L7 – Raster Algorithms

NGEN06(TEK230) –
Algorithms in Geographical Information Systems

by: Abdulghani Hasan, updated Nov 2016 by Per-Ola Olsson
Background

Store and analyze the geographic information: **Raster data** or **Vector data**.
Content

1. Introduction (raster data and analyses)
2. Local and regional operators
3. Morphological operators
4. Shaded relief
5. Resampling
6. Flow analysis in hydrology
Vector vs. Raster

Algorithms described for vector data (distances, area of polygon, point-in-polygon) are all fairly simple to perform also with raster data.

Result is often worse than in the vector case since the resolution of the raster data is not that good.
Example of raster data and raster analyses

Digital maps

Digital Terrain Models (DTM) and Digital Elevation Models (DEM)

Images (often aerial or satellite images)
Example of raster data and raster analyses

Digital maps

Digital Terrain Models (DTM) and Digital Elevation Models (DEM)

Images (often aerial or satellite images)
Example of raster data and raster analyses

Digital maps

Digital Terrain Models (DTM) and Digital Elevation Models (DEM)

Images (often aerial or satellite images)
Example of raster data and raster analyses

1. Digital maps

**Cost analyses** – finding the path with the lowest “cost” between locations. Raster version of Dijkstra.

**Location analyses** – finding suitable locations e.g. for private houses or factories.

**Overlay analyses** – e.g. landcover and soil map.

**Generalization** – e.g. morphological operators for simplification.
Example of raster data and raster analyses

2. Digital Terrain Models (DTM) and Digital Elevation Models (DEM)

**Viewshed analysis** – e.g. from which locations will a wind power park be visible?

**Flow analysis** – hydrology, how will rainfall flow.

**Shaded relief** – cartography, to enhance the feeling of topography.

**Image orthorectification** – photogrammetry, create orthophotos.
Example of raster data and raster analyses

2. Digital Terrain Models (DTM) and Digital Elevation Models (DEM)

Image orthorectification – photogrammetry, create orthophotos.
Example of raster data and raster analyses

3. Images (often aerial or satellite images)

- **Resampling** – georeferencing, transforming raster.
- **Image classification** – remote sensing, e.g. create land cover map based on spectral properties.
- **Information extraction** – e.g. extracting roads, buildings etc. in images.
Raster analyses in this lecture

1) Morphological operators
   This is a basic method for digital maps. Several manipulations and analyses of raster maps are based on morphological operators. *(shrink and expand)* → L9 generalization

2) Shaded relief
   A shaded relief is created (based on a DEM) for a fictitious sun. This relief is then added to a map to enhance the feeling of topography.

3) Resampling of images
   To use e.g. a satellite image in a geographic analysis it must be geocoded. Geocoding - resampling of the image to the new geometry.

4) Flow analyses in hydrology
   All of these operations are regional operations.
Local and regional operations

*Local operation:* In this case the pixel \((i,j)\) in the new raster is only dependent on pixel \((i,j)\) in the original raster.

All pixels multiplied with 1.02
Local and regional operations

Local operation: In this case the pixel \((i,j)\) in the new raster is only dependent on pixel \((i,j)\) in the original raster.

Regional operation: Here each pixel \((i,j)\) in the new raster is dependent on all the surrounding pixels to pixel \((i,j)\) in the original raster.

Kernel filter

Typically a filter is 3x3, 5x5 etc.
Local and regional operations

*Local operation:* In this case the pixel \((i,j)\) in the new raster is only dependent on pixel \((i,j)\) in the original raster.

\[
\begin{array}{cccc}
100 & 100 & 110 & 140 \\
100 & 110 & 120 & 140 \\
120 & 120 & 140 & 150 \\
140 & 140 & 150 & 200 \\
\end{array}
\times 1.02 =
\begin{array}{cccc}
102 & 102 & 112 & 143 \\
102 & 112 & 122 & 143 \\
122 & 122 & 143 & 153 \\
143 & 143 & 153 & 204 \\
\end{array}
\]

*Regional operation:* Here each pixel \((i,j)\) in the new raster is dependent on all the surrounding pixels to pixel \((i,j)\) in the original raster.

Kernel filter: \[
\begin{bmatrix}
0.25 & 0.25 \\
0.25 & 0.25 \\
\end{bmatrix}
\times \begin{array}{cccc}
100 & 100 & 110 & 140 \\
100 & 110 & 120 & 140 \\
120 & 120 & 140 & 150 \\
140 & 140 & 150 & 200 \\
\end{array} =
\begin{array}{cccc}
102 & 110 & 127 \\
112 & 122 & 137 \\
130 & 137 & 160 \\
\end{array}
\]
Local and regional operations

Local operation: In this case the pixel \((i,j)\) in the new raster is only dependent on pixel \((i,j)\) in the original raster.

Regional operation: Here each pixel \((i,j)\) in the new raster is dependent on all the surrounding pixels to pixel \((i,j)\) in the original raster.

Kernel filter
Morphological operators

**Expand**
(Dilation with 2x2 kernel)

**Shrink**
(Erosion with 2x2 kernel)

Note that a new raster is created
Shaded relief

Enhance the feeling of topography in a map.

DEM

Shaded relief
The sun placed in NW with a solar height angle of 45 degrees
Shaded relief

Shaded relief

DEM

Darker is lower

Ortophoto
Computing the shaded relief

\[
\text{ShadeValue}\left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right) = \frac{1}{2} + \frac{1}{2} \cdot \left(\frac{p+a}{b}\right)
\]
Computing the shaded relief

\[
\text{ShadeValue}\left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right) = \frac{1}{2} + \frac{1}{2} \cdot \left(\frac{p+a}{b}\right)
\]

\text{a and b are constants}

Normally the parameters \(a\) and \(b\) are set to \(a = 0\) and \(b = \frac{1}{\sqrt{2}}\); these values implies that the shaded values will be between 0 and 1.
Computing the shaded relief

\[
\text{ShadeValue} \left( \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right) = \frac{1}{2} + \frac{1}{2} \cdot \left( \frac{p + a}{b} \right)
\]

where

\[
p = \frac{p_0 \cdot \frac{\partial z}{\partial x} + q_0 \cdot \frac{\partial z}{\partial y}}{\sqrt{p_0^2 + q_0^2}}
\]

\[
p_0 = -\cos(\beta) \tan(\alpha),
q_0 = -\sin(\beta) \tan(\alpha),
\]

\(\alpha\) – solar height angle,
\(\beta\) – solar horizontal angle measured clockwise from North, and
the parameters \(a\) and \(b\) defines the domain of the ShadeValue.
Computing the shaded relief

\[ \text{ShadeValue}\left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right) = \frac{1}{2} + \frac{1}{2} \cdot \left(\frac{p + a}{b}\right) \]

where

\[ p = \frac{\left( p_0 \frac{\partial z}{\partial x} + q_0 \frac{\partial z}{\partial y}\right)}{\sqrt{p_0^2 + q_0^2}} \]

\( \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \) are the partial derivatives of the height \( z \) from West to East \( x \) respective from South to North \( y \).

Here we discuss two methods to approximate the partial derivatives:
- Steepest gradient approach
- Third order approach (Sobel filter)
Approximating the partial derivatives

Steepest gradient approach

$$\left. \frac{\partial z}{\partial x} \right|_{i,j} \approx \frac{\max \left[ z_{i+1,j+k} - z_{i-1,j+k} \right]}{2dx}, \quad k = -1, 0 \text{ or } 1$$

$$\left. \frac{\partial z}{\partial y} \right|_{i,j} \approx \frac{\max \left[ z_{i+k,j+1} - z_{i+k,j-1} \right]}{2dy}, \quad k = -1, 0 \text{ or } 1$$

Where

dx and dy are distances between pixels (resolution) in the DEM in the two coordinate axis direction respectively, and z is the height for a pixel.

Note that -100 is larger than +80. Sign is not important.
Approximating the partial derivatives
Steepest gradient approach

\[
\frac{\partial z}{\partial x} \bigg|_{i,j} \approx \max \left[ \frac{z_{i+1,j+k} - z_{i-1,j+k}}{2dx} \right], \quad k = -1, 0 \text{ or } 1
\]

\[i = \text{columns}, \quad j = \text{rows}\]

\[\begin{array}{ccc}
110 & 120 & 140 \\
120 & 140 & 150 \\
140 & 150 & 200 \\
\end{array}\]

\[
\begin{align*}
140 - 110 &= 30 \\
150 - 120 &= 30 \\
200 - 140 &= 60 \\
\end{align*}
\]

\[
\max \left[ z_{i+1,j+k} - z_{i-1,j+k} \right]
\]

\[
\text{max} = 60
\]
Approximating the partial derivatives
Steepest gradient approach

\[ \frac{\partial z}{\partial x} \bigg|_{i,j} \approx \frac{\max[z_{i+1,j+k} - z_{i-1,j+k}]}{2dx}, \quad k = -1, 0 \text{ or } 1 \]

\[ i = \text{columns, } j = \text{rows} \]

The partial derivative \( dz/dx \)

\[ \max dz = 60 \text{ m} \]
\[ dx = 1000 \text{ m} \]
\[ dz/dx = 60 / (2*1000) = 0.03 \]
Approximating the partial derivatives
Steepest gradient approach

\[
\frac{\partial z}{\partial y}_{i,j} \approx \max \left[ \frac{Z_{i+k,j+1} - Z_{i+k,j-1}}{2dy} \right], \quad k = -1, 0 \text{ or } 1
\]

The partial derivative \( dz/dy \)

max \( dz = -60 \) m
\( dy = 1000 \) m
\( dz/dx = -60/(2*1000) = -0.03 \)
Approximating the partial derivatives
Third order (Sobel filter) approach

\[
\frac{\partial z}{\partial x}_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i+1,j} + z_{i+1,j-1} - (z_{i-1,j+1} + 2z_{i-1,j} + z_{i-1,j-1})}{8dx}, \text{ and}
\]

\[
\frac{\partial z}{\partial y}_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i,j+1} + z_{i-1,j+1} - (z_{i+1,j-1} + 2z_{i,j-1} + z_{i-1,j-1})}{8dy}.
\]
Approximating the partial derivatives
Third order (Sobel filter) approach

\[ \frac{\partial z}{\partial x}_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i+1,j} + z_{i+1,j-1} - (z_{i-1,j+1} + 2z_{i-1,j} + z_{i-1,j-1})}{8dx} \], and

\[ \frac{\partial z}{\partial y}_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i,j+1} + z_{i-1,j+1} - (z_{i+1,j-1} + 2z_{i,j-1} + z_{i-1,j-1})}{8dy} \].

Which is the same as a Sobel filter:

\[ \frac{\partial z}{\partial x} \approx \frac{1}{8dx} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \]

\[ \frac{\partial z}{\partial y} \approx \frac{1}{8dy} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]
Approximating the partial derivatives
Third order (Sobel filter) approach

\[
\frac{\partial z}{\partial x}_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i+1,j} + z_{i+1,j-1} - (z_{i-1,j+1} + 2z_{i-1,j} + z_{i-1,j-1})}{8dx}, \text{ and}
\]

\[
\frac{\partial z}{\partial x} \approx \frac{1}{8dx} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>120</td>
<td>140</td>
</tr>
<tr>
<td>120</td>
<td>140</td>
<td>150</td>
</tr>
<tr>
<td>140</td>
<td>150</td>
<td>200</td>
</tr>
</tbody>
</table>

140 \(-\) 110 = 30
2\(\times\)(150 \(-\) 120) = 60
200 \(-\) 140 = 60
Approximating the partial derivatives
Third order (Sobel filter) approach

\[
\frac{\partial z}{\partial x}_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i+1,j} + z_{i+1,j-1} - (z_{i-1,j+1} + 2z_{i-1,j} + z_{i-1,j-1})}{8dx}, \text{ and}
\]

\[
\frac{\partial z}{\partial x} \approx \frac{1}{8dx} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}
\]

\[
\begin{array}{c|c|c}
110 & 120 & 140 \\
120 & 140 & 150 \\
140 & 150 & 200 \\
\end{array}
\]

\[
140 - 110 = 30 \quad \rightarrow \quad (30 + 60 + 60)/(8*1000)
\]

\[
2*(150 - 120) = 60
\]

\[
200 - 140 = 60
\]
Approximating the partial derivatives
Third order (Sobel filter) approach

\[
\left. \frac{{\partial z}}{{\partial x}} \right|_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i+1,j} + z_{i+1,j-1} - (z_{i-1,j+1} + 2z_{i-1,j} + z_{i-1,j-1})}{8dx}, \text{ and}
\]

\[
\frac{{\partial z}}{{\partial x}} \approx \frac{1}{8dx} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}
\]

<table>
<thead>
<tr>
<th></th>
<th>110</th>
<th>120</th>
<th>140</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>140</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>150</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

\[
140 - 110 = 30 \quad \Rightarrow \quad (30 + 60 + 60)/(8*1000)
\]

\[
2*(150 - 120) = 60
\]

\[
200 - 140 = 60 \quad \Rightarrow \quad 0.019
\]
Approximating the partial derivatives

Steepest gradient approach:

\[
\begin{array}{ccc}
110 & 120 & 140 \\
120 & 140 & 150 \\
140 & 150 & 200 \\
\end{array}
\]

The partial derivative \( \frac{dz}{dx} \)

\[
\text{max } dz = 60 \text{ m} \\
\text{dx} = 1000 \text{ m} \\
\frac{dz}{dx} = 60 / (2*1000) = 0.03
\]

Third order (Sobel filter) approach:

\[
\begin{array}{ccc}
110 & 120 & 140 \\
120 & 140 & 150 \\
140 & 150 & 200 \\
\end{array}
\]

The partial derivative \( \frac{dz}{dx} \)

\[
\text{dx} = 1000 \text{ m} \\
\frac{dz}{dx} = 0.019
\]
Approximating the partial derivatives - row order

\[ \frac{\partial z}{\partial y}_{i,j} \approx \frac{z_{i+1,j+1} + 2z_{i,j+1} + z_{i-1,j+1} - (z_{i+1,j-1} + 2z_{i,j-1} + z_{i-1,j-1})}{8dy} \]
Shaded relief – solar horizontal angle
Shaded relief – solar horizontal angle
Shaded relief – solar horizontal angle
Raster transformation

- **Original grid**
- **Transformed grid**
  (here: affine transformation, translation, rotation, different scales in x and y directions)
Raster transformation

But a raster shall be like this so we need to **resample** it.

![Original grid](image1)

![Transformed grid](image2)

How do we decide the values of the pixels in the resampled raster?

(here: affine transformation)
Resampling method

It is important to state what kind of raster data that is used and the application of the resampled data.

If the raster data is a digital map (or contain any type of data in nominal or ordinal scale) it is not allowed to do calculations on the raster data values. A recommended resampling method is then nearest neighbour.

If the data is an image e.g. remote sensing data (which contain continuous data that is in ratio or interval scale) it is allowed to do calculations on the raster data values.
Nearest neighbour resampling

Original grid

Transformed grid
(here: affine transformation)

How do we find the nearest pixel?
Nearest neighbour resampling

Compute the distances to all pixels and find the shortest.

Original grid

This is for one pixel only and for a small grid.
Nearest neighbour resampling

Instead do the inverse of the raster transformation.
Nearest neighbour resampling

Instead do the inverse of the raster transformation.
Nearest neighbour resampling

Instead do the inverse of the raster transformation.
Inverse distance resampling

\[
    z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k} \cdot z(x_i)}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}}
\]

where

- \( z(x_p) \) is the raster value for the pixel \( p \) (which is a pixel in the new grid),
- \( x_i \) is a pixel in the transformed raster data,
- \( d(\ldots) \) is the Euclidean metric (cf. L1),
- \( k \) is an integer value, and
- \( n \) is the number of pixels used in the interpolation.
Inverse distance resampling

\[ z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k} \cdot z(x_i)}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}} \]

Must determine the two parameters \( n \) and \( k \).
Normally, \( n \) is set to 4 or 9, which implies that the 4 or 9 closest pixels are used in the interpolation.
\( k \) is related to the spatial autocorrelation of the data.
\( k = 0 \) gives an ordinary mean value (among the \( n \) pixels)
\( k \to \infty \) gives a nearest neighbour interpolation.
Normally a value of \( k=2 \) is used.
Inverse distance resampling

Original grid

\[ z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k} \cdot z(x_i)}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}} \]

Transformed grid (here: affine transformation)
Inverse distance resampling

\[ n = 4 \]

\[ \frac{1}{6^k} \times 2 \]

\[ \frac{1}{6^k} \]

\[ z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k} \cdot z(x_i)}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}} \]
Inverse distance resampling

\[ n = 4 \]

\[ \frac{1}{6^k} \cdot 2 + \frac{1}{4^k} \cdot 3 \]

\[ \frac{1}{6^k} + \frac{1}{4^k} \]

\[ z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k \cdot z(x_i)}}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}} \]
Inverse distance resampling

\[ n = 4 \]

\[
\frac{1}{6^k} \times 2 + \frac{1}{4^k} \times 3 + \frac{1}{3^k} \times 1 + \frac{1}{1^k} \times 4
\]

\[
\frac{1}{6^k} + \frac{1}{4^k} + \frac{1}{3^k} + \frac{1}{1^k}
\]

\[
z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k} \times z(x_i)}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}}
\]
Inverse distance resampling

\[ z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k} \cdot z(x_i)}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}} \]

\[ \frac{1}{6^k} \cdot 2 + \frac{1}{4^k} \cdot 3 + \frac{1}{3^k} \cdot 1 + \frac{1}{1^k} \cdot 4 \]

\[ \frac{\frac{1}{6^k} + \frac{1}{4^k} + \frac{1}{3^k} + \frac{1}{1^k}}{\frac{1}{1^k} \cdot 2 + \frac{1}{1^k} \cdot 3 + \frac{1}{1^k} \cdot 1 + \frac{1}{1^k} \cdot 4} = \frac{10}{4} = 2.5 \]
Inverse distance resampling

\[ n = 4 \]

\[ k = 2 \]

\[
\frac{1}{6^k} \times 2 + \frac{1}{4^k} \times 3 + \frac{1}{3^k} \times 1 + \frac{1}{1^k} \times 4
\]

\[
\frac{1}{6^k} + \frac{1}{4^k} + \frac{1}{3^k} + \frac{1}{1^k}
\]

\[
\frac{\frac{1}{36} \times 2 + \frac{1}{16} \times 3 + \frac{1}{9} \times 1 + \frac{1}{1} \times 4}{\frac{1}{36} + \frac{1}{16} + \frac{1}{9} + \frac{1}{1}} \approx 3.62
\]
Inverse distance resampling

Original grid

Transformed grid (here: affine transformation)

\[ z(x_p) = \frac{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k} \cdot z(x_i)}{\sum_{i=1}^{n} \frac{1}{d(x_p, x_i)^k}} \]
Flow analysis in hydrology

**Flow distribution** and **flow accumulation** is crucial in hydrological modelling.

Based on a DEM it is possible to make more reliable estimations of flow over a surface:

Estimate the flow distribution from all cells in a DEM to its eight neighbours and compute the accumulated flow.

Distributed hydrological model:
Water flow is linked to water exchange with the atmosphere (precipitation and evapotranspiration), vegetation cover, and soil properties.
Flow analysis in hydrology

How to distribute the flow to other cells.

Single flow

The water from the center cell flows to the neighboring cell with the lowest elevation

Multiple flow

The water from the center cell is distributed proportionally to a number of neighboring cells with lower elevation
Surface flow algorithms over digital elevation models

### Single flow direction algorithm SFD

\[
\begin{align*}
    f_i &= \frac{(\tan \beta_i)^x}{\sum_{j=1}^{8}(\tan \beta_j)^x} \\
    \text{for all } \beta > 0
\end{align*}
\]

where \(i,j = \) flow directions (1…8), \(f_i = \) flow proportion (0…1) in direction \(i\), \(\tan \beta_i = \) slope gradient between the centre cell and the cell in direction \(i\), and \(x = \) variable exponent.

\(x = 1\) gives a flow proportional to slope gradients \((\beta_i)\)

\(x \rightarrow \infty\) gives a single directional flow.

### Multiple flow direction algorithm MFD
Form based flow direction algorithm

Pilesjö et al. (1998) developed a new model for flow analysis based on the following assumptions:

a. From any point in the terrain, the water flows according to the topographic form of the cell and its eight neighbour cells.

b. Water is evenly distributed over the grid cells (i.e. homogeneous precipitation).

c. The infiltration capacity over the surface is set to zero.

d. The surface is bare (e.g. no vegetation).

e. The evapotranspiration is set to zero.
Form based flow direction algorithm

The topographic form is described with a second order polynomial
Form based flow direction algorithm

A convex surface (flow is split)
Form based flow direction algorithm

A concave surface – one directional flow
Artefacts in real world data
Flow accumulation I

Stream network - Flow to one pixel only
Flow accumulation I

Area draining IN to a pixel - Flow to one pixel only

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>1</td>
<td>19</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Diagram: Area draining IN to a pixel with flow to one pixel only.
Flow accumulation I

Flow Accumulation
\[ \geq 10 \text{ Cell Threshold} \]

Stream Network for 10 cell Threshold Drainage Area
Flow accumulation I

Watershed Draining to Outlet

Total accumulated flow = 20
Flow accumulation II
computing flow accumulation as proposed by Pilesjö et al. (1998):

1. Identify highest (not-vistited) cell
2. Compute the output flow
   \[ F_{\text{out}} = F_{\text{in}} + F_{\text{local}} \]
3. Compute to which cells the flow will run

- \( F_{\text{in}} = \) incoming flow to the cell
- \( F_{\text{local}} = \) flow produced in the cell

Step 3. The flow is based on the topographic form of the cell and its eight neighbours. If the form is convex the flow is distributed according to:

\[ f_i = \frac{(\tan \beta_i)^x}{\sum_{j=1}^{8}(\tan \beta_j)^x} \quad \text{(In this case } x=1). \]

If the form is concave the flow is divided between the two cells that is in the direction of the steepest gradient of the second order polynomial surface.

Special treatment of flat areas and sinks in the DEM, see Pilesjö et al. (1998).
Flow accumulation comparison

Flow accumulation with all flow to the lowest neighbouring cell (ESRI ArcGIS).

Flow accumulation according to Pilesjö et al. (1998).

The water in the single flow model mainly flows in either X,Y or diagonal direction, while the Pilesjö model allows a more general water flow.