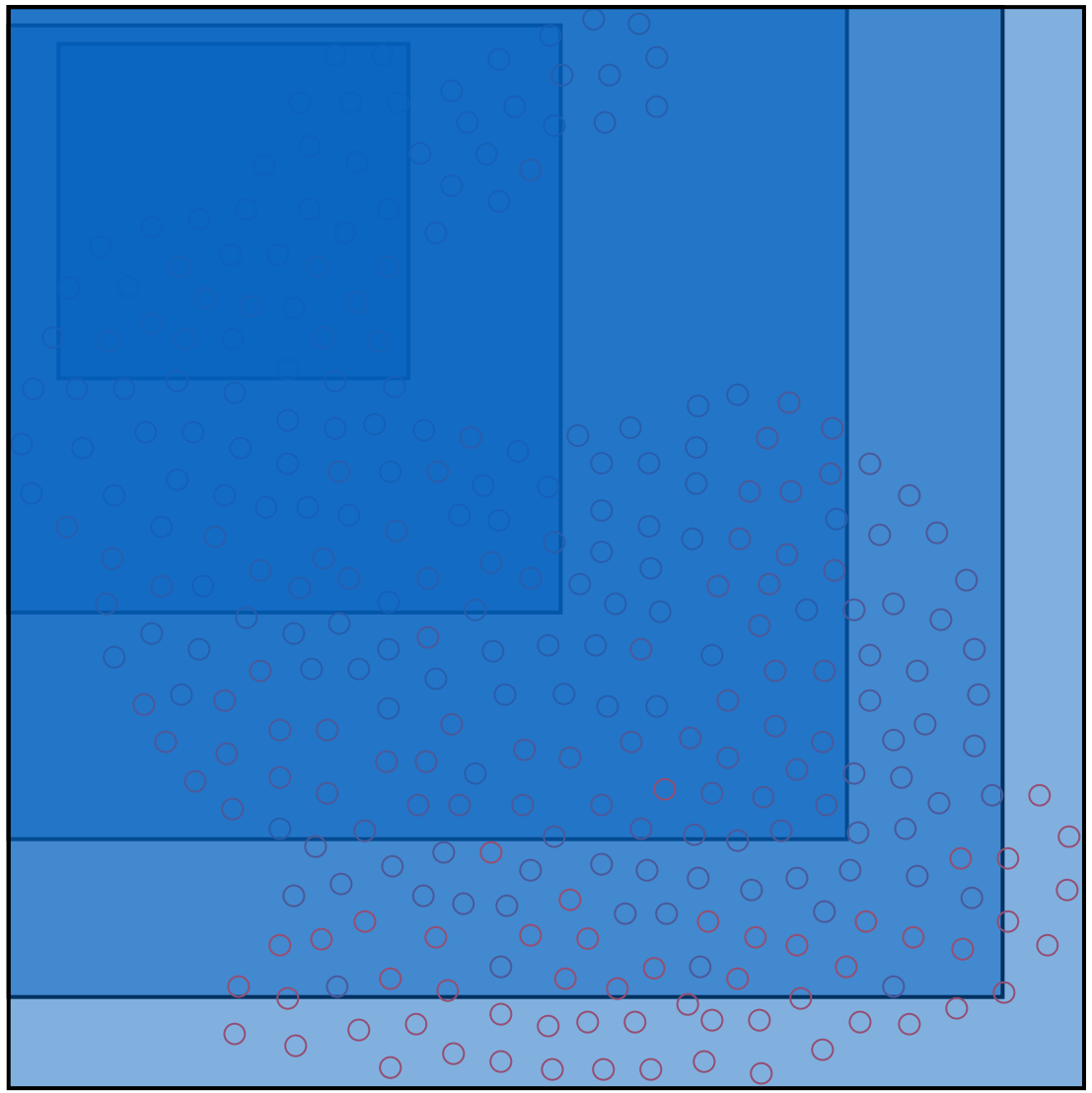
- find and remove all triangles with non-empty circumcircles
- triangulate by connecting new point

The Finalizer

```
(spfinalize)
```

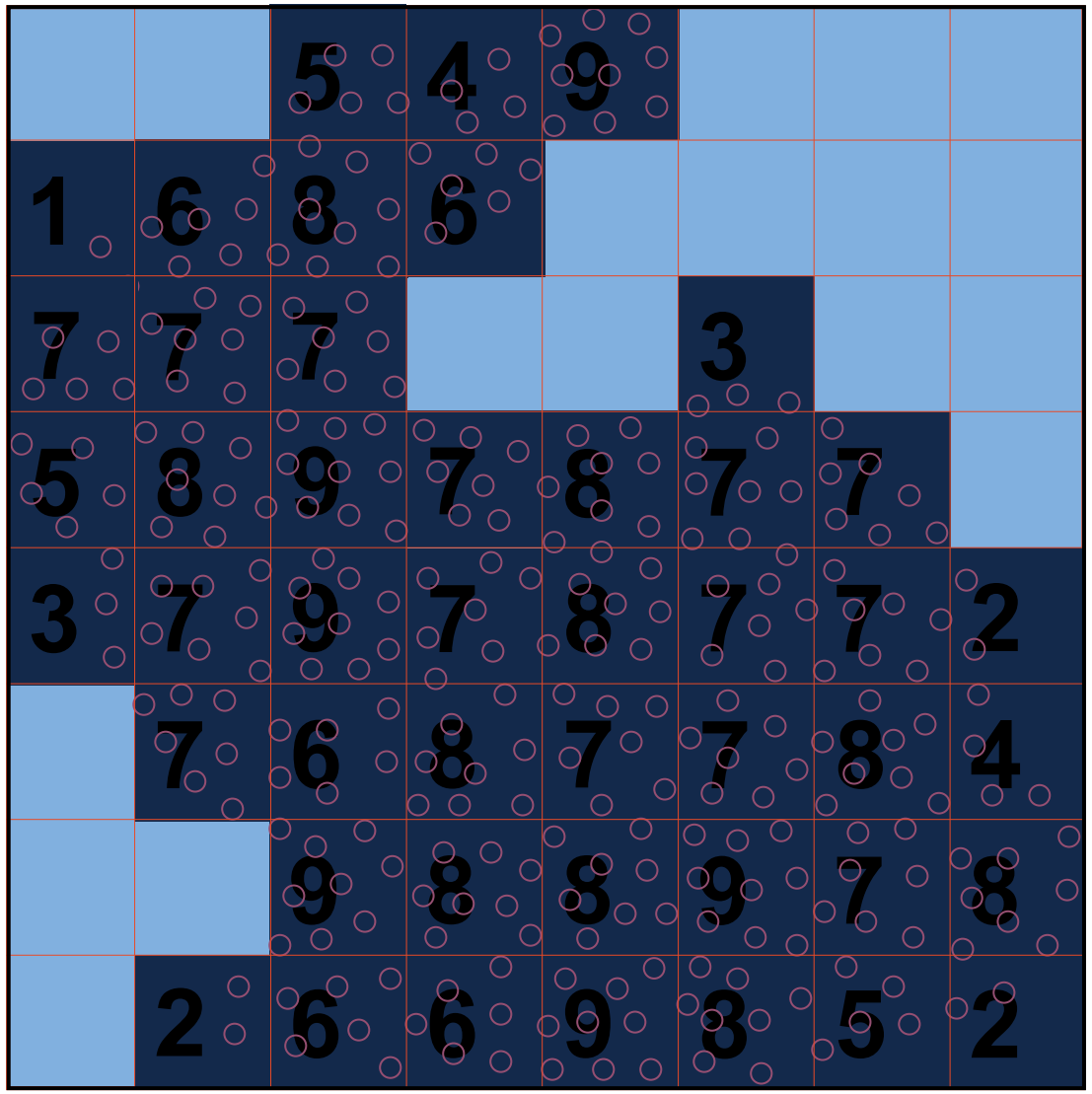
ILLINOIS

Spatial Finalization of Points



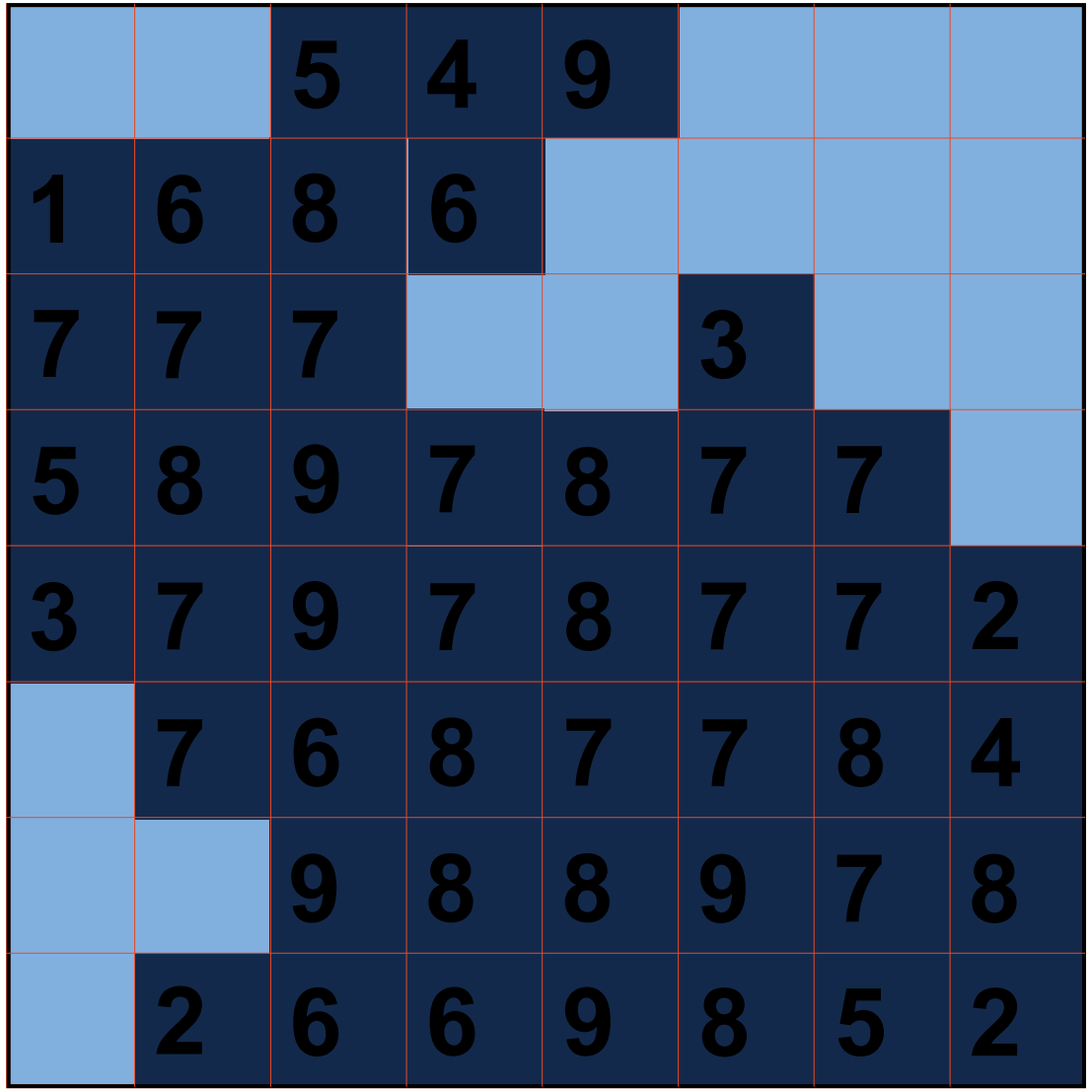
1. compute bounding box

Spatial Finalization of Points



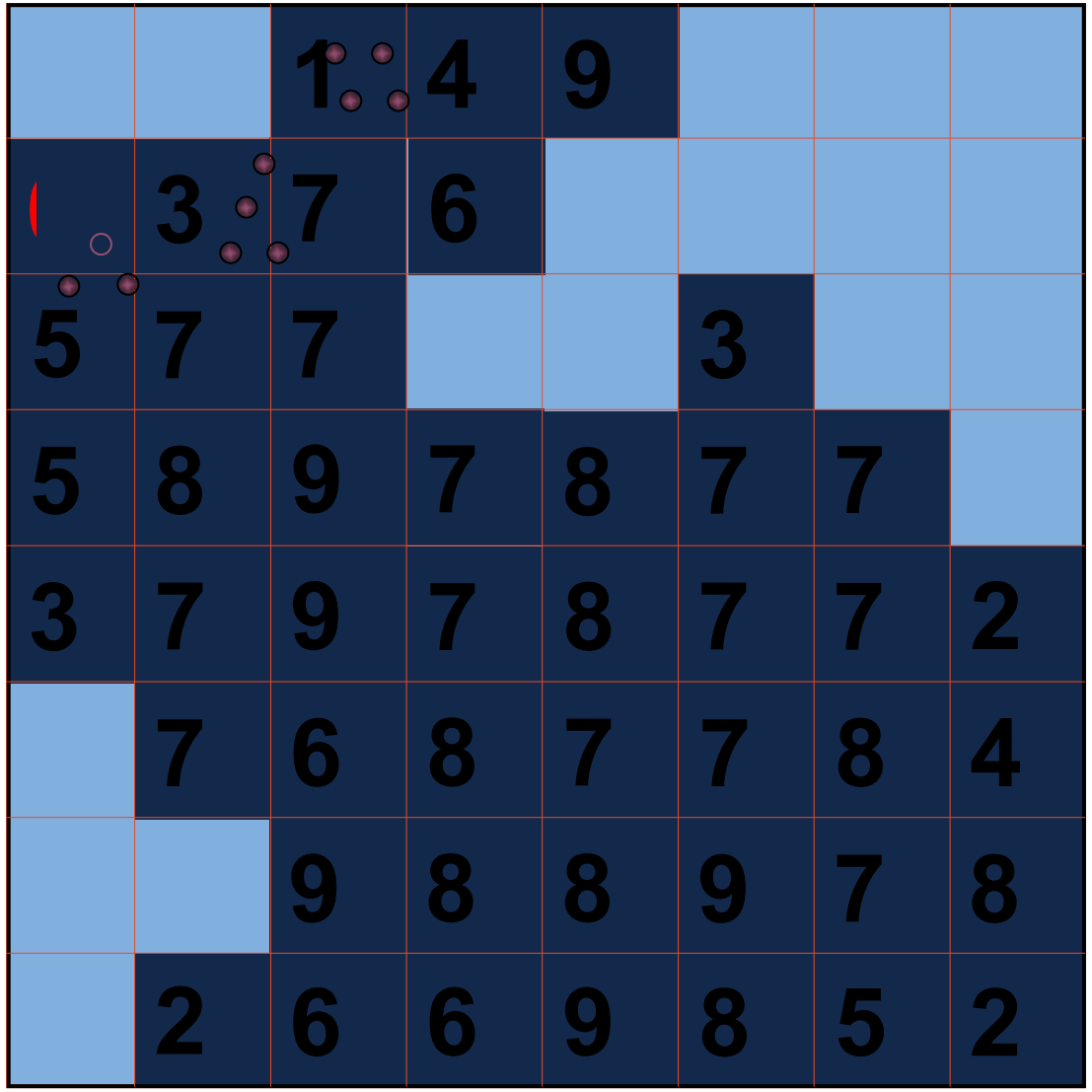
1. compute bounding box
2. create finalization grid
   - count number of points per cell

Spatial Finalization of Points



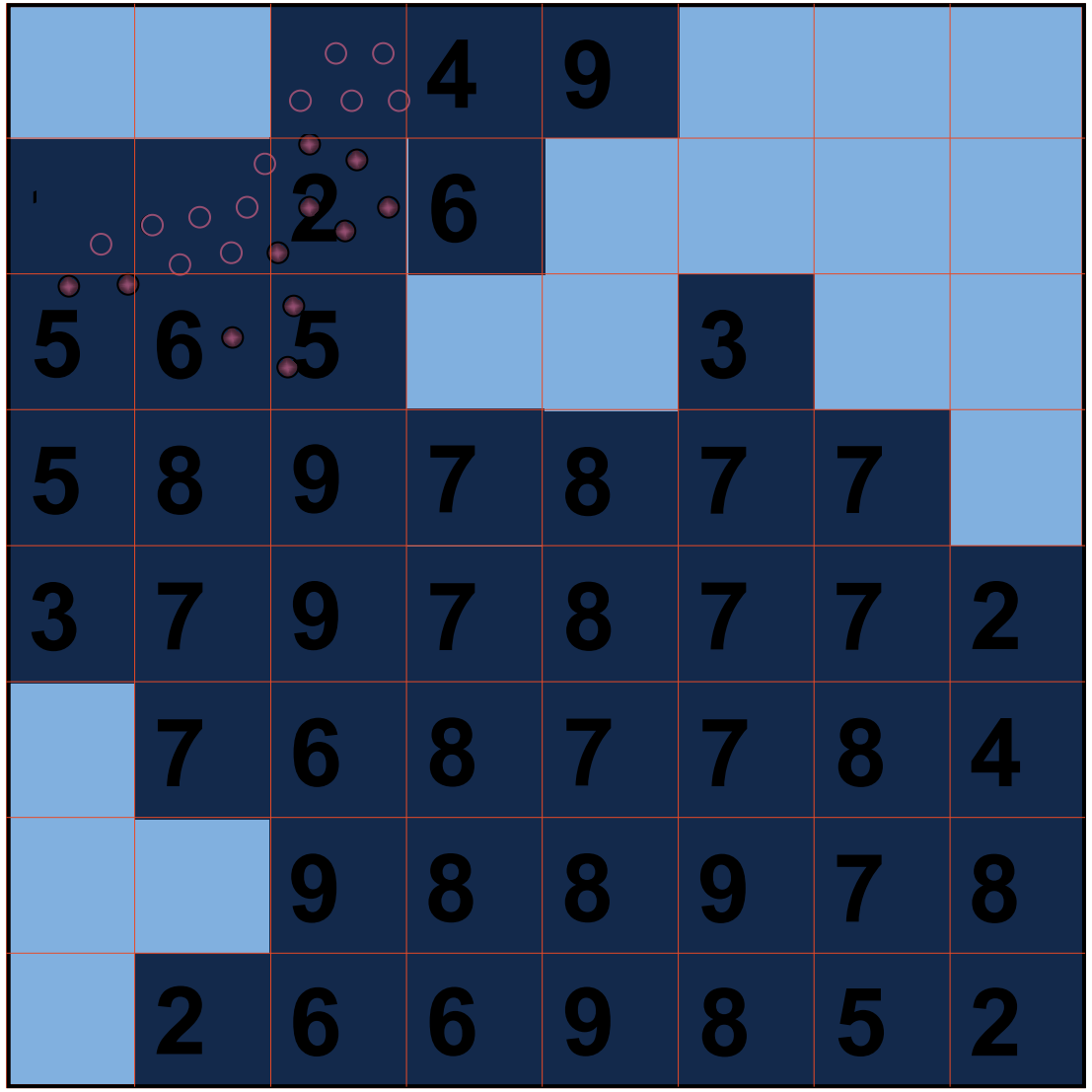
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

Spatial Finalization of Points



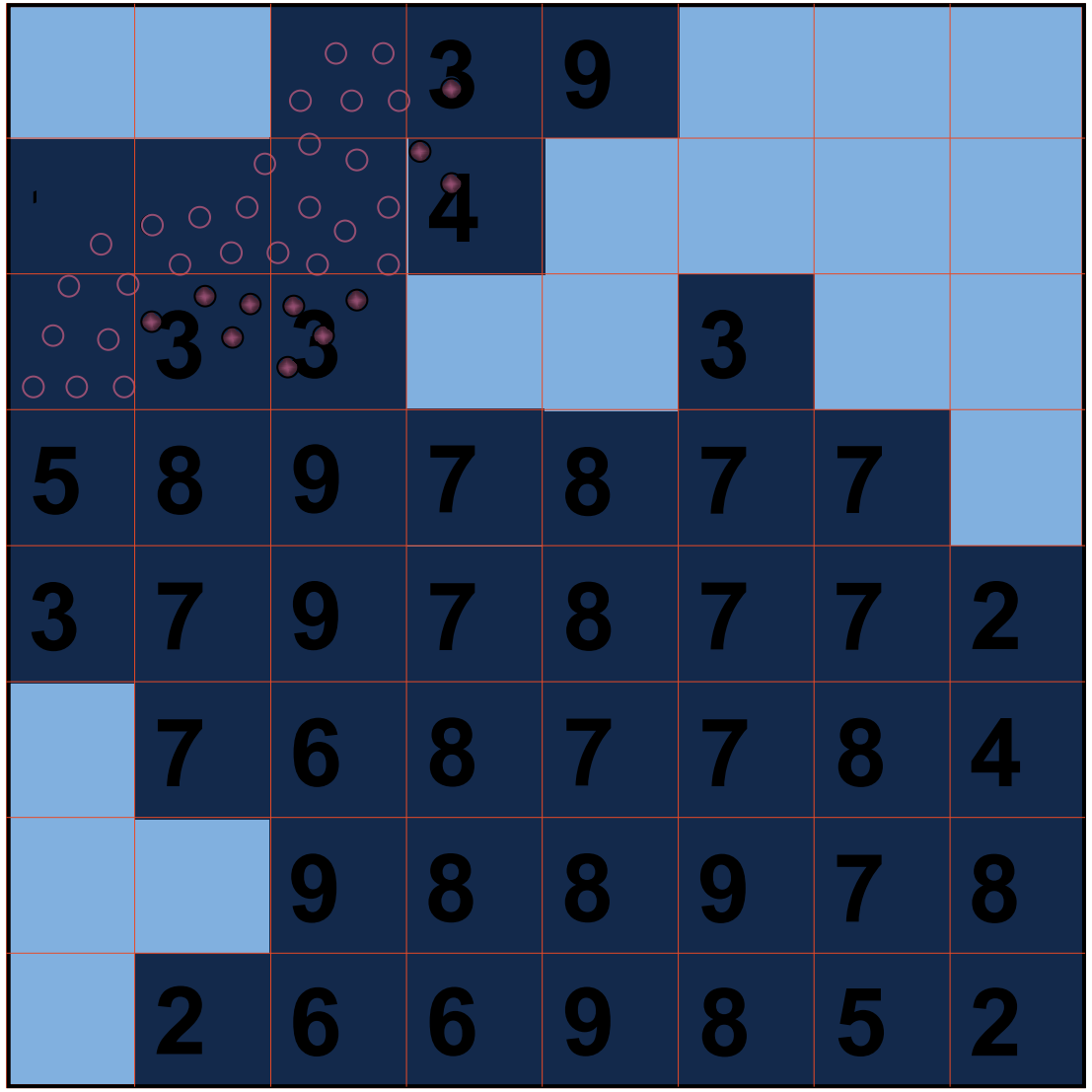
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

Spatial Finalization of Points



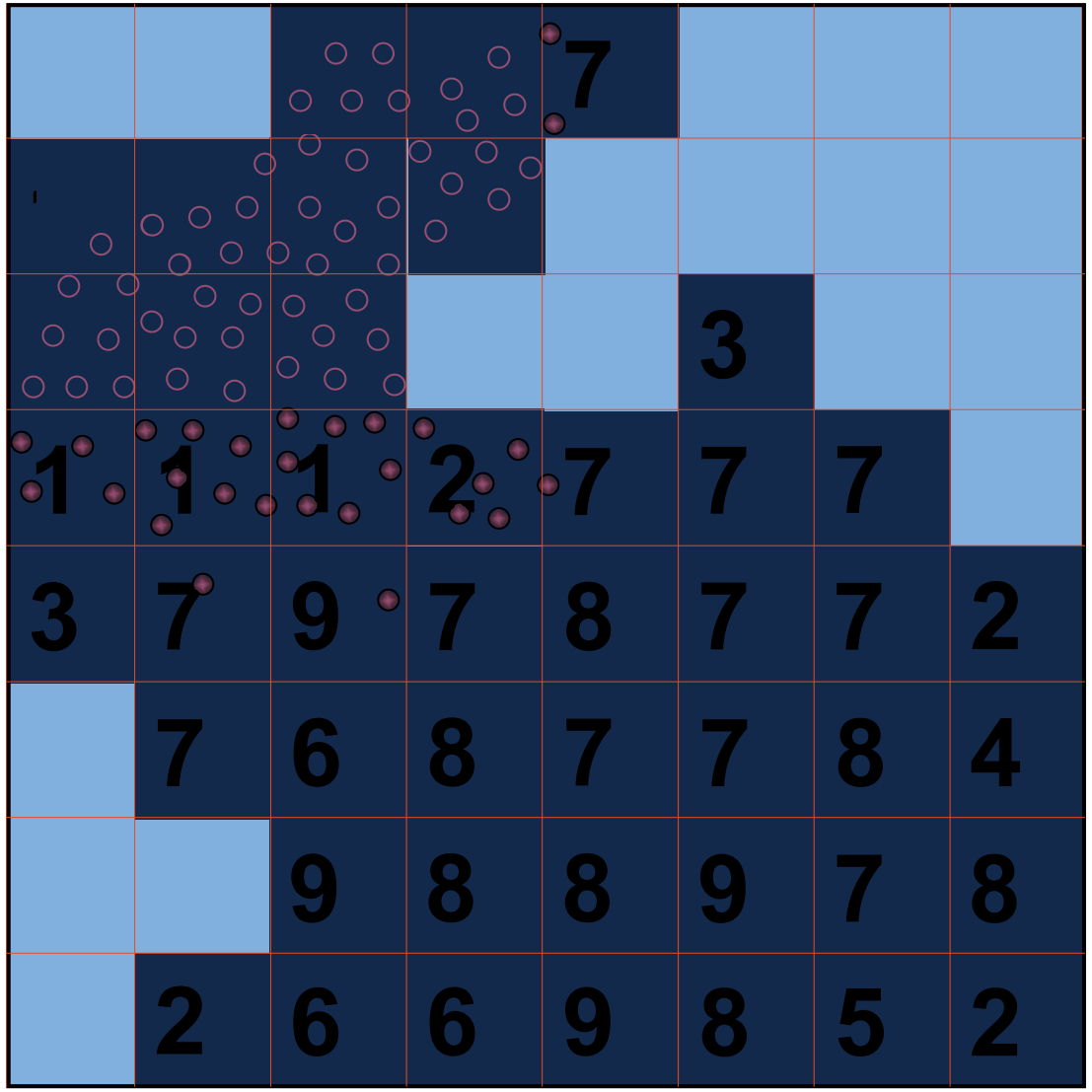
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

Spatial Finalization of Points



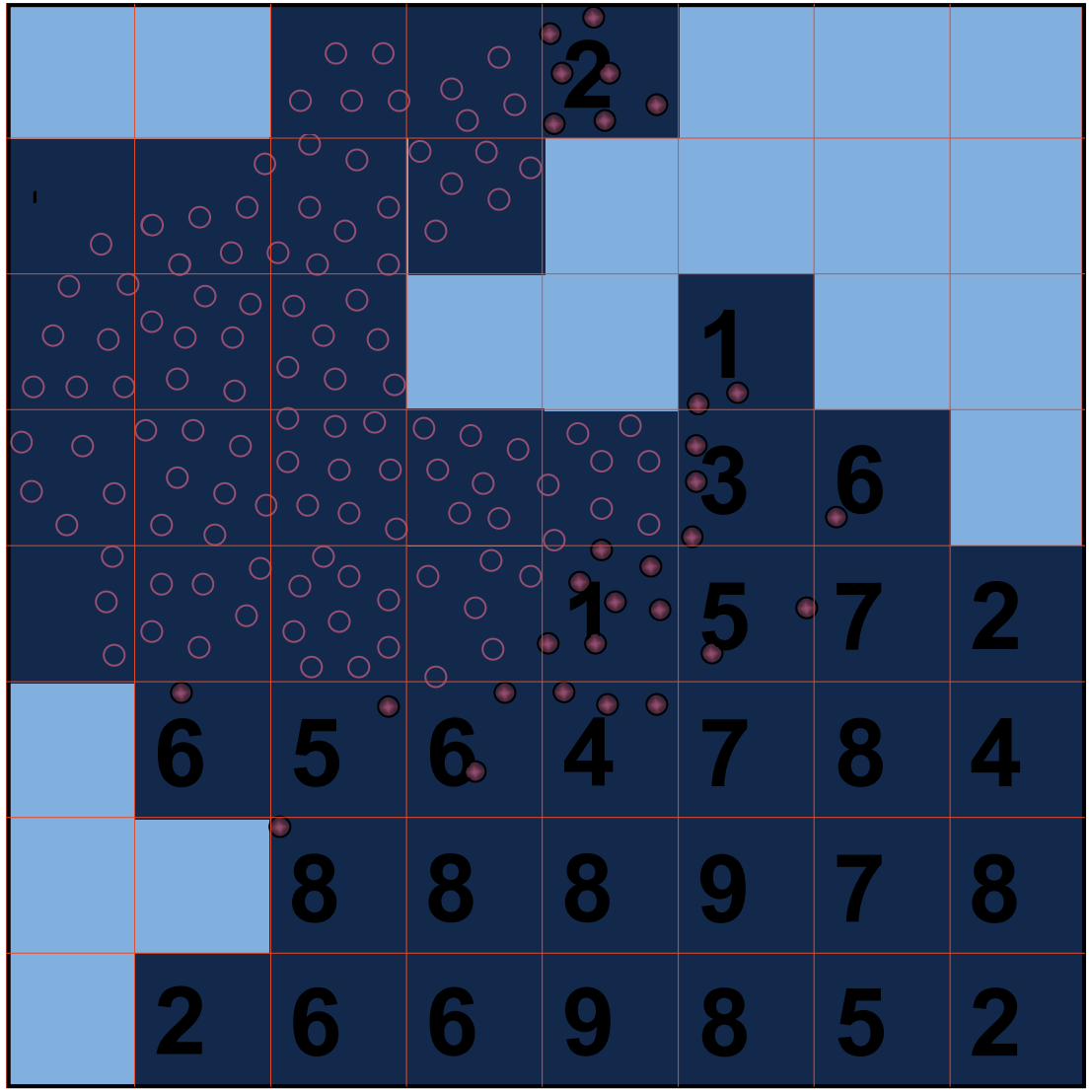
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

Spatial Finalization of Points



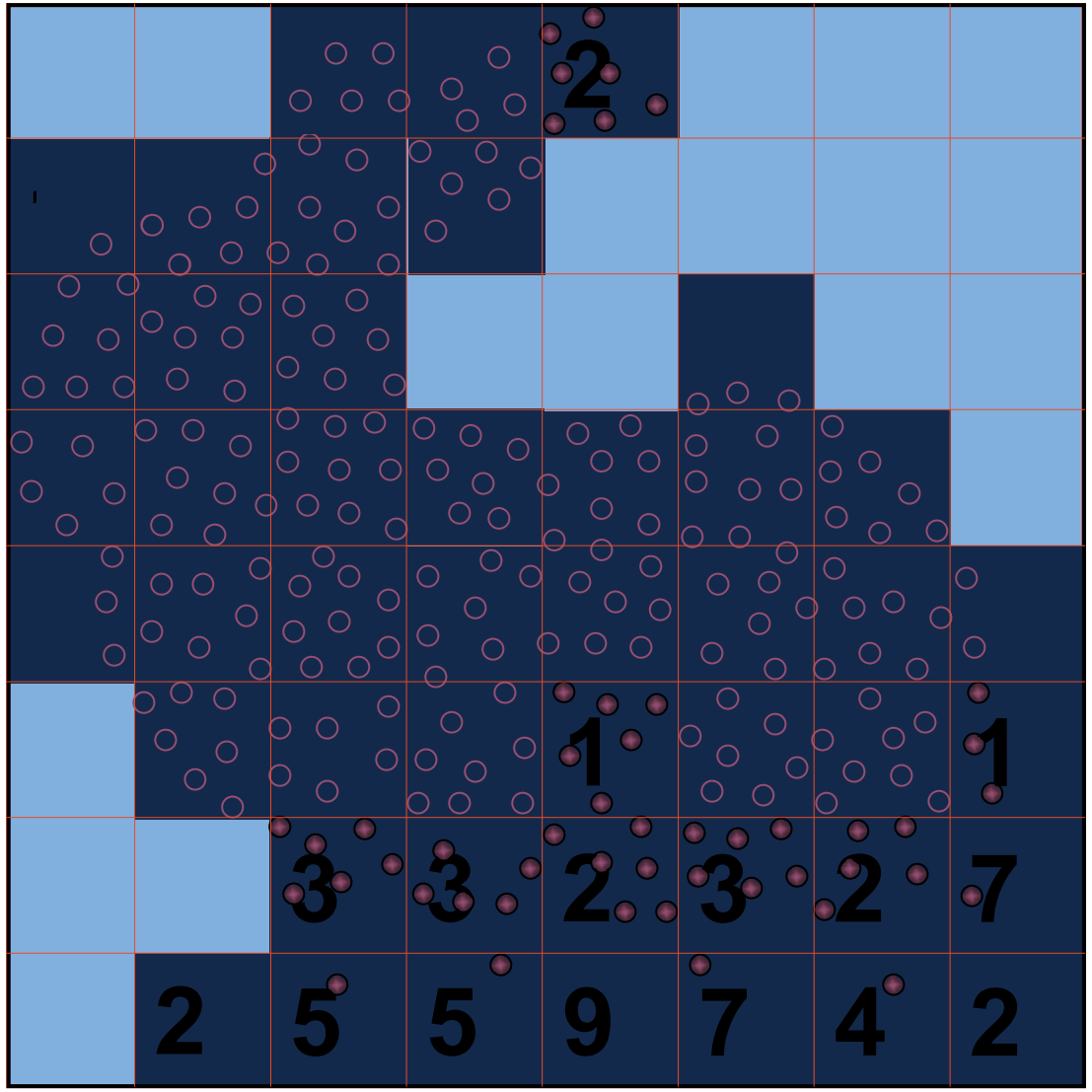
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

Spatial Finalization of Points



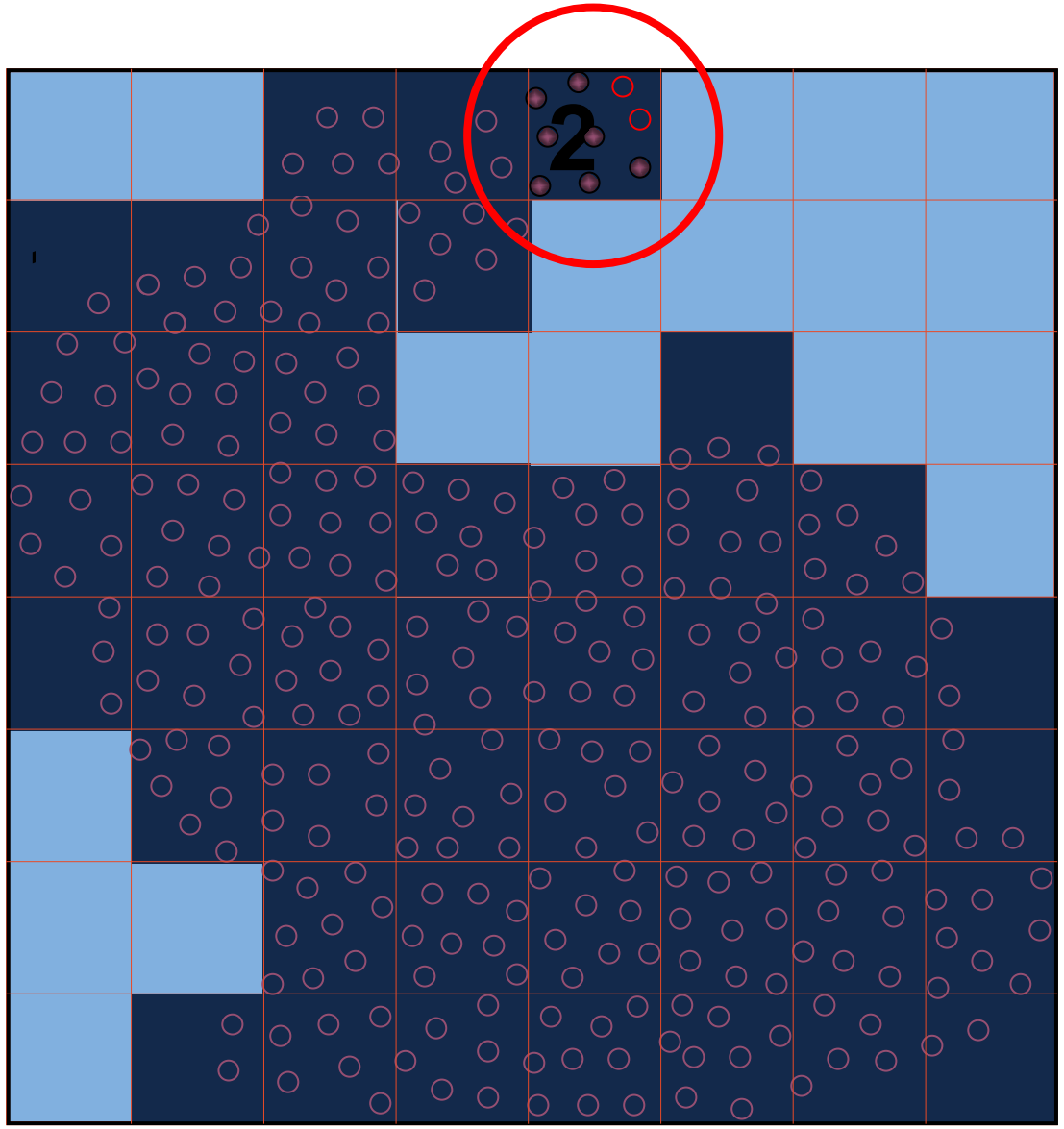
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

Spatial Finalization of Points



1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

Spatial Finalization of Points



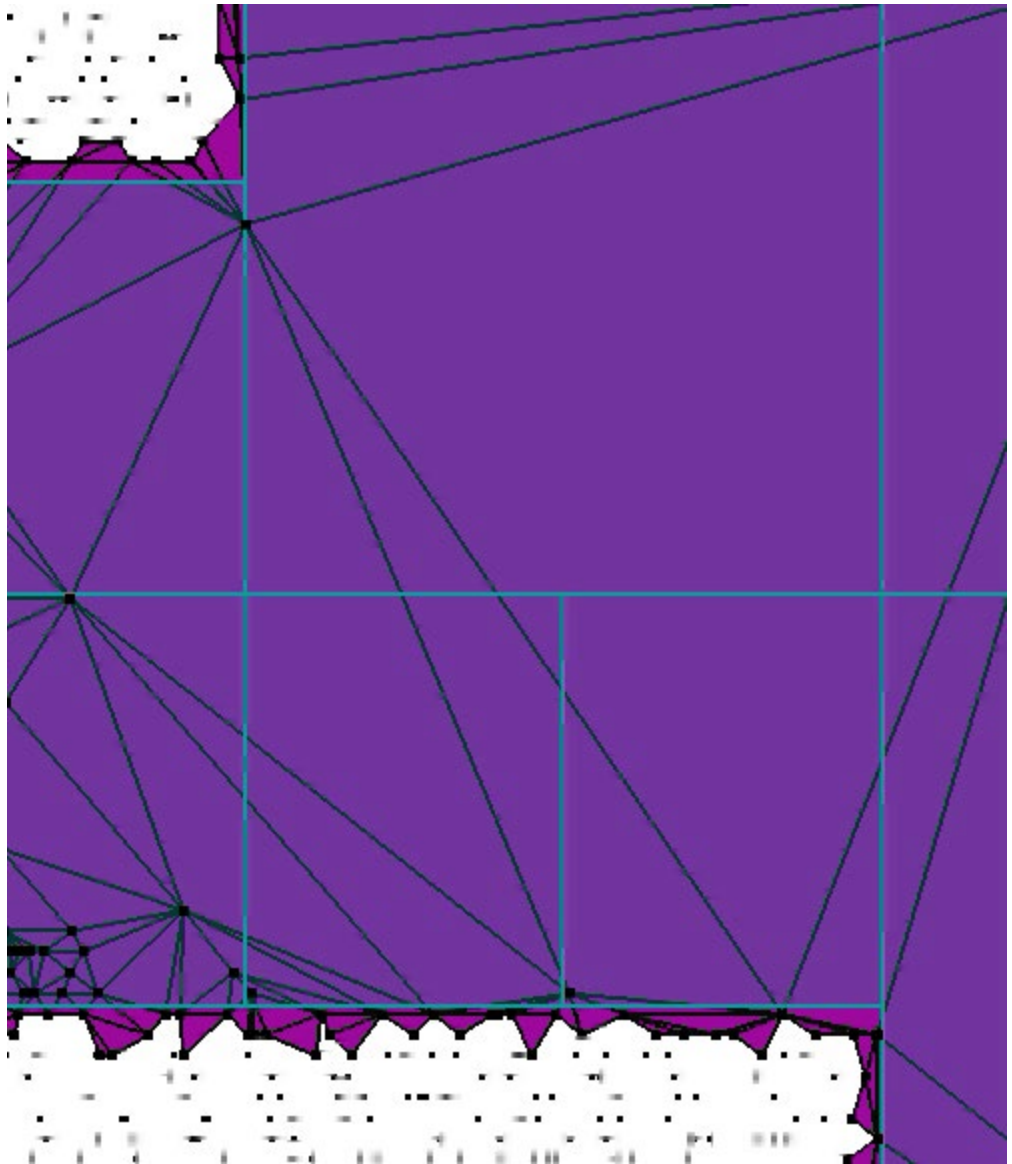
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

The Triangulator

(spdelaunay)

Modified Incremental Delaunay

- insert points
- when reading a finalization tag
  - find certified triangles
  - output them to disk or pipe
  - deallocate data structure

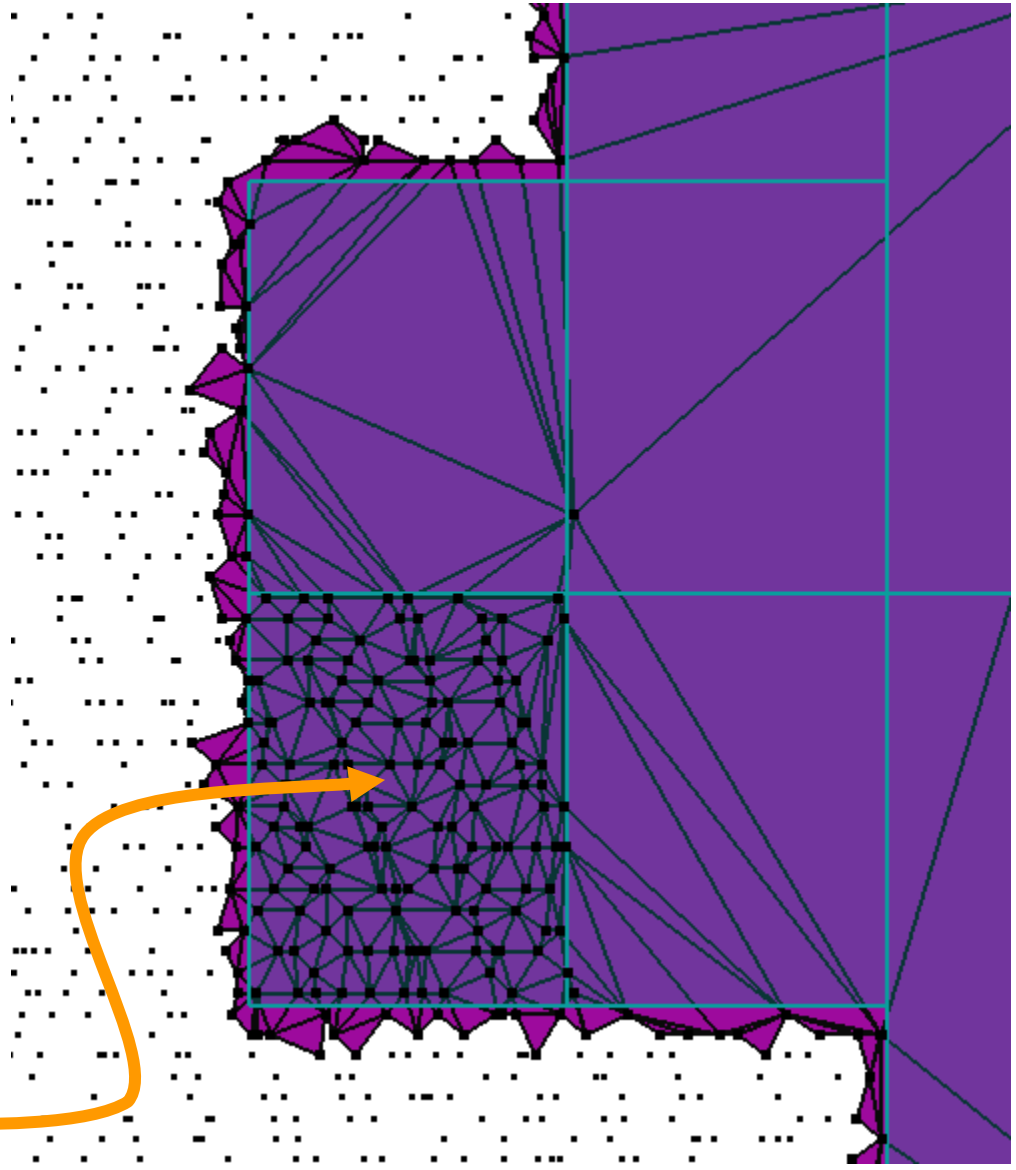


ILLINOIS

Modified Incremental Delaunay

- insert points

- when reading a finalization tag
  - find certified triangles
  - output them to disk or pipe
  - deallocate data structure

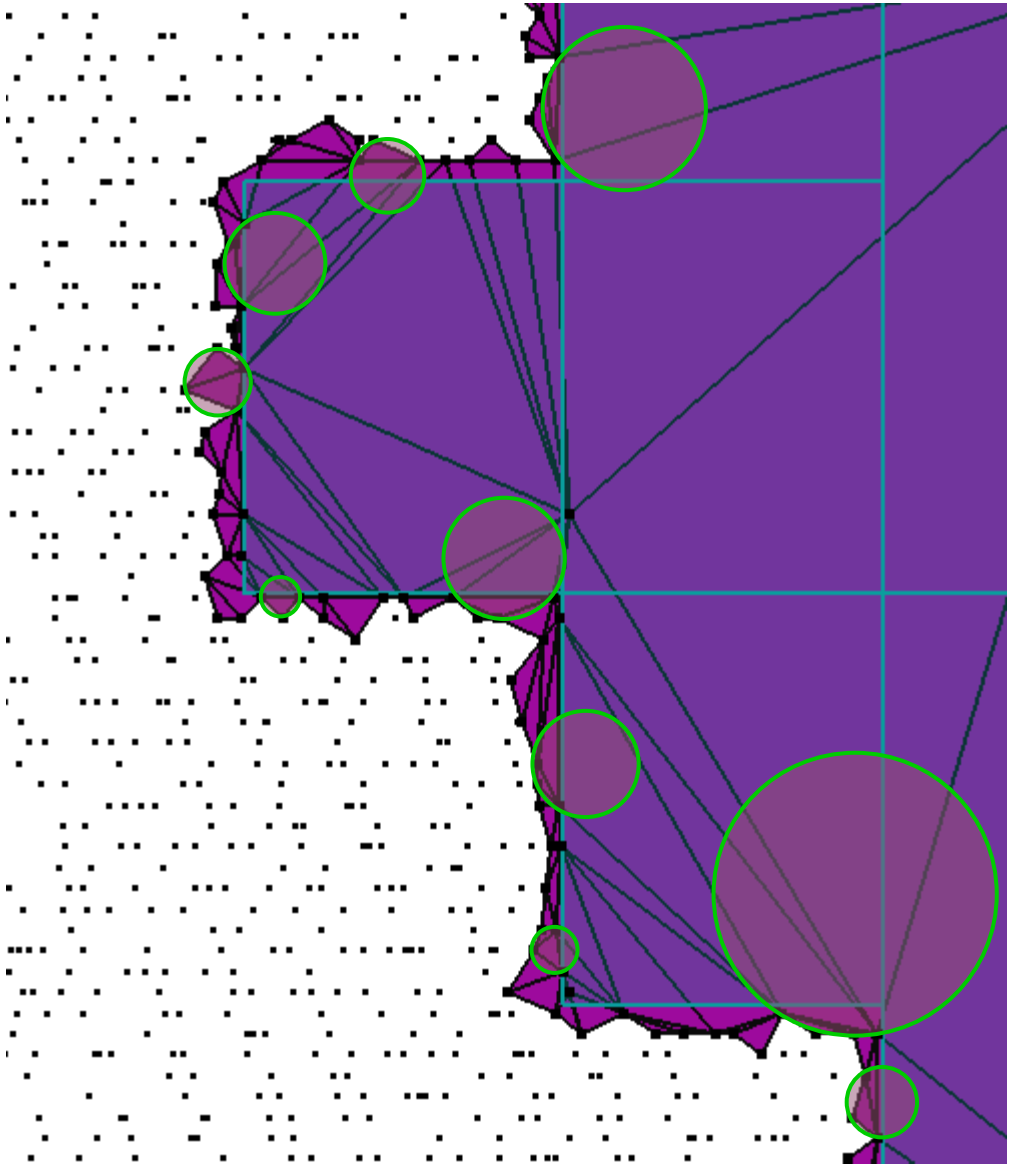


ILLINOIS

Modified Incremental Delaunay

- insert points

- when reading a finalization tag
  - find certified triangles
  - output them to disk or pipe
  - deallocate data structure



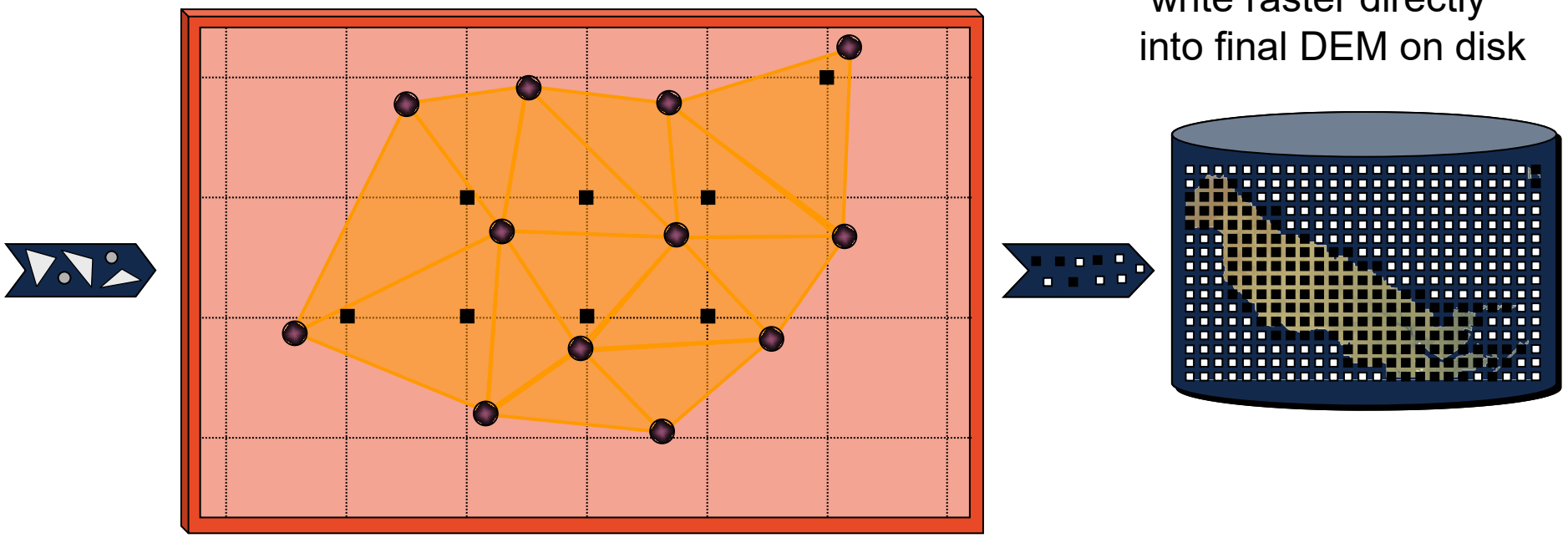
The Rasterizer

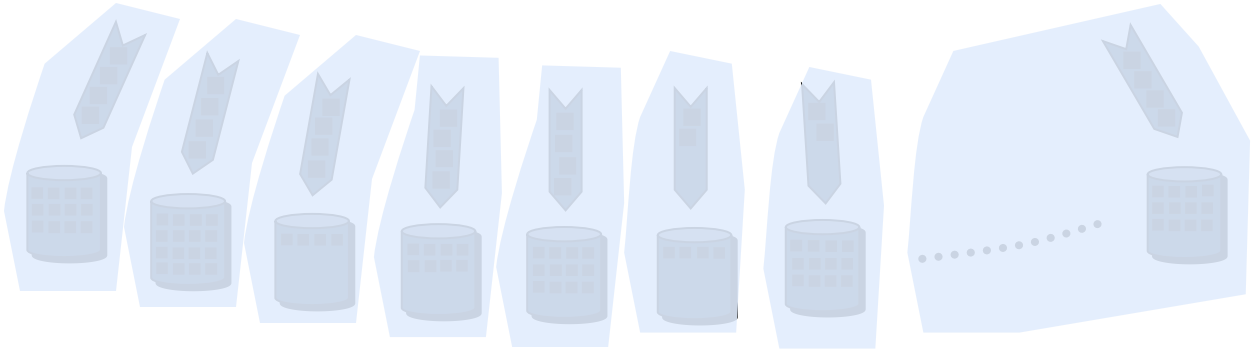
(tin2dem)

ILLINOIS

Rasterizing the Streaming TIN

write raster directly
into final DEM on disk

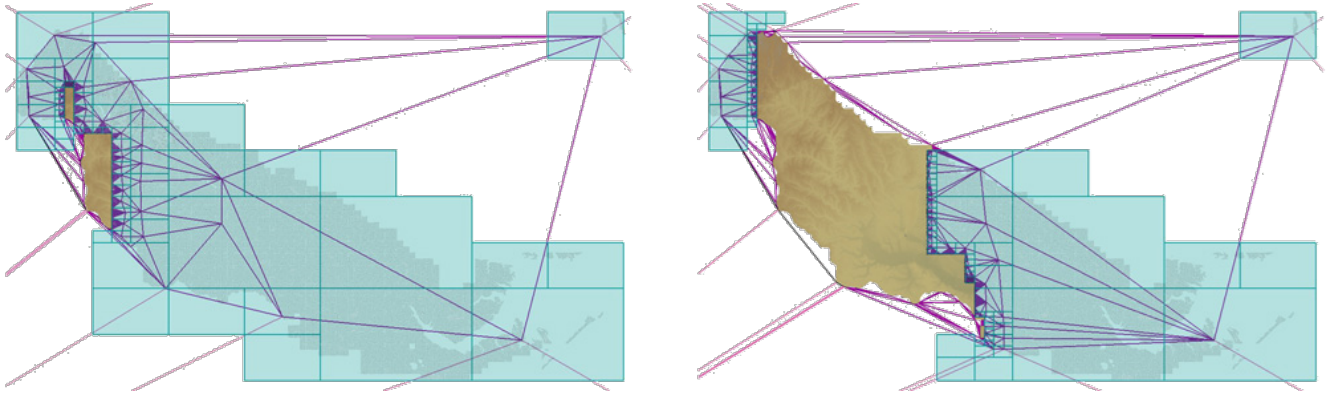




write raster to temporary files on disk
(use small buffer, compress when writing buffer)

Results

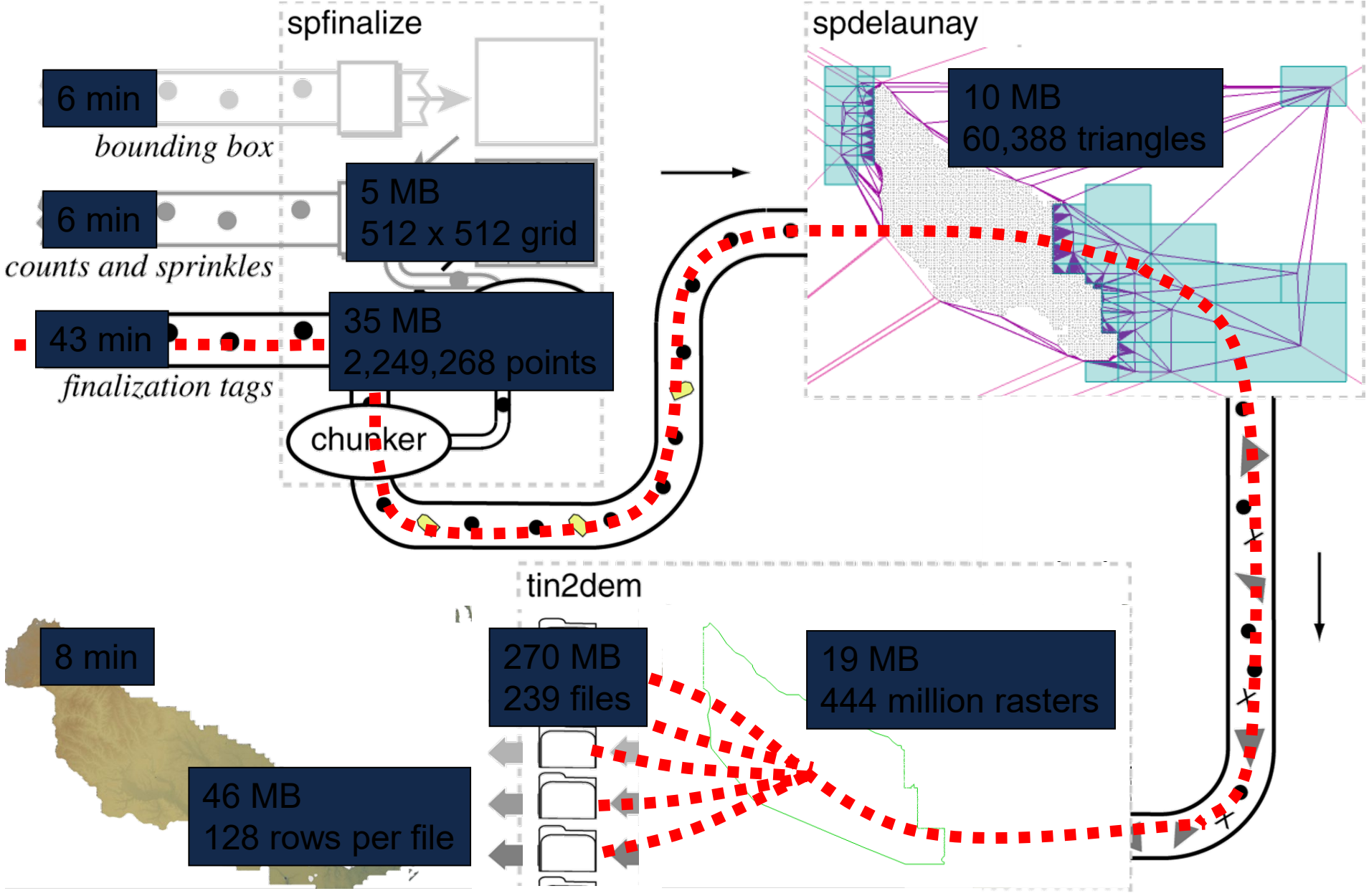
Neuse-River Basin



	number of points, file size	500,141,313	11.2 GB
LIDAR	$[\min_x; \max_x]$	[1,930,000;	2,937,860]
input mass points	$[\min_y; \max_y]$	[390,000;	1,000,000]
	$[\min_z; \max_z]$	[−129.792;	+886.443]

	grid spacing	40 ft	20 ft	10 ft
DEM	cols	25,198	50,394	100,788
output grid	rows	15,250	30,500	61,000
	output file size	750 MB	3.0 GB	12.0 GB
Time (hours:minutes)	**total**	**0:53**	**1:07**	**1:20**
Memory (MB)	**maximum**	**55**	**64**	**91**
Scratch Files (MB)	**total size**	**82**	**270**	**868**

Pipeline Details for the 20 ft DEM



Comparison to [Agarwal et al. '06] (20 ft)

- 53 hours
  - 3.4 GHz desktop
  - 10,000 RPM disks
  - scratch space:
    - quadtree: ~ 8 GB
    - rasters: ~ 2.5 GB
- 67 minutes
  - 2.1 GHz laptop
  - 5,400 RPM disks
  - scratch space:
    - compressed rasters: 270 MB
- 86 % of CPU time for expensive RST
  - remaining 7.5 hours
    - constructing quadtree: 1.1 hours
    - finding cell neighbors: 5.6 hours

ILLINOIS

Acknowledgements

- data
  - Kevin Yi, Duke
- support
  - NSF grants 0429901 + 0430065

    "Collaborative Research: Fundamentals and Algorithms for Streaming Meshes."
  - NGA award HM1582-05-2-0003
  - Alfred P. Sloan Research Fellowship

ILLINOIS

Paper Analysis

- Paper was chosen as a good introduction to tasks related to GIS visualizations
- But if we wanted to discuss it as a research paper:

- Paper relies on empirical evidence rather than theory
  - Which is not an issue…
  - …but some reviewers would likely object to drawing general conclusions from 1 data set
- System pieces together existing ideas…not innovative in that sense
  - Streaming Delaunay was published by authors previously...more ground-breaking
  - For some research area cultures this is a fine approach, less so for others
- Performance numbers are excellent
- Addresses a real problem…large data and people working with limited resources
- Excellent, pragmatic work IMO