Massive Terrain Data Processing: Scalable Algorithms

Pankaj K. Agarwal, Duke University
Helena Mitasova, NCSU
Supported by ARO W911NF-04-1-0278

STREAM Project

http://terrain.cs.duke.edu/

Scalable Techniques for hi-Resolution Elevation data Analysis & Modeling

Participants (PIs)

Pankaj K. Agarwal

Lars Arge

Helena Mitasova

Students

Andrew Danner

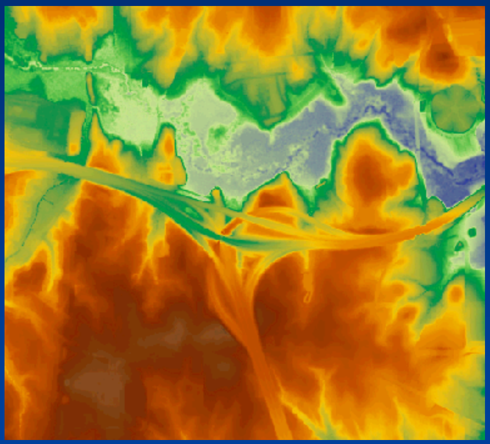
Thomas Molhave

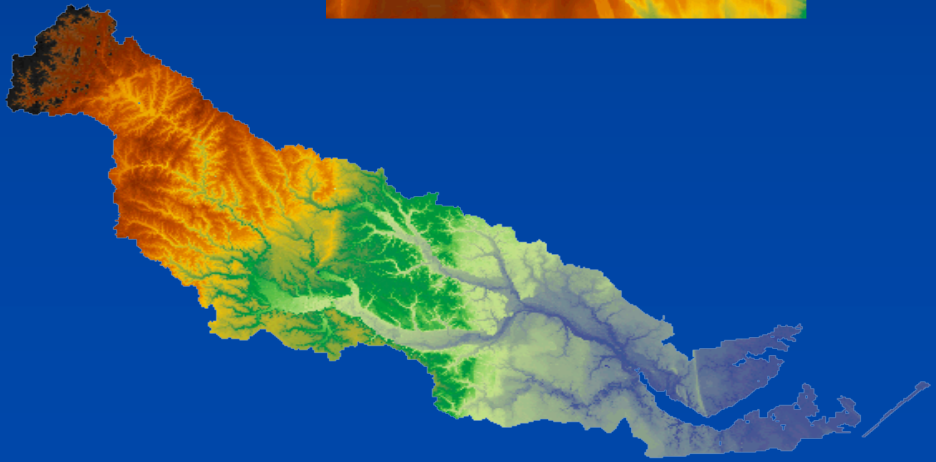
Amber Stillings

Ke Yi

Massive Data Sets

- LIDAR point clouds
  - late 90ies NC Coast: 200 million points – over 7 GB
  - Neuse River basin (NC): 500 million points – over 17 GB
- Raster DEMs are also large
  - 3m res. grid: 3 billion cells
- Data too big for RAM
  - Must reside on disk
  - Disk is slow

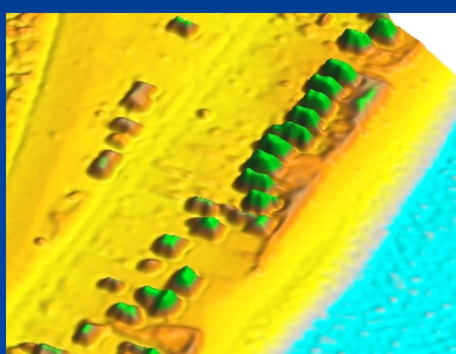




Increasing LIDAR point density

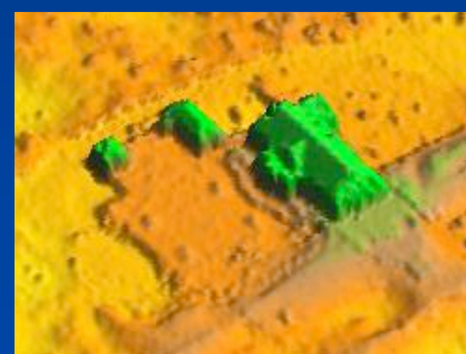
1998 **2004**

NC Coast:
from 1pt/3m to 1pt/0.3m
substantially improved representation of structures but
much larger data sets



1m resolution DEM
computed by RST

binned
2004 lidar 0.5m resolution DEM



computed by RST

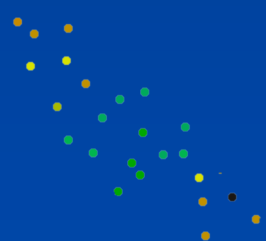
Terrain modeling and analysis workflow

All steps must run for massive data sets

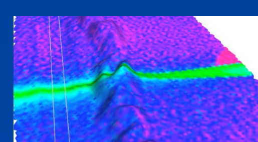
input -> **LIDAR Points**

Density, noise and accuracy analysis:
selection of resolution, approximation method, systematic error removal

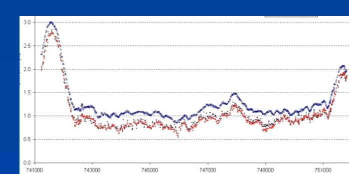
Spatial approximation:
smoothing of random noise, computation of grid DEM and its parameters



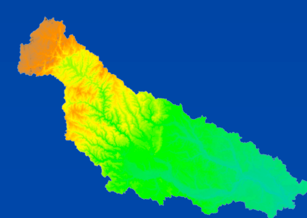
r.in.xyz



r.what, awk



v.in.ascii -b, v.surf.rst

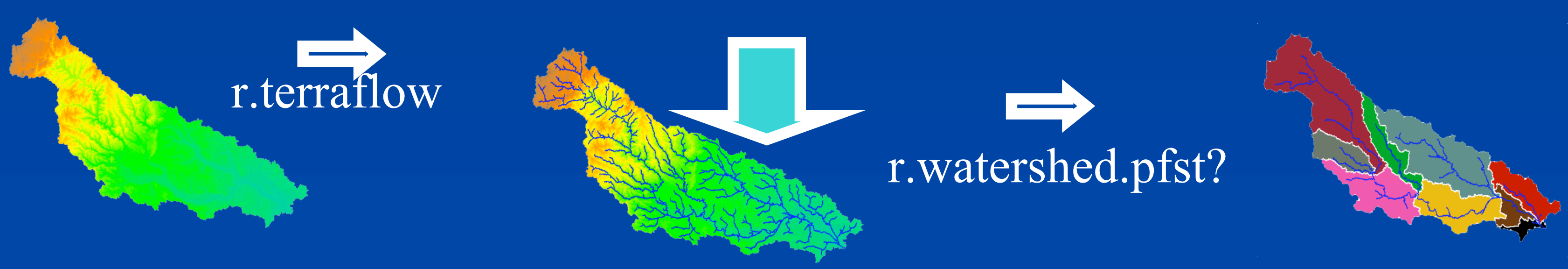


Terrain modeling and analysis workflow

Flow analysis:
sink removal, flow direction, flow accumulation

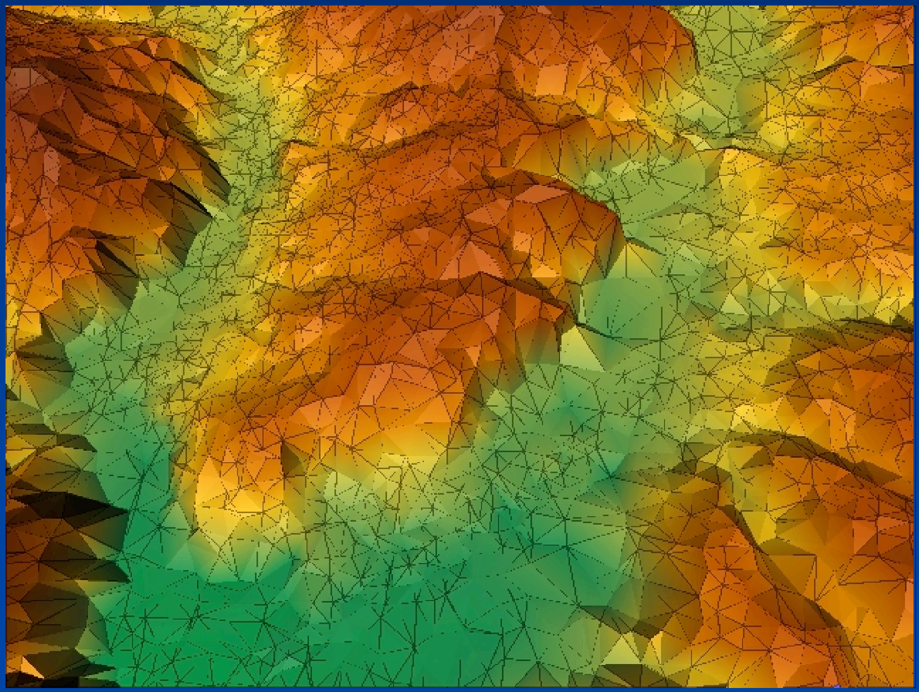
Watershed hierarchy:
Pfaffstetter labeling, watershed hierarchy

Vectorization:
streams and watershed boundary



Elevation points to TIN DEM

TIN: Triangulated Irregular Network



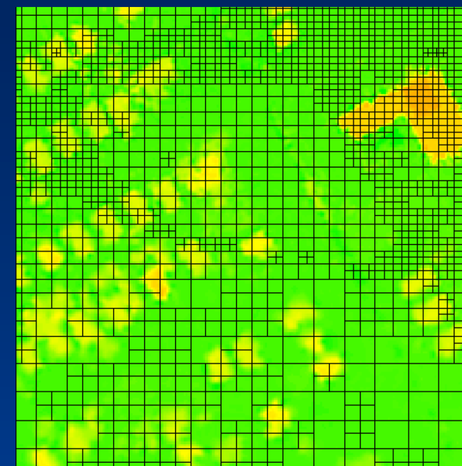
Constrained Delaunay Triangulation

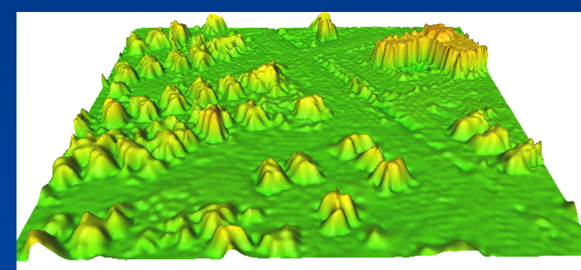
Developed an I/O-efficient algorithm: requires special vector data structure, stand alone module

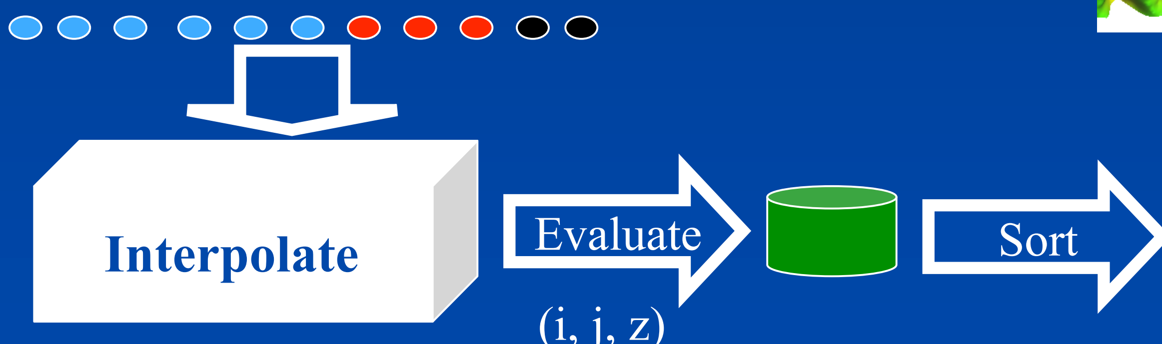
Construction of grid DEM

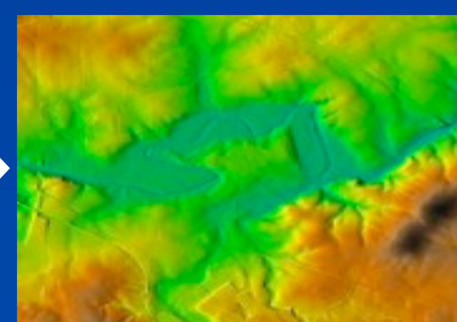
Modified I/O efficient approach

- Segment the space into small regions
- Interpolate within each segment, any interpolation/approximation method can be used
  - Evaluate at grid cells, write grid cell values as (i,j,z) as they are computed
  - Sort grid cells by raster order



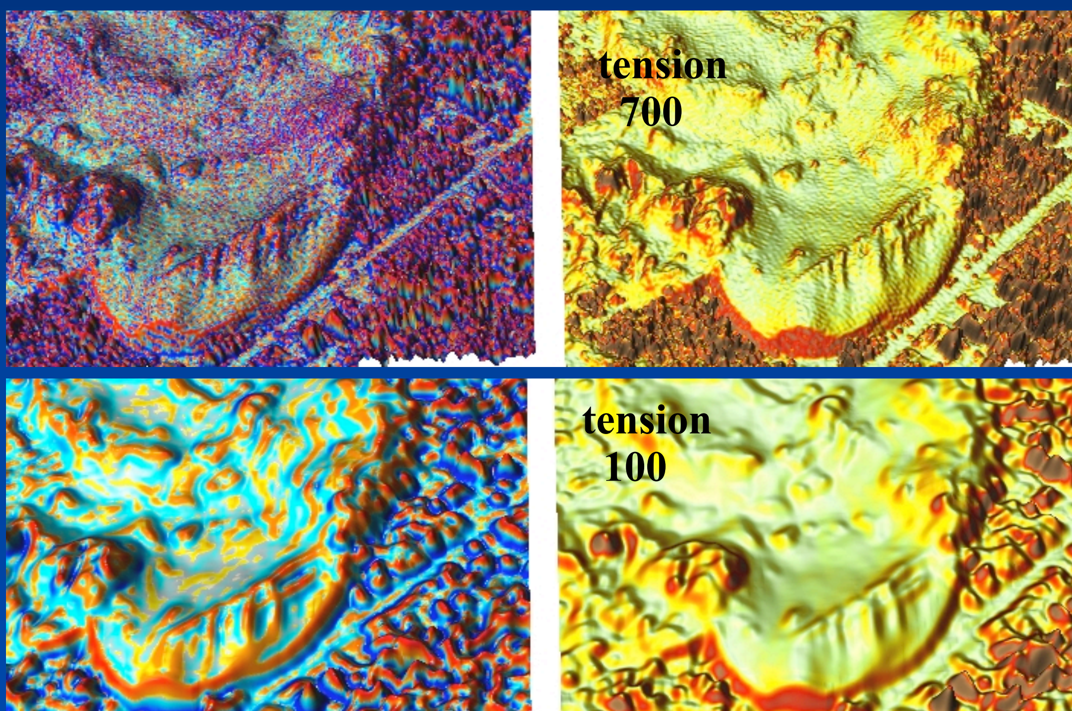




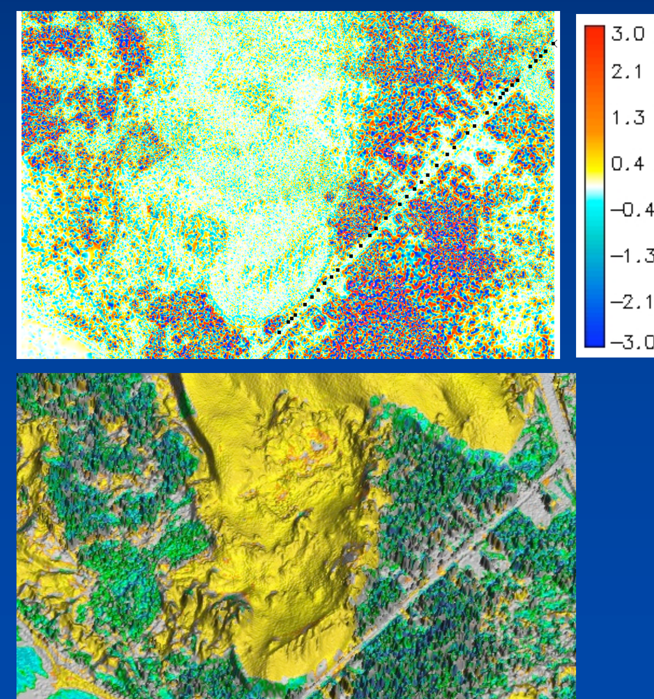


Coping with Noisy Data

- vegetation, natural roughness, lidar errors: noise (bumps and pits)
- in high resolution DEMs - difficulties extracting topo features
- smoothing during DEM construction (e.g. using RST) reduces noise and allows to extract some curvature based features

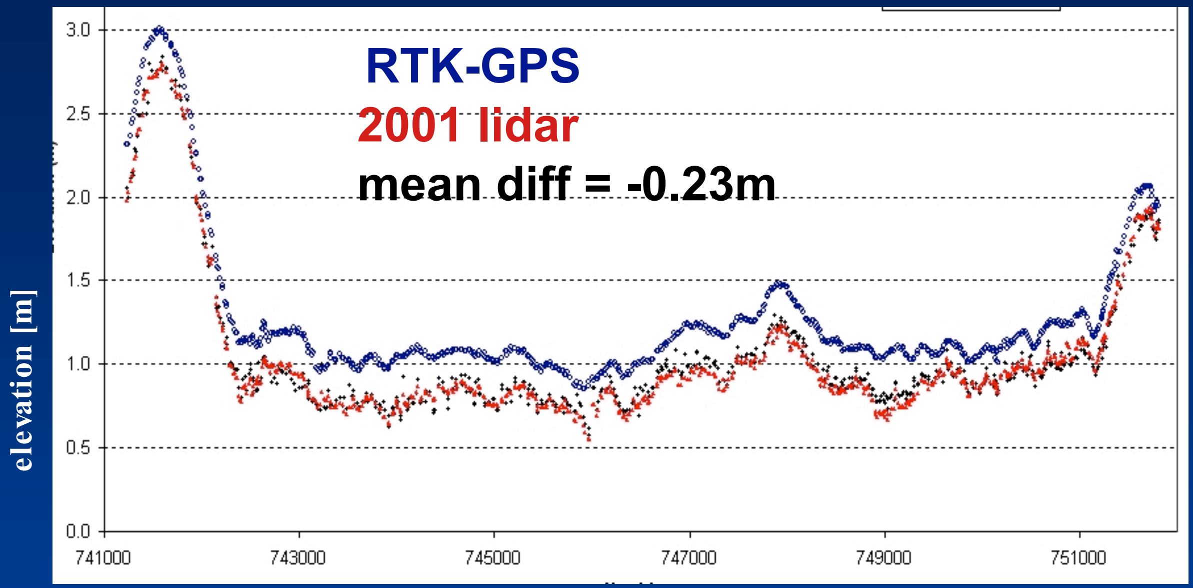


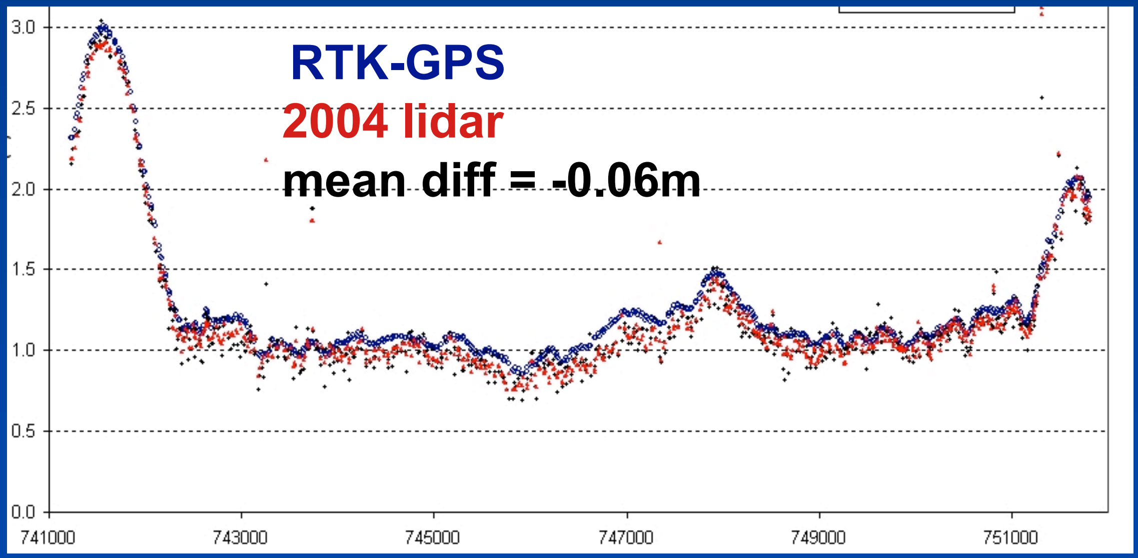
profile curvature slope



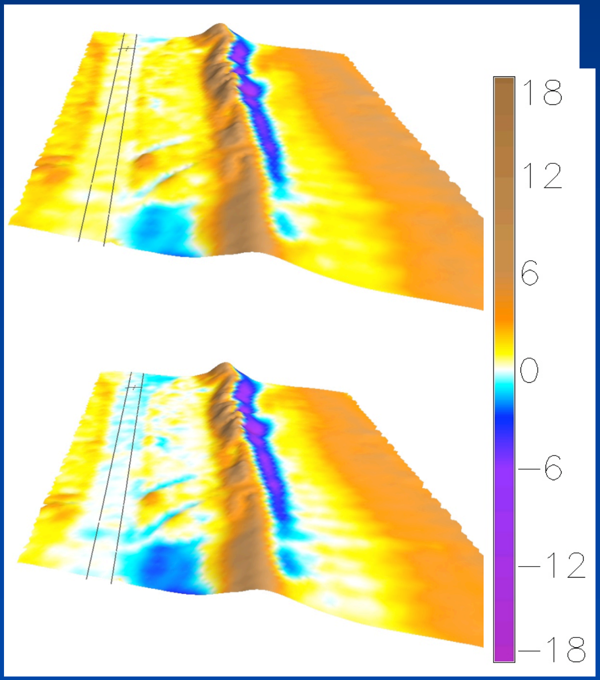
z-deviations, vegetation

Analysis of systematic error





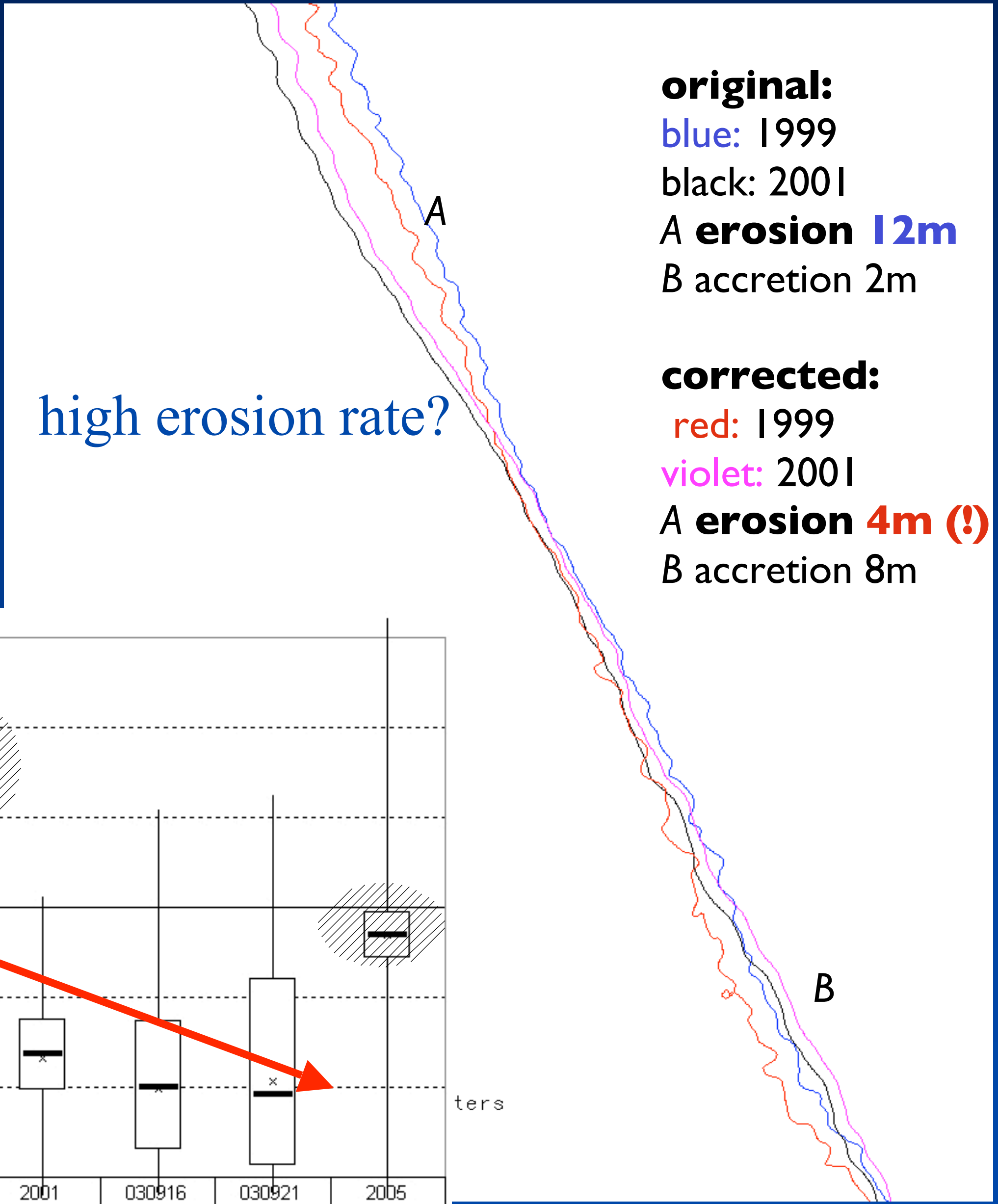
Often overlooked step in terrain analysis: Elevation difference between RTK-GPS survey (0.03m RMSE) and lidar data along centerline of a road.

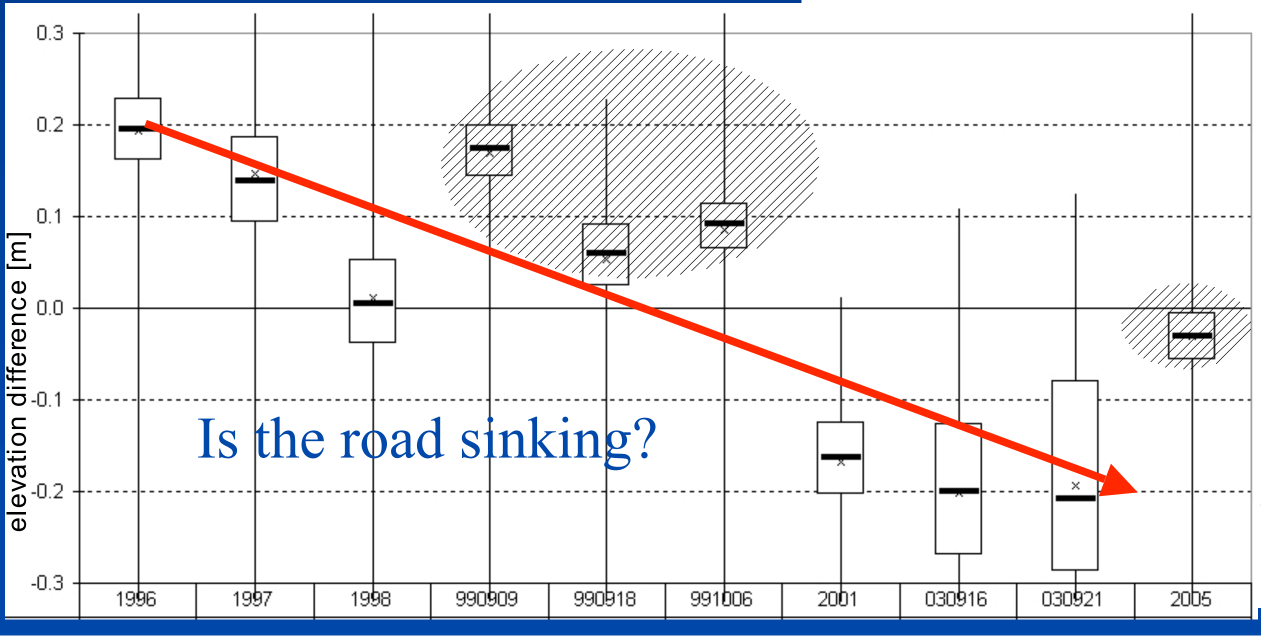


Spatial pattern of elevation difference: 2001 and 2004

Impact of systematic errors

systematic errors can lead to misleading results: examples from coastal terrain change analysis

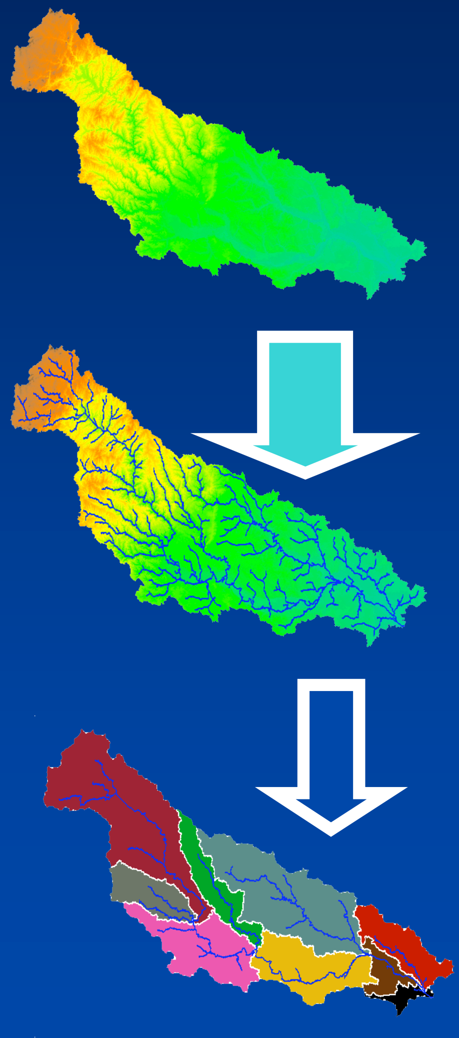




Watershed analysis

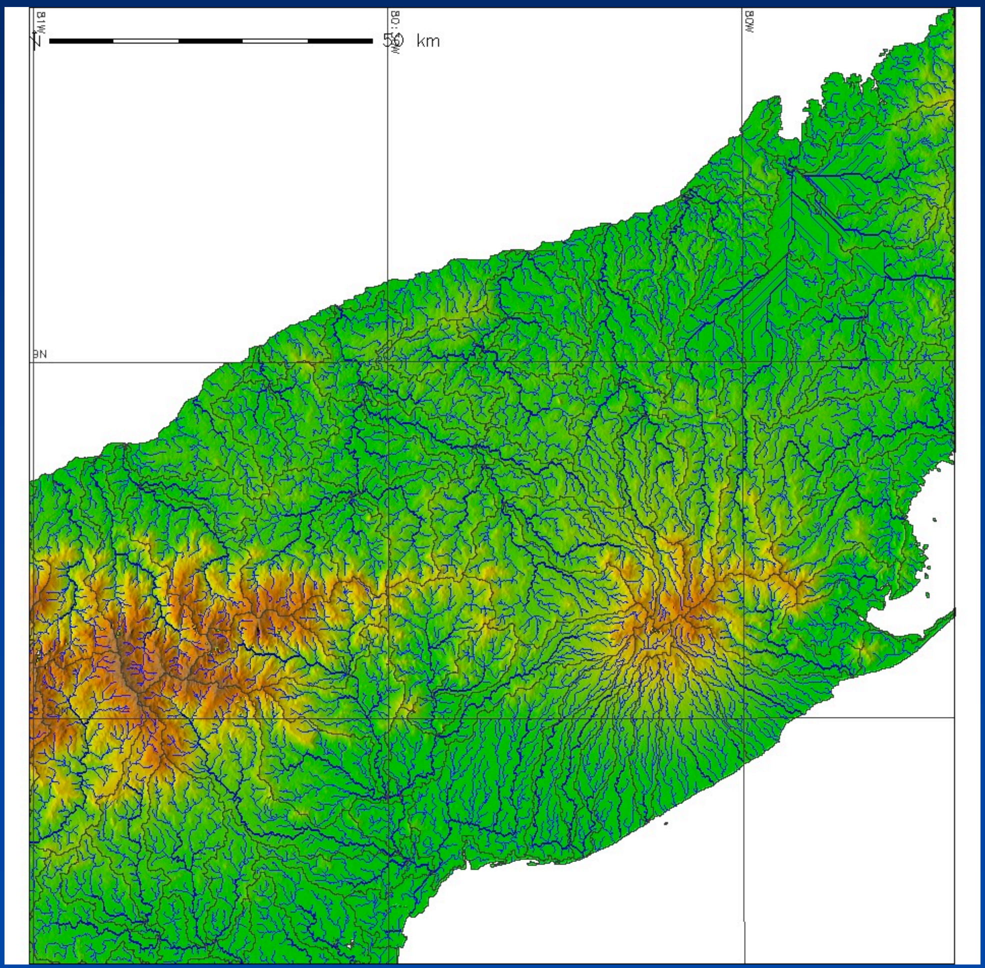
- spatial pattern of flow
- stream network extraction
- watershed boundaries

Many software tools exist,
most cannot handle massive DEMs.
As opposed to grid DEM construction,
problem cannot be solved easily
by splitting area into smaller segments

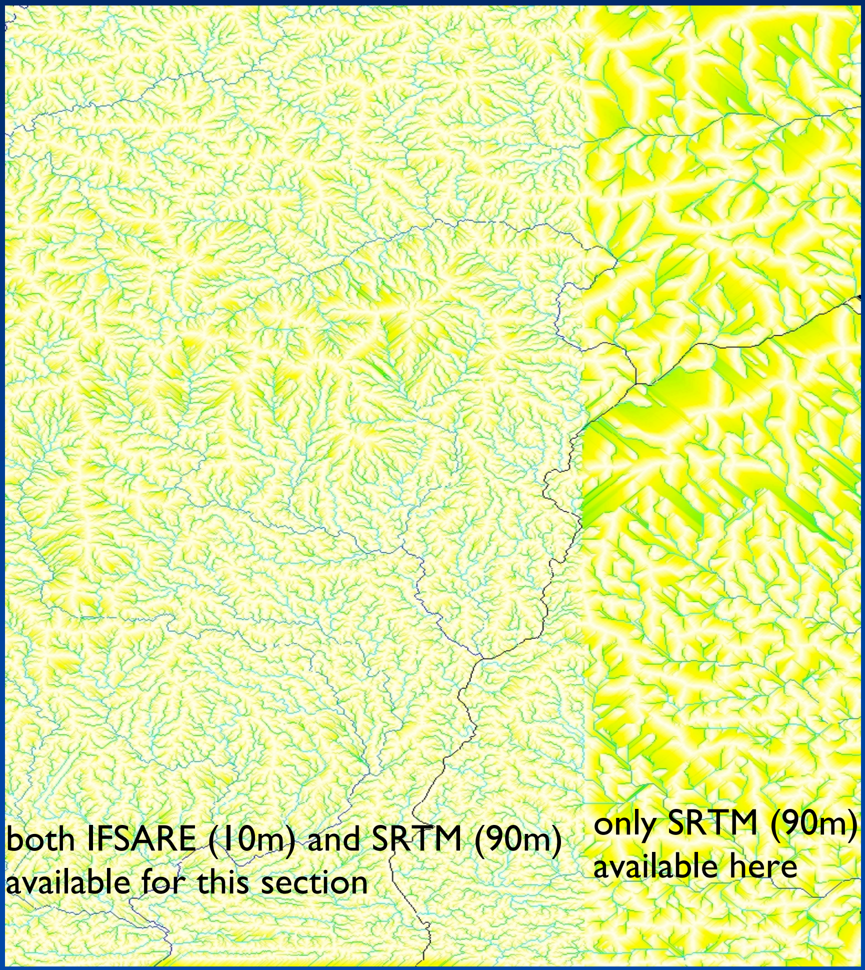


Stream networks from SRTM and IFSARE

Stream network and watershed boundaries from tiled SRTM DEM : r.watershed



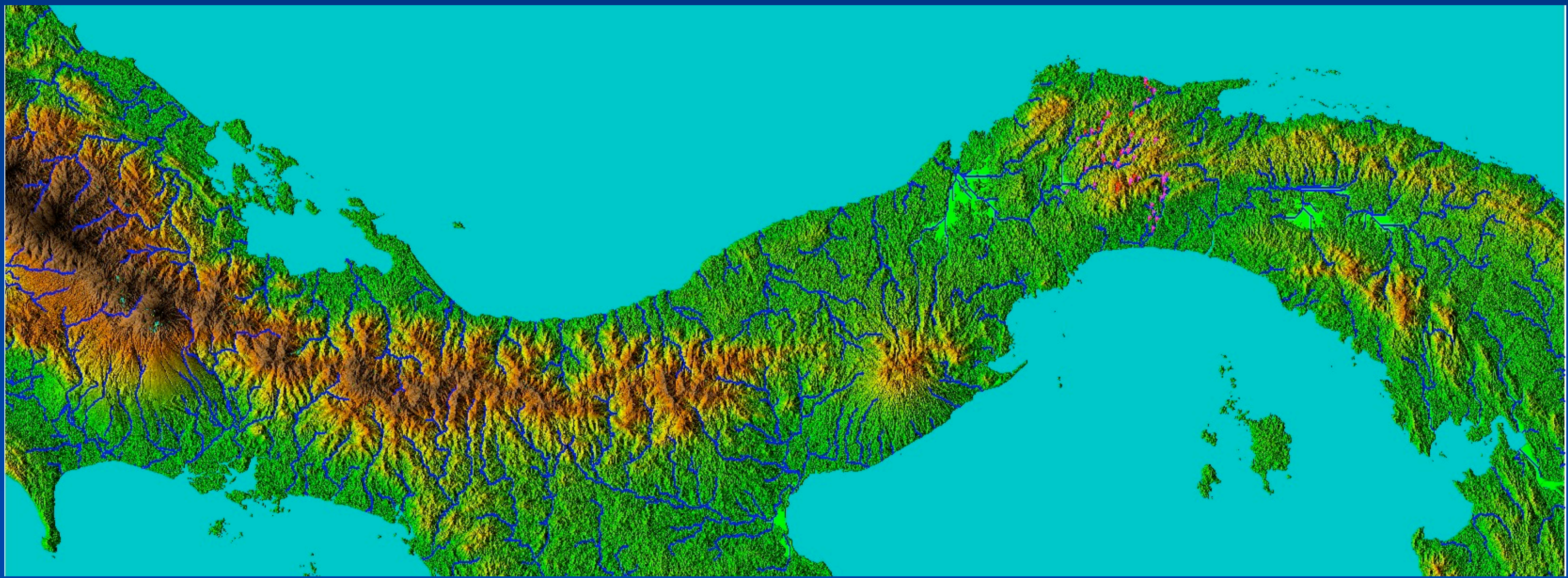
Detail of stream networks from SRTM 90m and IFSARE 10m DEMs patched together and reinterpolated to 30m resolution



time consuming procedure for entire Panama

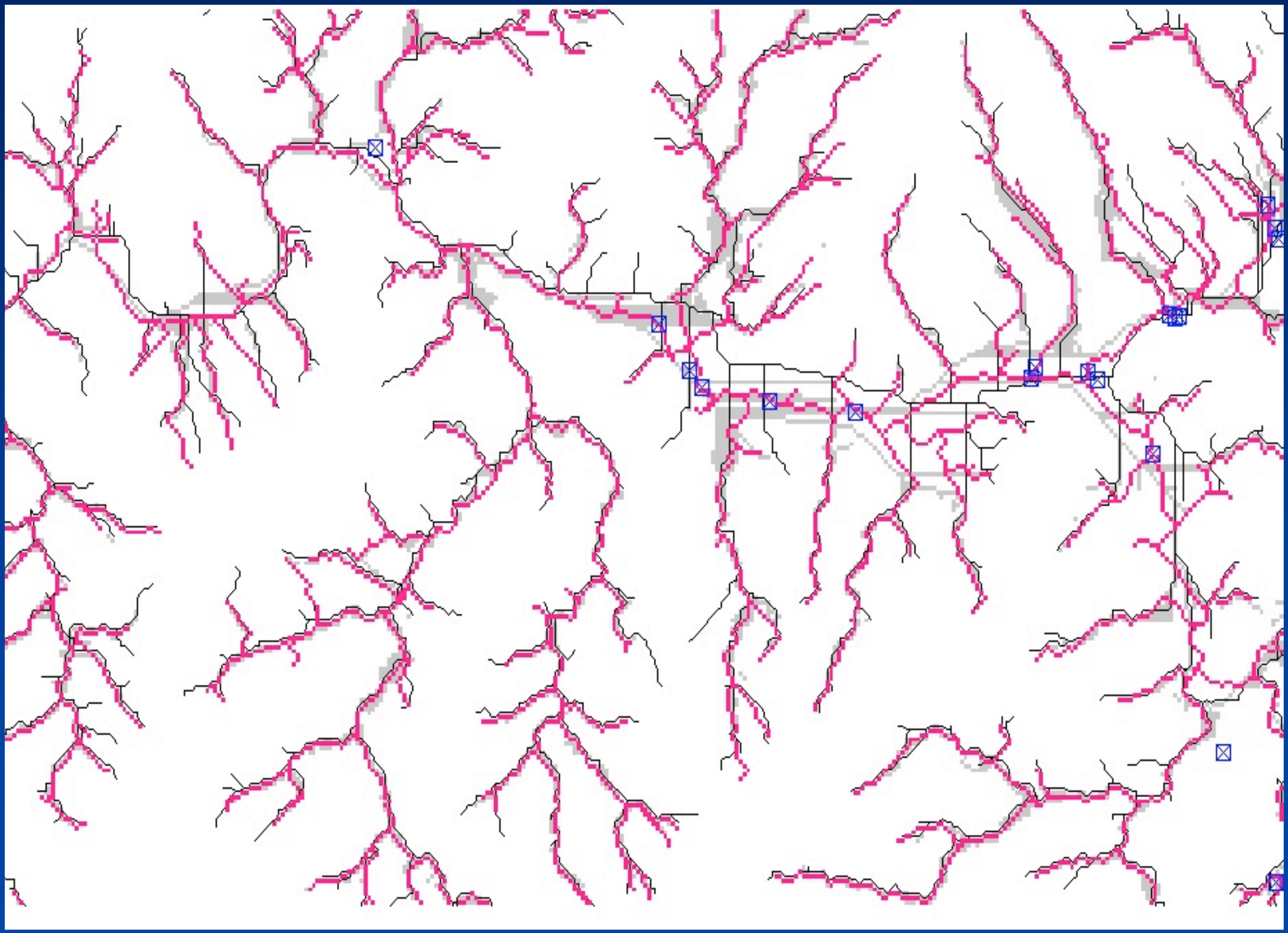
IFSARE and SRTM data analysis

Process the entire state in a single run :
SRTM - 7400x3600 DEM at 90m res. for entire Panama,
IFSARE - 10800x11300 DEM at 10m res. for the Panama canal section
Streams can be extracted in 3-4 hours:
r.terraflow, r.mapcalc, r.to.vect

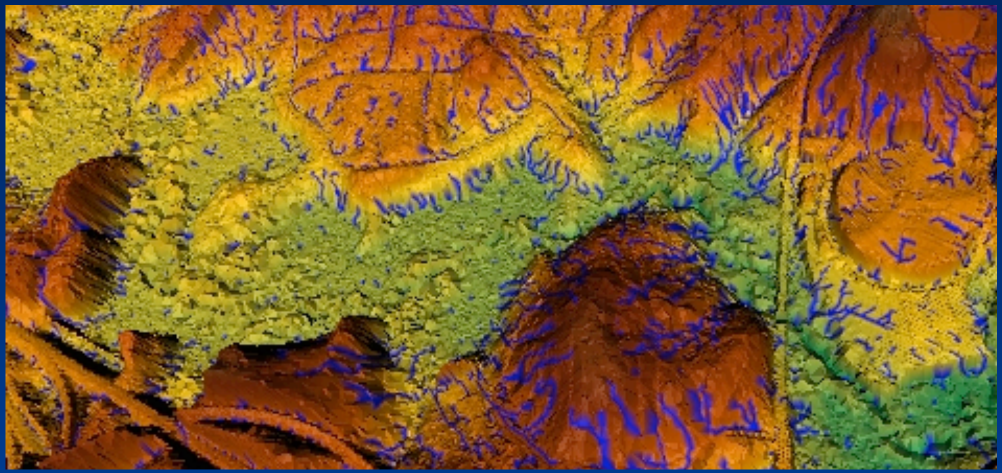


Impact of sink filling: SRTM

- r.watershed
- r.terraflow
- rivertools
- measured sites



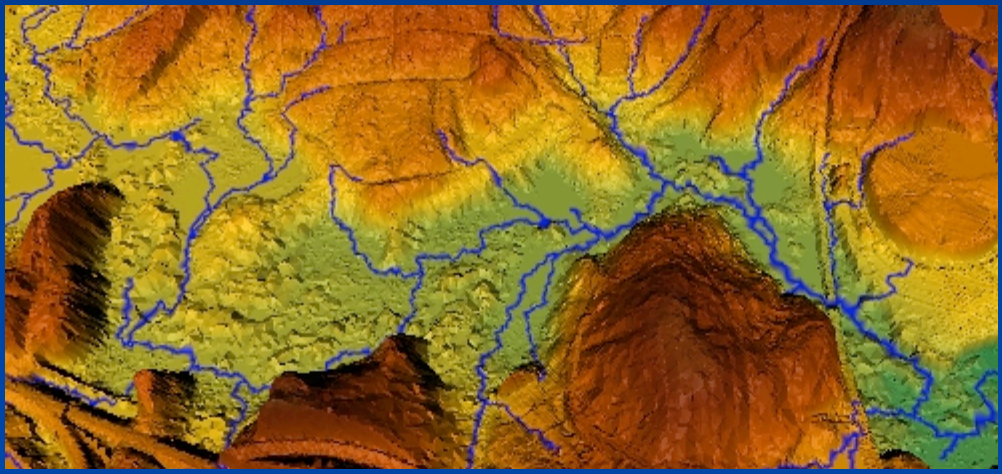
Coping with depressions: Lidar



natural and artificial

depressions and structures (bridges) impede

flow-routing



Most common approach:

depression filling

Flooding in Sort(N) I/Os

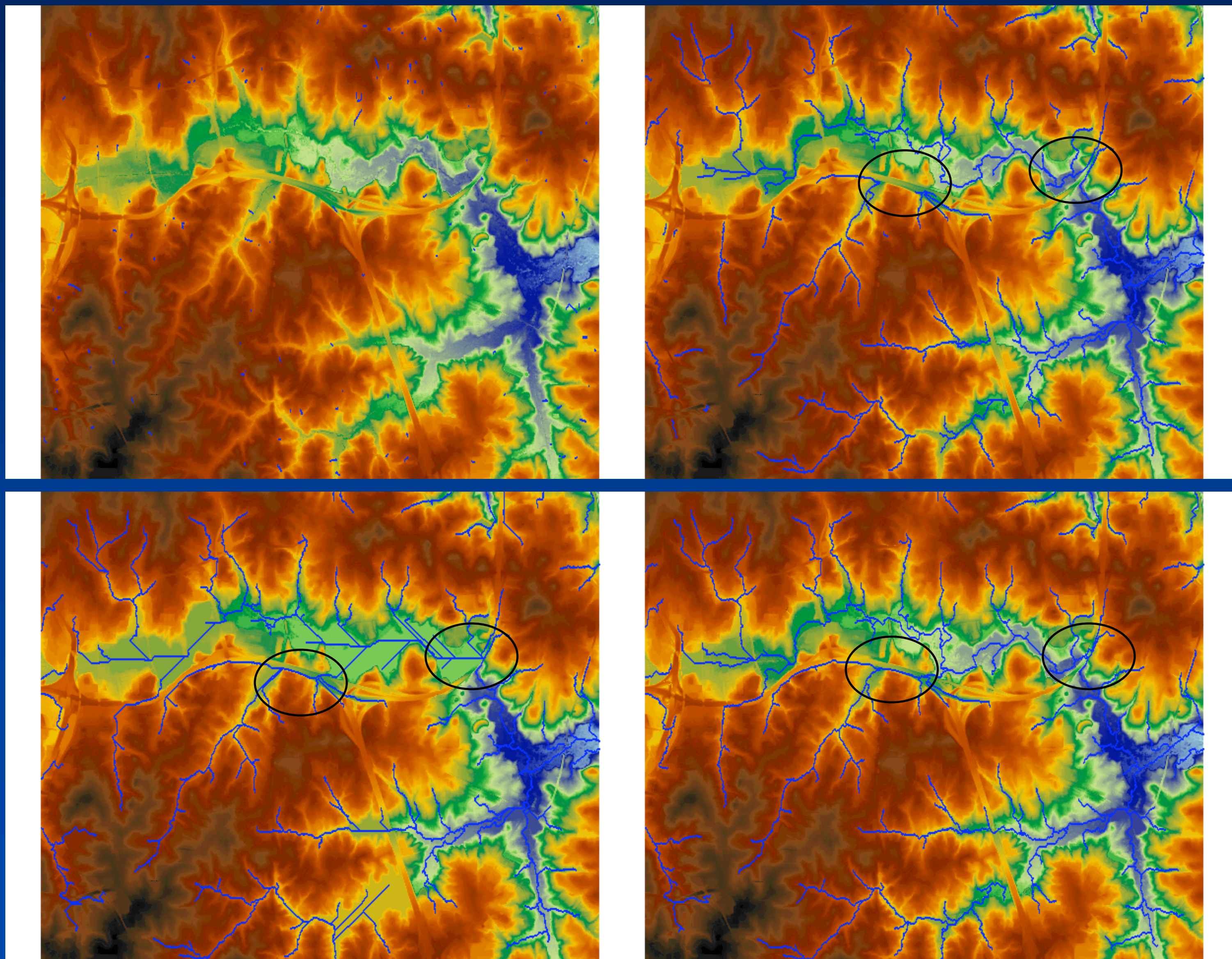
Depressions: real features and noise

- Identifying minima likely due to noise
- Don’t want to remove real features
  - Topological persistence [ELZ 02]
  - Computed in Sort(N) I/Os

Example of real depression type feature: quarry

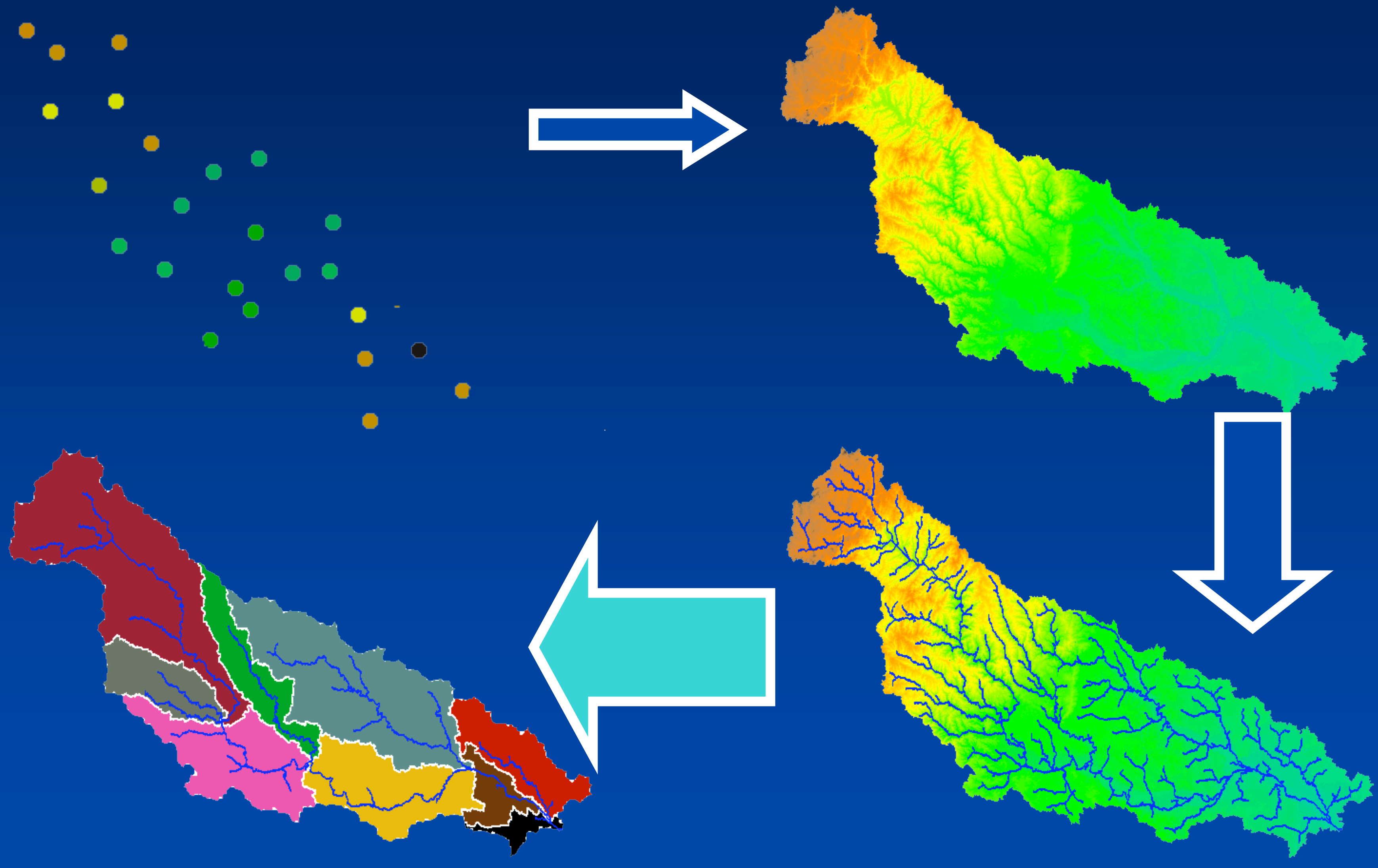
Flowrouting through structures

Filling



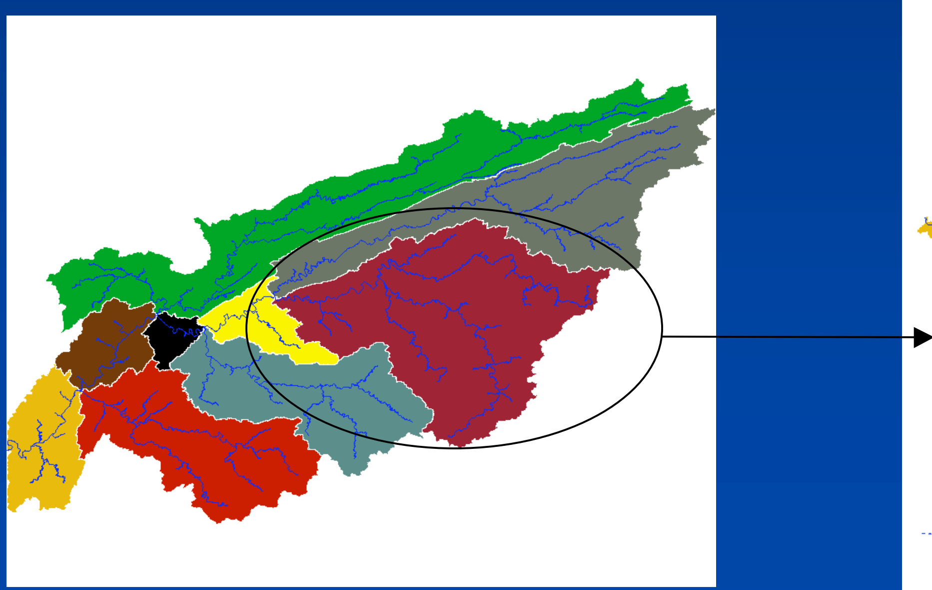
Carving

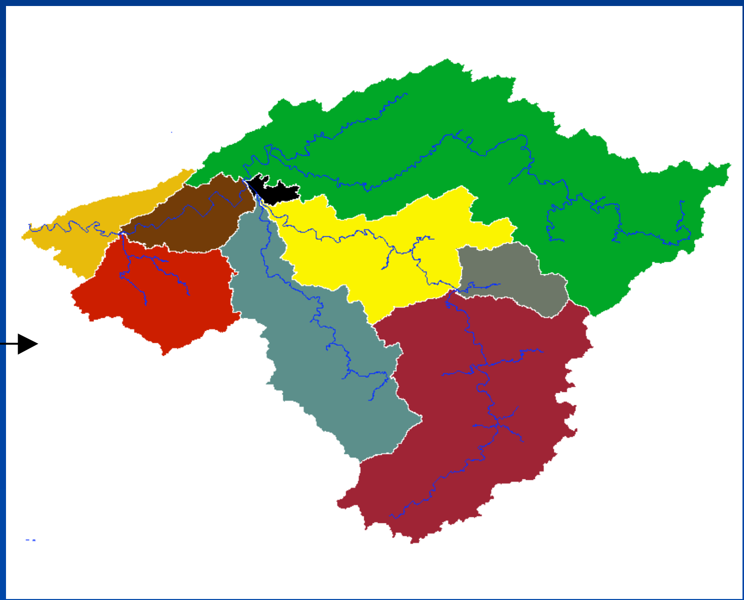
Hierarchical Watershed Decomposition



Watershed Hierarchies

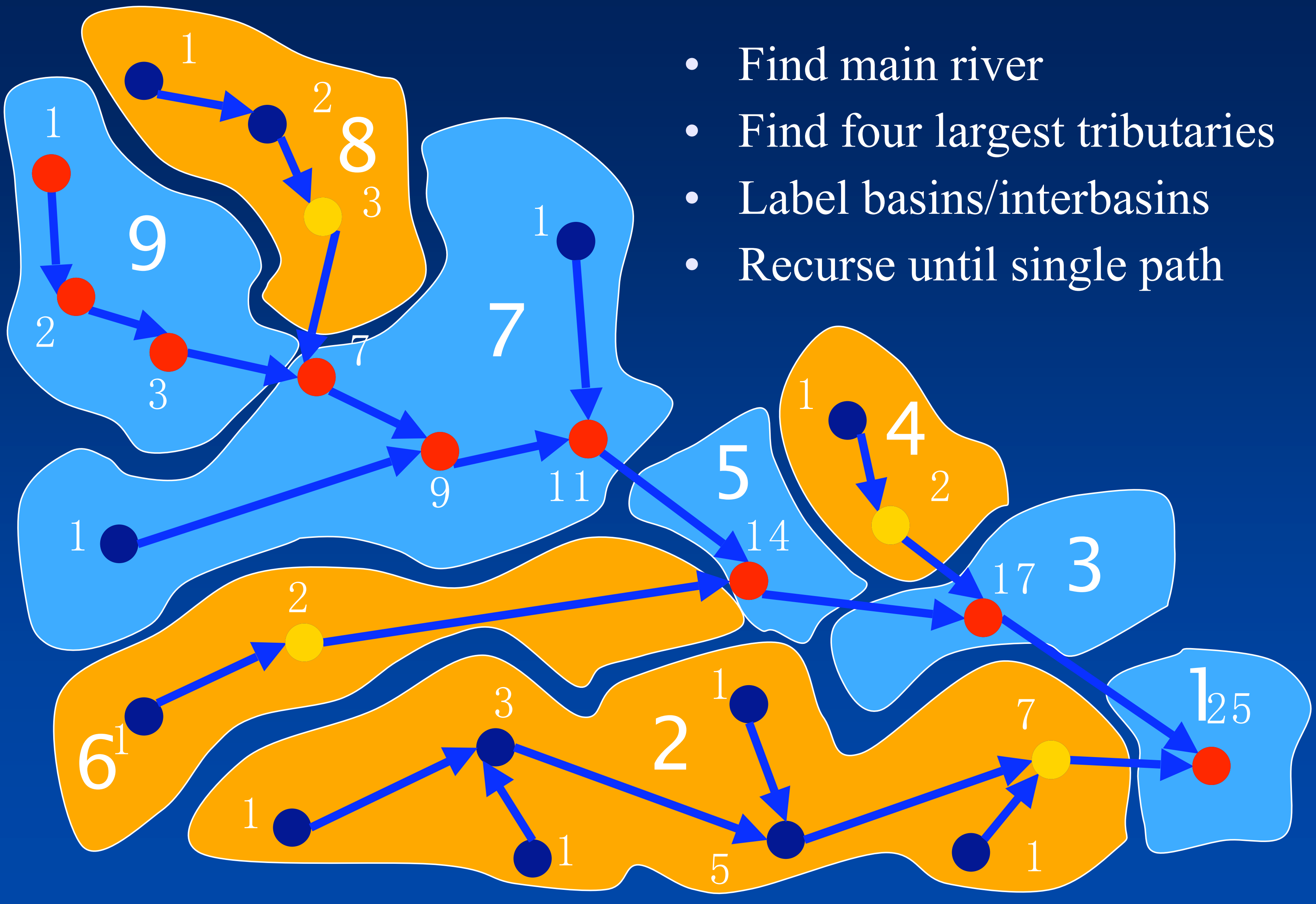
- Decompose a terrain into a hierarchy of hydrological units
- All water in HU flows to a common outlet
- Hierarchy provides tunable level of detail
- Method used: Pfafstetter [VV99]
- Want a solution scalable to large modern hi-res terrains



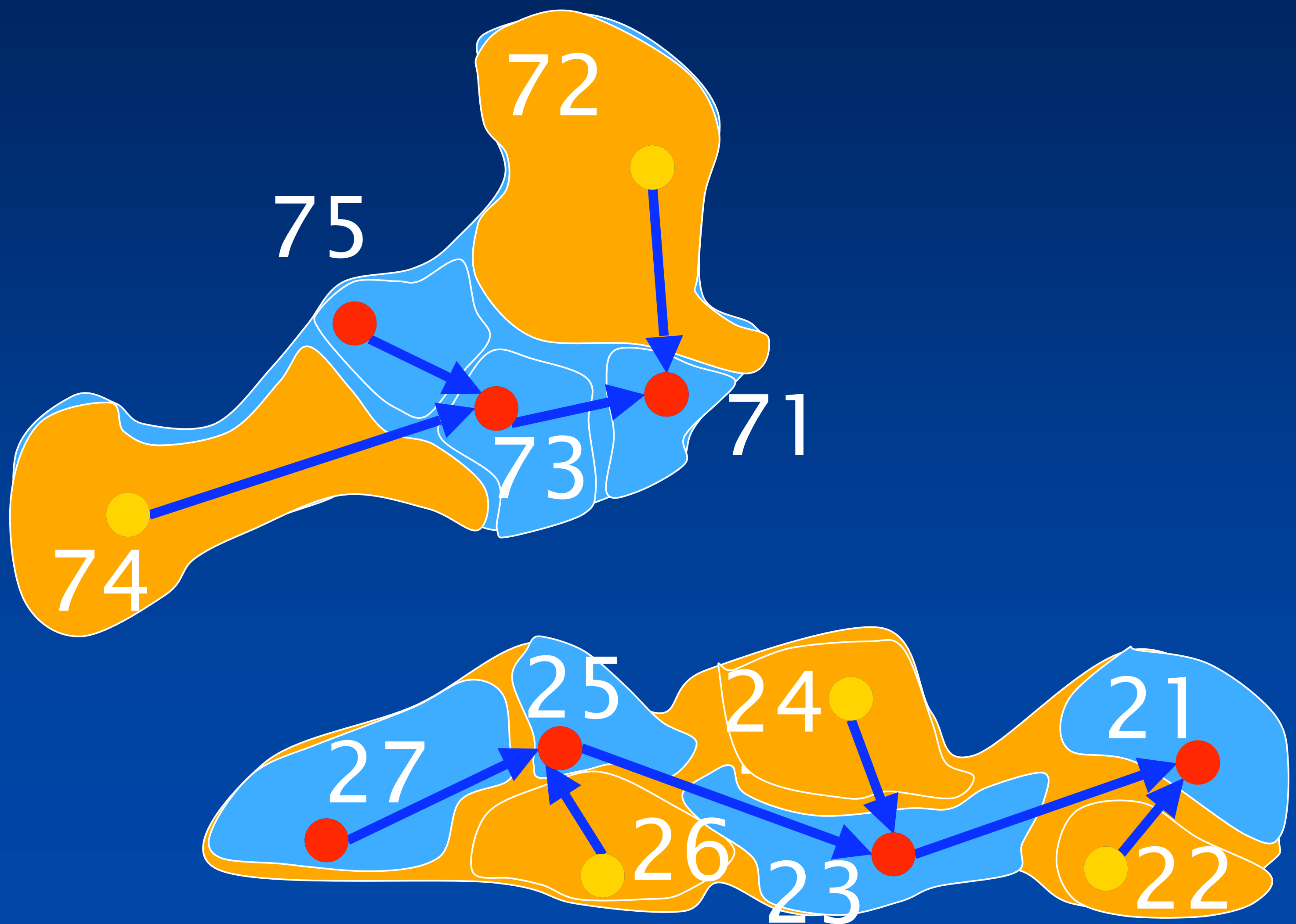


Pfafstetter

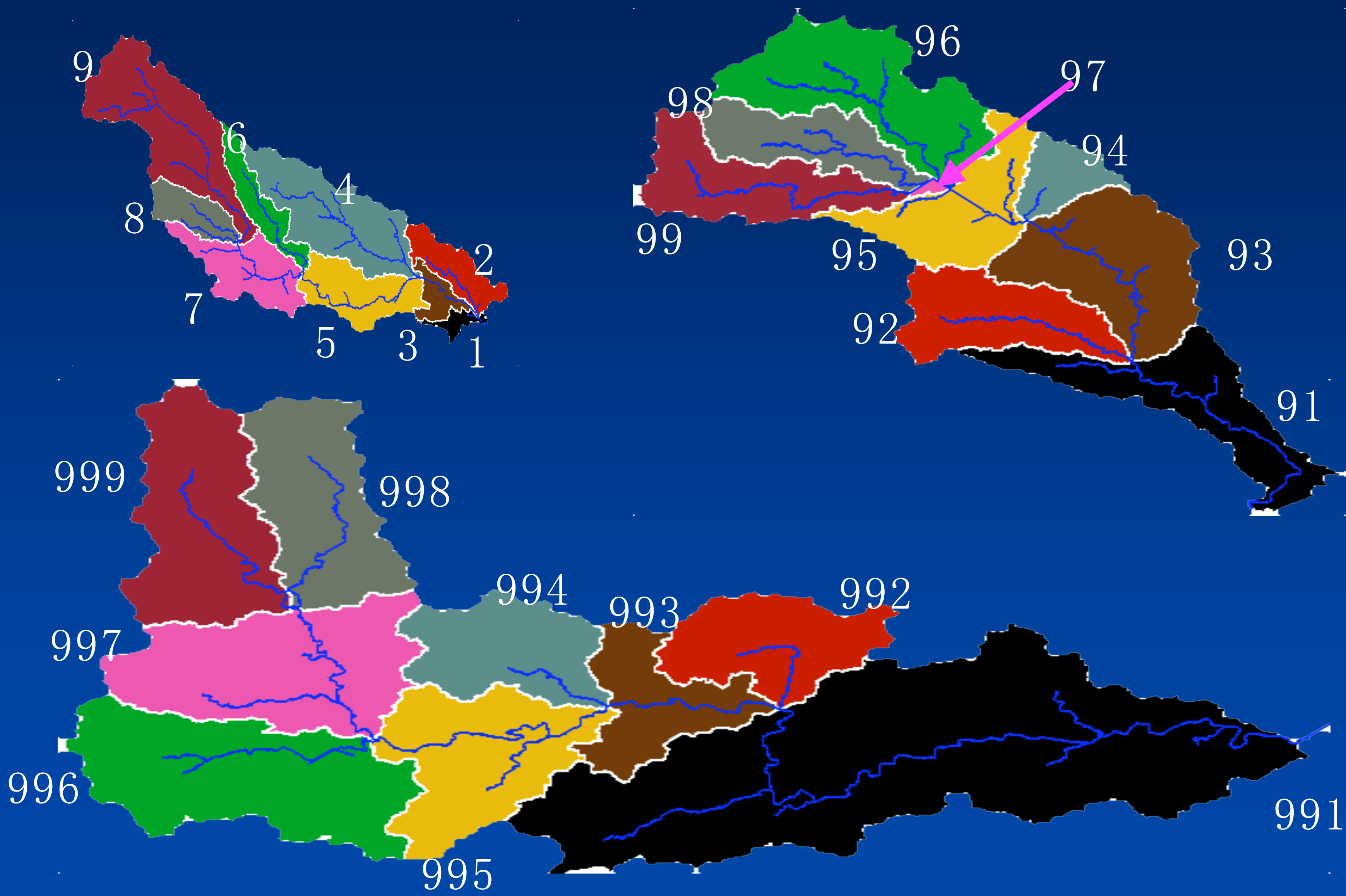
- Find main river
- Find four largest tributaries
- Label basins/interbasins
- Recurse until single path



Recurse



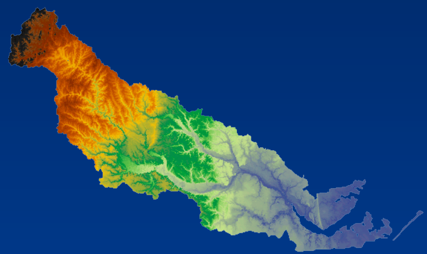
Example Watershed Boundaries



Implementation

- TPIE: C++ primitives for I/O-efficient algorithms
- GRASS: Open Source GIS
- Interpolation: Regularized spline with tension (in GRASS)
- Data:
  - North Carolina LIDAR
    - Neuse river basin: 400 million points (NC Floodmaps)
    - Outer banks coastal data : 128 million points (NOAA CSC)
  - USGS 30m NED

Grid Construction Results



Resolution (ft)	40	20	10
Grid cells x10^6	221	885	3542
Points x10^6	205	340	415
Total time	12h32m	14h46m	26h52m
Time spent(%)			
Build quad tree	8.9	7.1	5.7
Find neighbors	31.6	32.4	29.1
Interpolate	58.8	58.5	59.3
Write output	0.7	2	5.9

Sample Watershed Results

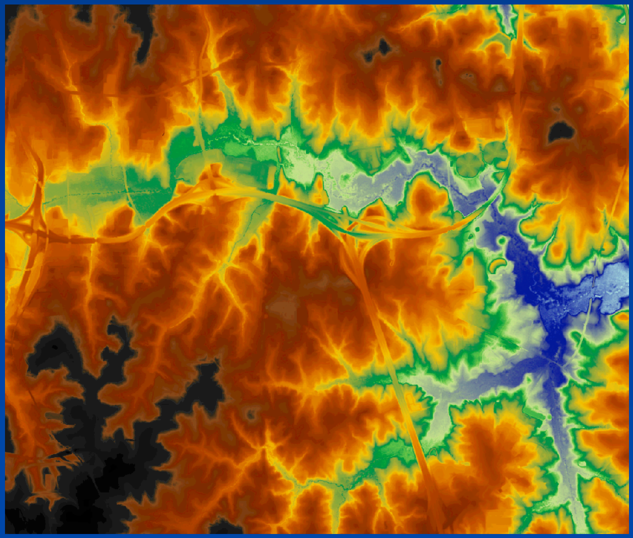
size (MB)	150	713	5819
size (mln cells)	30.8	147	396.5
total time	10m29s	58m10s	187m43s
Time spent… %			
importing data	8	7	16
sorting by flow	16	15	13
building river list	31	35	29
sorting river list	19	20	19
computing labels	7	6	6
sort by grid order	14	13	12
exporting data	5	4	5

Future Directions – Grid Construction

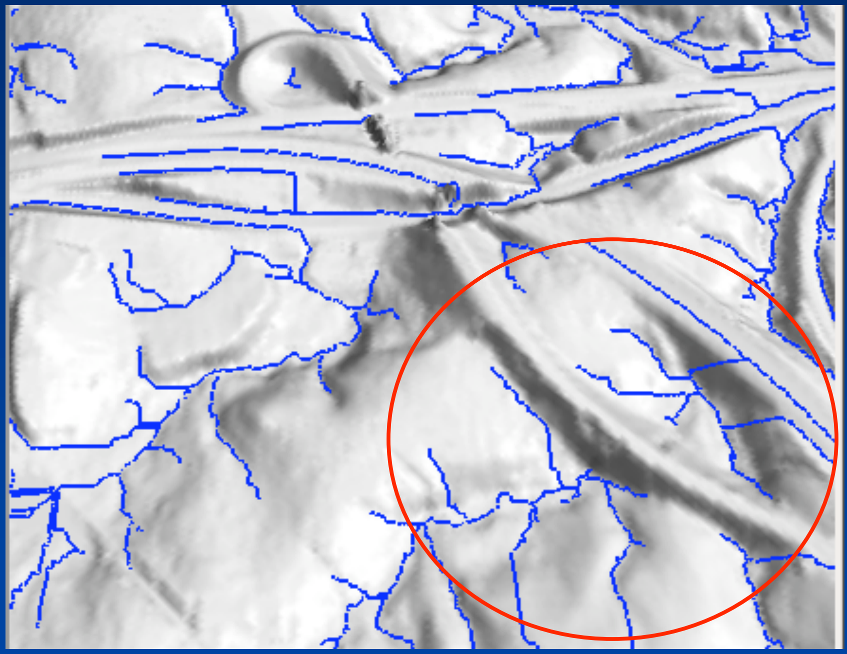
- Interpolate leaves in parallel (done for s.surf.rst in GRASS5 not in GRASS6)
- Test other interpolation methods
- Test with more data sources: much higher density (new coastal data, Phase II NCFlood)
- Finding the optimal resolution

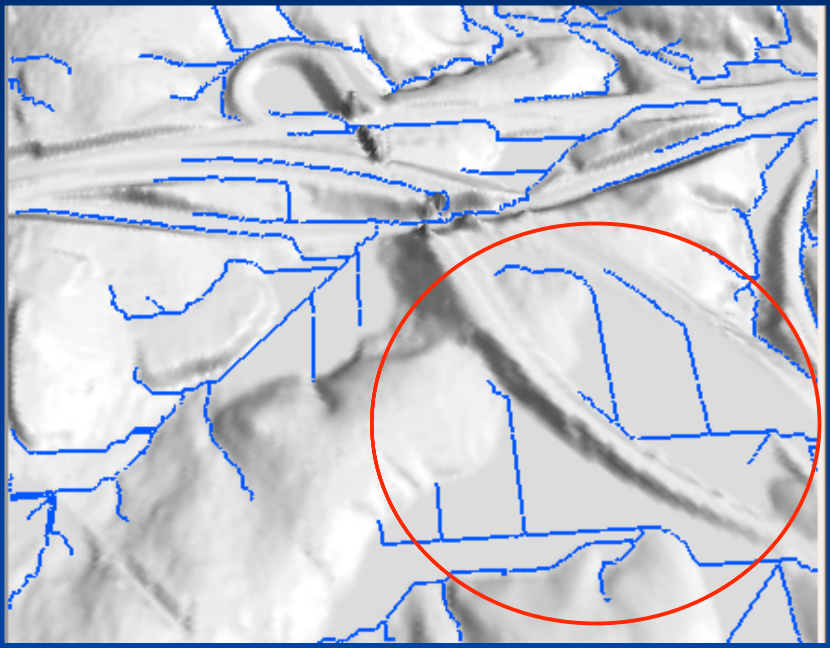
Future Directions – Flow Routing

- Bridge detection/removal
- Other flow routing methods
- Flow routing on flat surfaces
- Comparing flow networks



Flow Routing and Bridges





Future Directions – Watershed Hierarchies

- Comparison of hierarchies at different resolutions
- Terrain simplification
- Support for upstream downstream basin queries
- Point and click watershed extraction

Basic research tech. transfer

How to get from research code to robust, user friendly implementation ?

What works the best?

- integration with large open source project, e.g. GRASS
- linking with industry standard, proprietary software
- stand alone research program

Thanks!