GeoPUMMA

Urban and Peri-urban landscape representation tool
for hydrological distributed modeling

https://forge.irstea.fr/projects/geopumma/wiki

Tutorial

Geo-PUMMA v.1.1

Author: Tomás Ignacio Gómez

Revised by Geo-PUMMA Team

January 2016

Table of Contents

1 General Description 2
2 Data preparation 5
2.1 Installing Virtual Box Machine 5
2.2 Copying Location example in GRASS-GIS and data visualization 10
3 Step A: Pre-processing procedures 14
3.1 Import of the Geo-PUMMA routines 14
3.2 Step A.1: Clean Polygon Topology (p.A.clean_topology) 14
4 Step B.1: Delineation and characterization of UHE 16
4.1 Definition of sidewalks and streets associated to each urban lot 16
4.2 Definition of Urban Hydrological Elements 17
4.3 Average altitude calculation 19
4.4 Calculation of Green Areas on each UHE 22
4.5 Calculation of distance between the UHE and the street center lines 23
4.6 Calculation of Built Areas on each UHE 24
5 Step B.2: Segmentation of HRU 27
5.1 Identifying bad shaped elements 27
5.2 Extraction of bad-shaped HRU 29
5.3 Apply Triangle Plugin to the bad-shaped elements 32
5.4 Dissolving by Convexity Index Criterion 37
5.5 Dissolving by Form Factor Criterion (p.form_factor.py) 38
5.6 Dissolving big polygons 39
5.7 Update original mesh 41
5.8 Update HRU altitude info 42
6 Step B.3: Hydrological connectivity 45
6.1 Extracting all interfaces 46
6.2 River Segmentation 47
6.3 Water Table River Interface 48
6.4 Water Table Interface 49
6.5 OLAF (overland flow) 51
6.6 Geo-descriptors 53

1 General Description

This tutorial lets you use Geo-PUMMA, a GIS toolbox to generate vectorial meshes for terrain representation in distributed hydrological modeling, and to get hydrological connectivity in urban and peri-urban catchments (< 10 km^2). Geo-PUMMA generates well-shaped Hydrological Response Units (HRUs) and Urban Hydrological Elements (UHEs) in vectorial shapefile format. Geo-PUMMA can be used to represent the terrain in distributed hydrological modeling for applications at urban and peri-urban scales. This tutorial includes the GIS database for two real sub-catchments extracted from a natural area of Mercier (France) (Fig.1a) and from a mixed urban-natural area of El Guindo (Chile) (Fig.1b).

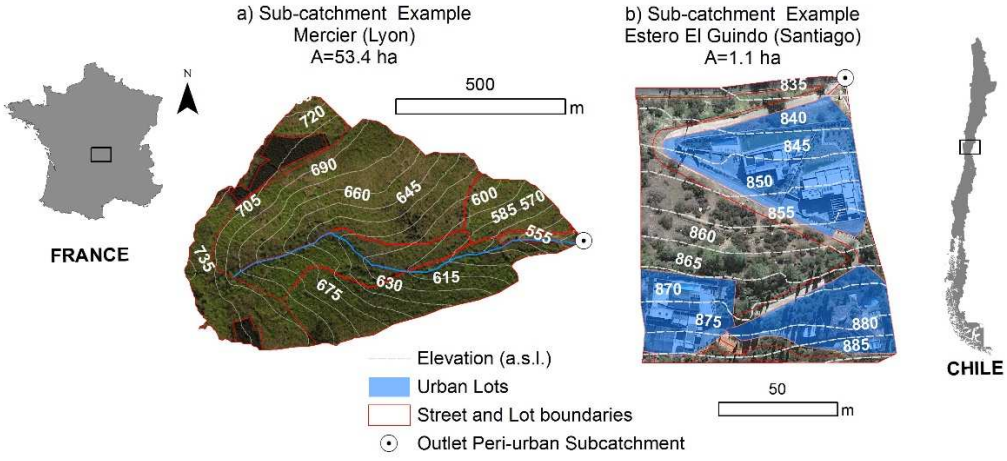


Figure 1 General view of sub-catchments at Mercier and El Guindo

This tutorial will guide the user through a basic and simplified example of the process required to develop and obtain the recommended mesh for two examples. To achieve this, four main steps are to be performed (Table 1):

i. Data preparation
ii. Step A: Pre-processing procedures
iii. Step B.1: Delineation of Urban Hydrological Elements
iv. Step B.2: Segmentation of HRU
v. Step B.3: Hydrological connectivity

Table 1. Tasks in each step of Geo-PUMMA and the corresponding scripts

	Script/Plugin	**Task (optional/compulsory)**
Step A	*p. clean_topology.py*	Cleaning topological polygons (compulsory)
	p. clean_polyline.py	Snapping, breaking and joining polylines (optional)
Step B.1	*p.sidewalk_street.py*	Segmenting part of sidewalk and street in front of each urban lot (compulsory)
	p.uhe.py	Creating the UHE shapefile (compulsory)
	p.a.average_altitude.py	Getting the mean altitude and statistical parameters of each UHE (compulsory)
	p c.wood_surface.py	Getting the green area percentage of each UHE (optional)
	p.length.py	Getting the distance from the centroid to the street centerline (compulsory)
	p.built.py	Getting the building percentage of each UHE (optional)
Step B.2	*p.polygons_holes.py*	Segmenting the HRU with island inside (optional)
	p.shape_factors.py	Calculating shape factors (convexity index, solidity index, form factor and compactness) (compulsory)
	Triangle Plugin	Segmenting the bad-shaped HRU using library Meshpy and Software Triangle implemented in QGIS (compulsory)
	p.convexity.py	Dissolving using convexity and area criterion recommended for highly non-convex polygons (compulsory)
	p.formfactor.py	Dissolving using form factor and area criterion, recommended for thin and needle-shaped polygons such as streets (compulsory)
	p.raster_segmentation.py	Segmenting units with high variability of a given property from raster information (optional)
Step B.3	*p.all_interfaces.py*	Identifying all the interfaces between polygons and/or linear features (WTI and WTRI) (compulsory)
	p.river_segm.py	Segmenting the river considering the WTI and WTRI elements (compulsory)
	p.wtri.py	Identifying all interfaces between HRU/river and UHE/river (compulsory)
	p.wti.py	Identifying all interfaces between HRU/UHE (compulsory)

	Script/Plugin	**Task (optional/compulsory)**
	p.olaf.py	Extracting the drainage network considering overland flow, natural streams and channelized infrastructures (compulsory)
	p.geo_descriptors.py	Updating the database from the model mesh, considering the update of distance and cumulative area as input for computing width and area functions (compulsory)
	p.river_direction.py	Changing all directions of the river's segments considering upstream (-1) or downstream (1) direction (optional)
	p.rebuild_ditch_segments.py	Dissolving all river segments, allows simplifying the number of final segments, keeping their properties uniform (optional)
	p.river_h_s.py	Getting the altitude and slope of each river segment (compulsory)

Geo-PUMMA was developed in GRASS 6.4 in a virtual machine with Ubuntu 14 (64b). Although programming skills are not needed, Geo-PUMMA requires some knowledge on spatial analysis, hydrology and hydrological modeling. It is necessary to be familiar with the use of commands and to have basic knowledge of urban hydrology that allows making decisions when representing urban features. You will learn with this tutorial how to apply the main scripts of Geo-PUMMA. For this, exercises must be completed in the described order.

If you have any suggestions or comment, please send an e-mail to: ppsanzana@uc.cl, isabelle.braud@irstea.cl, jgironas@ing.puc.cl

2 DATA PREPARATION

As it was mentioned on the general description, Geo-PUMMA requires several different programs to be used, including GRASS 6.4, QGIS and Python, running on Ubuntu 14 (64b).

SOFTWARE AVAILABILITY

Availability: https://forge.irstea.fr/projects/geopumma

Year First Available: 2016

Hardware Required: Desktop/Laptop with 2 GHz CPU, 4 GB RAM or more

Operating System Required: Ubuntu 14 (64b) or newer;

Software required: GRASS GIS 6.4, GRASS GIS 7.0, QGIS 2.12

Cost: Free

Program Language: Python

License: GNU General Public License

For those users who do not use this operating system, a good alternative is installing a Virtual Box Machine. This program allows creating a different computer within a Windows, Mac or other Unix environment.

Additionally, an already virtualized service is available for users to download, which will provide them with a work environment that already possess all the software required, in the operating system mentioned above.

2.1 INSTALLING VIRTUAL BOX MACHINE

To install the Virtual Box Machine, please visit their website (see Figure 2, <https://www.virtualbox.org/>) and download the corresponding file that matches your operating system from the download section (see Figure 3).

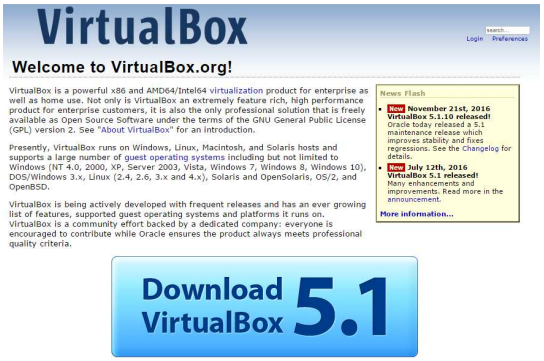


FIGURE 2 VIRTUAL BOX HOME WEBSITE

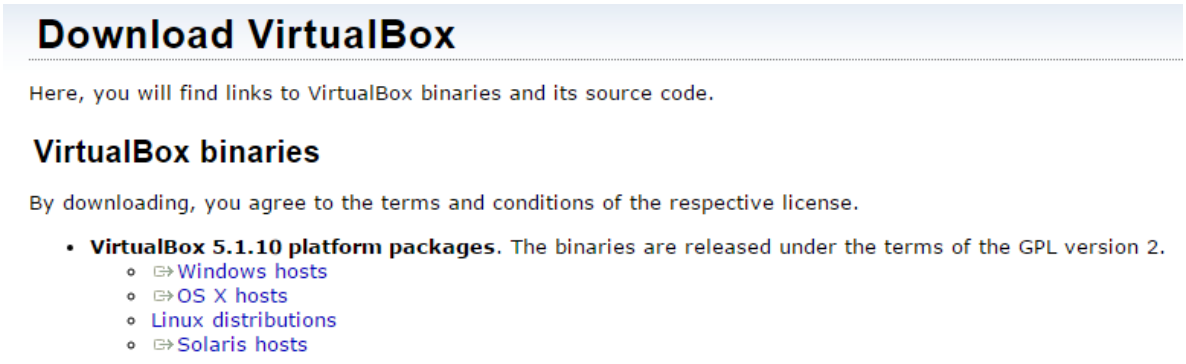


FIGURE 3 VIRTUAL BOX DOWNLOAD SECTION

To install Virtual Box, execute the downloaded installer and follow the instructions of the installation wizard, as is shown in the following figures. Figure 4 shows the first screen, just click next. The second screen (Figure 5) gives you alternatives on what to install. In this case, the default options are acceptable. Click next. Third screen (Figure 6) allows you to choose some shortcut alternatives, choose your preference then click next. Fourth screen (Figure 7) gives you a warning about losing connectivity for some periods during the installation, click yes to continue. The fifth screen (Figure 8) represents the final confirmation before starting the installation process. If everything is ok, click on Install. Windows-users may get a system warning, as shown in Figure 9, click on Install to continue the installation process.

FIGURE 4 FIRST SCREEN OF THE INSTALLATION WIZARD

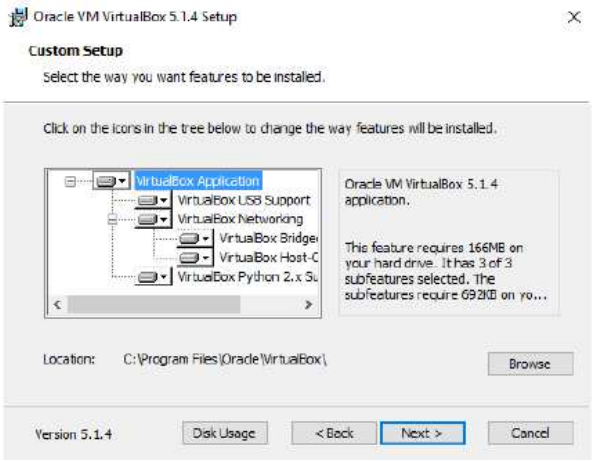


FIGURE 5 SECOND SCREEN OF THE INSTALLATION WIZARD

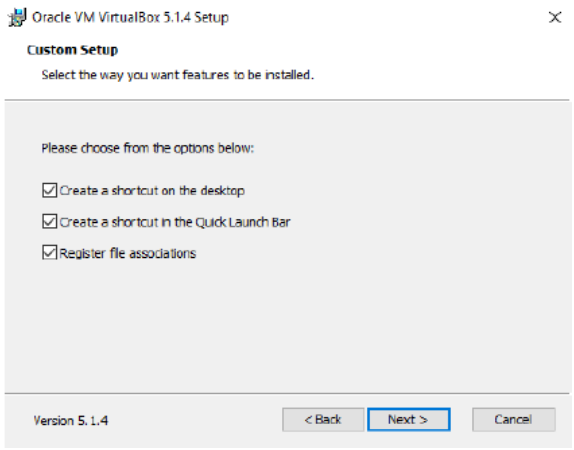


FIGURE 6 THIRD SCREEN OF THE INSTALLATION WIZARD

FIGURE 7 FOURTH SCREEN OF THE INSTALLATION WIZARD

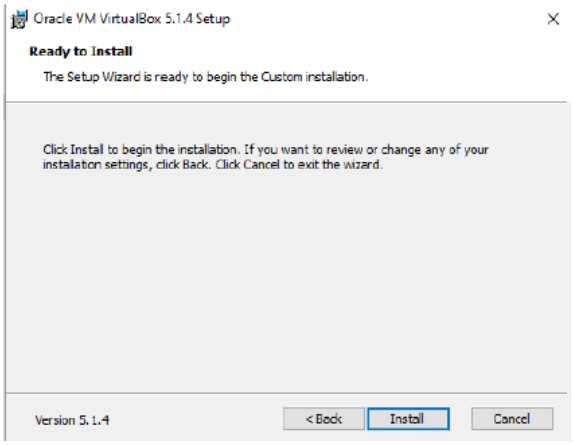


FIGURE 8 FIFTH SCREEN OF THE INSTALLATION WIZARD

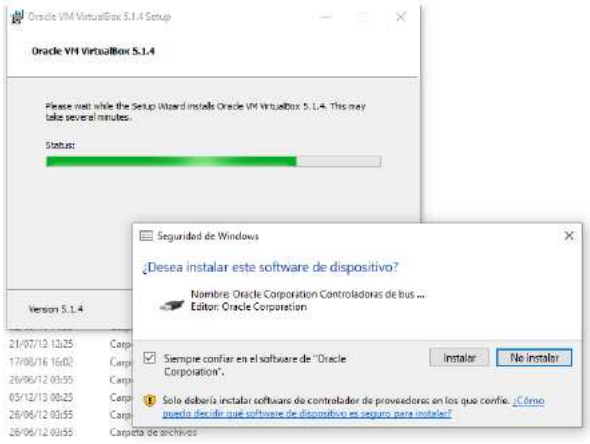


FIGURE 9 SIXTH SCREEN OF THE INSTALLATION WIZARD

Once the installation process has been finished correctly, open Virtual Box (Figure 10). What follows now is the installation of the virtual machine available at the repository[1]. To do this, click on "File", then select "Import virtual service". A screen like the one shown in Figure 11 will show up. Select the virtual machine file and click on next. It may take a few moments because of the file size.

[1] Official Repository https://forge.irstea.fr/projects/geopumma
Alternative Repository https://www.dropbox.com/s/paphvhb1fx3646o/GEOPUMMA_64B_3_1_VF.7z?dl=0

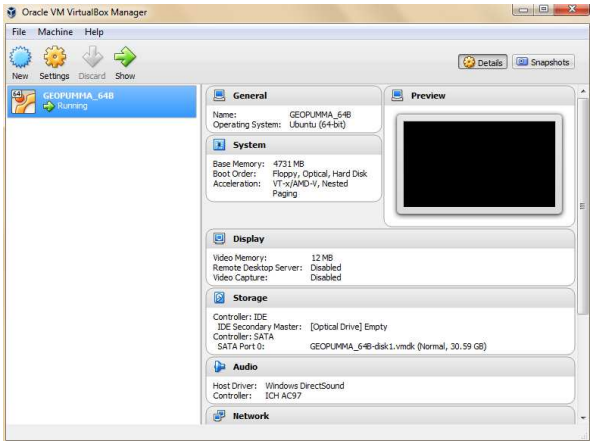


FIGURE 10 VIRTUAL BOX MAIN SCREEN

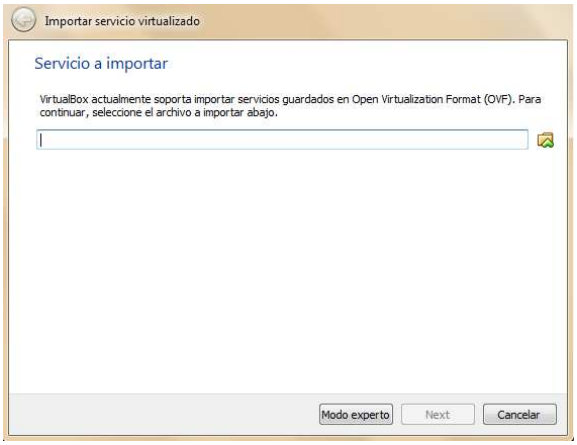


FIGURE 11 IMPORT VIRTUAL SERVICE SCREEN

Once the virtual machine has loaded, it is necessary to check if the operating system allows working with virtual machines of 64 bit. In the case that it is not allowed, it will only show Ubuntu (32-bit) among the alternatives (see Figure 12 for details on where to check whether you have this problem or not).

If the 64-bit option does not show, try with the following instructions:

i. Turn off your computer
ii. Turn it back on and immediately press ESC or F10 to access the BIOS menu (some computers may access this menu with a different key)
iii. In the BIOS menu, activate the option that allows virtual machines.
iv. Accept and then go to exit, saving the changes.

Once you restart the computer, the option for Ubuntu (64-bit) should be available.

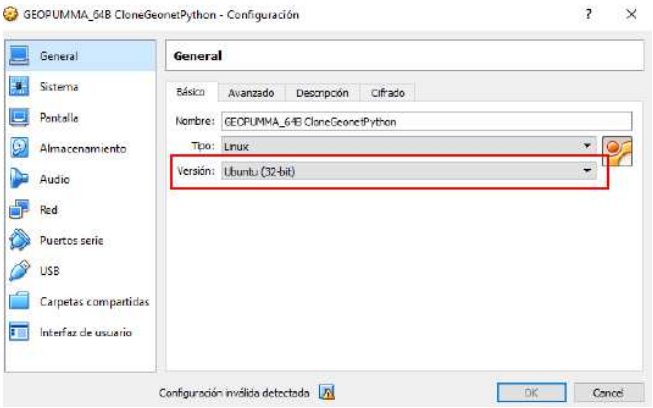


FIGURE 12 VIRTUAL MACHINE SETTINGS WINDOW

The hard disk drive space allocated to the virtual machine is not accessible from the original operating system. However, it is possible to set a shared folder that will be accessible from both the main

operating system and the virtual machine. To do this, on the settings menu, go to the "Shared folders" tab and select the desired location, by clicking on the add folder icon. Create a folder named "Lyon_2014_psanzana" on the hard drive and select it. Also, select the option "Auto-mount" and "Make permanent". Then click "Ok" to close the window and then click on "Ok" again to add this folder. See Figure 13 for a detailed visual representation of these steps.

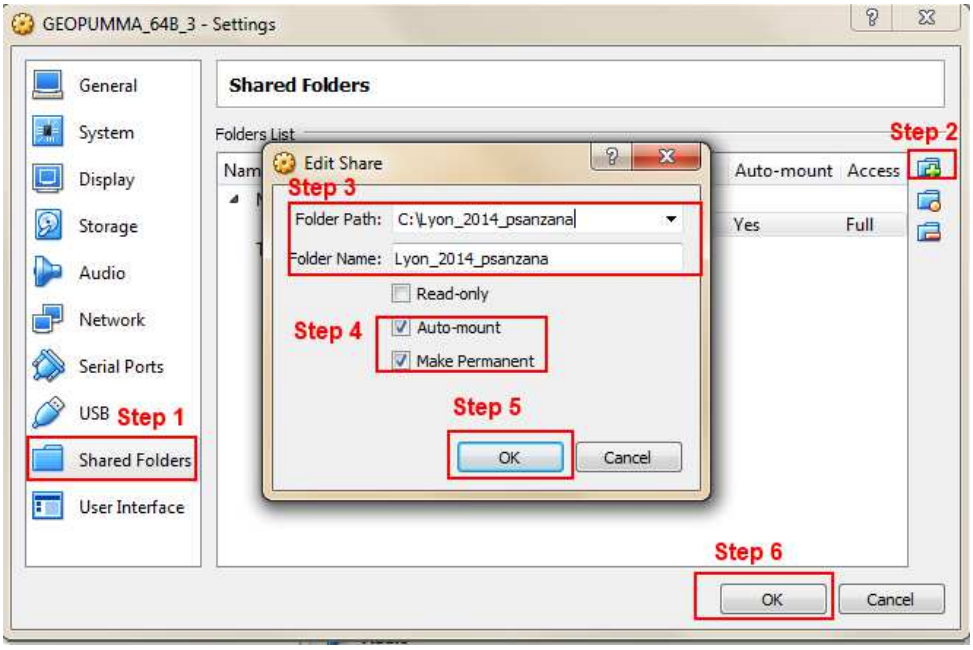


FIGURE 13 ADDING A SHARED FOLDER BETWEEN THE MAIN OPERATING SYSTEM AND THE HARD DISK DRIVE

The last remaining step is to adjust the options for the base memory. The base memory allocated should not exceed the indicated green limit, which varies according to the characteristics of each computer (Figure 14). Also, in the screen tab, the option for 3d acceleration must be enabled (Figure 15).

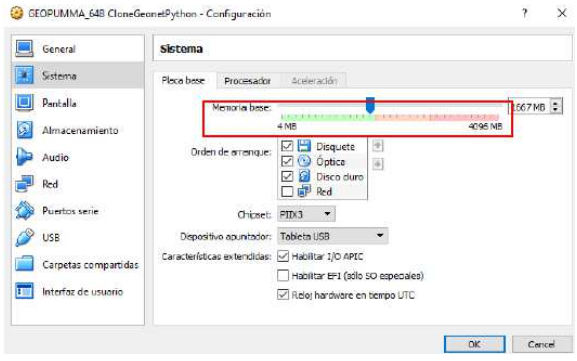


FIGURE 14 MEMORY ALLOCATION SCREEN

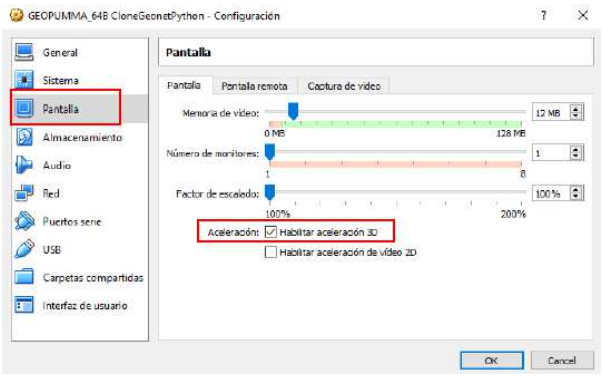


FIGURE 15 3D ACCELERATION CHECK

After this, click on show to start the virtual machine. Ubuntu works with a light and useful desktop simulator (Figure 16).

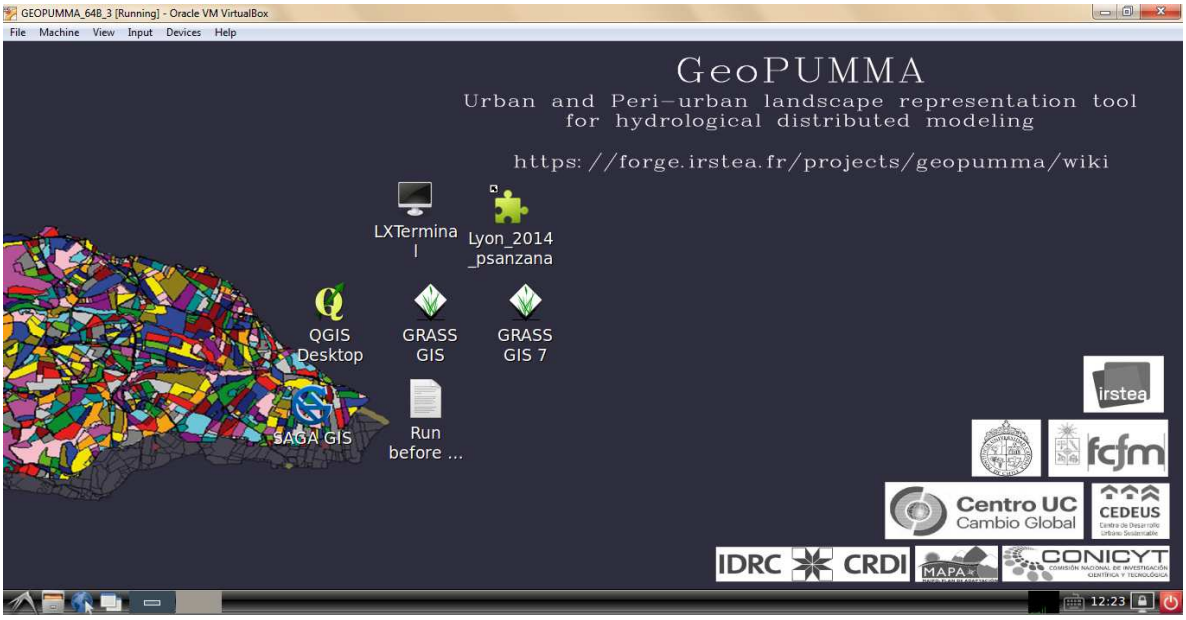


FIGURE 16 GEOPUMMA VIRTUAL MACHINE DESKTOP SCREEN SHOT

2.2 COPYING LOCATION EXAMPLE IN GRASS-GIS AND DATA VISUALIZATION

Data provided for this tutorial should be copied into the shared folder. In this case the folder is <C:\Lyon_2014_psanzana\>. Copy and paste the Mercier and the El Guindo files provided in the tutorial in this folder. Also, copy the files with all the required python routines in the path <C:\Lyon_2014_psanzana\geopumma\Trunk\>

Once these preparations have been done, it is time to move on to the virtual machine and start working there.

First of all, open GRASS GIS. Once the program is open, the VM should look like Figure 17. Note that two locations corresponding to Estero El Guindo and Mercier areas have been preset. To create another different location, please refer to the GRASS user guide and tutorial that guides you through this process.[2]

[2]Check https://**grass**.osgeo.org/documentation/**tutorials** for further information

FIGURE 17 GRASS INITIAL WINDOW.

From here, it is necessary to create a different mapset in each location, where the shapefiles to be processed must be installed. For this, click on the "Create mapset" button, then write down the name "initial_mapset" and finally click on "Ok" (see Figure 19 for a graphic explanation).

Additionally, an extra mapset named "hydrological_connectivity" must be created in the Mercier location. Once this has been done, the start-up window of GRASS should look as shown below.

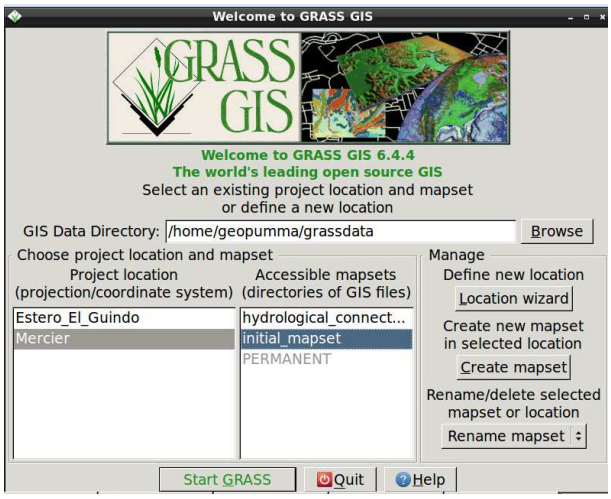


FIGURE 18 START-UP WINDOW OF GRASS AFTER CREATING THE MAPSET ON THE MERCIER LOCATION

After this, select the initial_mapset on the Mercier location, and click on the "Start GRASS" button to begin executing the program. After it finishes starting, three different windows should come up: the

layer manager window, the display window and the GRASS console. The layer manager window allows the management of different layers, where it is possible to choose which layers to show, or process them, if the "Command console" tab is selected. The display window shows the graphic representation of the selected layers (see Figure 20).

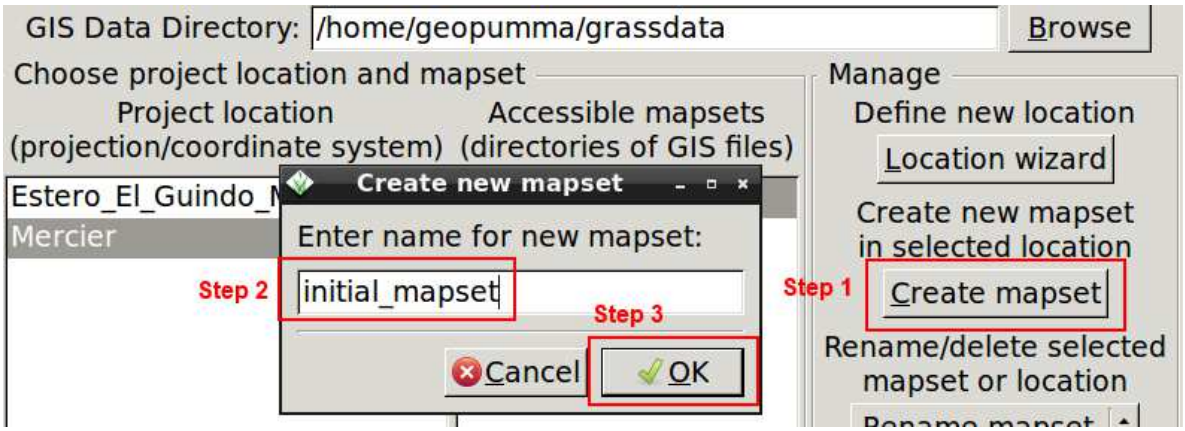


FIGURE 19 STEPS TO CREATE A NEW MAPSET

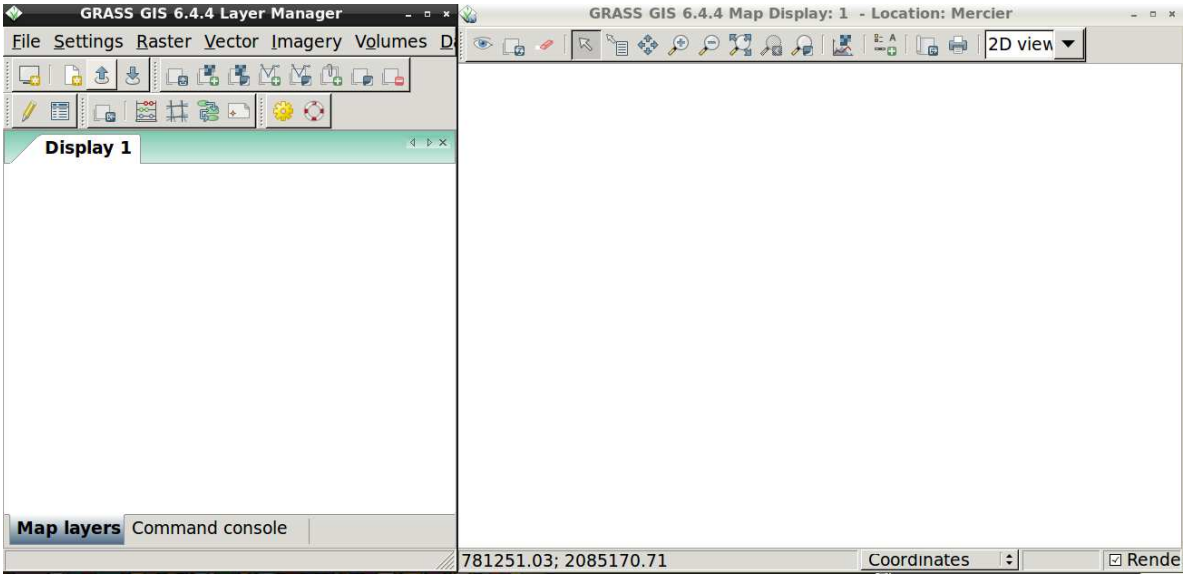


FIGURE 20 LAYER MANAGER AND DISPLAY WINDOWS OF GRASS

The console window is the window where instructions can be written and processed to perform different geospatial processes.

The first process to be performed is the import of the maps. For this, the command v.in.ogr is used. This command imports a vector shapefile into the GRASS mapset. Copy and paste the following instructions on the console to import the base map files that had previously been copied in the folder <C:\Lyon_2014_psanzana\>. Three maps must be imported for each location, representing the initial

mesh, the river and flow network (which may include different waste water networks) and the output point of the catchment.

For the "initial_mapset" in the Mercier area, the command to be written on the console is the following:

```
v.in.ogr dsn
=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_initial_mesh/merc
ier_initial_mesh.shp output=mercier_mesh
```

The command lines to be executed in the initial_mapset (to be created) of the El Guindo location are as follows:

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/el_guindo_cadastre/el_
guindo_cadastre.shp output=el_guindo_cadastre
```

```
v.in.ogr                                                                -z
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/el_guindo_folvoirie/el
_guindo_folvoirie.shp output=el_guindo_street_center_line
```

3 STEP A: PRE-PROCESSING PROCEDURES

An initial mesh of the catchment is required to implement the Geo-PUMMA processing method. A good approximation of what is required can be obtained from the Geo-MHYDAS procedure[3], if it is not already available. Additional to a first mesh of the catchment, it is also required a polyline layer representing the rivers and drainage ditches (the main water courses) and a layer with the outlet point of the catchment (which should be the end point of the ditch layer)

Once the initial mesh, the ditch network and the output point have been imported, it is possible to start working with them. However, a little preprocessing is required to ensure the information has been correctly formatted.

The importation of the library and a check of the topology of the mesh is performed in this section.

3.1 IMPORT OF THE GEO-PUMMA ROUTINES

In order to be able to use the Geo-PUMMA routines, these must be imported into GRASS every time a new session of the program is started. To do this, simply execute the following code on the GRASS console. This will add to the library the python routines that are in the specified location (in this case, the location specified is the one created on the previous section).

```
export PATH=/home/geopumma/geomhydas:/home/geopumma/geopumma:$PATH
```

It is important to note that these routines will require of an active interaction between the user and the console to specify the required input through the prompt; the prompt will ask for whatever input may be necessary.

3.2 STEP A.1: CLEAN POLYGON TOPOLOGY (P.A.CLEAN_TOPOLOGY)

This step uses a routine whose purpose is to eliminate any redundant line or border that may not be obvious to the eye. After the routine has been applied, it is necessary to create a column that identifies each element of the mesh with a unique identifier ID.

Routine:	p.A1.clean_topology.py
Required inputs:	• Initial mesh of the catchment • Snap threshold value • Map of initial columns
Output:	Initial mesh of the catchment with a clean topology

Copy and paste the following command on the console:

```
p.A1.clean_topology.py
```

[3] Check < http://community.openfluid-project.org/index.php/MHYDAS_Installation> for more information on this methodology.

This will initiate a routine, where the user will be asked to specify some inputs: name of the file to be processed, output name of the file to be created, a snap threshold value and the name of the map to get the initial columns (Figure 21).

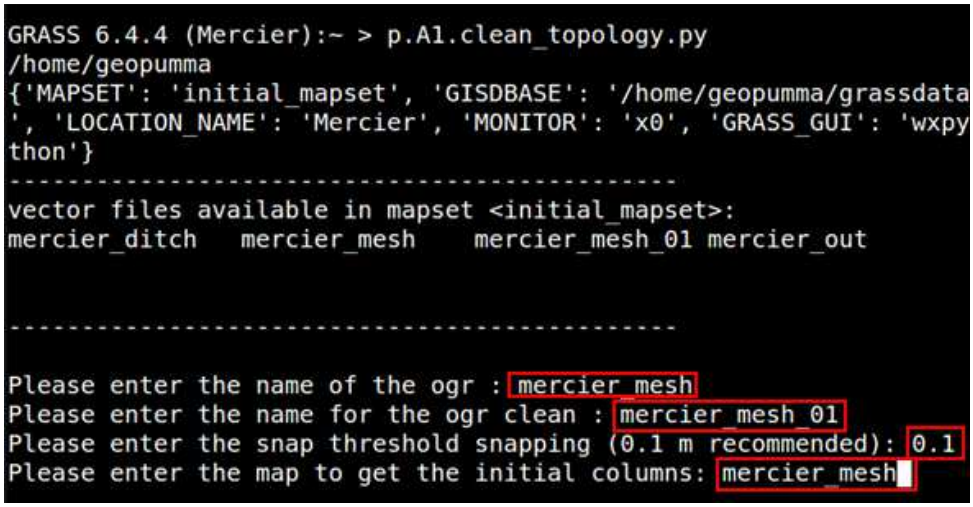


FIGURE 21 APPLICATION OF THE ROUTINE P.A1.CLEAN_TOPOLOGY

With this, a new layer with no topological errors named mercier_mesh_01 has been created. To finish preparing the layer, it is required that each element of the mesh must have a unique ID for identification purposes. To do this, write the following commands on the console:

```
v.db.addcol map=mercier_mesh_01 columns="id_mesh int"
v.db.update map=mercier_mesh_01 column=id_mesh qcolumn=cat
```

4 STEP B.1: DELINEATION AND CHARACTERIZATION OF UHE

A section of the El Guindo catchment, located in Santiago, Chile, will be used in this tutorial. A cadastral map has been provided and already imported into the GRASS location of El Guindo on a new mapset, as a result of the previous steps. It is important to note that while Geo-Pumma provides a methodological approach to process the UHE, the base information must be acquired from the land use layer of the study area, if this approach is to be used on a different location than this tutorial. This base information must specify not only that an element is of urban use, but also specify it is an urban lot, a land plot or street.

The first step to create the UHE is to extract urban lots, land plots and streets from the land use layer in which all the built elements are digitalized. This step is not included within this tutorial since there a several approaches than can be done and it is left for the user to decide in case it would require to do so. For the purpose of this tutorial, this information is provided and should have already been imported into the initial_mapset from the El Guindo location on GRASS.

4.1 DEFINITION OF SIDEWALKS AND STREETS ASSOCIATED TO EACH URBAN LOT

A relevant input for the definition of the UHEs is the polyline representing the axis of every street, from which the distance to each UHE is computed. Because no specific script is available in Geo-PUMMA, this polyline must be digitalized from the urban street layer either manually or using computer-assisted tools

Routine:	p.B1.sidewalk_street.py
Required inputs:	El Guindo cadaster layer El Guindo Street center line for the road involved
Output:	This routine will create a set of temporal layers that identify many features such as parcels that are connected to a road, sections of each road assignted to each of those parcels, among other things.

To run this routine, execute the following command:

```
p.B1.sidewalk_street.py
```

The execution of this routine will require two inputs that need to be provided manually. The first is the name of the cadastral layer and the second one is the name of the layer that contains the information of the streets center line. Both of these layers should be already in the mapset, if the instructions from the previous section were followed. An example of the application of this routine is shown in Figure 22.

```
GRASS 6.4.4 (Estero_El_Guindo):~ > p.B1.sidewalk_street.py
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_N
AME': 'Estero_El_Guindo', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
------------------------------------------
vector files available in mapset <initial_mapset>:
el_guindo_cadastre                    el_guindo_street_center_line


------------------------------------------

Please enter the name of the file 'cadastre' or 'occupation du sol' : el_guindo_
cadastre
Please enter the name of the file 'filvoirie' or 'center of the street' : el_gui
ndo_street_center_line
```

FIGURE 22 APPLICATION EXAMPLE OF ROUTINE PB1.SIDEWALK_STREET.PY

The result of this routine is a set of different layers that will be used in the following steps. The list of generated layers is shown in Figure 23.

```
vector files available in mapset <initial_mapset>:
el_guindo_cadastre                    parcelles_masquees
el_guindo_street_center_line          parcelles_viaires
elements_viaires                      trottoir
filvoirie                             voirie
parcelles                             voirie_d
```

FIGURE 23 LIST OF LAYERS RESULTING FROM THE P.B1.SIDEWALK_STREET.PY

4.2 DEFINITION OF URBAN HYDROLOGICAL ELEMENTS

Once the sidewalks and streets sections associated to each lot have been defined, it is possible to merge them to get an UHE. This UHE will be composed by a parcel that may contain built areas and green areas, as well as the corresponding sidewalk and street sections. This routine will merge geographically all these elements into one, that will be considered the basic unit for the urban areas analysis.

Routine:	p.B2.uhe.py
Required inputs:	Streets and sidewalks per parcel's layer Parcel's layer
Output:	Urban Hydrological Elements

To run this routine, execute the following command:

```
p.B2.uhe.py
```

The execution of this routine will require two inputs that need to be provided manually and the both are part of the set of layer generated in the previous step. An example of the execution of this routine is shown in Figure 24. The first one to be specified is "parcelles_viaires", which corresponds to the plots of the urban land and the second one corresponds to "trottoir" that corresponds to the street

elements, already processed from the routine on the previous steps. The result of this routine can be seen in Figure 25.

Note that there are some blank areas that are not filled. The reason for this is that those areas correspond to other land uses, such as vegetation, and therefore, it is not incorrect to not have the complete area develop as UHE.

```
GRASS 6.4.4 (Estero_El_Guindo):~ > p.B2.uhe.py
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_
': 'Estero_El_Guindo', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
-------------------------------------------
vector files available in mapset <initial_mapset>:
el_guindo_cadastre                      parcelles_masquees
el_guindo_street_center_line            parcelles_viaires
elements_viaires                        trottoir
filvoirie                               voirie
parcelles                               voirie_d


-------------------------------------------

Please enter the name of the layer for 'parcelles_viaires' : parcelles_viaires
Please enter the name of the layer for 'trottoir' : trottoir
```

FIGURE 24 APPLICATION EXAMPLE OF ROUTINE PB2.UHE.PY

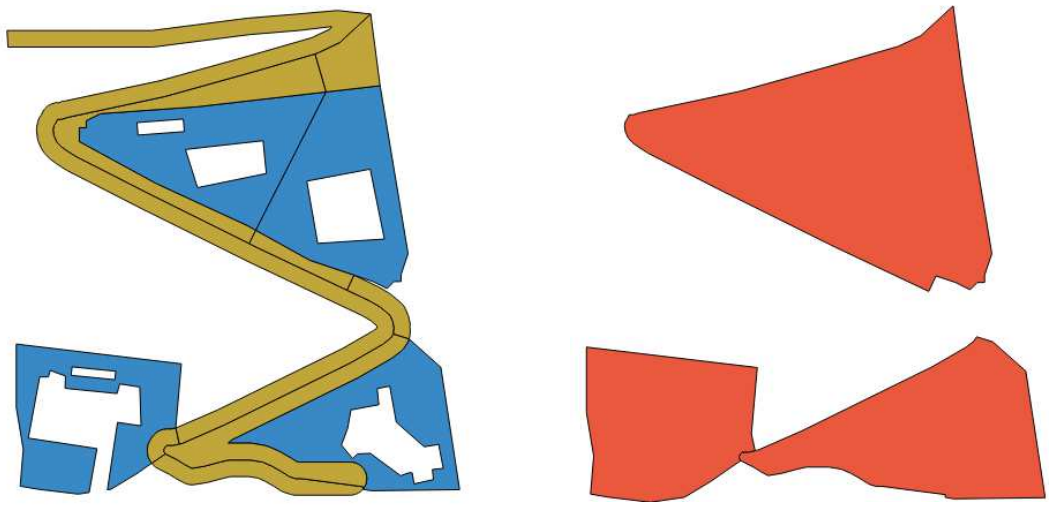


FIGURE 25 COMPARISON OF THE URBAN LOTS LAYER AND THE POST-PROCESSED STREET AND SIDEWALKS LAYER WITH RESULT OF THE APPLICATION OF THE P.B2.UHE.PY

The routine for cleaning topology (p.A1.clean_topology.py) is then executed with the new UHE layer, to avoid topological mistakes. This procedure is executed with the command:

```
p.A1.clean_topology.py
```

An example of its application can be seen in Figure 26. Once this routine has been executed, a new shapefile will have been created on the main directory under the name “uhe_clean”.

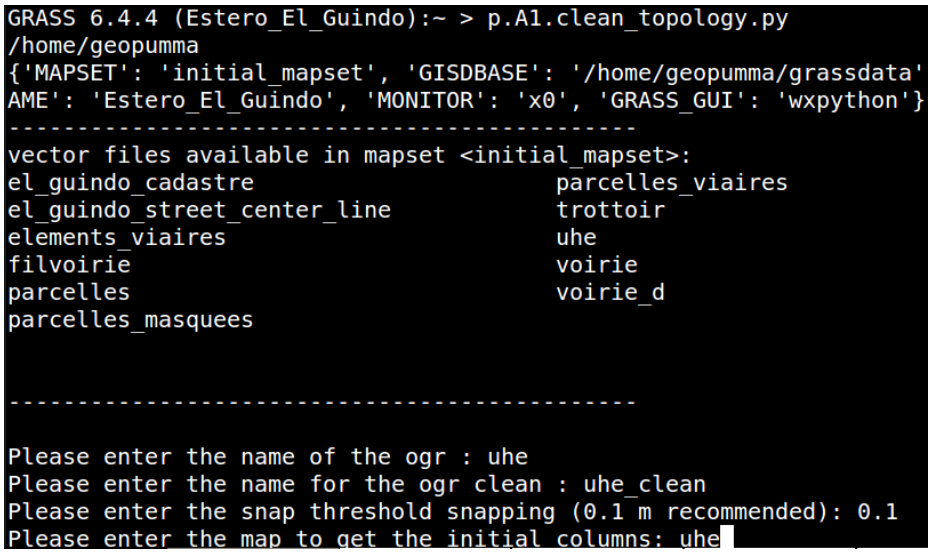


FIGURE 26 APPLICATION EXAMPLE OF THE ROUTINE P.A1.CLEAN_TOPOLOGY.PY

4.3 AVERAGE ALTITUDE CALCULATION

The definition of the UHE has been performed by joining lots and its corresponding sidewalks and streets associated with them. However, once these elements have been determined, it is necessary to define the geographical properties of each one of them.

This routine will calculate and add the information of average altitude (and other statistical parameters) to each of the UHE, based on the information provided by a DEM of the zone. However, since this process requires a raster layer, it will be developed on a newer version of GRASS than the one currently being used. This is because the latest version allows a better handling of raster information. To exit GRASS 6 by simply executing the following command on the console:

```
Exit
```

Then click on the GRASS 7 shortcut that is on the desktop. The startup window for this version is quite similar to that of GRASS 6. The same locations from GRASS6 will be available. Create a new mapset on the Mercier location and name it GRASS7. At this point, the screen should look like Figure 27.

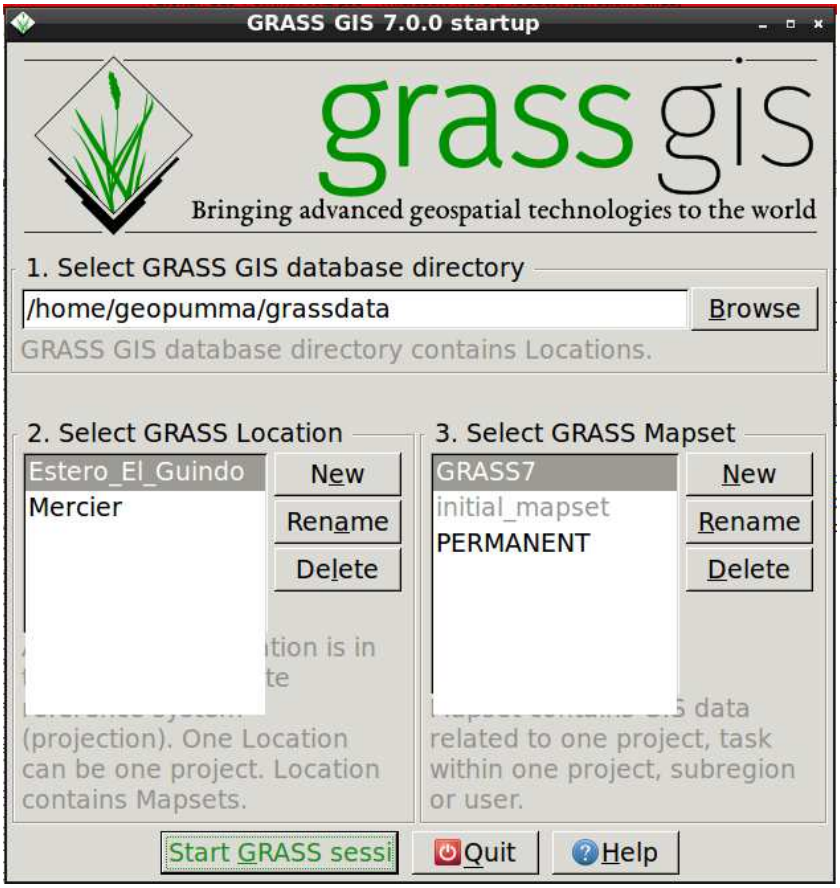


FIGURE 27 GRASS7 START SCREEN AFTER CREATING THE NEW MAPSET AT THE EL GUINDO LOCATION

Select the GRASS7 mapset recently created and start the program. The look of the program is almost identical to that GRASS 6, so the working environment should be quite familiar. The console is where the processing procedures are going to be developed.

Before any further action is taken, it is necessary to load all the Geo-PUMMA routines by running the following command line:

```
export PATH=/home/geopumma/geomhyda:/home/geopumma/geopumma:$PATH
```

After this, it is necessary to import the UHE layer and the raster map to the new mapset. Execute the following commands on the console to do this:

```
r.in.gdal
input=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/dem_el_guindo.tiff
output=el_guindo_dem
```

```
v.in.ogr input=/home/geopumma/uhe_clean/uhe_clean.shp layer=uhe_clean
output=uhe_clean geometry=None
```

It is necessary to specify the workspace region of the DEM. For this, the following command needs to run. It will define the work region as the area defined by the raster map that was previously imported.

```
g.region raster=el_guindo_dem
```

Now that the UHE layer and the DEM have been imported, and the work area properly specified, it is possible to update the elevations of each UHE using the following command:

```
p.B3.a.average_altitude.py
```

Routine:	p.B3.a.average_altitude.py
Required inputs:	El Guindo UHE layer El Guindo digital elevation model (DEM) A prefix name for the columns to be generated
Output:	Updated El Guindo UHE layer with altitudes

This routine will require the manual specification of the layers to be processed. In this case, three inputs are required. The first one is the name of the UHE layer. Next, it is required to specify the name of the DEM that contains the information of the elevation in the area. Finally, it is necessary to specify a prefix for the columns to be generated with the altitude information. An example of this process is shown in Figure 28.

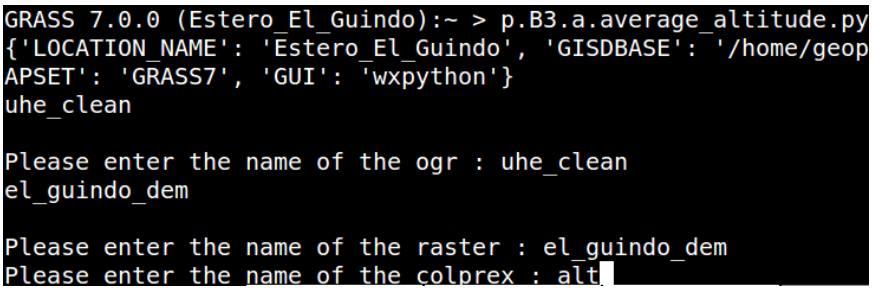


FIGURE 28 APPLICATION EXAMPLE OF THE P.B3.A.AVERAGE_ALTITUDE.PY ROUTINE

After this routine has been executed, the uhe_clean layer will be updated with altitude information for each of its units, as it is shown in Figure 29.

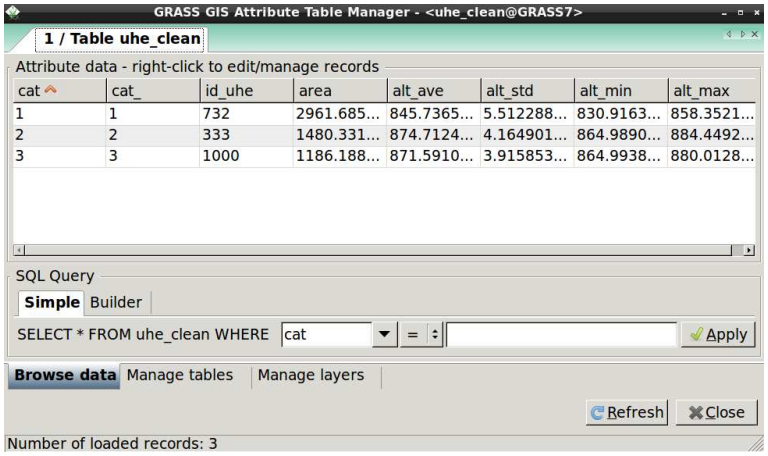


FIGURE 29 P.B3.A.AVERAGE_ALTITUDE.PY APPLICATION OUTPUT

4.4 CALCULATION OF GREEN AREAS ON EACH UHE

In the case that a detailed digitalization of each lot is available, this routine will calculate the area that is covered by green areas in each UHE. For this tutorial, this information is provided and must be imported into a new layer called **el_guindo_wooded_areas** through the following command line:

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/el_guindo_wooded_areas
/el_guindo_wooded_areas.shp output=el_guindo_wooded_areas
```

Once this information has been imported, the routine can be executed by the command:

```
p.B3.c.wood_surface.py
```

This routine will require the manual specification of the layers to be processed. In this case, two inputs are required. The first one is the layer with the UHE. The second one corresponds to the layer just imported, that contains the information about the wooded areas. The resulting layer will contain all the green areas within the defined UHE and will have an output name that is made of the name of the layer containing the wooded areas and the suffix _uhe (in this case el_guindo_wooded_areas_uhe). An example of the application of this routine is shown in Figure 30.

Routine:	p.B3.c.wood_surface.py
Required inputs:	UHE layer Wooded areas layer
Output:	UHE layer updated with the green percentage area on each unit

A visual example of the output of this routine is shown in Figure 31.

```
GRASS 6.4.4 (Estero_El_Guindo):~ > p.B3.c.wood_surface.py
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_
AME': 'Estero_El_Guindo', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
------------------------------------------
vector files available in mapset <initial_mapset>:
el_guindo_cadastre                      parcelles_viaires
el_guindo_street_center_line            trottoir
el_guindo_wooded_areas                  uhe
el_guindo_wooded_areas_uhe              uhe_clean
elements_viaires                        uhe_clean_boise
filvoirie                               voirie
parcelles                               voirie_d
parcelles_masquees


------------------------------------------

Please enter the name of the vector with UHE areas: uhe_clean
Please enter the name of the vector with wooded areas: el_guindo_wooded_areas
```

FIGURE 30. APPLICATION EXAMPLE OF THE ROUTINE P.B3.C.WOOD_SURFACE.PY

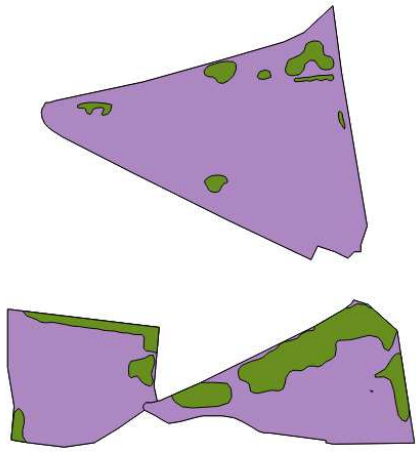


FIGURE 31. RESULT EXAMPLE OF THE ROUTINE P.B3.C.WOOD_SURFACE_PY. THE AREAS IN GREEN REPRESENT THE GREEN COVERED AREA IN EACH OF THE UHE

4.5 CALCULATION OF DISTANCE BETWEEN THE UHE AND THE STREET CENTER LINES

This routine will calculate and add the information of the distance between the centroid of each UHE and the closest center line of a street. To run this routine, execute the command:

```
p.B4.length.py
```

This routine requires two parameters to be specified: the name of the cadastral layer and the name of the UHE layer. An example of the routine being run is shown in Figure 32.

Routine:	p.B4.length.py
Required inputs:	UHE layer
Output:	UHE layer updated with distance from the centroids

```
GRASS 6.4.4 (Estero_El_Guindo):~ > p.B4.length.py
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_N
AME': 'Estero_El_Guindo', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
------------------------------------------
vector files available in mapset <initial_mapset>:
el_guindo_cadastre                      parcelles_viaires
el_guindo_street_center_line            trottoir
el_guindo_wooded_areas                  uhe
el_guindo_wooded_areas_uhe              uhe_clean
elements_viaires                        uhe_clean_boise
filvoirie                               voirie
parcelles                               voirie_d
parcelles_masquees


------------------------------------------

Please enter the name of the file 'cadastre' or 'occupation du sol' : el_guindo_
cadastre
Please enter the name of the file 'uhe' : uhe_clean
```

FIGURE 32 APPLICATION EXAMPLE OF ROUTINE P.B4.LENGTH.PY

An image showing the expected result of this routine is shown in Figure 33. It consists of a layer where each element within an UHE has a point associated with it an attribute table information about the distance from this point to the street center line.

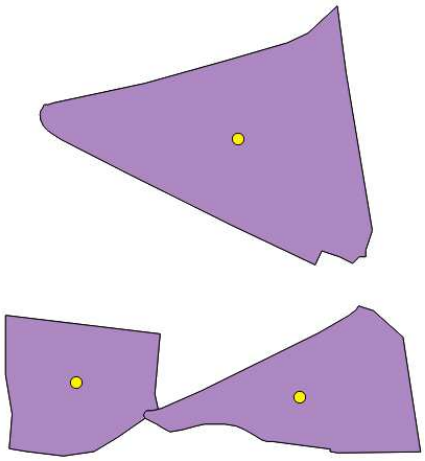


FIGURE 33 EXAMPLE OF THE OUTPUT OF THE P.B4.LENGTH.PY ROUTINE

4.6 CALCULATION OF BUILT AREAS ON EACH UHE

In the case that a detailed digitalization of each lot is available, this routine will calculate the area that is covered by built areas in each UHE. For this tutorial, this information is provided and must be imported into a new layer called **el_guindo_built_areas** through the following command line:

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/el_guindo_built/el_gui
ndo_built.shp output=el_guindo_built
```

Once this information has been imported, the routine can be executed by the command:

```
p.B5.built.py
```

This routine requires the manual input of two layers and an output name. The first one is the layer with the built areas information and the second one is the name of the input to be obtained. Finally, an output name for the resulting layer must be provided as well. An image showing this process is shown in Figure 34.

Routine:	p.B5.built.py
Required inputs:	UHE
Output:	UHE layer updated with distance from the centroids

```
GRASS 6.4.4 (Estero_El_Guindo):~ > p.B5.built.py
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_NAME
': 'Estero_El_Guindo', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
------------------------------------------
vector files available in mapset <initial_mapset>:
el_guindo_built                         el_guindo_wooded_areas_uhe
el_guindo_cadastre                      length
el_guindo_street_center_line            uhe
el_guindo_wooded_areas


------------------------------------------

Please enter the name of the built areas : el_guindo_built
Please enter the name of the file 'uhe' : uhe
Please entre the name of the built areas classified by UHE ID : el_guindo_uhe_built
```

FIGURE 34 APPLICATION EXAMPLE OF THE P.B5.BUILT.PY ROUTINE

An image showing the expected result of this routine is shown in Figure 35.

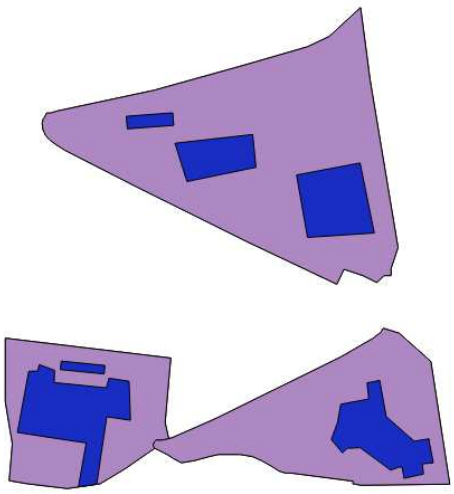


FIGURE 35 EXAMPLE OF THE OUTPUT OF THE P.B5.BUILT.PY ROUTINE. THE BUILT AREAS ARE SHOWN IN ORANGE AND THE UHE ARE SHOWN IN BROWN.

Please note that after running this last routine, most of the temporary layers created on the first step are removed automatically. Once this last step has been performed, the preparation process of the UHE and their required information for the PUMMA model have been successfully completed.

5 STEP B.2: SEGMENTATION OF HRU

Once the pre-processing of the data provides an initial mesh, using the base information available for the catchment, such as topography, land uses, geology, among others, it is possible to begin the process of refinement to determine appropriate Hydrological Response Units (HRU).

This process through its different commands aims to correct different geometric features that are not adequate for the purposes of representing units in hydrological modeling, such as non-convex shapes, that end up with their barycenter out of their boundaries, or shapes that are too long and that end up, in the actual exercise of modeling, acting "like a wall" that prevents appropriate routing of the hydrological network.

This section and all the routines and processing that involve, will be performed on the Mercier area. Although these routines are applied to the complete catchment, they rarely will have an impact on the UHE defined on the previous sections, due to inherent shape and properties that define them.

5.1 IDENTIFYING BAD SHAPED ELEMENTS

This routine involves the creation of several new columns with information about indexes that will help identifying problematic elements of the mesh. The indexes created by this routine include the convexity index and the form factor.

There are three criteria that define an element as problematic:

i. HRU shape is non-convex
   This kind of HRU tends to be troublesome since they usually have their barycenter outside of their boundaries, which leads to problems when defining the hydrological network between different HRU. They will be identified using the **convexity index.**
ii. HRU is too "slim" (rectangle shapes)
   This kind of HRU will generate problems when defining the hydrological network since long slim elements will act "as a wall", because each unit will have an average altitude and thus could mislead appropriate representation of the network. They will be identified using the **form factor.**
iii. HRU is too big
   This kind of HRU is considered bad for purposes of hydrological modeling since it will not represent appropriately the network. They will be identified using the **area.**

Routine:	p.B7.shape_factors.py
Required inputs:	• Mercier_initial_mesh_clean
Output:	• Segmented river layer

Write down the following code on the console. After entering it, it will show all the available vector files and ask for the name of the polygon map to process. The layer map that is specified will have

several new columns with information about the convexity index, the form factor, among other indexes (see Figure 36).

```
p.B7.shape_factors.py
```

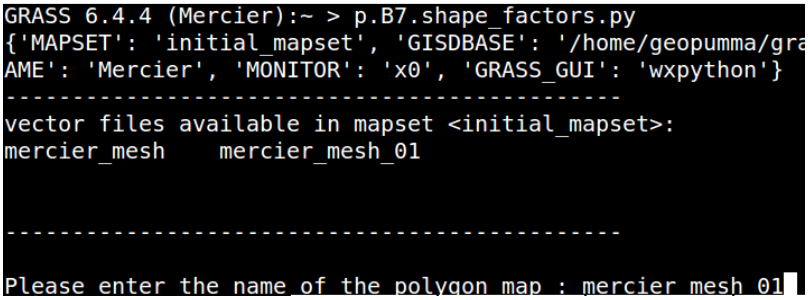


FIGURE 36 APPLICATION EXAMPLE OF P.B7.SHAPE_FACTORS.PY

It is possible to check if the routine performed correctly by checking if the columns were effectively added. To do this, open QGIS from the shortcut on the desktop, and add the map layer mercier_mesh_01. To do this, click on the icon for "add GRASS layer" (step 1) and then select the appropriate layer (step 2) and finally click on "OK" (Step 3) (see Figure 37).

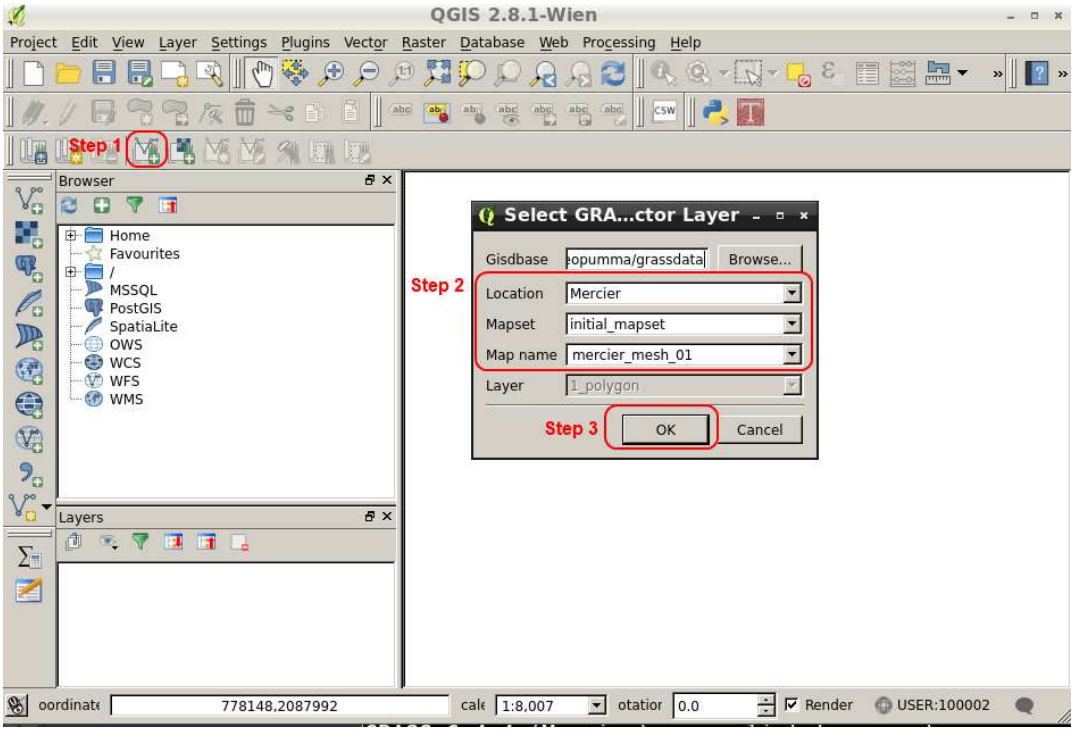
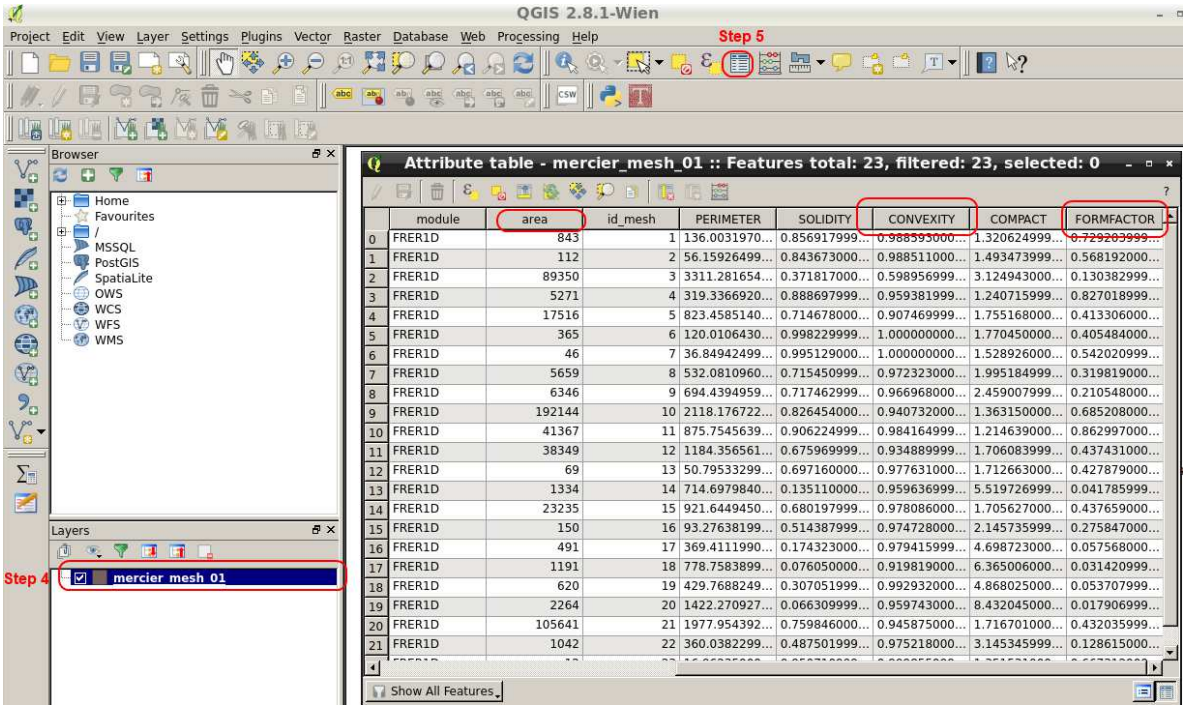


FIGURE 37 ADDING A GRASS LAYER ON QGIS

Once the layer has been added, select it (step 4), then click on the open attribute table (step 5) and check that indexes were indeed processed (see Figure 38).



	module	area	id_mesh	PERIMETER	SOLIDITY	CONVEXITY	COMPACT	FORMFACTOR
0	FRER1D	843	1	136.0031970...	0.856917999...	0.988593000...	1.320624999...	0.729203999...
1	FRER1D	112	2	56.15926499...	0.843673000...	0.988511000...	1.493473999...	0.568192000...
2	FRER1D	89350	3	3311.281654...	0.371817000...	0.598956999...	3.124943000...	0.130382999...
3	FRER1D	5271	4	319.3366920...	0.888697999...	0.959381999...	1.240715999...	0.827018999...
4	FRER1D	17516	5	823.4585140...	0.714678000...	0.907469999...	1.755168000...	0.413306000...
5	FRER1D	365	6	120.0106430...	0.998229999...	1.000000000...	1.770450000...	0.405484000...
6	FRER1D	46	7	36.84942499...	0.995129000...	1.000000000...	1.528926000...	0.542020999...
7	FRER1D	5659	8	532.0810960...	0.715450999...	0.972323000...	1.995184999...	0.319819000...
8	FRER1D	6346	9	694.4394959...	0.717462999...	0.966968000...	2.459007999...	0.210548000...
9	FRER1D	192144	10	2118.176722...	0.826454000...	0.940732000...	1.363150000...	0.685208000...
10	FRER1D	41367	11	875.7545639...	0.906224999...	0.984164999...	1.214639000...	0.862997000...
11	FRER1D	38349	12	1184.356561...	0.675969999...	0.934889999...	1.706083999...	0.437431000...
12	FRER1D	69	13	50.79533299...	0.697160000...	0.977631000...	1.712663000...	0.427879000...
13	FRER1D	1334	14	714.6979840...	0.135110000...	0.959636999...	5.519726999...	0.041785999...
14	FRER1D	23235	15	921.6449450...	0.680197999...	0.978086000...	1.705627000...	0.437659000...
15	FRER1D	150	16	93.27638199...	0.514387999...	0.974728000...	2.145735999...	0.275847000...
16	FRER1D	491	17	369.4111990...	0.174323000...	0.979415999...	4.698723000...	0.057568000...
17	FRER1D	1191	18	778.7583899...	0.076050000...	0.919819000...	6.365006000...	0.031420999...
18	FRER1D	620	19	429.7688249...	0.307051999...	0.992932000...	4.868025000...	0.053707999...
19	FRER1D	2264	20	1422.270927...	0.066309999...	0.959743000...	8.432045000...	0.017906999...
20	FRER1D	105641	21	1977.954392...	0.759846000...	0.945875000...	1.716701000...	0.432035999...
21	FRER1D	1042	22	360.0382299...	0.487501999...	0.975218000...	3.145345999...	0.128615000...

FIGURE 38 CHECK OF THE INDEXES

5.2 EXTRACTION OF BAD-SHAPED HRU

Once the mesh has been processed and the auxiliary indexes for identification of bad shaped elements have been calculated, it is possible to start correcting them.

Let's start with the non-convex items. It is necessary to create a layer with all the HRU that have a Convexity Index less than 0.75. This value is used as a threshold as it represents an empirical value that appropriately represents a balance between calculation time and determining the appropriate shapes for the HRU.

To select only those elements with a CI<0.75, the following code is used:

```
v.extract                                    input=mercier_mesh_01@initial_mapset
output=mercier_non_convex where="convexity<0.75"
```

This will create a new map layer named mercier_non_convex that should look as shown in Figure 39.

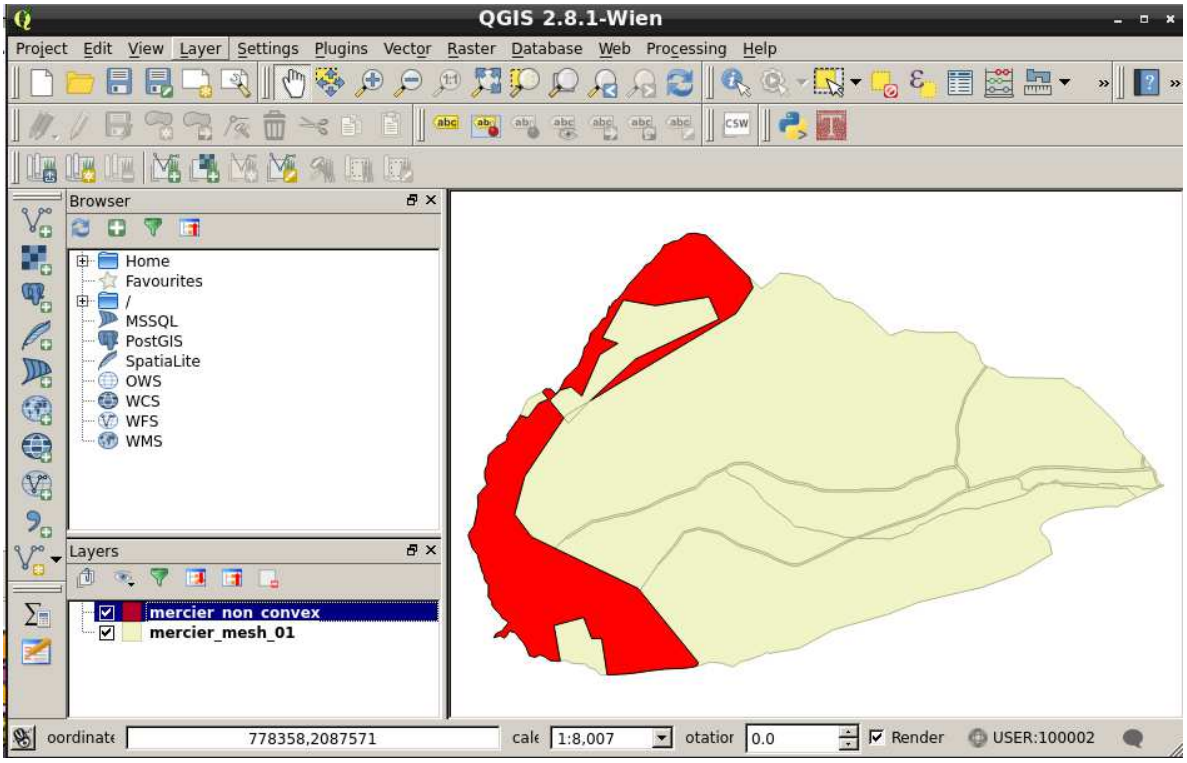


FIGURE 39 EXAMPLE OF THE OUTPUT OBTAINED FROM THE SELECTION OF NON-CONVEX ELEMENTS. NON-CONVEX ELEMENTS ARE SHOWN IN RED

Analogously, the same process is then performed to obtain the HRU with a form factor value less than 0.2. The code for this is:

```
v.extract input=mercier_mesh_01@initial_mapset
output=mercier_long_elements where="convexity>=0.75 AND formfactor<0.2"
```

Note that the conditions for selections are two: form factor less than 2 and convexity index equal or over 0.75. This last condition aims to not repeat elements, so they are not processed twice, which could lead to potential over resizing.

Finally, one last operation creates a layer for all the elements with size more than 20,000 m^2.

```
v.extract input=mercier_mesh_01@initial_mapset output=mercier_big_areas
where="convexity>=0.75 AND formfactor>=0.2 AND area>20000"
```

Note that this last code includes both of the conditions for convex elements and elements that are not too “slim” as to avoid duplicates, similar to the condition before.

A visual representation of the obtained new map layers with the bad shaped HRU is shown in Figure 40.

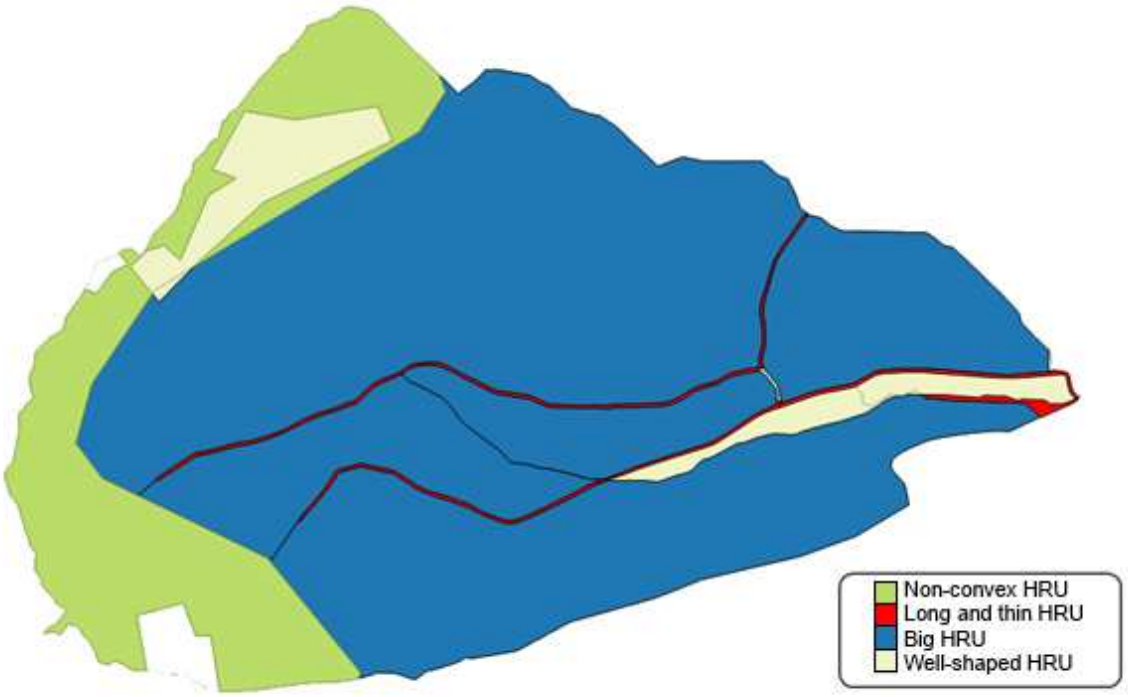


FIGURE 40 MAP LAYERS OF IDENTIFIED BAD-SHAPED HRU

A final command line to extract all the HRU that are correct is executed to identify all those elements that do not fit into the bad shaped criteria and to be patched together with the processed HRU.

```
v.extract  input=mercier_mesh_01@initial_mapset  output=mercier_correct
where="convexity>=0.75 AND formfactor>=0.2 AND area<=20000"
```

5.3 APPLY TRIANGLE PLUGIN TO THE BAD-SHAPED ELEMENTS

The next step to process the non-convex elements of the mesh is to divide it into smaller triangle shapes. To do this, QGIS and its triangle plug-in will be used (QGIS has it already installed).

However, there is one previous step that needs to be performed on the **mercier_long_elements** map layer. These long-and-thin elements must be split from big lines to small lines that present an appropriate number of vertexes for the triangulation process. To do this the following command should be applied:

```
v.split input=mercier_long_elements output=mercier_long_elements_split
length=5
```

This will create a new additional map layer only for the bad shaped HRU that have a form factor less than 0.2 (meaning, the long-and-thin ones).

Once this has been done, it is time to move on to do the triangulation process. On Figure 39, the GRASS layer of the non-convex elements has been added, in the same way the layer mercier_mesh_01 was. The triangle plug-in could be used directly over this layer. However, it is a good practice to transform this layer into a shapefile, which provides with more stable capabilities. To do this, simply do a right click on top of the layer and select “Save as” (Figure 41). After clicking on the “Save as” option a window like Figure 42 will pop up. Click on browse (step 1) to specify the location and name of the shapefile being created. Go to the location <media/ Lyon_2014_psanzana> (step 2), name the file mercier_non_convex_shp (step 3) and then click on “Save” (step 4). This will close the window. Also, click on the option “Add saved map” to immediately add the map to QGIS (step 5). Then, click on “Ok” (step 6) and the file will be created and added to the current QGIS project.

This process must also be performed for the map layers **mercier_long_elements** and **mercier_big_areas.** By the end of this process, the layers window on QGIS should look similar to Figure 43. Now that the three layers of bad shaped elements have been transformed into shapefiles, it is time to divide them into triangles that can be later processed.

To achieve this, we need to open the triangle plug in, which is circled on Figure 41.

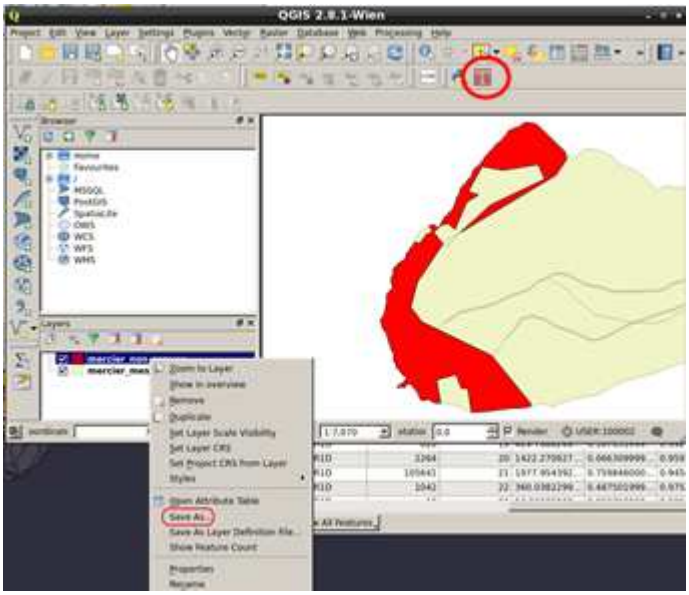


FIGURE 41 SAVING A GRASS LAYER TO SHAPEFILE (1/2)

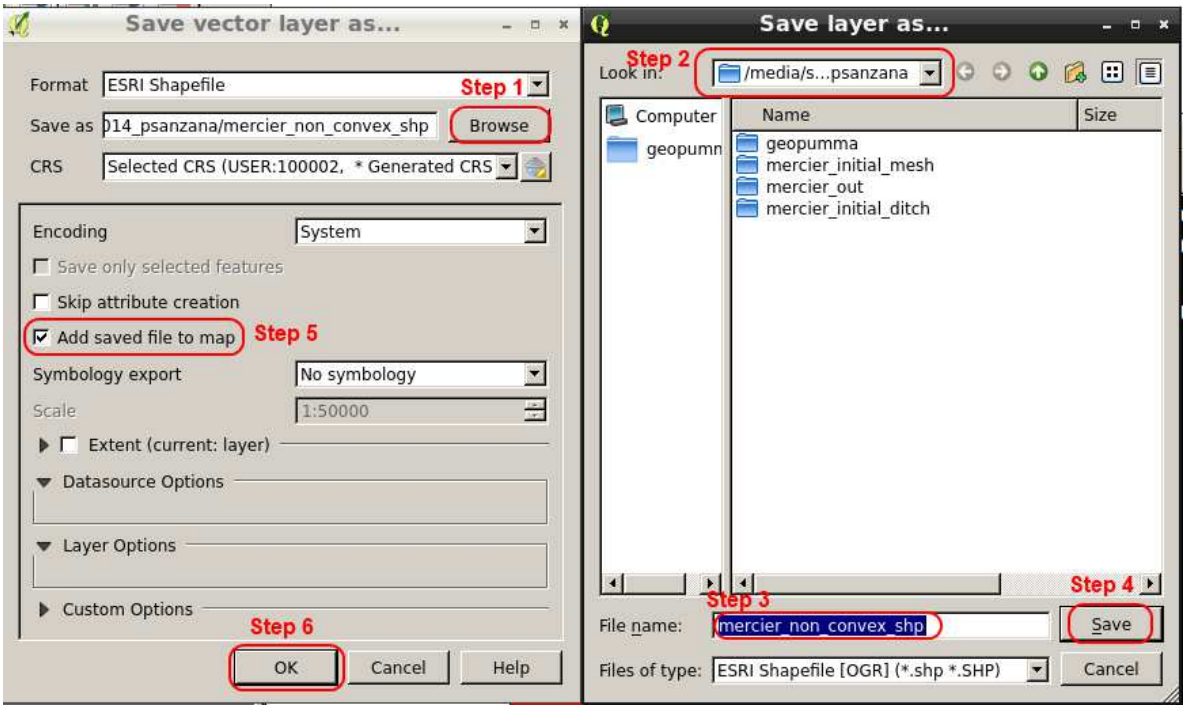


FIGURE 42 SAVING A GRASS LAYER TO SHAPEFILE (2/2)

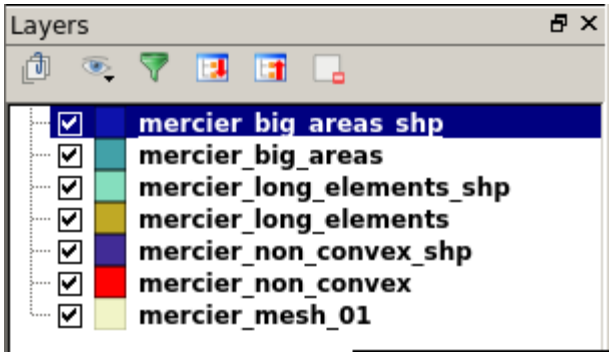


FIGURE 43 ALL LAYERS ON QGIS, AFTER SAVING EVERYTHING TO SHAPEFILE

The Triangle plug-in looks as Figure 44. On the same figure, it is shown as example the configuration for the processing of the non-convex elements layer. The configuration for each of the layers to process is shown on the table below.

Case	Non convex elements layer	Long elements layer	Big areas layer
Select layer to triangle	mercier_non_convex_shp	mercier_long_elements_split_shp	mercier_big_areas_shp
Minimum Angle	0	0	0
Maximum Area	*Leave empty*	*Leave empty*	20,000
Name of new Mesh	mercier_non_convex_triangle	mercier_long_elements_triangle	mercier_big_areas_triangle
Shape descriptor	No shape descriptor	No shape descriptor	No shape descriptor
Descriptor Threshold	0	0	0

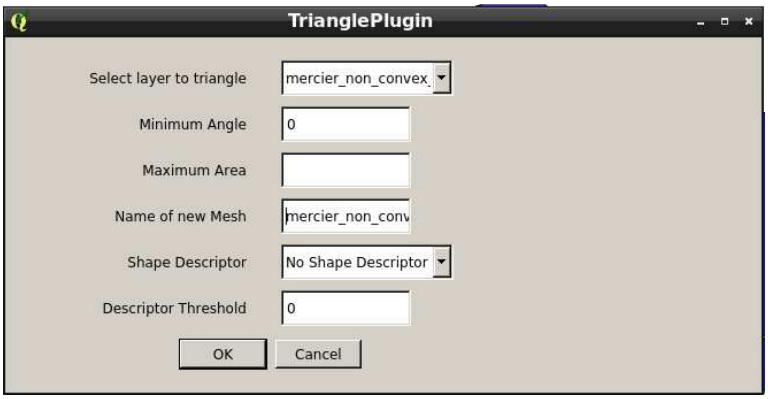


FIGURE 44 TRIANGLE PLUG-IN

An example of the outcome of the Triangle Process is shown in Figure 45.

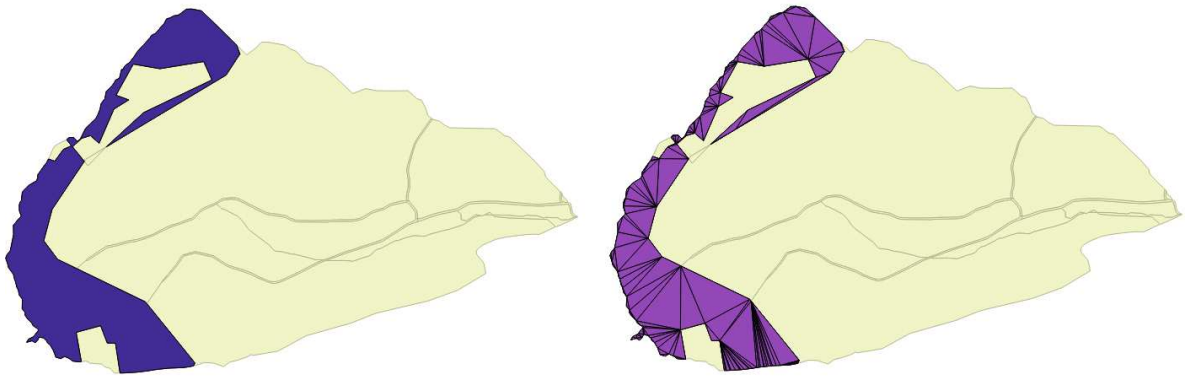


FIGURE 45 EXAMPLE OF THE RESULTS OF THE TRIANGLE PLUG IN, APPLIED TO THE NON-CONVEX ELEMENTS OF THE MESH

These new layer must be saved using the same process described on Figure 41 and Figure 42, on the same location as before, and then added to the GRASS project. To do this, we use the already familiar command of **`v.in.ogr`**. The commands to import these maps are stated next:

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_non_convex_tri
angle.shp output=mercier_non_convex_triangle
```

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_long_elements_
triangle.shp output=mercier_long_elements_triangle
```

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_big_areas_tria
ngle.shp output=mercier_big_areas_triangle
```

At this point, if we were to ask GRASS for a list of all the vector maps that are loaded in the mapset, it should show something similar to Figure 46, if the name suggestions have been followed.

```
------------------------------------------------
vector files available in mapset <initial_mapset>:
mercier_big_areas                       mercier_mesh
mercier_big_areas_triangle              mercier_mesh_01
mercier_ditch                           mercier_non_convex
mercier_long_elements                   mercier_non_convex_triangle
mercier_long_elements_triangle          mercier_out


------------------------------------------------
(END)
```

FIGURE 46 LIST OF MAPS AVAILABLE ON THE MAPSET

One final step to be performed to get the appropriate information on the triangle areas is to apply the p.A1.clean_topology routine, and to associate each of the triangles to its original HRU (to avoid creating HRU that are a mix of different original ones).

The application of this routine is started by the following command:

```
p.A1.clean_topology.py
```

This will start the routine, which will ask for several inputs. An example of the application is shown on Figure 47.

```
GRASS 6.4.4 (Mercier):~ > p.A1.clean_topology.py
/home/geopumma
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LO
AME': 'Mercier', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
------------------------------------------
vector files available in mapset <initial_mapset>:
mercier_big_areas                       mercier_mesh_01
mercier_big_areas_triangle              mercier_mesh_02
mercier_ditch                           mercier_non_convex
mercier_long_elements                   mercier_non_convex_triangle
mercier_long_elements_triangle          mercier_out
mercier_mesh


------------------------------------------

Please enter the name of the ogr : mercier_big_areas_triangle
Please enter the name for the ogr clean : mercier_big_areas_triangle_02
Please enter the snap threshold snapping (0.1 m recommended): 0.1
Please enter the map to get the initial columns: mercier_big_areas
```

FIGURE 47 APPLICATION OF P.A1.CLEAN_TOPOLOGY TO THE TRIANGLED SHAPED MAP LAYERS

Note that the proposed output names for these files are `mercier_non_convex_trianlge_02`, `mercier_big_areas_triangle_02`, `mercier_long_elements_triangle_02`.

5.4 DISSOLVING BY CONVEXITY INDEX CRITERION

Once the bad shaped HRU have been identified and subdivided into smaller triangles, it is possible to apply some routines to allow the creation of bigger units that comply with the desired geometric requirements.

Since the three different layers of bad shaped elements were defined in a way that allows no duplication of element, each can be now processed independently.

We will start processing the non-convex elements, using the routines provided for this purpose. Run the following command on the console.

```
p.B8.a.convexity_segmentation.py
```

Routine:	p.B8.a.convexity_segmentation.py
Required inputs:	• Mercier_non_convex_triangle layer (clean topology) • Mercier_non_convex layer (pre Triangle) • A convexity index threshold • A maximum area value • A form factor threshold
Output:	• Mercier_non_convex_processed

An application of this routine is shown in Figure 48.

```
GRASS 6.4.4 (Mercier):~ > p.B8.a.convexity_segmentation.py
/home/geopumma
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_N
AME': 'Mercier', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
------------------------------------------
vector files available in mapset <initial_mapset>:
mercier_big_areas                       mercier_long_elements_triangle_02
mercier_big_areas_triangle              mercier_mesh
mercier_big_areas_triangle_02           mercier_mesh_01
mercier_correct                         mercier_non_convex
mercier_long_elements                   mercier_non_convex_triangle
mercier_long_elements_split             mercier_non_convex_triangle_02
mercier_long_elements_triangle



------------------------------------------

Please enter the name of the polygon to dissolve : mercier_non_convex_triangle_0
2
Please enter the name of the output polygon : mercier_non_convex_processed
Please enter the name of polygon pre Triangle to get columns: mercier_non_convex
Please enter the Convexity Index Threshold (CIT 0.75-0.85) : 0.75
Please enter the Maximum Area (Amin rec = 20000 m2) : 20000
Please enter the Form Factor Threshold (FFT 0.20-0.40) : 0.2
```

FIGURE 48 APPLICATION EXAMPLE OF THE P.B8.A.CONVEXITY_SEGMENTATION

The result of this operation is shown in Figure 49.

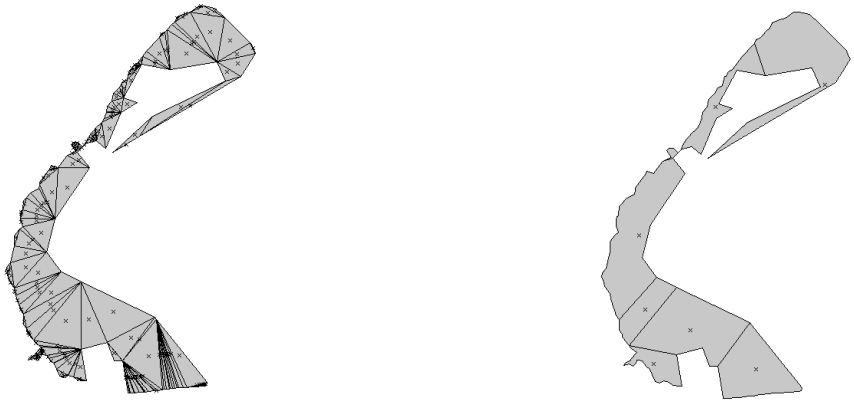


FIGURE 49 COMPARISON OF BEFORE AND AFTER PROCESSING FOR THE TRIANGLED MAP LAYER OF THE NON-CONVEX ELEMENTS

5.5 DISSOLVING BY FORM FACTOR CRITERION (P.FORM_FACTOR.PY)

The triangled-layer map of HRU polygons that do not fit the form factor criteria is processed in this step. This allows the creation of polygons that do fit the reference criteria.

Routine:	p.B8.b.formfactor_segmentation.py
Required inputs:	• Mercier_long_elements_triangle layer • A maximum area value • A form factor threshold
Output:	• Mercier_ long_elements _processed

The command line to use this routine is:

```
p.B8.b.formfactor_segmentation
```

Same as the routine before, it requires some input to be manually specified. An example of this operation is shown on Figure 50.

```
GRASS 6.4.4 (Mercier):~ > p.B8.b.formfactor_segmentation.py
/home/geopumma
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_NAME':
x0', 'GRASS_GUI': 'wxpython'}
-----------------------------------------------
vector files available in mapset <initial_mapset>:
mercier_big_areas                       mercier_mesh
mercier_big_areas_triangle              mercier_mesh_01
mercier_big_areas_triangle_02           mercier_mesh_02
mercier_ditch                           mercier_non_convex
mercier_long_elements                   mercier_non_convex_processed
mercier_long_elements_triangle          mercier_non_convex_triangle
mercier_long_elements_triangle_02       mercier_non_convex_triangle_02
                                        mercier_out


-----------------------------------------------

Please enter the name of the polygon to dissolve : mercier_long_elements_triangle_02
Please enter the name of the output polygon : mercier_long_elements_processed
Please enter the Form Factor Threshold (0.20-0.40) : 0.2
Please enter the Maximum Area (Amin rec = 20000 m2) : 20000
```

FIGURE 50 EXAMPLE OF USE OF THE P.B8.FORMFACTOR_SEGMENTATION.PY ROUTINE

An example of the result of this process can be seen in Figure 51.

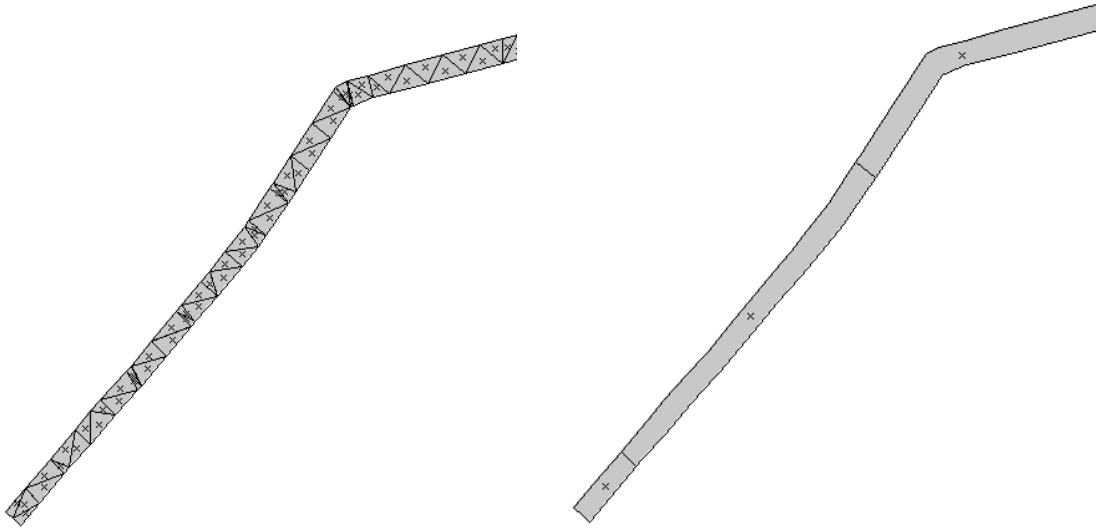


FIGURE 51 COMPARISON OF PRE- AND POST-PROCESSING OF THE LONG ELEMENTS MAP LAYER

5.6 DISSOLVING BIG POLYGONS

To process the polygon that are bigger than 20,000 m^2, the same routine that was used to process the non-convex HRU will be used. Depending on the user requirements, the specification of the values may vary. In this example, the same values as before will be used.

Routine:	p.B8.a.convexity_segmentation.py
Required inputs:	• Mercier_big_areas_triangle layer

	• Mercier_ big_areas layer (pre Triangle) • A convexity index threshold • A maximum area value • A form factor threshold
Output:	• Mercier_ big_areas _processed

The command line for activation is the following:

```
p.B8.a.convexity_segmentation.py
```

This routine, as was seen when dissolving by convexity index criterion, will require some inputs that have been provided by the user. An application of this routine is shown in Figure 52.

```
GRASS 6.4.4 (Mercier):~ > p.B8.a.convexity_segmentation.py
/home/geopumma
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCATION_NA
: 'x0', 'GRASS_GUI': 'wxpython'}
-----------------------------------------------
vector files available in mapset <initial_mapset>:
mercier_big_areas                       mercier_mesh_02
mercier_big_areas_triangle              mercier_non_convex
mercier_big_areas_triangle_02           mercier_non_convex_processed
mercier_ditch                           mercier_non_convex_triangle
mercier_long_elements                   mercier_non_convex_triangle_02
mercier_long_elements_triangle          mercier_out
mercier_long_elements_triangle_02       out_poly_area_1
mercier_mesh                            out_poly_area_1_disolved
mercier_mesh_01


-----------------------------------------------

Please enter the name of the polygon to dissolve : mercier_big_areas_triangle_02
Please enter the name of the output polygon : mercier_big_areas_processed
Please enter the name of polygon pre Triangle to get columns: mercier_big_areas
Please enter the Convexity Index Threshold (CIT 0.75-0.85 : 0.75
Please enter the Maximum Area (Amin rec = 20000 m2) : 20000
Please enter the Form Factor Threshold (FFT 0.20-0.40) : 0.2
```

FIGURE 52 APPLICATION EXAMPLE OF THE P.B8.A.CONVEXITY_SEGMENTATION.OY ROUTINE TO PROCESS THE BIG AREAS

An example of the result of this process can be seen in Figure 53.

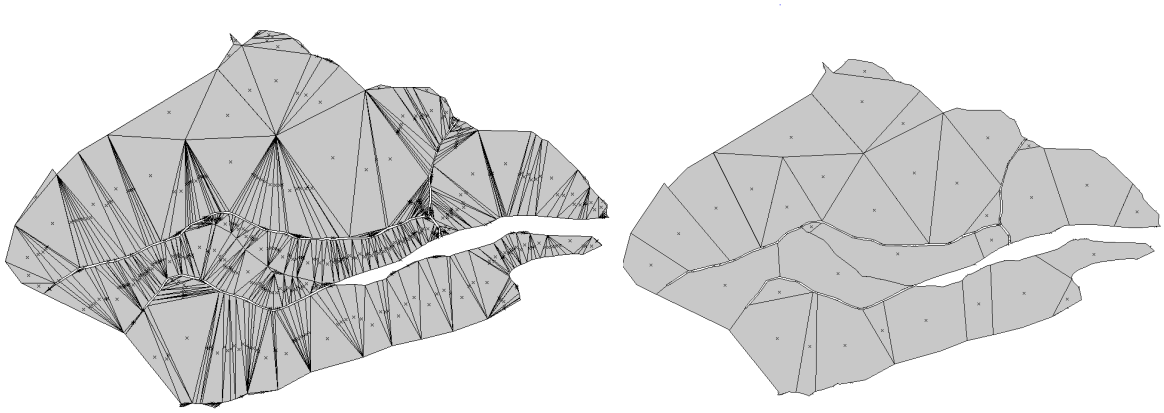


FIGURE 53 COMPARISON OF PRE- AND POST-PROCESSING OF THE BIG ELEMENTS MAP LAYER

5.7 UPDATE ORIGINAL MESH

Once all the layers have been processed, it is necessary to assemble the updated mesh, patching together the processed HRU layers and the original correct HRU. To do this, two steps are required: first, all the layers must be put together; second, it is necessary to clean the topology once again to avoid overlapping lines.

```
v.patch
input=mercier_correct,mercier_non_convex_processed,mercier_long_elemen
ts_processed,mercier_big_areas_processed output=mercier_mesh_02
p.A1.clean_topology.py
```

The use of the second command line will, as before, require of additional inputs that need to be provided manually. An image of how it should look is shown in Figure 54.

```
GRASS 6.4.4 (Mercier):~ > p.A1.clean_topology.py
/home/geopumma
{'MAPSET': 'initial_mapset', 'GISDBASE': '/home/geopumma/grassdata', 'LOCA
AME': 'Mercier', 'MONITOR': 'x0', 'GRASS_GUI': 'wxpython'}
------------------------------------------
vector files available in mapset <initial_mapset>:
mercier_big_areas                       mercier_long_elements_triangle_02
mercier_big_areas_processed             mercier_mesh
mercier_big_areas_triangle              mercier_mesh_01
mercier_big_areas_triangle_02           mercier_mesh_02
mercier_correct                         mercier_non_convex
mercier_ditch                           mercier_non_convex_processed
mercier_long_elements                   mercier_non_convex_triangle
mercier_long_elements_processed         mercier_non_convex_triangle_02
mercier_long_elements_split             mercier_out
mercier_long_elements_triangle


------------------------------------------

Please enter the name of the ogr : mercier_mesh_02
Please enter the name for the ogr clean : mercier_mesh_03
Please enter the snap threshold snapping (0.1 m recommended): 0.1
Please enter the map to get the initial columns: mercier_mesh_01
```

FIGURE 54 USE OF P.A1.CLEAN_TOPOLOGY.PY TO CLEAN THE TOPOLOGY OF THE PATCHED MAP.

The final distribution of the HRU in the Mercier catchment should look like Figure 55.

FIGURE 55 POST-PROCESSED DISTRIBUTION OF THE HRU IN THE MERCIER CATCHMENT

5.8 UPDATE HRU ALTITUDE INFO

Since the original HRU have changed as a result of the processes done, it is required to update the elevation of each of them. In order to do this, it is necessary to switch from GRASS 6 to GRASS 7. This is because the latest version allows a better handling of raster information.

However, before switching on to a newer version, it is necessary to export the update mesh layer as a shapefile that will be imported later into GRASS7. Since the p.A1.clean_topology.py routine has

already been executed, there is a shapefile already exported on the home folder named "mercier_mesh_03".

Having done this, exit GRASS 6 by simply executing the following command on the console:

```
Exit
```

Then click on the GRASS 7 shortcut that is on the desktop. The startup window for this version is quite similar to that of GRASS 6. The same locations from GRASS6 will be available. Create a new mapset on the Mercier location and name it GRASS7. At this point, the screen should look like Figure 56.

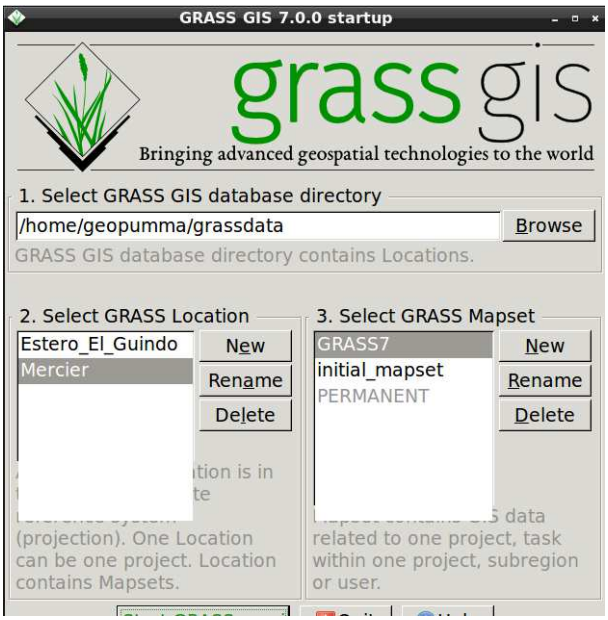


FIGURE 56 GRASS7 START SCREEN AFTER CREATING THE NEW MAPSET

Select the GRASS7 mapset recently created and start the program. The look of the program is almost identical to that of GRASS 6, so the working environment should be quite familiar. The console is where the processing procedures are going to be developed.

Before any further action is taken, it is necessary to load all the Geo-PUMMA routine by running the following command line:

```
export PATH=/home/geopumma/geomhyda:/home/geopumma/geopumma:$PATH
```

After this, it is necessary to import the processed mesh and the raster to the new mapset. Execute the following commands on the console to do this:

```
r.in.gdal
input=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_lidar2m_exte
nd.tiff output=mercier_dem
```

```
v.in.ogr input=/home/geopumma/mercier_mesh_03/mercier_mesh_03.shp
layer=mercier_mesh_03 output=mercier_mesh_03 geometry=None
```

The region must be set accordingly to the raster that has been just imported. To do this, simply execute the following command:

```
g.region raster=mercier_dem
```

Now that the processed mesh and the DEM have been imported, it is possible to update the elevations of each HRU using the following command:

```
p.B3.a.average_altitude.py
```

Routine:	p.B3.a.average_altitude.py
Required inputs:	• Mercier digital elevation model (DEM) • A prefix name for the columns to be generated
Output:	• Updated Mercier mesh with corrected altitudes for each of the HRU

Once the command is executed, it is necessary to specify some inputs that the routine requires for its development. These inputs and its correct filling are shown on Figure 57.

```
GRASS 7.0.0 (Mercier):~ > p.B3.a.average_altitude.py
{'MAPSET': 'GRASS7', 'GISDBASE': '/home/geopumma/gra
ercier', 'GUI': 'wxpython'}
mercier_mesh_03

Please enter the name of the ogr : mercier_mesh_03
mercier_dem

Please enter the name of the raster : mercier_dem
Please enter the name of the colprex : alt
```

FIGURE 57 USE OF P.B3.A.AVERAGE_ALTITUDE.PY TO CALCULATE THE ALTITUDE OF EACH HRU

After finishing this step, the process of segmentation and correction of the bad shaped HRU has been successfully completed.

However, to use and develop the hydrological connectivity routines, it is necessary to run them in GRASS6. It is because of this that the GRASS layer is exported using the following command:

```
v.out.ogr input=mercier_mesh_03@GRASS7
output=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_mesh_04.shp
format=ESRI_Shapefile
```

6 STEP B.3: HYDROLOGICAL CONNECTIVITY

The purpose of this step is to establish the hydrological connectivity between the different elements that compose the mesh (HRU, UHE or a mix of them as is most probable).

To achieve this, a series of steps must be completed, which will be explained in the following points. Most of these steps are performed on the GRASS console.

In this tutorial, an example of application will be shown on the Mercier region. If this tutorial has been followed from the previous step, then the user will already have produced the mesh that is provided as an input for this section on the step before.

However, all the necessary files to go through this section are provided so the user can do this section independently from the former two. Despite that, users are encouraged to try to use their own files when working on this section, as a mean to get familiar with the working environment of GRASS.

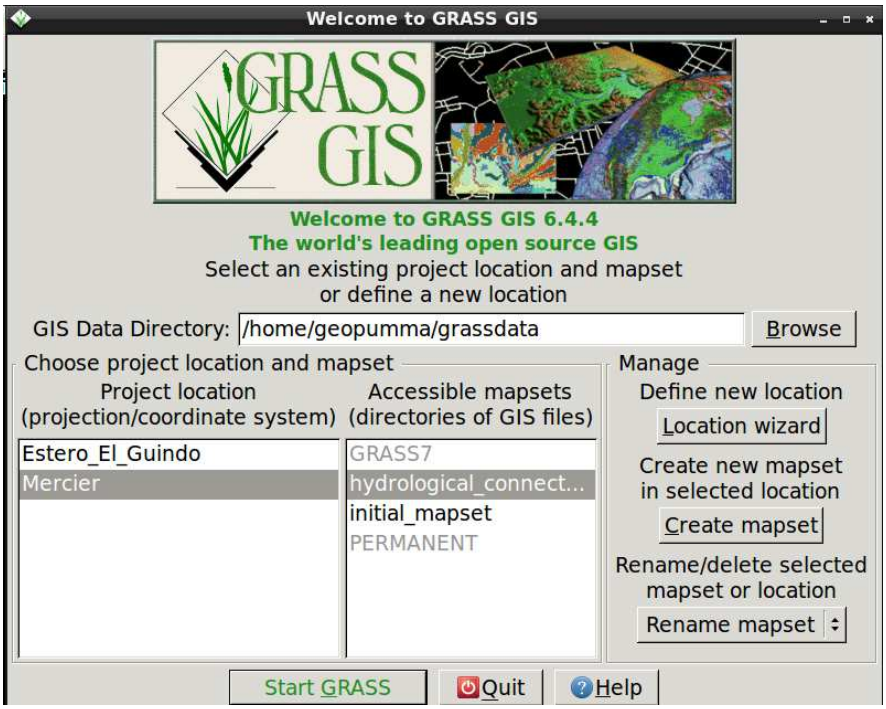


FIGURE 58 EXAMPLE OF A NEW MAPSET CREATED UNDER THE NAME HYDROLOGICAL_CONNECTIVITY

To import the necessary files for this section, start GRASS6 with the hydrological_connectivity mapset that was created on the Data preparation Section and execute the following commands:

```
export PATH=/home/geopumma/geomhyda:/home/geopumma/geopumma:$PATH
```

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_mesh_10/mercier_mesh_10.shp output=mercier_mesh
```

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_initial_ditch/mercier_initial_ditch.shp output=mercier_ditch
```

```
v.in.ogr
dsn=/home/geopumma/Database_Tutorial_GeoPUMMA_v1/mercier_out/mercier_out.shp output=mercier_out
```

The first command line is to import all the GeoPUMMA routines, while the other three provide the initial required shapefiles of the processed mesh, the river ditch and the outlet point.

6.1 EXTRACTING ALL INTERFACES

This routine requires as an input the post-processed mesh of the catchment. It processes the different interactions between all the interacting units, either HRU, UHE or river.

Routine:	p.B10.all_interfaces.py
Required inputs:	Post-processed mesh of the catchment
Output:	General interface's layer

The command line required to start this routine is:

```
p.B10.all_interfaces.py
```

This routine will require the manual specification of the layers to be processed. In this case, the only required input is the post-processed mesh of the catchment and an output name for the file, as can be seen in Figure 59.

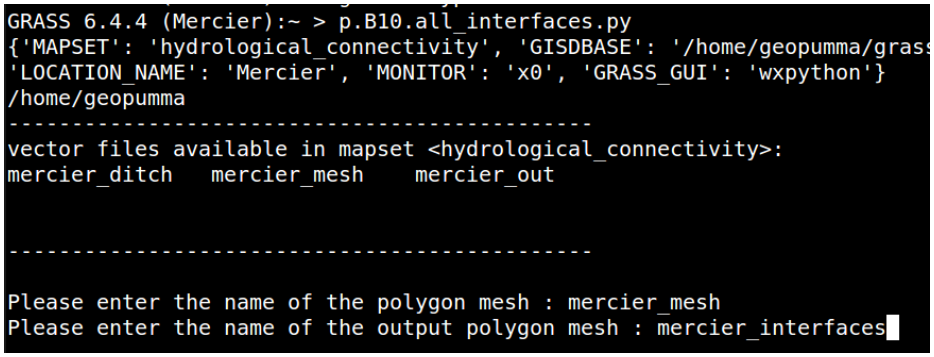


FIGURE 59 APPLICATION EXAMPLE OF THE P.B10.ALL_INTERFACES.PY ROUTINE

An example of the results is shown on Figure 60.

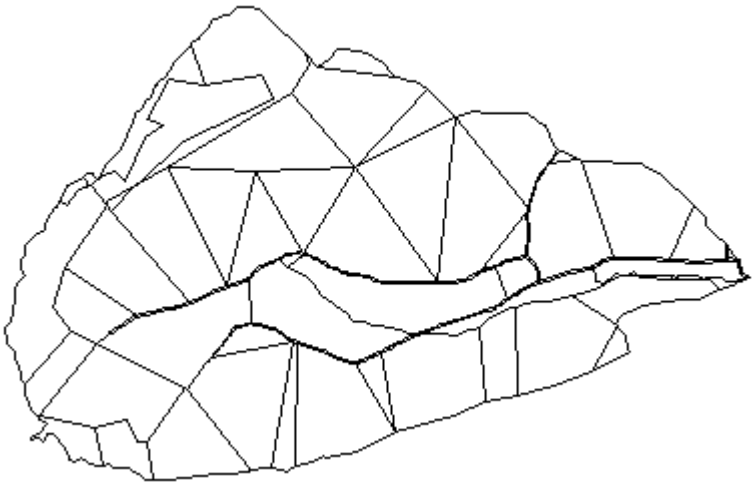


FIGURE 60 ALL INTERFACES MAP LAYER EXAMPLE

6.2 RIVER SEGMENTATION

This routine requires as an input the river vector map and segments it accordingly to the different river interfaces that interact with it. It requires the definition of a minimum length for each segment of the river to avoid the creation of non-representative interactions.

The output of this routine is a segmented polyline of the river, divided into different segments, based on the interactions with the different interfaces for each element of the mesh.

Routine:	p.B11.river_seg.py
Required inputs:	River polyline layer (Mercier_ditch) General interfaces' layer (Mercier_all_interfaces) Value for minimum length for each segment (suggested value of 5m)
Output:	Segmented river layer

The command line required to start this routine is:

```
p.B11.river_segm.py
```

This routine will require the manual specification of the layers to be processed. In this case, the river polyline map layer, the general interfaces' layer generated on the previous step and a value for the minimum length of each segment is required (a recommended value of 1m is suggested).

```
GRASS 6.4.4 (Mercier):~ > p.B11.river_segm.py
/home/geopumma
---------------------------------------------
vector files available in mapset <hydrological_connectivity>:
mercier_ditch       mercier_interfaces  mercier_mesh        mercier_out


---------------------------------------------

Please enter the name of the map with all interfaces : mercier_interfaces
Please enter the name of the river : mercier_ditch
Please enter the minimum length to segment the river (1 m) : 1
Please enter the name of the output segmented river : mercier_segmented
```

FIGURE 61 APPLICATION EXAMPLE OF ROUTINE P.B11.RIVER_SEG.PY ROUTINE

An image showing the expect layer acquired from this step is shown in Figure 62.

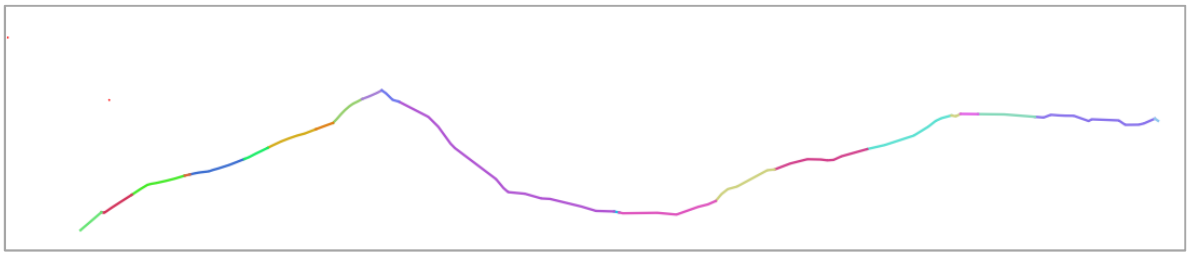


FIGURE 62 EXPECTED RESULT OF THE RIVER SEGMENTATION PROCESS.

6.3 WATER TABLE RIVER INTERFACE

This routine takes the interfaces map that was previously processed, and together with the post-processed mesh and the river polyline, calculates the interfaces between the river and the units that interact with it. It is necessary to specify the attribute column for the mesh that represents the ID of each element.

The output of this routine is a set of interfaces, that consider only the units in the mesh that interact with the river.

Routine:	p.B12.wtri.py
Required inputs:	• Post-processed mesh of the catchment • River polyline layer • General interfaces' layer
Output:	• Water table river interfaces layer

The command line required to start this routine is:

```
p.B12.wtri.py
```

This routine will require the manual specification of the layers to be processed. In this case, several input are required. The map of all the interfaces is the first one to be specified. Then, the layer representing the river must be specified. After this layer has been submitted, it will show all the columns of the attribute table of this layer and it will ask to specify the column that contains the ID for each segment of the river. Also, a minimum length of the water table river interfaces must be specified (a recommended value of 5m is suggested). Finally, an output name for the new layer must be provided (see Figure 63 for a graphic example).

```
GRASS 6.4.4 (Mercier):~ > p.B12.wtri.py
/home/geopumma
-------------------------------------------
vector files available in mapset <hydrological_connectivity>:
mercier_ditch       mercier_mesh        mercier_segmented
mercier_interfaces  mercier_out


-------------------------------------------

Please enter the name of the map with all interfaces : mercier_interfaces
Please enter the name of the river : mercier_ditch
Displaying column types/names for database connection of layer 1:
INTEGER|cat
DOUBLE PRECISION|id
DOUBLE PRECISION|river_id
DOUBLE PRECISION|manning_id
DOUBLE PRECISION|section_id
DOUBLE PRECISION|slope
DOUBLE PRECISION|elevation
DOUBLE PRECISION|water_leve
DOUBLE PRECISION|subbasin2

Please enter the name of the id reach : id
Please enter the minimum length to extract the wtri segments (5 m): 5
Please enter the name of the output wtri : mercier_wtri
```

FIGURE 63 APPLICATION EXAMPLE OF THE P.B12.WTRI.PY ROUTINE

An example of the output layer is shown in Figure 64. Note that there are some segments missing because of the minimum length restriction.

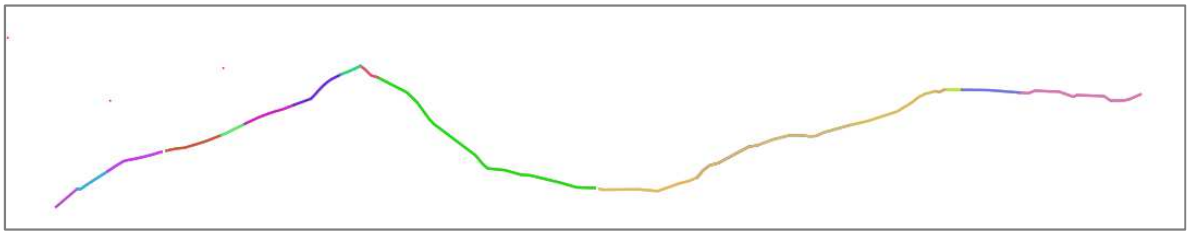


FIGURE 64 EXAMPLE OF THE WATER TABLE RIVER INTERFACE OBTAINED FROM THE P.B12 ROUTINE.

6.4 WATER TABLE INTERFACE

This routine takes the interfaces map that was previously processed and the water table for the river interface, and calculates the interfaces between the different units that are not directly connected to the river. It is necessary to specify the attribute column for the interfaces layer that represents the ID of each interaction.

The output of this routine is a set of interfaces, that consider only the interactions between different units of the mesh, excluding the interfaces between any two units that both interact with the river, or between any given unit and the river.

Routine:	p.B13.all_wti.py
Required inputs:	General interfaces' layer Water table river interfaces layer (specifying the attribute column that contains an ID for the interfaces already defined).
Output:	Water table interfaces layer

The command line required to start this routine is:

```
p.B13.wti.py
```

This routine will require the manual specification of the layers to be processed. In this case, several input are required. The map of all the interfaces is the first one to be specified. Then, the layer with the water table river interfaces from the previous step. For this layer, it is necessary to specify the id of each interface (by default id_interf). Finally, a file name must be specified for the output layer.

```
GRASS 6.4.4 (Mercier):~ > p.B13.wti.py
-----------------------------------------------
vector files available in mapset <hydrological_connectivity>:
mercier_ditch       mercier_mesh        mercier_segmented
mercier_interfaces  mercier_out         mercier_wtri


-----------------------------------------------

Please enter the name of the map with all interfaces : mercier_interfaces
Please enter the name of the wtri : mercier_wtri
Displaying column types/names for database connection of layer 1:
INTEGER|cat
INTEGER|id_interf
DOUBLE PRECISION|length
DOUBLE PRECISION|id_riv
CHARACTER|mod_mesh
INTEGER|id_mesh
CHARACTER|id_subb

Please enter the name of the column ID of each interfaces (id_interf) : id_interf
Please enter the name of the output wti : mercier_wti
```

FIGURE 65 APPLICATION EXAMPLE OF THE P.B13.WTI.PY ROUTINE

An example of the output layer is shown in Figure 66.

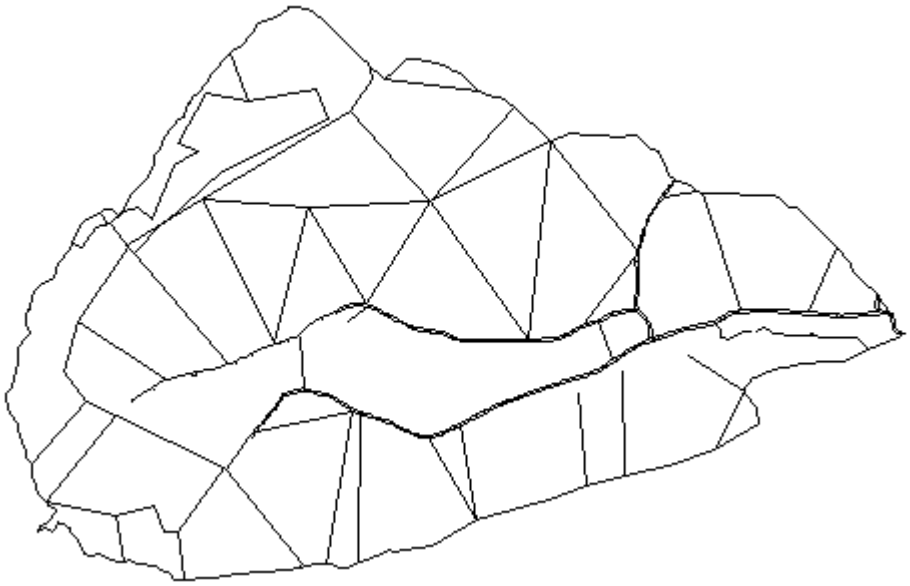


FIGURE 66 EXAMPLE OF THE WATER TABLE INTERFACE FROM THE P.B13 ROUTINE

6.5 OLAF (OVERLAND FLOW)

This routine establishes the flow direction network of the mesh. It requires as an input the post-processed mesh of the catchment, specifying which of its attributes contains information about the average altitude, the segmented river polyline (from step 6.2), the water table river interfaces layer (from step 6.3) and the water table interfaces layer (from step 6.4).

As a result, a layer specifying to where each of the units flows, either another unit or the river, is created.

Routine:	p.B14.olaf.py
Required inputs:	post-processed mesh of the catchment (specifying the attribute column for the ID of each unit) Segmented river layer Water table river interfaces layer Water table interfaces layer.
Output:	Flow direction network

The command line required to start this routine is:

```
p.B14.olaf.py
```

This routine will require the manual specification of the layers to be processed. In this case, several input are also required. The map of the catchment mesh is the first. For this layer, it is required to

specify the column with the average altitude for the HRU. After that, in the following order, it is necessary to specify the water table river interface layer, the water table interface layer, and the segmented river layer. An example of this process can be seen in Figure 67.

```
Please enter the name of the map with polygon mesh : mercier_mesh
Displaying column types/names for database connection of layer 1:
INTEGER|cat
DOUBLE PRECISION|new_cod_1
DOUBLE PRECISION|id_subb_2
DOUBLE PRECISION|soil_id_3
DOUBLE PRECISION|geol_id_4
CHARACTER|module
INTEGER|area
INTEGER|id_mesh
DOUBLE PRECISION|PERIMETER
DOUBLE PRECISION|SOLIDITY
DOUBLE PRECISION|CONVEXITY
DOUBLE PRECISION|COMPACT
DOUBLE PRECISION|FORMFACTOR
DOUBLE PRECISION|AltMer_ave
DOUBLE PRECISION|AltMer_std
DOUBLE PRECISION|AltMer_min
DOUBLE PRECISION|AltMer_max

Please enter the name of the column with altitude value : AltMer_ave
Please enter the name of the wtri : mercier_wtri
Please enter the name of the wti : mercier_wti
Please enter the name of the segmented river : mercier_segmented
Please enter the name of the olaf output vector : mercier_olaf
```

FIGURE 67 APPLICATION EXAMPLE OF THE P.B14.OLAF.PY ROUTINE

Finally, a name must be specified for the output layer. The output of this process is show in Figure 68. Note that only the black lines are the elements generated by this routine and that the background mesh is shown only for reference purposes.

FIGURE 68 OUTPUT OF THE P.B14 ROUTINE, SHOWING THE OVERLAND FLOWS FROM ONE HRU TO OTHER. OVERLAND FLOWS ARE SHOWN IN PURPLE, BLUE LINE REPRESENTS THE RIVER

6.6 GEO-DESCRIPTORS

The p.B15.geo_descriptors.py routine can be used to perform a detailed analysis of the spatial distribution of the connected area. This routine stores the area -or any other property whose value is spatially distributed- and distance to the outlet point. This information can then be used to compute the width and area function, two geomorphological functions utilized later to characterize the drainage networks generated by Geo-PUMMA.

Routine:	p.B15.geo_descriptors
Required inputs:	Post-processed mesh of the catchment
Output:	Interfaces between the different components of the mesh.

The command line required to start this routine is:

```
p.B15.geo_descriptors.py
```

This routine will require the manual specification of the layers to be processed. In this case, several input are also required. The map of the catchment mesh is the first. Next is the overland flow for this

mesh (the OLAF calculated in the previous step). After that, the river layer (mercier_ditch) and the outlet point of the river (mercier_out). After this info has been added, some calculation parameters are required: a threshold value to check that olaf snaps to the river (an initial value of 1 meter is suggested). After that, the routine asks if it is desired to route travel time (this information can be used later to determine the instantaneous unit hydrograph (IUH); some suggested velocity values for the flows are presented as a reference in case there is no information available).

Finally, there is the option to specify if the river path to be used (the olaf layer) has been manually modified and adjusted to avoid unconnected nodes of the network. Since during the development of the tutorial this has not been done, a value of 0 is used as an input.[4]

```
Please enter the name of the model mesh : mercier_mesh_03
Please enter the name of the map with olaf path way : mercier_olaf
Please enter the name of the river : mercier_ditch
Please enter the name of the out : mercier_out
Please enter the iterative threshold to check the olaf snapping to river (first try with 1.0 m) : 1
Please select option to route travel time (Yes=1;No=0) : 1
Please enter the velocity in channel (vel aprox = 1.85 m/s) : 1.85
Please enter the velocity in natural hillslopes areas (vel aprox = 0.05 m/s) : 0.05
Please enter the velocity in urban areas (vel aprox = 0.27 m/s) : 0.27
Please select option to use river_path manually modified (Yes=1;No=0) : 0
```

FIGURE 69 APPLICATION EXAMPLE OF THE P.15.GEO_DESCRIPTORS.PY ROUTINE

This will produce the following message (Figure 70)

```
WARNING: 90 features without category skipped
v.select complete. 1 features written to output.
################################################################################
# WARNING: The nodes of the shape file nodes_outside must be manually snaped #
# Please edit the olaf nodes and snap manually to river                        #
# Be careful do not create  boucles                                            #
# Be careful do not create  boucles                                            #
################################################################################
WARNING: OGR layer <nodes_outside_100_cm> already exists and will be
         overwritten
Exporting 1 geometries...
 100%
v.out.ogr complete. 1 features written to <nodes_outside_100_cm>
(ESRI Shapefile).
```

FIGURE 70 ERROR FROM THE APPLICATION OF P.15.GEO_DESCRIPTORS.PY

What this error means is that there are points from the drainage network that are within the specified snapping distance (in this case, the value of 1 meter), that are not connected to the river. This point can be identified in the newly-created layer named **nodes_outside_100_cm.** When opening this layer, it is possible to graphically identified the problematic points (see Figure 71). On the left side of this figure, an image of the whole mesh is presented. On the right side, there is a zoom in of the problematic point. As it can be seen, the error is not an actual error but a connection of the drainage

[4] The user is encouraged to explore and use its own methods to rectify their own drainage networks when applying Geo-PUMMA to other basins where alternatives can result in better options.

network from an HRU directly to the river, that because of its size and location was misinterpreted as an “non-snapped” point to the river.

FIGURE 71 GRAPHICAL IDENTIFICATION OF PROBLEMATIC POINTS. THE DRAINAGE NETWORK (OLAF LAYER) IS PRESENTED IN GREEN, THE RIVER LAYER IS PRESENTED IN BLUE, WHILE THE ORANGE DOT REPRESENTS THE PROBLEMATIC POINT

Since this is the only point that turned up as an error, the p.B15 routine is executed once more, specifying a smaller value for the snapping. In this next iteration, a value of 0.1 m is applied.

```
Please enter the name of the model mesh : mercier_mesh_03
Please enter the name of the map with olaf path way : mercier_olaf
Please enter the name of the river : mercier_ditch
Please enter the name of the out : mercier_out
Please enter the iterative threshold to check the olaf snapping to river (first try with 1.0 m) : 0.1
Please select option to route travel time (Yes=1;No=0) : 1
Please enter the velocity in channel (vel aprox = 1.85 m/s) : 1.85
Please enter the velocity in natural hillslopes areas (vel aprox = 0.05 m/s) : 0.05
Please enter the velocity in urban areas (vel aprox = 0.27 m/s) : 0.27
Please select option to use river_path manually modified (Yes=1;No=0) : 0
```

FIGURE 72 APPLICATION OF THE P.B15.GEO_DESCRIPTORS.PY WITH A SMALLER SNAPPING THRESHOLD

This time a new layer called `geo_descriptors_vect` is created. This layer is presented in Figure 73, where on the left the geometry is shown. The attribute table of this layer provide all the information to generate the width and area function. The length of starting node and ending node of each segment to the outlet are represented by the columns “Up_str” and “Do_str” (in meters), the cumulative area is represented by colum “a_1” (in meter squared) and the travel time of each starting node and ending node are represented in the columns “t_Up_str” and “t_Do_str” (in seconds).

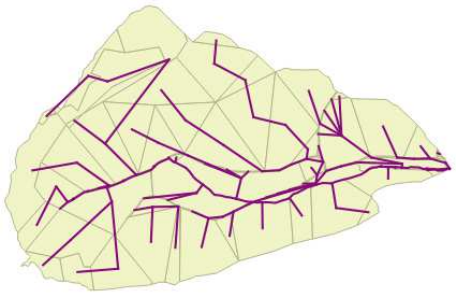


FIGURE 73 P.B15.GEO_DESCRIPTORS.PY RESULTS