

# Catchment delineation and stream network: from a raster DEM to a geospatial database

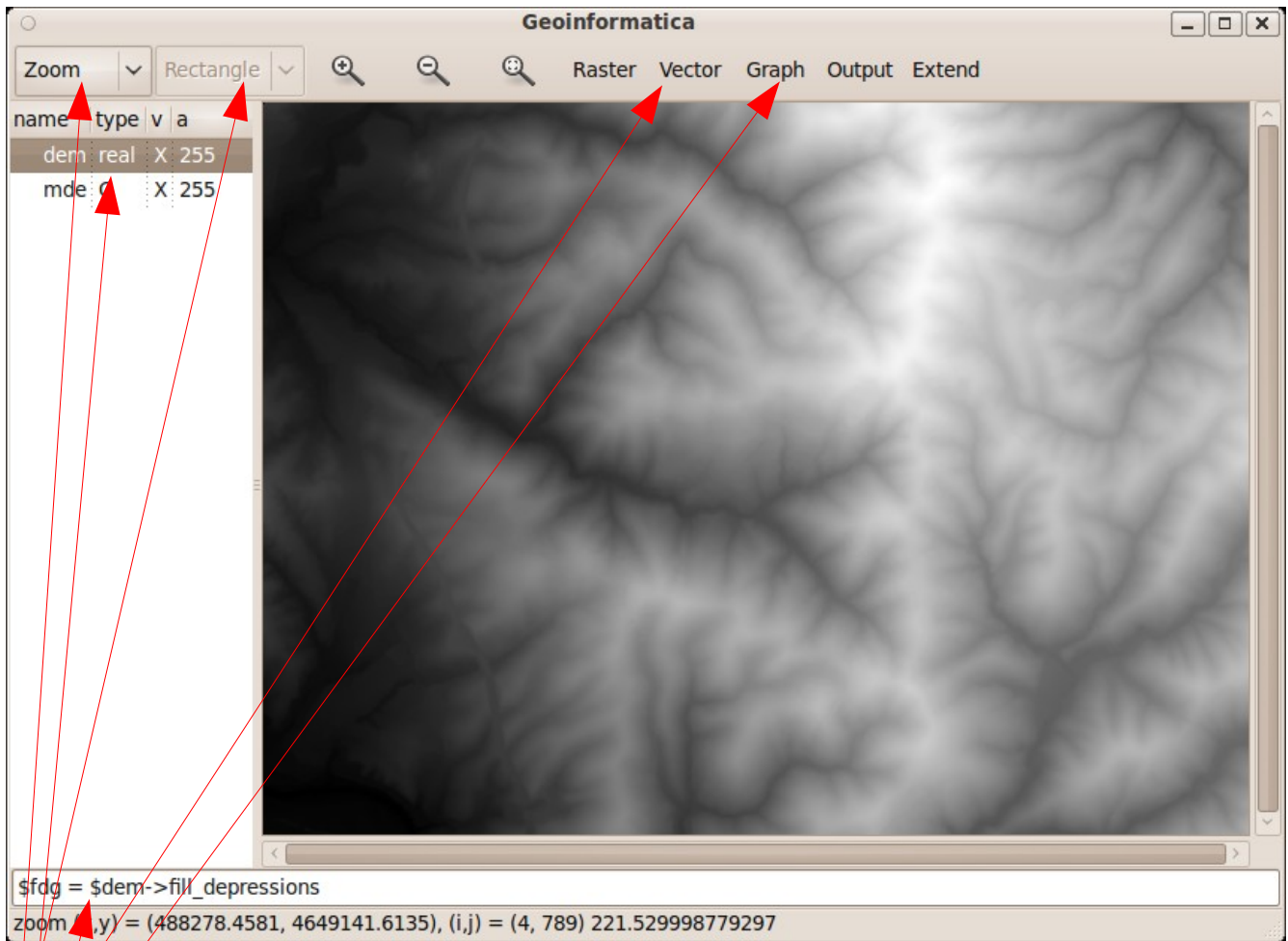
Ari Jolma, 18.6.2010

The goal of this exercise is to create a topological database of a catchment using a raster DEM. The catchment database will contain the stream network as a collection of topologically connected stream segments stored as linestrings, and the catchment structure as a collection of topologically connected subcatchments stored as polygons. The topological connections are stored as attributes of the stream segments and subcatchments.

The exercise introduces the FOSS4G stack Geoinformatica, its graphical user interface, and geoprocessing with GDAL using the Perl programming language. The exercise also introduces the basics of spatial hydrology.

The workflow:

1. Open Geoinformatica – this should be as simple as clicking on a button in the top panel of your Ubuntu desktop
2. Open the raster DEM (mde.tif) – click on the "Raster", select "Open ..." and navigate to the file
3. Zoom into a catchment within the area
4. Copy the visible area into a libral raster – click on the mde layer entry in the list with right mouse button and select "Copy ..."; in the dialog box select libral driver, write name "dem", select "<Current view>" as the region to copy, and press "OK"
5. Initiate the algorithm, which creates a depressionless flow direction raster: enter `$fdg = $dem->fill_depressions` into the command line entry, the process of the algorithm is shown in the output window (Open the output window by clicking the "Output" button)
6. Compute the upstream area raster: enter `$uag = $fdg->upslope_count` into the command line entry. The upstream area raster is more informative if visualized using a logarithm: `$x = ($uag+1)->log`
7. Compute the stream network raster: `$s = $uag > N`, select a suitable N by visual inspection and testing different values
8. Give each stream segment a unique id: `$s->number_streams($fdg)`
9. Compute the subcatchments raster and the topology:  
`($subs,$topology) = $s->subcatchments($fdg)`
10. Save the subcatchments, the stream segments, and the topology into a postgis database: `$cv, $sv) = $subs->vectorize_catchment($topology, $fdg, $s, data_source=>'PG: dbname=postgis')`, select a suitable database name and optionally add user and password parameters for the data\_source named parameter of the subroutine.



- Interaction mode
- Interaction geometry
- Layer class menus
- Open output window
- Layer list
- Command line

**Geoinformatica**

Zoom ▾ Rectangle ▾ 🔍 🔍 🔍 Raster Vector Graph Output Extend

name	type	v	a
sv	OGR U	X	255
cv	OGR U	X	255
subs	int	X	255
s	int	X	255
x	real	X	255
uag	real	X	255
fdg	int	X	255
dem	real	X	255
mde	G real	X	255

```
($cv, $sv) = $subs->vectorize_catchment($topology, $fdg, $s, data_source=>'PG: dbname=dev')
```

```
zoom (x,y) = (487690.6194, 4646961.7858)
```

**Edit Data - localhost ajalma (localhost:5432) - dev - subcatchmen**

File Edit View Tools Help

No limit ▾

	ogc_fid	wkb_geon	element	type	down	type_down
	[PK] seria	geometry	integer	character	integer	character
1	1	010300002(	1	sub	1	stream
2	2	010300002(	0			
3	3	010300002(	0			
4	4	010300002(	2	sub	2	stream
5	5	010300002(	0			
6	6	010300002(	0			
7	7	010300002(	1	sub	1	stream
8	8	010300002(	3	sub	5	stream
9	9	010300002(	4	sub	6	stream
10	10	010300002(	5	sub	4	stream
11	11	010300002(	9	sub	3	stream
12	12	010300002(	0			

Scratch pad

24 rows.

## Discussion

This procedure, and, indeed, partly Geoinformatica itself, is by no means "production ready", it is a product of an on-going research. Especially the last step, the `vectorize_catchment` subroutine, was written mainly for this exercise.

The workflow is implemented using a software stack, which mainly consists of three layers: (i) `libral` / `GDAL`, (ii) the Perl modules and methods, and (iii) the graphical user interface.

The first layer provides low level and fast geoprocessing methods. Some of the methods used in this workflow are generic, such as `polygonize` (`GDAL`) and `map algebra` (`libral`), and some are more specific like the depression filling algorithm (`libral`).

The second layer provides an interactive, ad-hoc access to these methods. This is very useful in research environments, as it allows fast development and testing of the workflow.

The third layer provides graphical interaction and visualization of the results. It also provides a platform for implementing the finished workflow with user-friendly dialog boxes and wizards.

All the source code is in your Ubuntu installation. You are welcome and even invited to make changes in any of them and test those changes. The whole stack can be compiled also on other operating systems, including Windows. However, Linux-based Unix systems are by far the easiest systems to develop on. Compiling and installing libraries written in low level programming languages may be very difficult in Windows for example. Libraries written in high level languages such as Perl and Python are typically installed as source code. Thus, it is even possible to modify the Perl libraries of binary distribution of Geoinformatica for Windows.

The basic spatial hydrology methods of this workflow are implemented using the C programming language in source code file `catchment.c`:

[http://trac.osgeo.org/geoinformatica/browser/libral/trunk/ral\\_catchment.c](http://trac.osgeo.org/geoinformatica/browser/libral/trunk/ral_catchment.c)

The flow direction method that is used by default is the classical D8. There are many others and `Dinf` is probably nowadays preferred over D8. Many of the functions in `catchment.c` are complex because they attempt to manage terrains with lakes. In our case we had only subcatchments and stream segments, but in general subcatchments and streams may drain into lakes, and lakes may have no, one or many outlets, which are stream segments. Also, if lakes are interpreted from remote sensing data, the flow direction raster may actually have flow paths that start and end in the same lake.

The `polygonize` subroutine<sup>1</sup> that is used by the `vectorize_catchment` subroutine<sup>2</sup> uses a method from `GDAL`. The `polygonize` method of `GDAL` was only very recently enhanced to work with 8-connectedness<sup>3</sup>, thus this workflow may produce one-pixel subcatchments. Also, the stream segments that this workflow produces are zero-width lines, which may not be the optimal choice in some cases.

---

1 <http://trac.osgeo.org/geoinformatica/browser/Geo-Raster/trunk/lib/Geo/Raster/Algorithms.pm>

2 <http://trac.osgeo.org/geoinformatica/browser/Geo-Raster/trunk/lib/Geo/Raster/TerrainAnalysis.pm>

3 <http://trac.osgeo.org/gdal/ticket/3619>