The Ottawa GRASS Users Group Presents

Tutorial # 2

Using GRASS and GPS to Create a Coverage

Introducing v.in.GPSBabel

Prepared by:
Dave Sampson
HB Outdoor Recreation, Parks Tourism
BA Geography

For:
Ottawa GRASS Users Group
Canadian GRASS Community
International GRASS Community

Table of Contents

About the Document

Document History

Version	*Date*	*Revisions*	*Authors*	*Sponsor*
1	01/15/06	2	Dave Sampson	Ottawa GRASS Users Group http://cemml.carleton.ca:8080/ogug
1.2	02/01/06	1	Dave Sampson	Ottawa GRASS Users Group http://cemml.carleton.ca:8080/ogug
1.4	02/04/06	1	Dave Sampson	Ottawa GRASS Users Group http://cemml.carleton.ca:8080/ogug

Contacts

- Dave Sampson (dr.sampson at sympatico dot ca)

Online Linkage:

http://cemml.carleton.ca:8080/ogug

GRASS version

6.x

Intro

Welcome:

Welcome to OGUG's second GRASS Tutorial. This tutorial is designed for GRASS 6.x. Any reference to the commands or programs should be the same although menu mapping may vary between versions.

The Document is released under the GPL. Please feel free to distribute, copy and append to the document. But remember to offer credit where credit is due. In turn we will try the same within this document. If you make updates and let us know this document can be improved.

The most current version of this tutorial will be housed at (http://cemml.carleton.ca:8080/OGUG) the home of the Ottawa GRASS Users Group.

The purpose of this tutorial is to cover two areas of interest. First how to get data from your GPS to a GRASS session. Second once you have some vector data, what needs to be done to it to make it work nicely within a GRASS GIS. We will cover basic concepts and will not exhaust the concepts of vector management.

Dedication:

This tutorial is dedicated to Claudio Porta & Lucio Davide Spano, students of Computer Science, University of Pisa (Italy). Commission from Faunalia Pontedera (PI) [http://www.faunalia.it/] for contributing the v.in.gpsbabel script and Hamish for putting up with my nagging and delivering the script from the developers list to those that needed it. We also dedicate this tutorial to all those that will make use of GPS technology in GRASS.

Preface:

In the open source and GIS world there are always multiple wasy to attack a problem. This tutorial uses some independent tools. Those who are new to open source GIS may find this annoying. “Why isn't everything contained in one package?” might be a nagging question. Well, I have no simple answer for that. But many would argue that it's nice that the same tool does not get re-invented.

In the mean time try to appreciate the effort of multiple people to contribute these tools you can use separately or all together. Take it as a chance to explore new tools. Just relax and have fun.

Syntax of the Tutorial:

At the start we lay things out pretty explicitly. By the end we move towards bare bones procedures. To help with this we have included two types of syntax. Either a menu command structure using the GUI or command line structure.

A menu map looks like this:

Parent > Child > Sub Child

A GRASS command line structure looks like:

# command [option] [...]

Exploring GPS with GRASS

Connecting the GPS.

Your GPS should remain off for the first stage (just to be sure). Start by pluging in the GPS cable to the computer (serial or USB) and then to the GPS.

Power up your GPS. Because we are indoors (probably) then you will not have GPS reception. Use the unit with GPS OFF, if error window appears.

All GPS Units

Ensure the GPS uses the following Units so we are all using the same data types. (outside the tutorial learn what the appropriate settings for your project are).
Position: hddd.ddddd (decimal degrees)
Datum: WGS84
Distance: Metric
Depth, Elevation: meters (won't matter too much)
Pressure: Millibars (for the climatologists, but again don't worry)

Proceed to the next step to setup your connection protocol so the computer and GPS can talk to each other.

Garmin (Etrex) settings

Navigate to your interface menu. For the Garmin Etrex Vista this is located on the main menu (press the page key 6 times). Scroll to SETUP. Select INTERFACE. Ensure GARMIN interface is selected.

Other GPS (Lowerence / Magellan).

Find where the equivalent settings are. [if you have instructions for some other comon GPS let us know]

Finding the serial port

From the command line try finding the serial port your GPS is connected to. (windows com port 1 is *nix /dev/ttyS0)

USB to serial coverters may use ttyUSB0. In any case check out the see if anything is happening on the port.

```
# cat /dev/(device) Or
# cat /dev/tts/USB(0 or 1) Or
# cat /dev/tts/(0 0r 1)
```

If there is a bunch of junk flowing across your screen then you found it. Depending on the protocol it might now have junk, but a blank line is also good. If it screams at you complaining of no file then move to the next one (eg ttys1).

If you can't initiate the port, try accessing it as root. If you don't know how ask the instructor (or RTM).

Start a GRASS Session

At the command line start GRASS (or icon on desktop if one exists)
# grass61

Create OTTAWA Location/Mapset

1. Click on CREATE REGION USING EPSG CODE.
2. Enter the name of the new region "OTTAWA_WGS84"(or select the one provided)
3. Ensure the path to share-proj is correct.
   Check by clicking EPSG-Codes or BROWSE. A long table should appear. If it doesn't change the directory.

Either
/usr/local/share/proj/epsg or /usr/share/proj/epsg

4. We will be using WGS84 / TM 6 NE. Or EPSG# 2311 Click OK.
5. You will restart GRASS and find the new region if you had to create it.

6. Select the new location. Create a new mapset (in right hand dialog box) using your last name.
7. Press CREATE. Select your location. Click ENTER GRASS.

Window Management

Now that we have a region to work with we can import some data. Get your environment set how you like it.
1. Start by opening a monitor, move your windows how you like them. Then we'll continue.

# d.mon start=x0

v.in.GPSBabel

From GRASS command line
# v.in.gpsbabel

Name:	trk	
Port:	/dev/[device]	
reprojection (cs2cs):	[blank]	
file type:	[gps protocol]	(eg garmin OR nmea
Options:	verbose	
	Upload tracks	

rename and repeat for routes (rts) and waypoints (wp).

If all goes well you should have files

wp	-	waypoints
rts	-	routes
trk	-	tracks

Forcing the data into points you get all the points of the track or route instead of lines. Might be usefull for some applications. Not here.

GRASS Database Error....

Depending on how your GRASS environment is setup you might get a database error. Makesure that GRASS is pointing to the right database. Read the manuals to learn how to do this. The Carleton University tutorial lab should have this set properly. We hope.

Viewing the Imported Track Log

Tracks are the most important issue for this tutorial. A track is a series of points plotted either in time intervals (eg. every 30 sec) or distance (eg. every 5 metres) depending how your GPS is setup.

To view the tracks downloaded into grass start a graphic terminal, open them as a normal vector. You may need to zoom in

This can all be done from the display pannel. If you're a command line junky then..

```
# d.mon select=x0
# d.vect
# d.zoom
# d.region
```

1. In the display panel select the vector (if you loaded from the command line load it through the panel for the next part).
2. Once the vector is loaded we will limit the DISPLAY to SHAPES and POINTS. This is the minimum to view your data.
3. As general interest de-select SHAPES and select CATEGORIES.
4. Check out TOPOLOGY and LINE DIRECTIONS too. We won't discuss these all now but they will make sense some day. What do you think each one represents?

Setting the Region

1. lets make sure the region reflects this new vector.

    GIS > Region > manage Region

    ```
    #g.region
    ```

2. Now display the vector. Display from Saved Region Settings (icon)

    ```
    # d.mon
    ```

3. Now it is time to check out the vector. Zoom into the area.

4. Now lets see what type of tabular data is associated with the DBF.

    Vector > Querry with mouse

    ```
    # d.what.vect
    ```

Move the cross hair over the feature or line in question. Left click to see the results. Yippee this should show what data we have. Now we are just looking at the raw data and there might be some cleaning up to do. Lets start a digitizing session.

Digitizing and Vector Editing in GRASS

The vector capabilities in GRASS are relatively new compared to all the other tools. The tools and functions are improving with each release. A word of warning for those used to other GIS packages. As of the time of this reading there are no UNDO options within a digitizing session. Be forwarned you should make a backup copy of any vector layer before digitizing or editing. Need help, read the manuals.

V.Digit: Manual Edits to Vectors

Vector > Develop Map > Digitize
# v.digit

Use some of the tools to clean up your dataset. [provide some description of tools]

Digitize a new point:
Digitize a new line:
Digitize a new polygon:

Move vertex:
Add vertex:
Remove Vertex:

Split Line:
Move feature:
delete feature:
Zoom In:
Zoom Out:

Pan Screen:
Zoom to default region
Zoom to region:
Redraw:

Display Categories:
Display Attributes:
Open Settings:

Save and Exit:

Lets not spend too much time hacking away at the dataset manually. Lets see what processes are in place that can do some basic things.

v.clean: Automated vector Cleaning

Some basic cleaning can be done with v.clean. Sorry no menu item in 6.1.
This tool takes an input vector and creates two new vectors. The first vector is where all error free features land. The second file is where all errors land. The errors are then dealt with individually. An error would be a vector that does not follow basic topology (discussion). Because we are creating new shapes here we can play around with different functions and combinations of functions. And of course we are working on a copy, not the original. Right?

Lets try a simple break and then look at it again through the digitizing tool. The v.clean tool offers the following functions.

```
# man v.clean                              (Documentation)
```

From the manual, these are the options:

break: break lines at each intersection
rmdupl: remove duplicate lines (pay attention to categories!)
rmdangle: remove dangles, threshold ignored if < 0
chdangle: change the type of boundary dangle to line, threshold ignored if < 0, input line type is ignored
rmbridge: remove bridges connecting area and island or 2 islands
chbridge: change the type of bridges connecting area and island or 2 islands from boundary to line
snap: snap lines to vertex in threshold
rmdac: remove duplicate area centroids ('type' option ignored)
bpol: break (topologically clean) polygons (imported from non topological format (like shapefile). Boundaries are broken on each point shared between 2 and more polygons where angles of segments are different
prune: remove vertices in threshold from lines and boundaries, boundary is pruned only if topology is not damaged (new intersection, changed attachement of centroid), first and last segment of the boundary is never changed
rmarea: remove small areas, the longest boundary with adjacent area is removed
rmsa: remove small angles between lines at nodes

Having cleaned up the vector lets look at what happened.

```
#v.digit
```

Categories and Attributes

Use the category or attributes function. You'll notice that any place where two lines crossed the line was split. But this still might not divide your perimeter from your trails. So lets now separate your trails from the perimeters using the digitizing tools. This is where it is handy to know what the track log was supposed to look like.

This is also an example of how saving track logs separately might help. GPSBabel does a good job of keeping separate tracks separate (unlike the older v.in.garmin)

Editing a Feature to Assign New Categories

As you might be aware there are many ways in GRASS to solve an problem. This approach may not be the best or simplest for extracting features but it demonstrates many principles at the same time.

First we must agree that vector coverages have many features. These features can be points, lines or polygons. For good form, each coverage should only contain one type of feature. Furthermore we should classify features into theme specific categories. Themes may include park boundry, path, park infrastructure or points of erosion.

At present your vector shape may contain point and line data. We will extract two types of features. First we will isolate the lines (eg the path) and then we will isolate the polygons (eg park boundary).

We will isolate them using the v.digit tool and assign each group a specific category. Open a digitizing session if one is not already open. (see above for commands and menu).

Select the DISPLAY CATEGORIES icon for the V.Digit command toolbar. Select a line. A categories window should appear. In the window should be a single entry. Mine includes:

Layer: 1
Cat: 5

We will replace this with our own category depending on the feature.

Cat	Description
1	Path
2	Boundary

Close out of the Categories window without editing it. Take a close look at the tracks (lines) on your screen and try to determine what is a path (cat 1) and boundary (cat 2).

Lets start with choosing a boundary. Select the CATEGORIES icon and select a feature that is a boundary. The feature is selected if it is colored yellow. And the CATEGORIES window appears.

In the Categories window enter the following information

layer: 1
Cat: 2

Click add new. Delete the original category (for me it was layer 1, cat 5). Repeat for all the boundaries. And then use the same procedure for all the paths. Voila the categories are edited and we can separate the features.

V.Build: Rebuild Vector Topology After Digitizing Sessions

Save and exit from the v.digit tool bar. Close the v.digit dialog

Now rebuild your topography. This is a recommended step after each digitizing session. It ensure that vectors and vertexes and points that were modified or removed are in fact reflected.

Vector > Develop Map > create / rebuild topography

# v.build

V.Extract: Extracting features based on category attributes

Right now we have a single vector coverage with soon to be line and polygon features. This is topologically incorrect. Not that we'll get errors in GRASS, but we will run into geo processing issues down the road. Its good habit to have each coverage contain one data type (line, polygon, point).

Now we will create two new vector containing only one feature type. We will extract the feature data base don the new categories we just finished assigning.

Vector > Query By Attributes

# v.extract

Create you path vector coverage.

Input: the cleaned vector we were digitizing	
name: the new vector name (eg Path)	
Type: Line	
layer: 1	
New category value: -1	(we want to keep this category)
Category range: 1	(we only have one category)
text file:	(leave this blank)
SQL:	(not needed, leave blank)

Creating your boundary coverage is the same process, except this time change Type from LINE to BOUND.

At this stage we have two vectors that have been cleaned, re-categorized, and separated into line and boundary.

Closing Polygons

This is the manual method of closing your polygons.
1. Open the boundaries layer in a digitizing session
2. Notice the end of the vectors. There will be either a red or green 'x'. If its red something is wrong
3. To snap one red vertex to another use the MOVE VERTEX tool
4. select the vertex to move, then choose its new location

you might want to cut a line or add a vertex to round out a corner or maybe you have too many vertices and want to delete a few. Be our guest.

Whats Next?

Well explore what you can do with vectors a bit more. All the tools you need are available at the command line or through the menu. Some things to try.

1. Measure the length of a line
2. create a buffer (internal, external, booth)
3. Open the DBF of a vector in Open Office and add some tabular data to your vector
4. create a vector coverage by using a raster map as a backdrop

Conclusion:

This tutorial is meant as an intro to GPS in GRASS, so some concepts were only touched on. We expect that the introduction of these tools will prompt some playing around. If you create a vector specific tutorial, the GRASS community could really use one.

Please contribute to filling out some of the sparse parts of this tutorial. Especaly when it becomes out of date.

Thanks for playing with us today.

About the Contributors

Dave Sampson

Dave is a Geographer specializing in Geograpic Information Systems (GIS). Dave has achieved an Honours of Outdoor, Recreation, Parks and Tourism and a Bachelor of Arts in Geography from Lakehead University in Thunder Bay, Ontario, Canada. He is particularly interested in the applications of GIS and Remote sensing in the management of parks and protected areas.

Dave was exposed to GRASS during university even though it was not in the curriculumn and few people had ever heard of it. He picked it up along with an interest in other open source projects. Dave is still very interested in its continued evolution getting closer to mainstream commercial acceptance. He formed the Ottawa GRASS (GIS) Users Group (OGUG) with the help of Scott Mitchel – a professor at Carleton University.

On line profile: http://cemml.carleton.ca:8080/OGUG/Members/drsampson/emp/