Research Summary: Siting and ODETLAP

Wenli Li\textsuperscript{1*}, W. Randolph Franklin\textsuperscript{1}, Salles V. G. Magalhães\textsuperscript{12}, Marcus V. A. Andrade\textsuperscript{2}

\textsuperscript{1}Rensselaer Polytechnic Institute
\textsuperscript{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 \textit{stride}: 1, 2, 4, 8, ...
• Increasing \textit{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

<table>
<thead>
<tr>
<th>stride</th>
<th>30 targets</th>
<th>120 targets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Coverage</td>
</tr>
<tr>
<td>1</td>
<td>87</td>
<td>95.5</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>95.5</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>95.5</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>95.5</td>
</tr>
<tr>
<td>$2^i$</td>
<td>11</td>
<td>96.0</td>
</tr>
</tbody>
</table>
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 $R$
    
    $$4Rz(x, y) - Rz(\ldots) - Rz(\ldots) - Rz(\ldots) - Rz(\ldots) = 0$$
  • Known-value equation
    
    $$z(x_i, y_i) = v_i$$

• Overdetermined system

    $$A_{(mn+k)\times mn}x = b$$

    $$A^T A x = A^T b$$
ODETLAP –
Lossy compression


• 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


ODETLAP – Application 2

- Li et al. 3D oceanographic data compression using 3D-ODETLAP. *SIGSPATIAL Special*, 2010

360 × 180 × 24 × 12 (× 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
| | | | / | | | | / \ |
B B B B B B B B B B

A A A A A A A A A A
| | | | | | | | | | |
```
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 $A$ and $b$ on CPU
  • Part 2: calculating $A^T$, $A^T A$ and $A^T b$
  • Part 3: solving $A^T A x = A^T b$

• Data
  • N43W072, 1200 $\times$ 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 $A$, device $A$ and $A^T$: COO; varying device $A^T A$
  - Solver: Conjugate Gradient (CG) method

<table>
<thead>
<tr>
<th></th>
<th>COO</th>
<th>CSR</th>
<th>ELL</th>
<th>HYB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>0.1 s</td>
<td>0.1 s</td>
<td>0.1 s</td>
<td>0.1 s</td>
</tr>
<tr>
<td>Part 2</td>
<td>0.3 s</td>
<td>0.3 s</td>
<td>0.3 s</td>
<td>0.3 s</td>
</tr>
<tr>
<td>Part 3</td>
<td>20.6 s</td>
<td>19.5 s</td>
<td>10.2 s</td>
<td>10.2 s</td>
</tr>
<tr>
<td>Memory</td>
<td>493 MB</td>
<td>428 MB</td>
<td>422 MB</td>
<td>422 MB</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>BiCG</th>
<th>CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>0.1 s</td>
<td>0.1 s</td>
</tr>
<tr>
<td>Part 2</td>
<td>0.3 s</td>
<td>0.3 s</td>
</tr>
<tr>
<td>Part 3</td>
<td>16.3 s</td>
<td>10.2 s</td>
</tr>
<tr>
<td>Memory</td>
<td>422 MB</td>
<td>422 MB</td>
</tr>
</tbody>
</table>
GPU-accelerated ODETLAP – Preconditioners

• Preconditioners
  • Smoothed aggregation preconditioner
  • Approximate Inverse (AINV) preconditioner
  • Diagonal preconditioner

• ELL matrix format and CG solver

<table>
<thead>
<tr>
<th></th>
<th>Smoothed aggregation</th>
<th>AINV</th>
<th>Diagonal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1</td>
<td>0.1 s</td>
<td>0.1 s</td>
<td>0.1 s</td>
</tr>
<tr>
<td>Part 2</td>
<td>0.3 s</td>
<td>0.3 s</td>
<td>0.3 s</td>
</tr>
<tr>
<td>Part 3</td>
<td>4.7 s</td>
<td>12.0 s</td>
<td>10.4 s</td>
</tr>
<tr>
<td>Memory</td>
<td>730 MB</td>
<td>533 MB</td>
<td>428 MB</td>
</tr>
</tbody>
</table>

Memory

730 MB

533 MB

428 MB
GPU-accelerated ODETLAP – Speedup

• Single-thread CPU
  • Storing $A$, $A^T$, and $A^T A$ on host

<table>
<thead>
<tr>
<th>Part 1</th>
<th>CPU</th>
<th>GPU</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 2</td>
<td>$0.0 \text{ s}$</td>
<td>$0.1 \text{ s}$</td>
<td>$0.6$</td>
</tr>
<tr>
<td>Part 2</td>
<td>$0.9 \text{ s}$</td>
<td>$0.3 \text{ s}$</td>
<td>$2.9$</td>
</tr>
<tr>
<td>Part 3</td>
<td>$37.8 \text{ s}$</td>
<td>$4.7 \text{ s}$</td>
<td>$8.1$</td>
</tr>
</tbody>
</table>
\[ 4Rz(x, y) - Rz(...) - Rz(...) - Rz(...) - Rz(...) = 0 \]
\[ z(x_i, y_i) = v_i \]