Computing Visibility on Large Rasters: A new GRASS module r.viewshed

Laura Toma  Will Richard
Bowdoin College
USA

FOSS4G 2008
Cape Town, South Africa
Visibility: the Basic Problem

- **Input**
  - T: an elevation model of a terrain
  - v: a viewpoint (location, height)
- **Compute the viewshed of v**
  - the set of points of T visible from v
Visibility: the Basic Problem

- **Input**
  - $T$: an elevation model of a terrain
  - $v$: a viewpoint (location, height)

- **Compute the viewshed of $v$**
  - the set of points of $T$ visible from $v$

- **Applications**
  - path planning and navigation
    - find a scenic path with overall maximum visibility
    - find location for a pipe/construction with least visibility
    - find location with minimum visibility in the terrain
    - find best hide place
  - guarding:
    - placement of fire towers, radar sites, cell phone towers
Visibility on Rasters

- Line-of-sight model
  - \( a, b \) visible if the line segment \( ab \) does not intersect the terrain

Interpolate to find out height of \( T \) along the line
Visibility on Rasters

- **Line-of-sight model**
  - $a, b$ visible if the line segment $ab$ does not intersect the terrain

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

**interpolate** to find out height of $T$ along the line
Nearest-Neighbor Interpolation
Nearest-Neighbor Interpolation
Nearest-Neighbor Interpolation
Nearest-Neighbor Interpolation

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>23</td>
<td>25</td>
<td>26</td>
<td>32</td>
<td>46</td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>21</td>
<td>23</td>
<td>31</td>
<td>32</td>
<td>46</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>24</td>
<td>27</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>32</td>
<td>22</td>
<td>29</td>
<td>30</td>
<td>35</td>
<td>34</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>33</td>
<td>34</td>
<td>36</td>
<td>37</td>
</tr>
</tbody>
</table>
# Nearest-Neighbor Interpolation

![Image of a grid with values and a diagonal line indicating nearest-neighbor interpolation.](image-url)
Visibility with Nearest-Neighbor Interpolation

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

```
  b
  a
  22  24  27  31  36  41  45
  a
  b
```
Visibility: Related Work

Theoretical
- Computational geometry/graphics
  - ...
- GIS
  - Direct algorithm: $O(n^{3/2})$
  - In-memory: $O(n \log n)$ [van Kreveld]
  - Disk: $O(sort(n))$ [HTZ07]

GIS
- Fisher, Franklin et al, Izraelevitz
- Wang et al
  - ...

Surveys:
- De Floriani & Magillo [FM94]
- Cole & Sharir [CS89]
Computing on Very Large Terrains

Why?

- Large amounts of data are becoming available
  - SRTM: 30/90m resolution of entire globe (~10TB)
  - LIDAR: sub-meter resolution

Traditional algorithms designed assuming
- data fits in memory
- has uniform access cost
- ..... don’t scale

Buy more RAM?
- Data grows faster than memory

Data does not fit in memory, sits on disk
- Disks are MUCH slower than memory

=> disk I/O bottleneck
Large Data: What To Do?

- **Very large data => needs efficient algorithms**
  - small data: 1 sec vs 3 sec
  - large data: 1 hour vs 1 day (or worse)

- **Massive data: bottleneck is disk I/O**
  - ==> Design algorithms that specifically minimize disk I/O

- **I/O-efficient algorithms**
  - **Idea:**
    - Do not rely on virtual memory!
    - Instead, change the data access pattern of the algorithm to increase spatial locality and minimize the number of blocks transferred between main memory and disk
This project: r.viewshed

- **r.viewshed**
  - Efficient visibility computation on very large grids
    - uses improved algorithm (both I/O- and CPU-efficient)
  - Can process very large grids fast
  - Available in GRASS add-ons

**Outline**

Visibility, model and related work

- **r.viewshed**
  - Overview
  - Efficiency and experimental results
  - Algorithm

Related and future work
Description:
IO-efficient viewshed algorithm

Keywords:
raster, viewshed, line of sight

Usage:
r.viewshed [-rcbe] input=name output=name viewpoint_location=lat,long
[observer_elevation=value] [max_dist=value] [memory_usage=value]
[--overwrite] [--verbose] [--quiet]

Flags:
-r Use row-column location rather than latitude-longitude location
-c Consider the curvature of the earth (current ellipsoid)
-b Output format is {0 (invisible) 1 (visible)}
-e Output format is {NODATA, -1 (invisible), elev-viewpoint_elev (visible)}
--o Allow output files to overwrite existing files
--v Verbose module output
--q Quiet module output
Parameters:

- **input**  
  Name of elevation raster map

- **output**  
  Name of output viewshed raster map  
  default format:  
  \{NODATA, -1 (invisible), vertical angle wrt viewpoint_location (visible)\}

- **viewpoint_location**  
  Coordinates of viewing position in latitude-longitude (if -r flag is present then coordinates are row-column)

- **observer_elevation**  
  Viewing elevation above the ground  
  default: 0.0

- **max_dist**  
  Maximum visibility radius. By default infinity (-1).  
  default: -1

- **memory_usage**  
  The amount of main memory in MB to be used  
  default: 500
Sierra Nevada, 30m resolution (10 million elements)
r.viewshed -c viewpoint=1000, 1000

obs=0

obs=100

obs=1000

obs=5000
r.viewshed -c view=1000, 1000  obs=5000
max=30,000
max=20,000
max=10,000
r.viewshed Efficiency
r.viewshed Efficiency

- **Experimental Platform**
  - Apple Power Mac G5
  - Dual 2.5 GHz processors
  - 1 GB RAM

- **Run analysis on various terrains**

- **Compare with current visibility module in GRASS**
  - GRASS r.los

- **Other (open-source) modules??**

<table>
<thead>
<tr>
<th>Gris Terrains</th>
<th>Grid Size (million points)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kaweah</td>
<td>2</td>
</tr>
<tr>
<td>Sierra Nevada</td>
<td>10</td>
</tr>
<tr>
<td>Cumberlands</td>
<td>67</td>
</tr>
<tr>
<td>Lower New England</td>
<td>78</td>
</tr>
<tr>
<td>East Coast USA</td>
<td>246</td>
</tr>
<tr>
<td>Midwest USA</td>
<td>280</td>
</tr>
<tr>
<td>Washington</td>
<td>1,066</td>
</tr>
</tbody>
</table>
current visibility module in GRASS: r.los

r.los -help

Description:
Line-of-sight raster analysis program.

Usage:
r.los input=name output=name coordinate=x,y [patt_map=name] [obs_elev=value] [max_dist=value]

Parameters:
input Raster map containing elevation data
output Raster map name for storing results
coordinate Coordinate identifying the viewing location
patt_map Binary (1/0) raster map
obs_elev Height of the viewing location
default: 1.75
max_dist Max distance from the viewing point (meters)
options: 0-99999
default: 1000
### r.viewshed Efficiency

<table>
<thead>
<tr>
<th>Gris Terrains</th>
<th>Grid Size (million points)</th>
<th>r.los</th>
<th>r.viewshed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kaweah</td>
<td>2</td>
<td>30 minutes</td>
<td>5 sec</td>
</tr>
<tr>
<td>Sierra Nevada</td>
<td>10</td>
<td>4 hours</td>
<td>1 min</td>
</tr>
<tr>
<td>Cumberlands</td>
<td>67</td>
<td>&gt; 40 hours</td>
<td>3.3 min</td>
</tr>
<tr>
<td>Lower New England</td>
<td>78</td>
<td></td>
<td>4.8 min</td>
</tr>
<tr>
<td>East Coast USA</td>
<td>246</td>
<td></td>
<td>16 min</td>
</tr>
<tr>
<td>Midwest USA</td>
<td>280</td>
<td></td>
<td>45 min</td>
</tr>
<tr>
<td>Washington</td>
<td>1,066</td>
<td></td>
<td>4.5 hours</td>
</tr>
</tbody>
</table>
The Underlying Algorithm

- When terrain fits in memory
The Underlying Algorithm

- When terrain fits in memory
The Underlying Algorithm

- When terrain fits in memory
The Underlying Algorithm

- When terrain fits in memory
The Underlying Algorithm

- When terrain fits in memory
The Underlying Algorithm

- When terrain fits in memory
The Underlying Algorithm

- When terrain fits in memory ==> line sweeping [vK 2001]
- Efficiency:
  - 3n events, $O(\lg n)$ per event --> $O(n \lg n)$ CPU time
The Underlying Algorithm

- When input does not fit in memory [HTZ’07]
  - divide terrain into equal-sized sectors
  - compute visibility in each sector recursively
  - handle sector interactions
- Efficiency: $O(\text{sort}(n))$ disk block transfers
Visibility grid: compute size of viewshed for each point in a terrain

Efficiency: for a grid of n points: \( n \times O(n \lg n) = O(n^2 \lg n) \) CPU time
Visibility Grid

- **Dataset**: $472 \times 391 = 184,552$ points
- **one viewshed**: 1.2 seconds
- **visibility grid**: 45 hours !!!
Visibility Grid

**Future work: approximation**
- compute approximate viewshed of each point?
  - compute exact viewshed in a small neighborhood
  - sample terrain to compute what points are visible
- compute exact viewshed of a sample of points?

**Future work: find point of largest/smallest visibility**
- without computing entire visibility grid

**Future work: guarding**
- find the locations of a minimal number of observers so that together they cover/see the entire terrain
- even slower: need to compute visibility grid repeatedly
Related Projects
Scalable modules for processing massive terrain data

- **r.terraflow** [2000]
  - flow modeling

- **r.refine** [2005]
  - terrain simplification

- **r.terracost** [2006]
  - least-cost surface

- **r.viewshed** [2008]
  - visibility
Related Projects
Scalable modules for processing massive terrain data

- r.terraflow [2000]: flow modeling
  - compute multiple flow directions, flooding and flow accumulation
  - input: DEM
  - output: FD grid, FA grid, filled DEM
  - see GRASS r.terraflow

![DEM](image1.png) ![flow accumulation](image2.png)
Related Projects
Scalable modules for processing massive terrain data

- r.refine [2005] : terrain simplification
  - simplify a grid into a TIN within a desired accuracy \( e \) such that
    distance(simplified TIN, grid) < \( e \)
  - input: grid terrain + error threshold
  - output: a Delaunay-triangulated TIN
r.terracost [2006]: least-cost surface
- **input:** cost surface + set of source points
- **output:** a least-cost surface, where each point represents the shortest path to a source
- similar to r.cost
Thank you.

GRASS-addons/raster/r.viewshed/

Laura Toma
Bowdoin College
Maine, USA
ltoma@bowdoin.edu

http://www.bowdoin.edu/~ltoma/