Viewshed Explorer

User Guide v.1

School of Geography
University of Leeds

Contents

1. Introduction

Most proprietary off-the-shelf GIS software packages provide basic visibility analysis tools, but these are generally slow, making real-time animation of viewsheds impossible and detailed analysis of whole landscapes using high resolution DEMs impractical. The Viewshed Explorer is a simple, easy to use and highly efficient viewshed algorithm and desk top tool for interrogating viewsheds in real-time using large high resolution digital surface models and batch processing of exhaustive cumulative viewshed analyses for landscape visualisation and assessment.

The following sections of this manual describe the principles used in developing the Viewshed Explorer tool, the tool interface and its use.

2. Underlying principles

Visibility analyses in GIS are used to calculate the theoretical area visible from an observation point or series of observation points across a given landscape taking terrain surface height and other barriers to intervisibility such as man-made structures and tree cover into account. There are many applications of visibility analyses in GIS ranging from landscape visualisation studies (Ghadirian and Bishop, 2008), landscape perception (Habron, 1998) and optimum siting of infrastructure and facilities (Kim et al., 2004) through to impact assessment and landscape evaluation (Gulinck et al., 2001). Other examples from the literature include evaluation of the inter-visibility between archaeological sites (Fisher et al., 1997), siting of mobile communications towers (Oda et al., 2000), assessment of Zones of Visual Influence (ZVIs) from wind farm developments (Bishop and Miller, 2007) and modelling absence of human artefacts in developing wilderness quality indices (Carver et al., 2012).

One of the principal problems with visibility analysis is that they are computationally very intensive and times taken to calculate a viewshed for even a single observation point across large, high resolution terrain models can be long, especially if wanting to take distance decay effects into account. When dealing with a small numbers of observation points (npoints < x103) then off-the-shelf visibility analyses provided by proprietary GIS are more than adequate. However, when dealing with much larger numbers of observation points (npoints >> x103) with significant search radii (r > x103 cells) across very large, high resolution DEMs (ncells > x 106) the speed at which these standard tools run can become a significant bottleneck in the analysis workflow. This was the case with a recent project to develop indices of wildness for the Scottish national parks where it was necessary to model the absence of modern human artefacts within the landscape based on visibility measures taking distance decay effects into account at a resolution of 20m. Estimated run times measured in years rather than days or weeks using proprietary GIS software meant a more efficient approach had to be found (Carver et al., 2012). An innovative solution was created utilising ray-casting methods and a voxel-based viewshed transform to produce a practical tool for calculating millions of viewsheds on a standard desktop PC. This paper describes the voxel-viewshed transform approach adopted, software development and testing, and a number of example applications within the context of landscape assessment.

A number of authors have previously identified speed and efficiency of the algorithms used as a possible area for improvement. A variety of efficient algorithms have therefore been developed and reported in the geocomputation literature. These include examples such as R2, R3, XDraw, line-rasterisation, tracking-in, tracking-out, etc., many of which rely on making crucial trade-offs between accuracy and speed (e.g. Andrade et al., 2011). While some authors have experimented with algorithm efficiency, others have looked towards advances in computing power to address the problem, such as the application of grid computing and parallel computing

architectures (Mills et al., 1992; Ware et al., 1996; Kidner et al., 1997; Ware et al., 1998; Rana and Sharma, 2006). Despite these improvements, few if any, seem to have developed into widely used tools or found their way onto users' desktops or into proprietary GIS toolkits.

The algorithm employed here is very similar to the R2 ray casting algorithm described by Franklin & Ray (1994). Such ray casting algorithms have been implemented in computer gaming software since the early 1990s (e.g. Wimmer et al., 1999), but have now been largely usurped by polygon-based approaches supported by modern graphics hardware (e.g. Krüger & Westermann, 2005). It is unclear whether the brevity of their success in the gaming field has any bearing on why they have been seemingly overlooked by developers of mainstream GIS software and tools, but they can be shown to be of enormous practical benefit for terrain based visibility analyses linked to GIS.

Theoretically, the computation time of algorithms like R2 is linear in the number of cells which constitute the region of interest for which visibility is to be computed. However the performance can in practice be sub-linear, owing to the fact that as a ray is cast further and further from an observer point, it is increasingly likely to have encountered a feature substantial enough to constitute a "horizon", at which point the ray can be terminated. This effect is most pronounced in the presence of rugged terrain, and can be further capitalised upon by employing pre-computed tables and dynamic programming techniques for the rapid identification of regions of potential visibility.

In the voxel surface model used here, each of the raster cells in a digital terrain model are projected as a series of vertical columnar elements of single-cell size in the plan view whereby vertical and horizontal surfaces can be independently checked for full or partial visibility. Cells, or more correctly "voxels" (for volume cells), are traversed from the observer to each perimeter voxel of the terrain model. For each voxel lying on the perimeter of the potentially visible area surrounding the observer, a ray is traced from the observer voxel. This is done in a sequential fashion so that only once such a ray-cast completes, is the next perimeter voxel considered. Voxels already encountered in previous traversals do not have their visibility re-assessed, even if the portion of the voxel visible may be different for subsequent ray-casts. This process is illustrated in Figure 1.

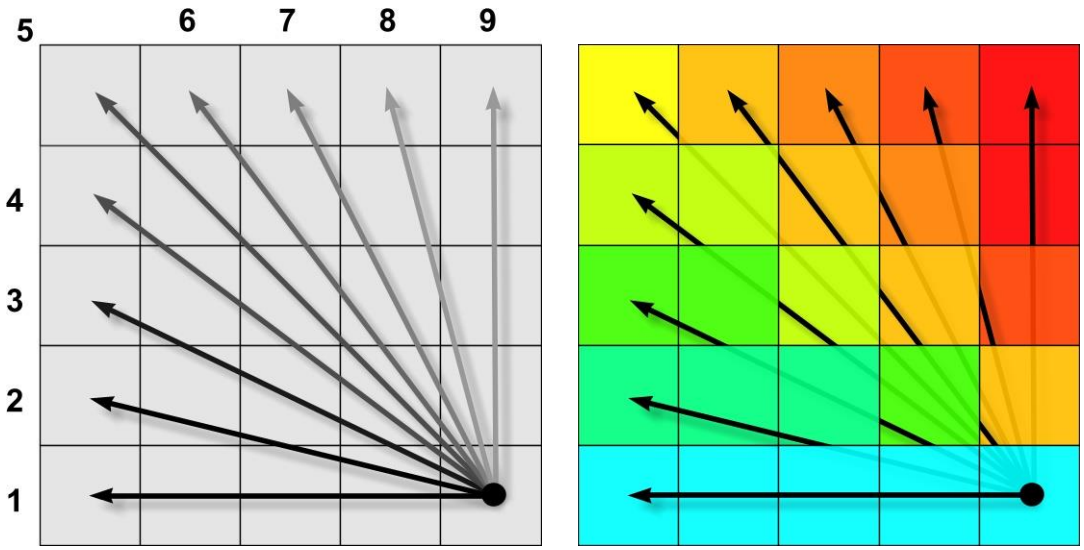

Figure 1. Ray casting and checking for partial/full visibility

Although we use the term “ray”, each is better thought of as a “sheet”, defined by an upper and lower bound, which fans out vertically from the observer location. As a ray propagates outwards from the observer and visible voxels are encountered, the lower bound of this sheet is updated so as to inform the visibility of subsequent voxels. Subsequent voxels occluded by this lower bound can be ignored, and the casting of a ray can then be terminated early if it can be determined that no more voxels can possibly intersect with the sheet. Any voxels which intersect a propagating sheet are deemed visible, with the extent of visibility depending upon where on the voxel's column the lower bound of the sheet intersects, and also whether the upper surface of the voxel is visible (i.e. whether the observer is looking from above or below). This visible portion is “read off” and its height and area in the observer's field-of-view calculated and stored. This process is illustrated in figure 2.

Notable efficiencies over the R2 method (Franklin & Ray, 1994) are introduced by two related assumptions:

- If a voxel is found to be visible, then although only a given portion of its column may be visible and its upper surface may or may not be visible, the voxel is always considered wholly visible in the planar sense. In other words, we do not concern ourselves with what quadrant or which sides of a voxel may be visible.
- If a voxel has been found to be visible to a previously cast ray, then its visibility need not be recomputed for subsequent rays. This is because it is logically impossible for a cell to be more than wholly visible to the observer. The standard R2 implementation makes a similar logical assumption, but re-calculates feature visibility for every ray which

encounters a feature and then retains that belonging to the single most proximate ray (Franklin & Ray, 1994).

Further efficiencies are generated by tiling the data and working on each tile in RAM. This minimizes time consuming disk access with input/output occurring only when reading source data into the model and when writing out the final results, and allows tiles to be farmed out to multiple parallel processors. The resulting efficiencies realized mean that the voxel viewshed transform is able to out-perform the equivalent calculations using proprietary GIS software by roughly 3 orders of magnitude (before parallelization).

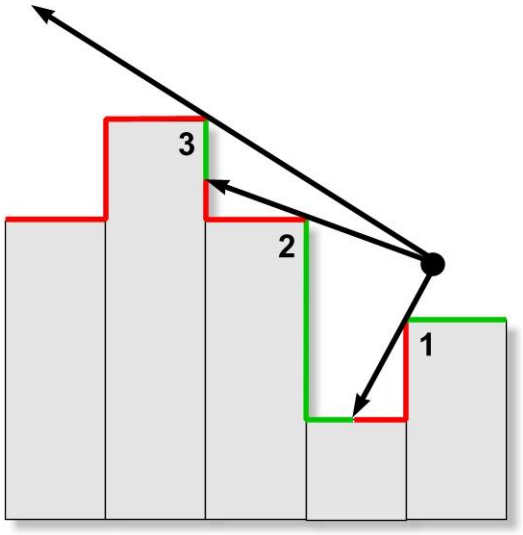

Figure 2. Determining observer's FOV and vertical portion of voxel visible

3. Tool interface

The algorithm has been embedded within a user-friendly GUI and can be run either from the desktop on a single PC or across the network on multiple machines to further speed up processing times for very large areas. A tiling-tool has been added to facilitate this process. The interface allows users to input a large DEM as a floating point raster and a feature layer containing separate feature classes. Observer height and maximum/minimum search radius can also be set. Distance decay effects can be calculated as 1/d or $1/d^2$ giving the relative height and surface area of a feature cell, respectively (where d = distance from the observer). The size of the DEM used is limited only by the RAM available. If the DEM is too large to fit in RAM, the tiling-tool can be used to divide up the DEM and run the transform across 2 or more tiles using an appropriate overlap distance to take into account visibility from outside and into adjacent tiles. Figure 3 shows the main GUI and real-time viewshed explorer and Figure 4 shows the tiling-tool. When run in batch mode, the viewshed of every grid cell in the input DEM is calculated and the relative proportion of the viewshed occupied by a feature or the background terrain is stored. Figure 5 shows the output window with an example completed viewshed transform. Output viewshed transforms from the tool can be generated either as separate feature layers or combined layers and saved on logged and unlogged scales in floating point grids. These can then be input into a GIS package for combination with other data layers.

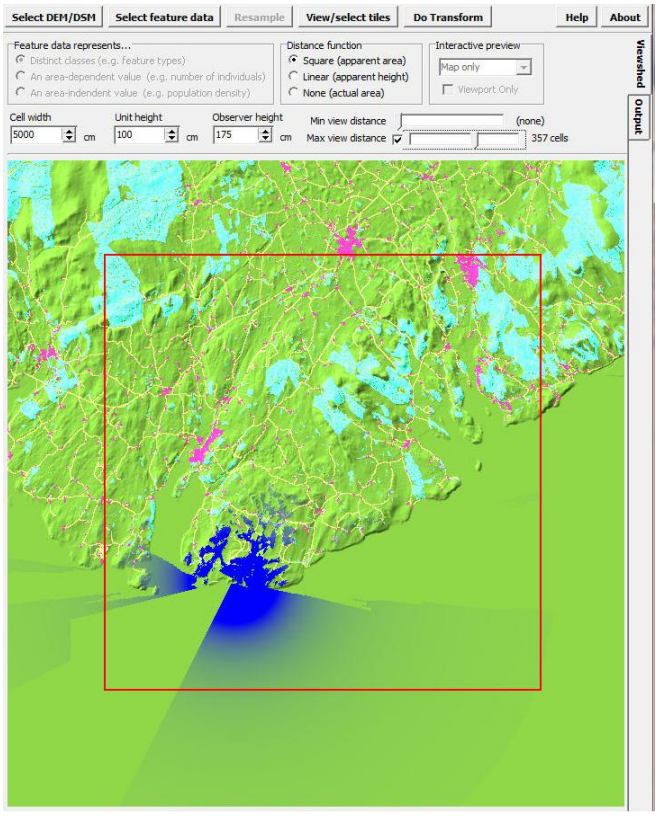

Figure 3. Main GUI and real-time viewshed explorer

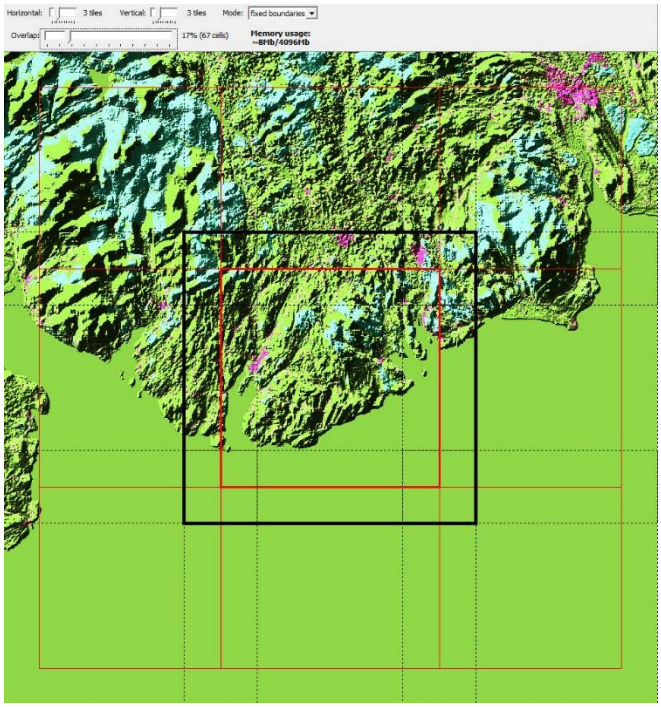

Figure 4. Tiling-tool

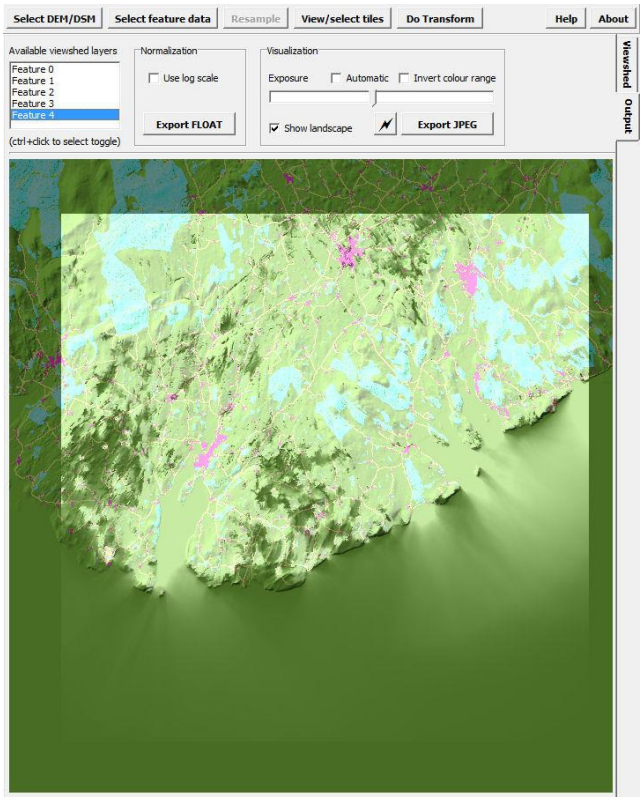

Figure 5. Output window and completed viewshed transform

4. Use

The following sections detail how to use the Viewshed Explorer tool in detail.

a. Preparing datasets

Datasets to be used with the Viewshed Explorer are to be imported as floating point grid format. Most GIS software provide conversion tools for converting between various raster formats including floating point grids. For example, ArcGIS can convert from any raster format to floating point grid using:

ArcToolBox > Conversion Tools > From Raster > To float

The Viewshed Explorer software requires 2 basic input grids:

- A raster digital elevation model (DEM). This can be a floating point grid (i.e. values in the cells can have significant figures after the decimal)
- A raster feature layer containing the locations of cells designated as a feature of interest which is to have its visibility modelled. This must be an integer grid (i.e. values in the cells must be rounded to the nearest whole number). The feature layer can contain up to seven classes number 1 through to 7 plus a “background” cell value of 0 (zero).

Both floating point grids should be exactly the same dimensions in terms of both cell size and row/column numbers, and should have exactly the same origin (i.e. cover the same area at the same resolution).

Each floating point grid will consist of two files with the suffixes .flt (the binary floating point grid) and .hdr (the header file).

The header file should be checked (e.g. in Notepad or other text editor) to make sure the xllcorner, yllcorner and cellsize values are all integer (i.e. have no significant figures after the decimal). If they do contain significant figures after the decimal these should be edited out including the decimal point as follows:

Original version		Edited version	
ncols	4959	ncols	4959
nrows	7201	nrows	7201
xllcorner	10507.935	xllcorner	10507
yllcorner	515183.564	yllcorner	515183
cellsize	100.1	cellsize	100
NODATA_value	-9999	NODATA_value	-9999
byteorder	LSBFIRST	byteorder	LSBFIRST

When creating input data layers you need to make sure the area of interest is surrounded by an additional “edge” distance of twice the maximum search radius you will be using in the viewshed calculations. This is to ensure that no edge effects will be seen in your area of interest (see below under section c. tiling for further details)

Check list:

1. *Two floating point rasters: one DEM grid (can be floating point), one feature grid (must be integer with maximum of 7 classes numbered 1 to 7 plus 0 background class) making sure an edge of 2x maximum viewshed is included around the area of interest.*
2. *Same dimensions and resolution*
3. *Same extent*
4. *No decimal figures after xllcorner, yllcorner and cellsize in .hdr files*

b. Loading data

Data is loaded into the Viewshed Explorer tool using the two “Select…” buttons on the top left of the tool window. Use “Select DEM/DSM” to input the floating point grid for the raster digital elevation model, and use “Select feature data” button to input the integer floating point grid for the raster layer containing the feature locations. The “Select feature data” button is grey out and unavailable until a DEM has been loaded (see Figure 3). Clicking on the buttons will open a folder/file selection window. Use this to navigate to and select the required .flt file for the respective inputs.

Check list:

1. *Use “Select DEM/DSM” button to select and load raster DEM grid*
2. *Use “Select feature data” button to select and load raster feature grid*

c. Tiling

If the data is a small enough area to fit in its entirety into the computer’s RAM you can just run the model in a single tile. However, if the dataset is large (i.e. covering a large area and/or at high resolution) it is better to tile the data and run the model over a series of tiles. This also facilitates running the model on several computers and/or utilising the computer’s multicore CPU to run a number of tiles at once. This can be done simply by running the Viewshed Explorer software multiple times. The software allows you to view/select tiles from 1x1 (a single tile) and split the data and model runs over up to 10x10 tiles.

The “View/select tiles” button is used to views and selected the tiles. This generates a new window showing a map of the data and the tile boundaries/overlaps and the tiling controls (see Figure 4).

Use the Horizontal and Vertical slider bars to specify the number or columns and rows used to tile the data.

When viewing/selecting you also need to specify the overlap distance using the slider bar. Fine adjustments to the overlap distance can be made using the left/right arrow keys on the keyboard. This is the maximum search radius used in the viewshed calculations specified in cells.

Note: because the software uses the overlap distance to specify the search radius you will need to make sure the input data includes an additional edge of 2x the maximum search radius to avoid edge effects in the area of interest (see above under section a. preparing datasets).

There are two modes to the tiling process: fixed boundaries, and all tiles equal. Fixed boundaries is the default.

Select the tile to run by clicking on the tile. Tiles which are currently running with show a progress time on the tile and tiles which have already completed are indicated by a tick. Clicking on a tile will load this data ready for the model run and return to the main menu with the map data showing. Moving the mouse over the loaded map shows an interactive animation of the viewshed using the parameters set in the tiling tool.

Check list:

1. *Use "View/select tiles" to see loaded data and control tiling*
2. *Use Horizontal and Vertical sliders to specify number of columns and rows in tiling the loaded data*
3. *Use the Overlap slider to specify the maximum search radius to be used and specify the overlap between tiles*
4. *Click on the tile to run to load this and prepare for model run*

d. Terrain offsets

The terrain offset for the viewer eye-level height can be specified in the main menu using Observer Height. The default is 175cm. Variable terrain offsets for feature data can be added to the DEM/DSM grid prior to using the Viewshed Explorer. It is recommended you do this in your GIS. If all the features are the same height this can be set using Unit Height. The default is 100cm (see Figure 3).

e. Distance decay functions

Three options are provided for distance decay functions (see Figure 3). These are none (calculates simple areas from which a feature is visible with no distance decay), linear (calculates distance decay based on apparent height of the feature), and squared (calculates distance decay based on apparent vertical area of the feature). These are selected using the radio buttons. Additionally the minimum and

maximum search radius can be specified using the slider bars but in most instances these should be unchanged from that set in the tilling tool.

f. Model runs

Model runs are initiated by clicking on the “Do Transform” button. A warning saying that this could take some time and slow PC performance is shown. Click on OK to accept and start the model run.

Model runs can be monitored by clicking on the “Output” tab at the right of the map window. This shows the progress of the viewshed being calculated. The views can be refreshed by clicking on the “lightening” button (see Figure 5).

The top of the window shows the current cell row/column number, cells per second being processed and the time left to completion. These numbers are approximate.

Once completed, the “Transform complete” window is displayed. Click OK to return to the output tab window.

There are two options for saving the output: either as a JPEG (the output is generated as a simple JPEG image format file), or as a floating point grid. The latter can be imported back into GIS for further analysis. Layers for each input feature type can be saved separately or together by selecting which layers to output in the “Available viewshed layers” sub window suing CTRL + CLICK to select. Data can be output as normal or log scale. Click on “Export FLOAT” to export the selected layers.

As a general rule, export layers using log scale for better visualisation in GIS.

Check list:

1. *Use “Do Transform” to run the model*
2. *Use “Output” tab to monitor progress*
3. *When complete use “Available viewshed layers” sub window to select layers for output*
4. *Export selected layers using “Export FLOAT” button*

g. Interpreting output

The output from model runs gives the proportion of the total viewshed that is occupied by the background value in the feature layer (i.e. the zero class). Therefore the numbers in the output vary showing high numbers where very little of the input features (classes 1 – 7) are visible and the viewshed is mainly background, and very low numbers where the viewshed is dominated by the input features. A value of zero (0) is given in the output for areas where no input features are visible at all.