Research Summary: Siting and ODETLAP

Wenli Li[1*], W. Randolph Franklin[1], Salles V. G. Magalhães[12], Marcus V. A. Andrade[2]

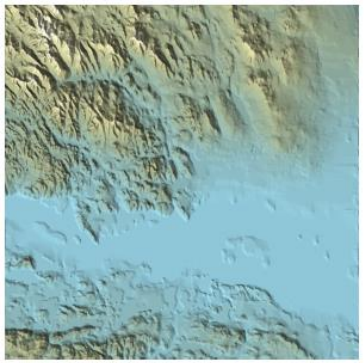
[1]Rensselaer Polytechnic Institute

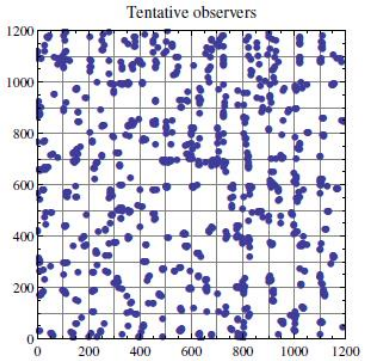
[2]Universidade Federal de Viçosa

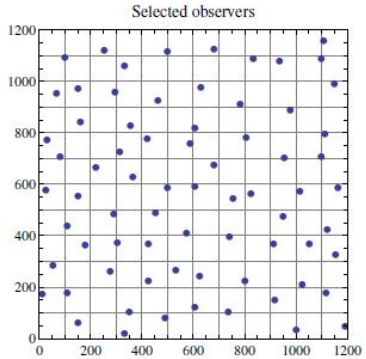
[*]liw9@rpi.edu

Multiple observer siting – Workflow

- Purpose: placing observers to cover targets on a terrain
- Workflow: *VIX* -> *FINDMAX* -> *VIEWSHED* -> *SITE*

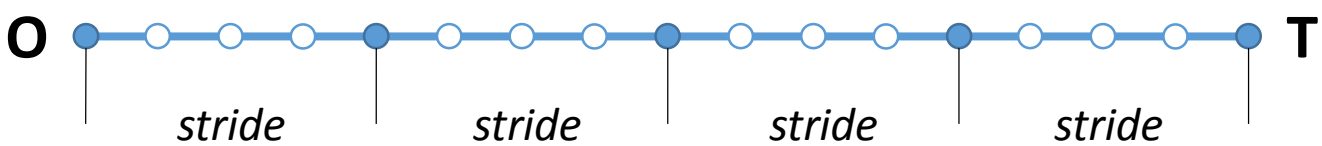






Multiple observer siting – Optimization 1

- *VIX*
  - Algorithm
    1. For each point
    2. Pick a number of random targets
    3. Compute the ratio of visible targets
  - Approximate visibility



  - Fixed *stride*: $1, 2, 4, 8, \ldots$
  - Increasing *stride*: 2^i

Multiple observer siting – Optimization 2

- *SITE*
  - Algorithm
    1. While not stop
    2. Compute the area of $V \cup C$ for the viewshed V of each unused observer
    3. Add the observer with the largest area
    4. Update the cumulative viewshed $C = C \cup V$
  - $Area(V \cup C)$: $O(n^2)$
  - $Area(V - C_V)$: $O(roi^2)$
  - Compute for unused observers within $2 \times roi$ of the last addition

Multiple observer siting – Parallelization

- OpenMP: compiler directives
- CUDA
  1. Compute visibility indices
  2. Select tentative observers
  3. Compute observer viewsheds
  4. Find observers within $2 \times roi$ of the last addition
  5. Compute the extra area of an observer viewshed
  6. Find the observer for addition
  7. Update the cumulative viewshed

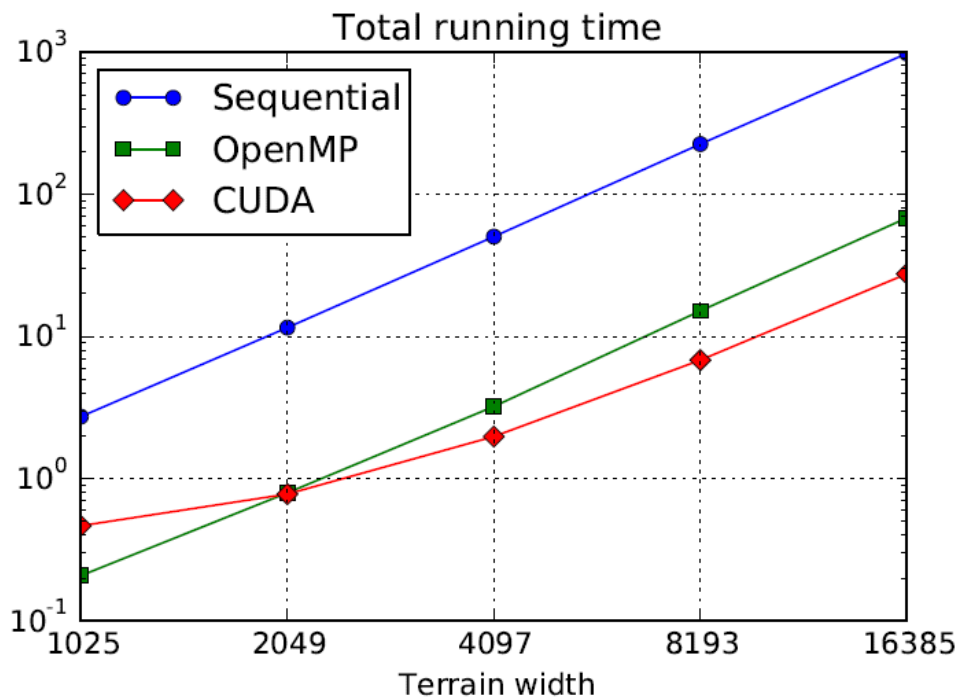
Multiple observer siting – Results 1

- 16K DEM, 26896 tentative observers
  - Running time of CUDA *VIX*
  - Percentage coverage
  - Number of selected observers

	30 targets			120 targets		
stride	Time (s)	Coverage	Observers	Time (s)	Coverage	Observers
1	87	95.5	25224	340	96.5	25108
2	50	95.5	25228	193	96.5	25130
4	30	95.5	25272	114	96.5	25137
8	19	95.5	25298	71	96.4	25214
2^i	11	96.0	25353	37	96.8	25239

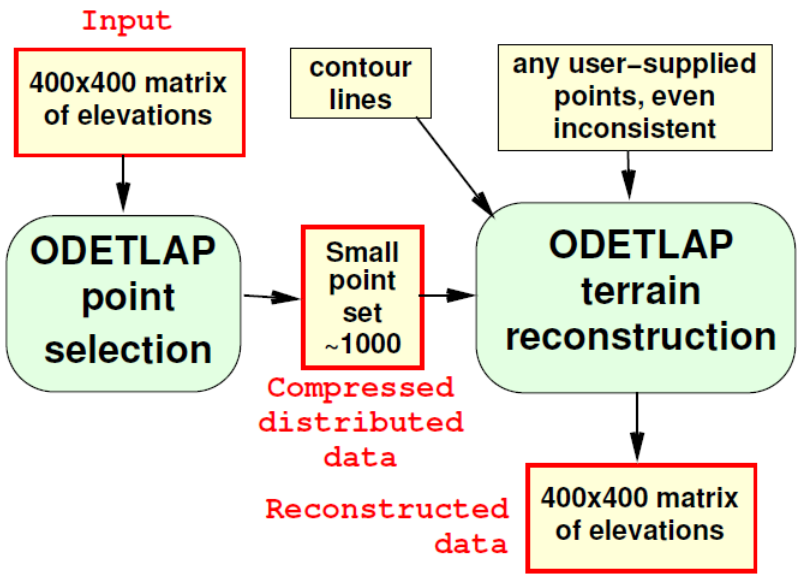
Multiple observer siting – Results 2

- Running time
  - Dataset: 1K, 2K, 4K, 8K, 16K DEMs
  - Hardware: two 8-core Xeon E5, Tesla K20Xm
- Speedup
  - OpenMP: 13—16
  - CUDA: 6—35



ODETLAP – Overview

- **Overdetermined** Laplacian Partial Differential Equations
- Two components: interpolation and lossy compression



(Franklin et al. CUDA-accelerated HDODETLAP: Lossy high dimensional gridded data compression. *Modern Accelerator Technologies for Geographic Information Science*, 2013.)

ODETLAP – Interpolation

- Two types of equations
  - Given a domain $m \times n$ and known points $\{(x_i, y_i, v_i)\}_k$
  - Averaging equation

$$4z(x,y) - z(x-1,y) - z(x+1,y) - z(x,y-1) - z(x,y+1) = 0$$

  - Smoothing factor R

$$4Rz(x,y) - Rz(\dots) - Rz(\dots) - Rz(\dots) - Rz(\dots) = 0$$

  - Known-value equation

$$z(x_i, y_i) = v_i$$

- Overdetermined system

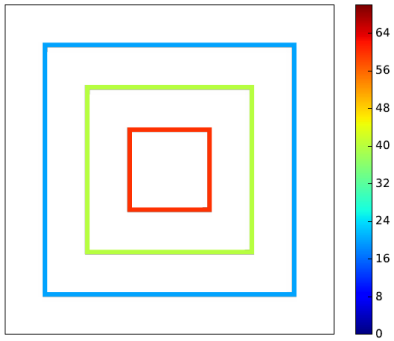
$$\boldsymbol{A}_{(mn+k)\times mn}\boldsymbol{x} = \boldsymbol{b}$$
$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$$

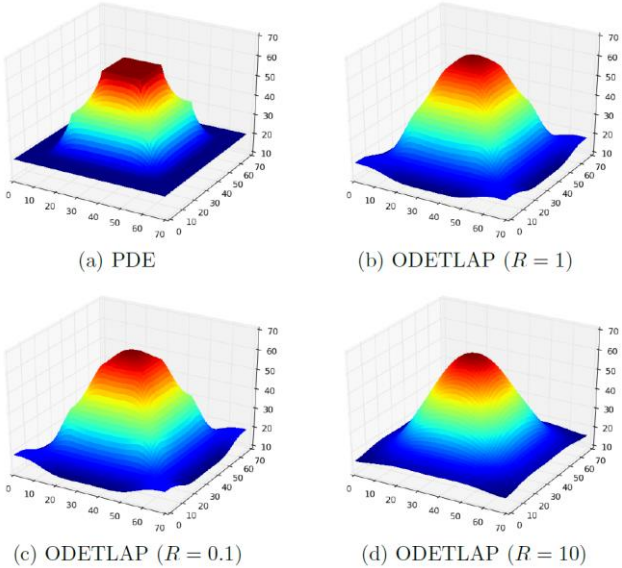
ODETLAP – Lossy compression

- Xie et al. Surface compression using over-determined Laplacian approximation. *SPIE 6697, Advanced Signal Processing Algorithms, Architectures, and Implementations XVII*, 2007.
- ODETLAP-based compression
  1. Select a set of initial points P using TIN construction
  2. Interpolate P using ODETLAP
  3. While stop condition is not satisfied
  4. Add a number of important points to P
  5. Interpolate P using ODETLAP

ODETLAP – Advantages over PDE

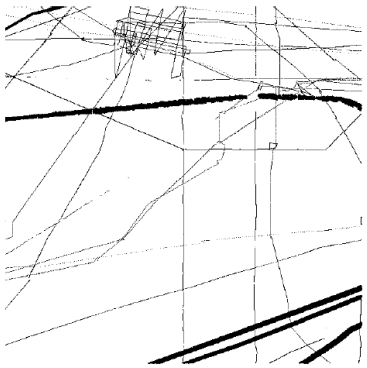
- Advantages
  - ODETLAP is overdetermined
  - ODETLAP can infer local extrema
  - Result is smoother across known points
  - R trades off accuracy vs. smoothness

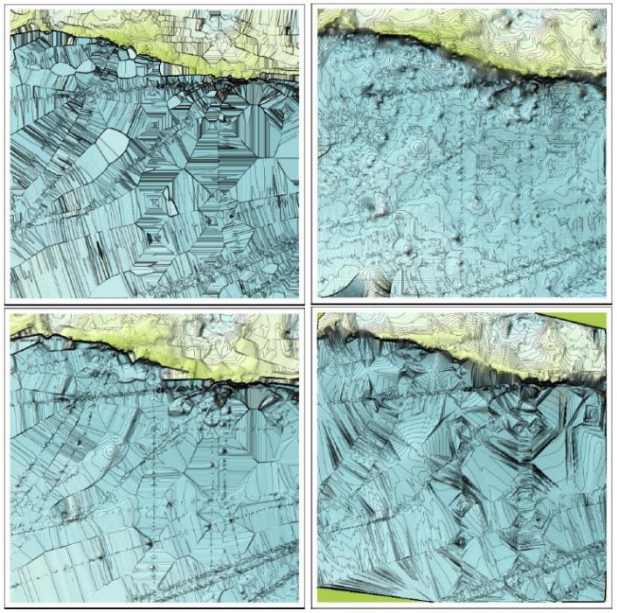


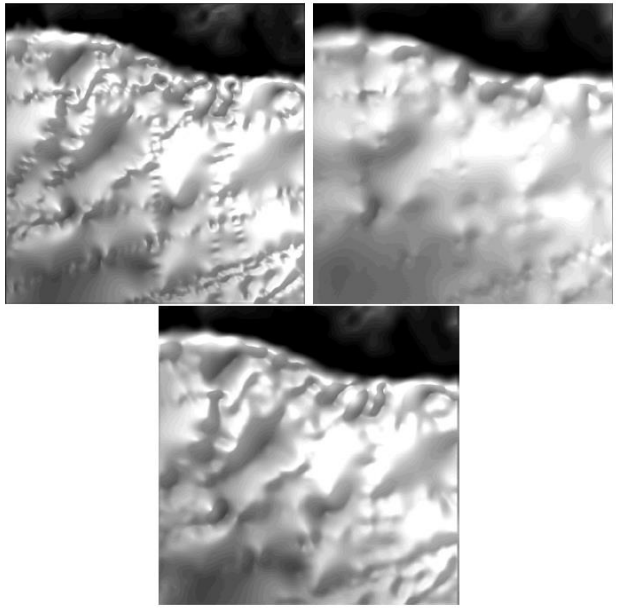


ODETLAP – Application 1

- Lau et al. Sea floor bathymetry trackline surface fitting without visible artifacts using ODETLAP. *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009.
- Lau and Franklin. Automated artifact-free seafloor surface reconstruction with two-step ODETLAP. *SIGSPATIAL Special*, 2012.

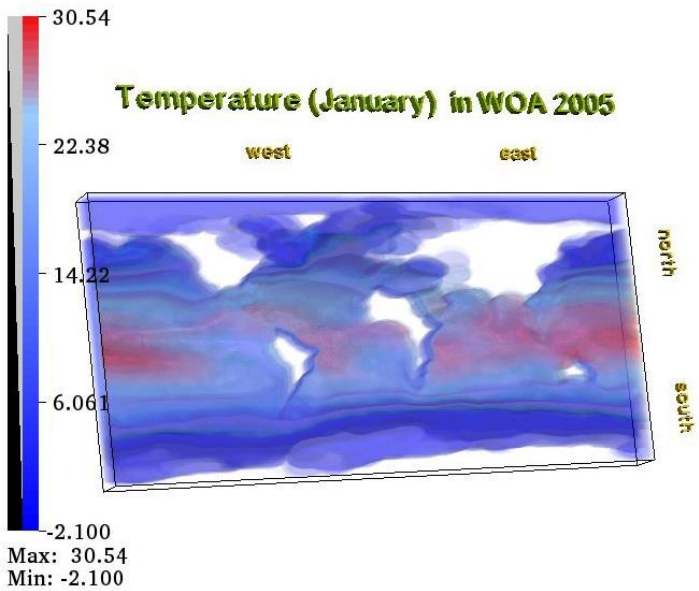




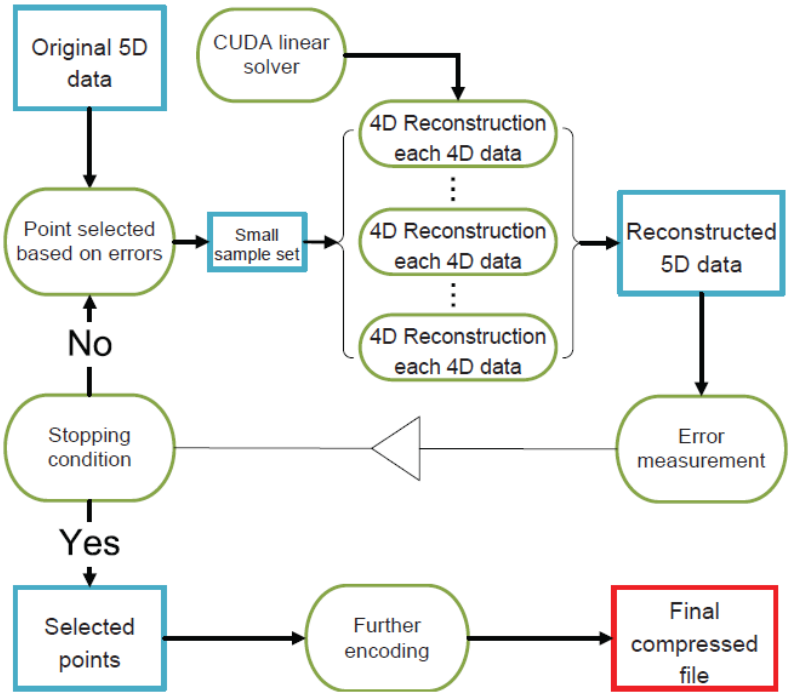


ODETLAP – Application 2

- Li et al. 3D oceanographic data compression using 3D-ODETLAP. *SIGSPATIAL Special*, 2010
- Li. CUDA-accelerated HD-ODETLAP: a high dimensional geospatial data compression framework. *PhD thesis*, 2011

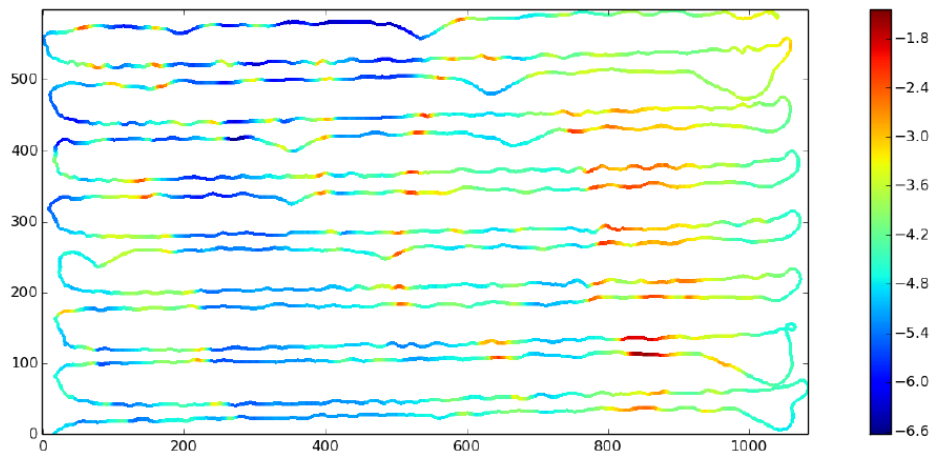


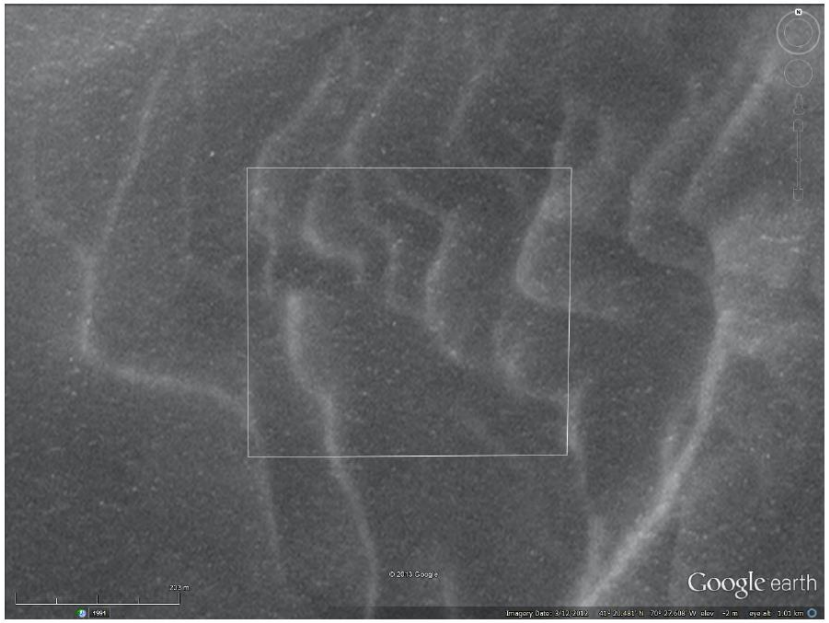
$360 \times 180 \times 24 \times 12\ (\times 4)$



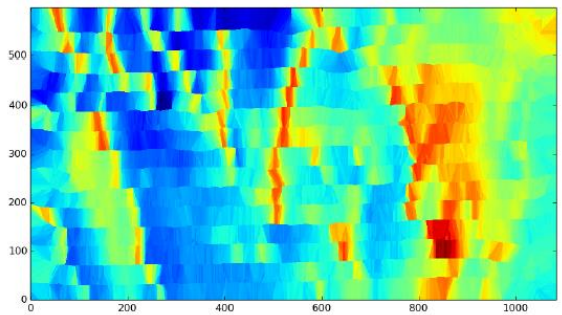
Bathymetry interpolation – A special case

- Data and image courtesy of Peter Traykovski at Woods Hole Oceanographic Institution

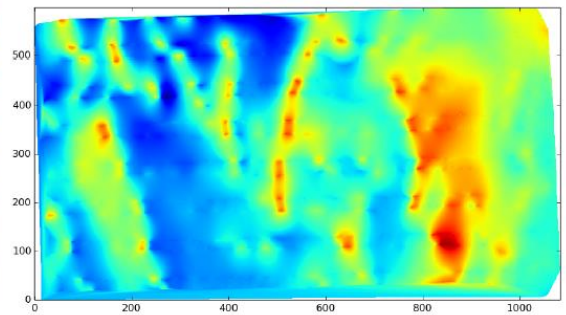




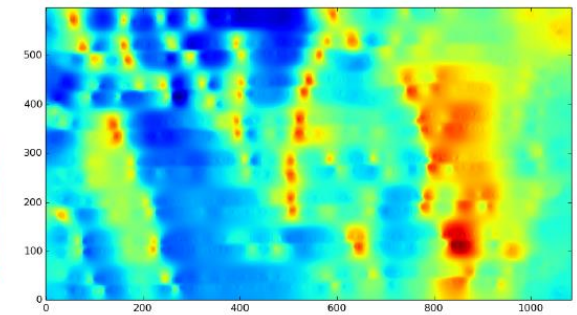
Bathymetry interpolation – Common methods



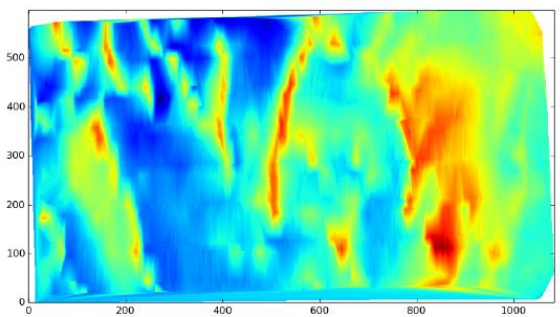
(a) Nearest neighbor interpolation



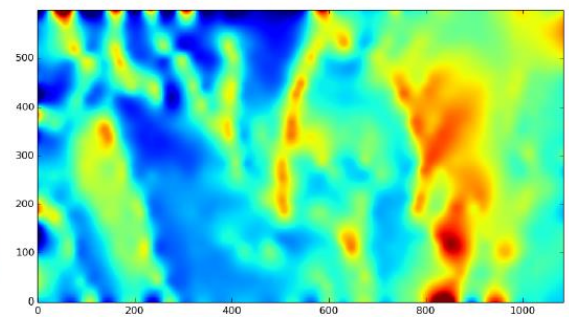
(b) Natural neighbor interpolation



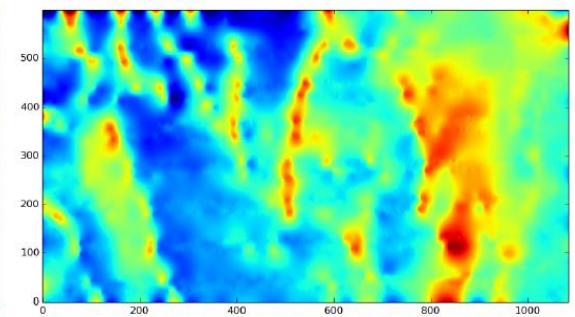
(c) Inverse distance weighting



(d) Linear interpolation



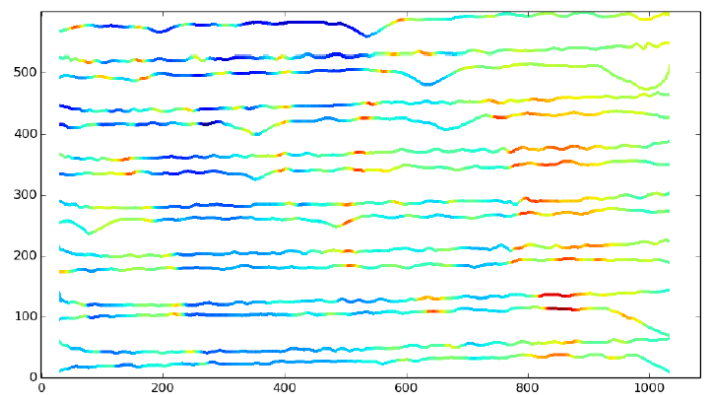
(e) ODETLAP $R = 10$

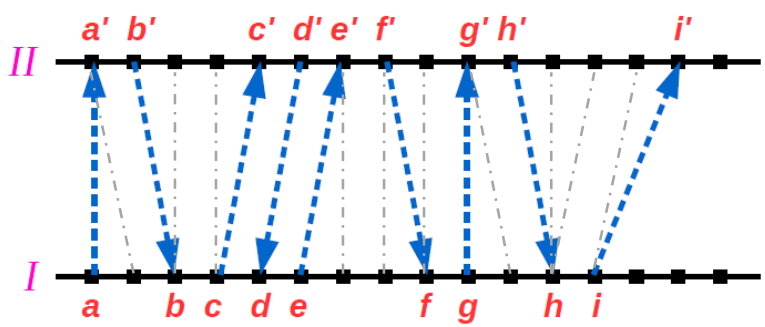


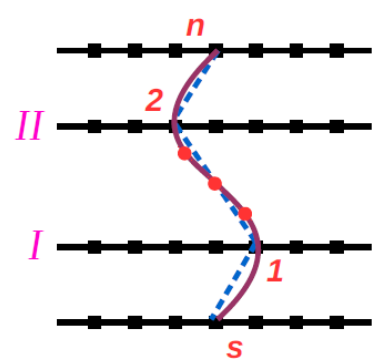
(f) ODETLAP $R = 0.1$

Bathymetry interpolation – Proposed method 1

- Computing intermediate tracklines

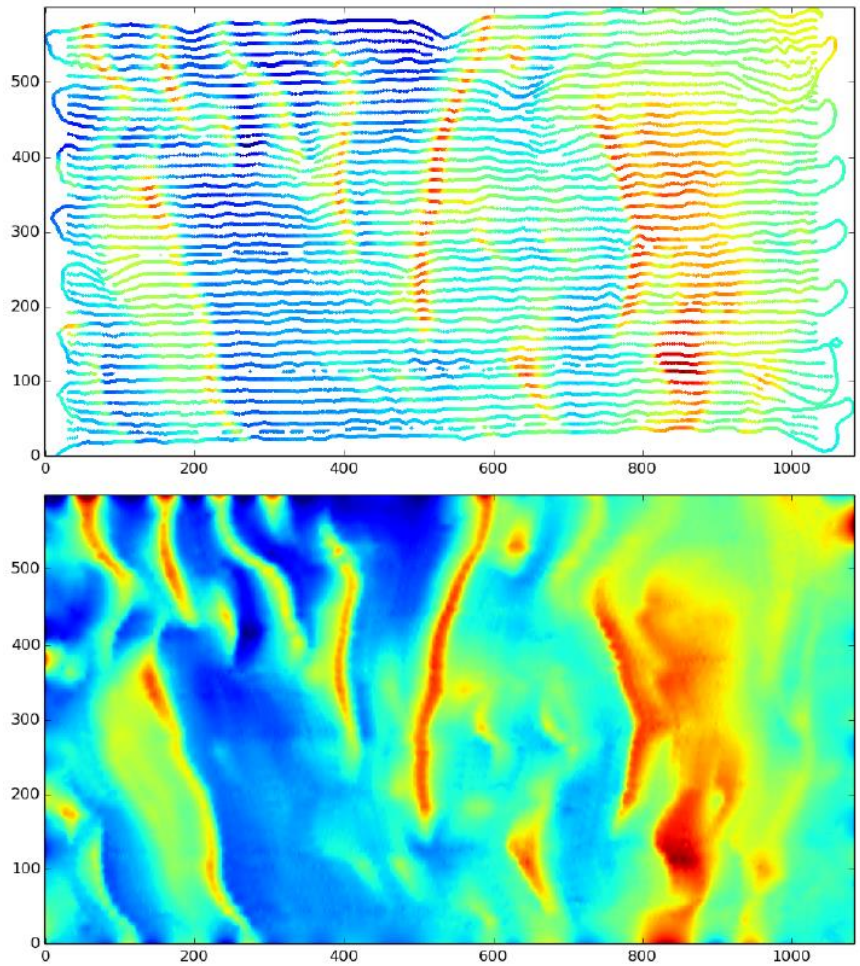






Bathymetry interpolation – Proposed method 1

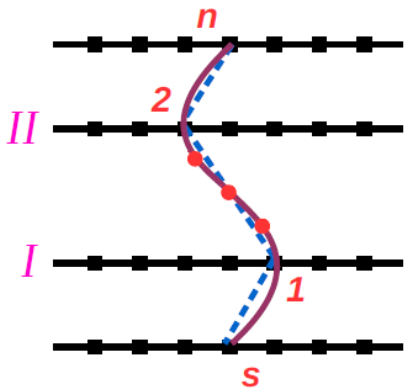
- Results



Bathymetry interpolation – Proposed method 2

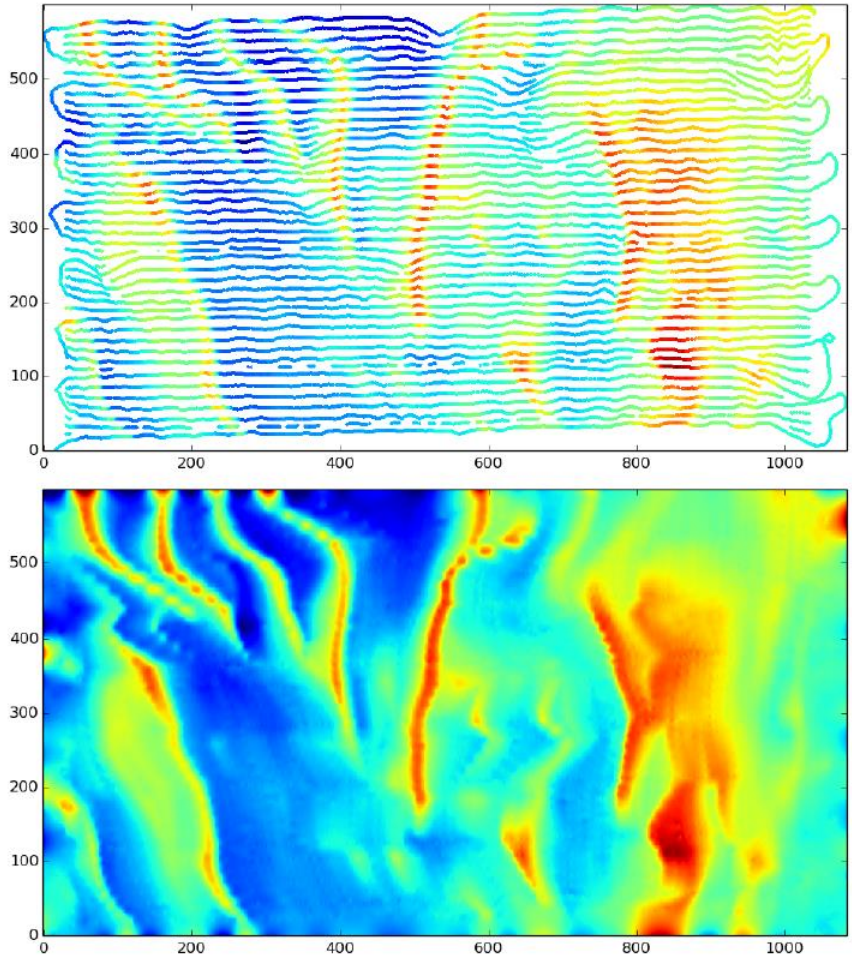
- Sequence alignment

```
G A A T T C A G T T A
| | | | | | | | | | |
G G A _ T C _ G _ _ A
0 1 0 2 0 0 2 0 2 2 0

A A A A A A A A A A A
| | | / | | / | / \ |
B B B   B B   B     B

A A A A A A A A A A A
| | | | | | | | | | |
B B B . B B . B . . B
```



Bathymetry interpolation – Proposed method 2

- Results

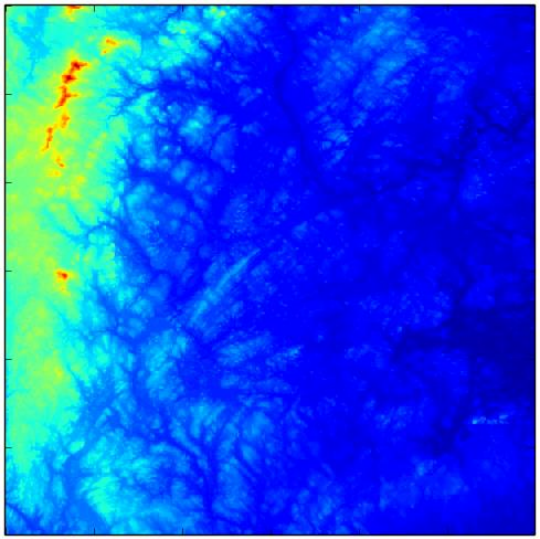


GPU-accelerated ODETLAP – Cusp

- Cusp: a parallel sparse matrix library
  - Based on Thrust: a parallel algorithms library
  - Matrix formats
  - BLAS
  - Iterative solvers: relaxation methods and Krylov subspace methods
  - Preconditioners

GPU-accelerated ODETLAP – Implementation

- ODETLAP interpolation
  - Part 1: building $\boldsymbol{A}$ and $\boldsymbol{b}$ on CPU
  - Part 2: calculating $\boldsymbol{A}^T$, $\boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{A}^T\boldsymbol{b}$
  - Part 3: solving $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$
- Data
  - N43W072, 1200×1200
  - 1% (14400) random points



GPU-accelerated ODETLAP – Sparse matrix formats

- Sparse matrix formats
  - Coordinate matrix format (COO)
  - Compressed Sparse Row matrix format (CSR)
  - ELLPACK/ITPACK matrix format (ELL)
  - Hybrid ELL/COO matrix format (HYB)
- Evaluation
  - Host $\boldsymbol{A}$, device $\boldsymbol{A}$ and $\boldsymbol{A}^T$: COO; varying device $\boldsymbol{A}^T\boldsymbol{A}$
  - Solver: Conjugate Gradient (CG) method

	COO	CSR	ELL	HYB
Part 1	0.1 s	0.1 s	0.1 s	0.1 s
Part 2	0.3 s	0.3 s	0.3 s	0.3 s
Part 3	20.6 s	19.5 s	10.2 s	10.2 s
Memory	493 MB	428 MB	422 MB	422 MB

GPU-accelerated ODETLAP – Iterative solvers

- Relaxation methods
  - Gauss-Seidel and SOR: very slow
  - Jacobi: diverged
- Krylov subspace methods
  - BiCGstab and CR: diverged
  - GMRES: slow
  - BiCG and CG

	BiCG	CG
Part 1	0.1 s	0.1 s
Part 2	0.3 s	0.3 s
Part 3	16.3 s	10.2 s
Memory	422 MB	422 MB

GPU-accelerated ODETLAP – Preconditioners

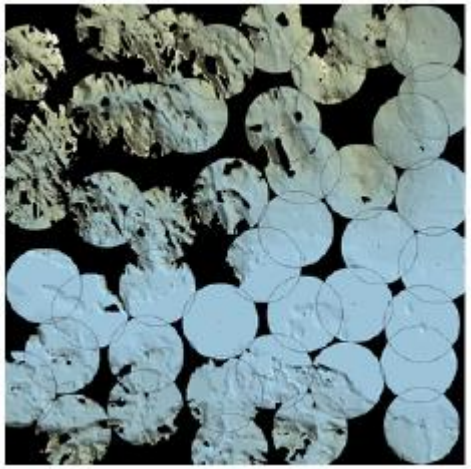
- Preconditioners
  - Smoothed aggregation preconditioner
  - Approximate Inverse (AINV) preconditioner
  - Diagonal preconditioner
- ELL matrix format and CG solver

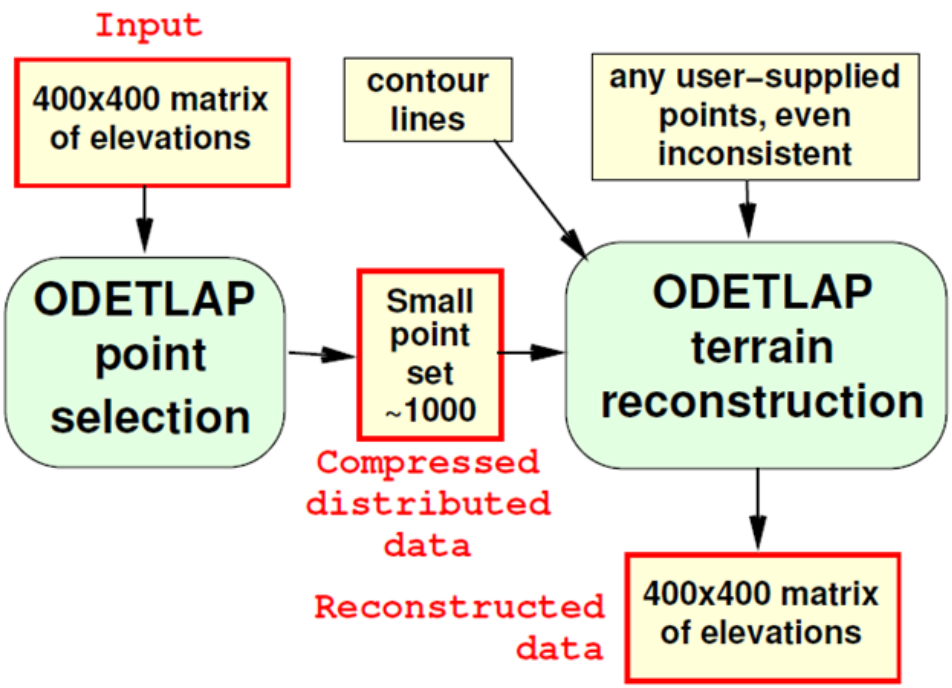
	Smoothed aggregation	AINV	Diagonal
Part 1	0.1 s	0.1 s	0.1 s
Part 2	0.3 s	0.3 s	0.3 s
Part 3	4.7 s	12.0 s	10.4 s
Memory	730 MB	533 MB	428 MB

GPU-accelerated ODETLAP – Speedup

- Single-thread CPU
  - Storing $\boldsymbol{A}$, $\boldsymbol{A}^T$, and $\boldsymbol{A}^T\boldsymbol{A}$ on host

	CPU	GPU	Speedup
Part 1	0.0 s	0.1 s	0.6
Part 2	0.9 s	0.3 s	2.9
Part 3	37.8 s	4.7 s	8.1





Thank you

$$4Rz(x,y) - Rz(\dots) - Rz(\dots) - Rz(\dots) - Rz(\dots) = 0$$
$$z(x_i, y_i) = v_i$$