Open-source tool for automatic import of coded surveying data to multiple vector layers in GIS environment

Eva Stopková

Institute of Oriental Studies,
Slovak Academy of Sciences
Klemensova 19, 813 64 Bratislava, Slovak Republic
eva.stopkova@gmail.com

Abstract

This paper deals with a tool that enables import of the coded data in a single text file to more than one vector layers (including attribute tables), together with automatic drawing of line and polygon objects and with optional conversion to CAD. Python script v.in.survey is available as an add-on for open-source software GRASS GIS (GRASS Development Team). The paper describes a case study based on surveying at the archaeological mission at Tell-el Retaba (Egypt). Advantages of the tool (e.g. significant optimization of surveying work) and its limits (demands on keeping conventions for the points' names coding) are discussed here as well. Possibilities of future development are suggested (e.g. generalization of points' names coding or more complex attribute table creation).

Keywords: surveying; automatic import of coded data; GIS.

Introduction

Standard functionality of current *Geographic Information System* (GIS) software includes also a tool for vector layer import from a text file. However, it is required to use just one text file for one vector output, as concurrent import of multiple layers seems to be not supported yet.

Output files from data collectors that enable setting up field codes during the data acquisition (e.g. total station[1]) usually contain the data that should be distributed to several vector layers. In case of need to import a text file containing this type of the data, it is necessary to perform the task for each layer separately. This option might be quite time-consuming and, above all, an advantage of automatic import of coded data is lost. Alternatively, the data can be imported to CAD automatically at first and then CAD drawing can be converted to vector layers in GIS environment.

The purpose of this paper is to provide an overview of a script that has been developed to create multiple vector layers in GIS environment directly from acquired data (and eventually to convert them to CAD).

[1]Total station is an electronic device for spatial data acquisition. It measures distances, horizontal and vertical angles and returns coordinates of the objects in three-dimensional space.

State of Art

There are several CAD or specialized graphical software products for land surveyors that provide functionality for automatic data import and drawing creation. This section will mention examples of the most commonly used options.

AutoCAD [1] in version *Civil 3D* provides *description keys* [2] that enable to distribute the data to multiple layers according to points' names. But, at first it is necessary to create layers manually and to edit the description keys. To automatically draw a line from imported data, there are available *Linework Code sets* [3] and the extension *Civil Express Tools* [7].

MicroStation [4] enables automatic import of points according to their codes using extensions *Mgeo* [13] or *iNGs_Geo* [19]. The last one is probably known primarily in the region of Central Europe, as well as another software, **Kokeš** [10], that provides similar functionality.

As there has not been found any similar tool for direct import of coded data in GIS environment, a script *v.in.survey* has been designed and developed as an add-on for open-source software **GRASS GIS** [14].

Python script *v.in.survey*

The main purpose of the newly developed tool was to import coded data (e.g. output file from total station) into vector layers as automatically as possible. To deal with this task:

- the data should be distributed into vector layers without any manual editing. If the data codes reflect desired layer structure (names and geometry types), automatic setup of vector layers may spare plenty of time, especially if the data should be imported to many different layers.
- line or polygon objects should be drawn automatically just connecting the points according to code that indicate particular object and following the order of the points.
- separated line or polygon objects that belong to one layer should be merged at last.

All these tasks are supported in *GRASS GIS* [14] by existing modules. Script development dealt mainly with enabling communication between them (based on a quite simple system of point naming), but some functionality was modified as well to match specific needs of the data import.

Input data format

Points' names play a substantial role in data import using *v.in.survey.* They provide basic information for automatic layer creation, for connecting points with lines and for polygon creation, for updating attribute table and for merging desired layers if necessary. Thus each point name should consist of three parts that are separated by dots:

```
layer_name.vector_code.point_id
```

Layer name should be short but descriptive. If the layers should be merged, then the point name should contain base string with a suffix that indicates separated objects in the layer.

Vector code gives information about object type. Currently, points, lines and polygons are supported. Vector code string may reflect geometry (i.e. `point`, `line` or various abbreviations), but it can be descriptive as well (e.g. `tree`, `well` etc.). The script translates vector code from the rules given by the user.

Point ID is recommended to be an integer to enable automatic number increase during the data acquisition and to keep the order of the points to draw the features properly.

Process description

Script performance is shown in *Fig. 1*. The diagram was created using the Python profiler *cProfile* [24] and *Gprof2Dot* [9] according to [15].

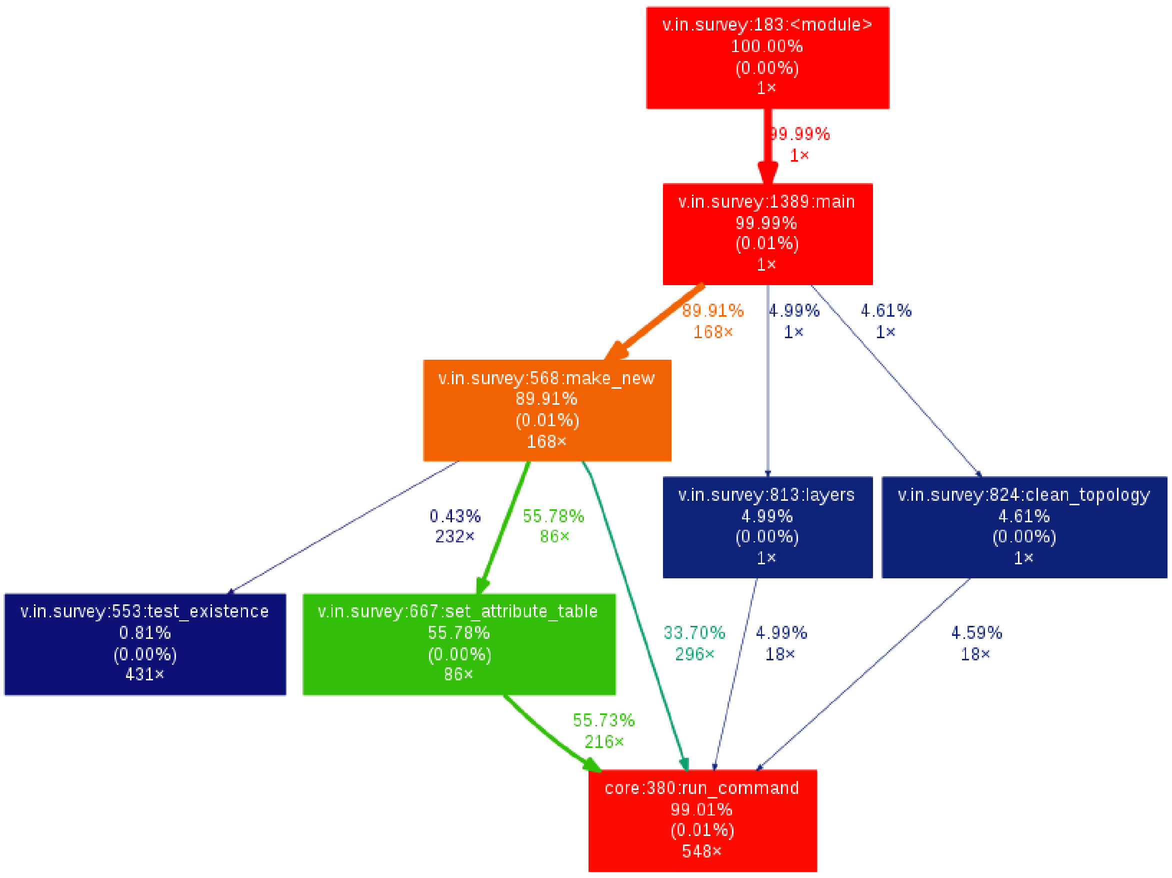

Figure 1: Performance profile of the script. The call function represents percentage of time $n\%$ spent in each function and its children [20], percentage of time *($n\%$)* spent in each function itself and the call count $n x$.

Pre-processing by the script includes sorting the data according to the point names (to keep points from separated layers together), data separation for vector layer import and conversion to standard format[2].

[2]In GRASS GIS, there are two modes for ASCII data import: *point* format for simple list of coordinates and *standard* for GRASS native vector format. For more detailed information, see [17].

The functionality of *make_new* consists of creating vector layers using *v.in.ascii* [17]. If necessary, conversion to the polygon layer is performed using *v.centroids* [6]. Attribute table for line and polygon layers is created using *v.db.addtable* [22] (categories of line layer objects are added at first using *v.category* [5]). Then, column that stores layer name is updated using *v.db.update* [21]. Point layers are imported in the *point format* together with the attributes.

Section *layers* supports merging the features by *v.patch* [11] using various patterns to detect desired layers. For more details, see the case studies in the following section. If any layers have been merged, *v.clean* [12] is performed to clean the topology. More information on the script processes is provided in the script manual available among *GRASS GIS* add-ons [16].

Conversion to CAD

Although *GRASS GIS* [14] and *QGIS* [25] provide tools for export to CAD, the script optionally supplies this conversion too due to the needs of the archaeological documentation. This requires especially 3D polylines support and exporting attributes to the text layers as well. Necessary functionality is based on modified parts of existing modules. Compared to the original functions, the conversion does not use imported vector layers, but the same text files with separated data that have been created in pre-processing for import into vector layers in GIS. The other differences between the output DXF files are summarized in *Tab. 1*.

Table 1: The differences between DXF export by *v.out.ogr* [8] and *v.in.survey*

Module	*v.out.dxf* [8]	*v.in.survey*
DXF content	separated layer	all the layers
3D polyline	no	yes
Layer content	point, line, boundary, centroid, point_label, centroid_label	layer (vector object) layer_label (label of the object) layer_elev_pts (point elevations) layer_label_pts (point labels)
Layer label	layer number	layer name

In fact, the tool enabling export of the whole project into the DXF file instead of separated ones is available in QGIS [25]. But we needed to export 3D features and the attributes as well and our another intention was to create complete archaeological map without any additional steps.

Case study: Tell el-Retaba

The script has been tested on spatial data from archaeological missions at Tell el-Retaba (Egypt). The data represents structures and small findings excavated during the seasons of 2014 and 2015.

The archaeological site of Tell el-Retaba is situated in the eastern Nile Delta, in the valley called Wadi Tumilat. The research is focused on the Pharaonic fortresses of the New Kingdom, but excavations prove presence of Hyksos settlement in 18^{th} - 16^{th} century BC and the likely presence of older fortress seems to be indicated here. More information on the Polish-Slovak mission can be found in [18] and in other scientific papers that have been referred there.

Coding system at the archaeological excavation

Archaeological documentation includes also maps of the excavations. This requires spatial determination of each structure and each finding (e.g. pottery, piece of metal, bone etc.). Each object is given an unique name (a number of stratigraphical unit or a number of finding) and it can be described by one of items recorded on quite a short list of finding types. During the season of 2014, a system of codes was developed for the purposes of effective surveying.

Table 2: Coding system at Tell el-Retaba

Layer name	Specification	ID	Description
SU0001	brd	01	a boundary of an archaeological layer
SU0001	elev	01	points within an archaeological layer
SU0001	PP	01	photogrammetric points for an archaeological layer
SF		0001	small finds
miscellaneous			fixed points, stake-out etc.

All the mentioned above means large datasets daily, that should be distributed to separate layers in CAD or GIS. That is the reason why we consider this data as an interesting sample for testing of the script.

Input data

Surveying has been performed using the total station *Trimble M3*. Positions of the measured objects were determined in three-dimensional local coordinate system *Retaba2014* [26], see the sample from the season of 2015 (*Fig. 2*, trench supervisors: L. Hulková and J . Marko):

```
# point_name,easting,northing,elevation
...
SU1716_1.brd.01,119.053,114.865,5.472
SU1716_1.brd.02,119.137,114.869,5.469
SU1716_1.brd.03,119.193,114.876,5.458
...
```

All the daily measurements were merged to one file. This dataset had to be modified in common text editor, as there were some points that did not respect sorting rules of the script. There might happen also another issues that require manual editing of the point names (see *Sect. 4.4*). If the data has been named correctly during the acquisition, just typos correction and small edits can be expected here.

Test of efficiency

The script was tested on the laptop, parameters of which are summarized in *Tab. 3*.

Table 3: Parameters of the computer that was used for the script testing

Operating system	Ubuntu 14.04 LTS, 64 bit
Processor	Intel Core i5, 2.60 GHz $\times$ 4
RAM	8 GiB

Following lines demonstrate the script commands that defined rules for distribution of the archaeological data into GIS project and CAD drawing. The first line contains general settings for import. Then, geometry types are coded and prefixes that indicate layers to be merged are summarized. At last, properties of output DXF file (name and text size) are included.

```
v.in.survey input=Retaba2014.csv outdir=Retaba2014 separator=tab \
easting=2 northing=3 elevation=4 \
pt_rules=pt,PP,elev ln_rules=profile,ln poly_rules=brd,brick \
merge_lyrs=SU1309_brd,SU1311_brick,SU1315_brd,SU1331_brick_2,\
SU1331_brick_3,SU1357_brick,SU1390_ln,SU1391_ln,flatM,trench \
dxf_file=Retaba2014.dxf text=0.05 -z -x
```

```
v.in.survey input=Retaba2015.csv outdir=Retaba2015 separator=comma \
easting=2 northing=3 elevation=4 \
pt_rules=pt,R14,PP,ebr,geo,pot,ch,elev ln_rules=line,nail,ctrl,bot \
poly_rules=brd,brick \
merge_lyrs=SU893_bot,trench,SU1678_brd,SU1681_brd,SU1682_2l, \
SU1691_brd,SU1691_brick,SU1695_1_brd,SU1718_brd,SU1723_brd, \
SU1728_brick,SU1750l_1,SU1751_brick,SU1759l \
dxf=Retaba2015.dxf textsize=0.05 -z -x
```

The criteria for automatic data import might be quite diverse, especially for complex datasets. Criteria definition requires good knowledge of the data purpose and some planning before the acquisition, but it might be worth considering the difference between the time taken by script performance (see *Tab. 4*) and the time taken by importing the data in other ways outlined above.

To demonstrate how the dataset structure may influence processing time, the tutorial dataset [23] from North Carolina was included into this test as well. This data differs from the archaeological data significantly: while the excavation data consists of plenty of separate layers that can be merged occasionally, the line vector layer in the North Carolina dataset has been merged from many components. How this has affected processing time, may be seen in *Tab. 4*. More detailed information on the case study is available in the script manual.

Table 4: Performance time on various datasets

Dataset	Number of:				Processing time	
	points	point layers	line layers	polygon layers	no DXF	DXF
NC clipped	71157	$1 \rightarrow 1$	$355 \rightarrow 1$	$25 \rightarrow 1$	$11^{m}15^{s}$	$17^{m}03^{s}$
Retaba 2014	3587	$155 \rightarrow 155$	$26 \rightarrow 20$	$160 \rightarrow 150$	$4^{m}31^{s}$	$7^{m}07^{s}$
Retaba 2015	3494	$111 \rightarrow 111$	$20 \rightarrow 17$	$106 \rightarrow 88$	$3^{m}35^{s}$	$4^{m}41^{s}$

Processing time rapidly increases with number of objects merged to one vector layer. This step includes time-consuming process of removing temporary layers that store the objects before patching. The dataset from the season of 2014 contains only 100 points more than the dataset from 2015, but the processing takes much more time. In 2014, there were 160 measured polygon objects that should be stored in 150 vector layers (i.e. 10 layers should be merged), while 2015, there were only 18 layers to be patched. Merging the layers increases processing time significantly for DXF conversion as well.

Validation of the results

Imported vector layers were compared with existing CAD drawings. Just issues depending on the input data needed to be emerged (*Tab. 5*): broken geometry and wrong layer name. *Fig. 2 – 4* illustrate the most frequented issues.

Table 5: Summary of the issues in automatic multiple layer import

Issue	Number of layers		Solution (modify input data)
	2014	2015	
messy shape	19	7	reorder input points
wrong connection	12	23	distinguish objects to be separated
wrong vector type	4	7	modify vector code in point name
small artificial shape	13	6	remove duplicate starting vertex
artificial closing of the area	1		edit manually or create line layer
missing layer	5	6	split artificially merged objects
correct layers	116	56	no need to modify
Total sum	170	105	

An archaeological layer in *Fig. 2* consists of three vertically stratified parts that have not been distinguished in the original data file. Thus, the object has been drawn just following order of the points and the shape (red hatched area) looks unrealistic. To get correct polygons (separate blue areas), different suffixes have been added to the point names in the input data.

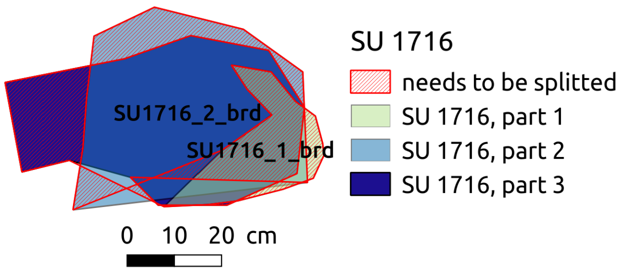

Figure 2: Layer to be split. Trench supervisors: L. Hulková and J. Marko, 2015.

The structure in *Fig. 3* was measured very effectively, without a need to go around it twice. There is nothing wrong about the way how it has been measured, but automatic drawing creation requires measurement according to topological rules to keep the order of the points.

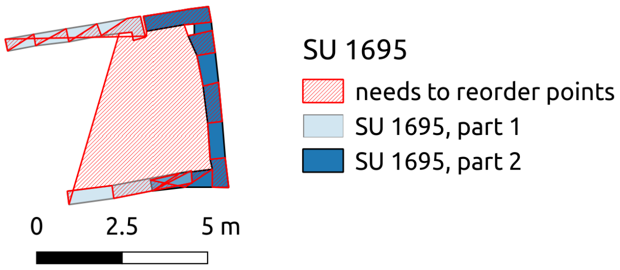

Figure 3: Layer to be reordered. Trench supervisor: K. Smoláriková, 2015.

Fig. 4 represents a stratigraphical unit with a small dangle in the corner. This can be avoided simply by not observing starting (or ending) point twice. Polygons are closed automatically.

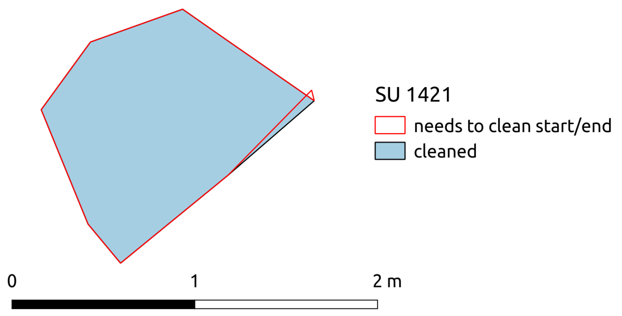

Figure 4: Layer to be cleaned. Trench supervisor: V. Dubcová, 2014.

Imported layers have been compared with the surveying journal to identify any missing layers. A few cases happened (see *Tab. 5*), but all of them have been caused by inappropriate merging when the layers that should stay separated have been merged (as in *Fig. 2*).

New DXF drawing has been overlaid with existing documentation as well. *Fig. 5* shows an archaeological layer labelled with point names, elevations and with the number of stratigraphic unit located on the layer centroid. The layer is visualized using exported labels (purple text) and the object (grey line) from original documentation.

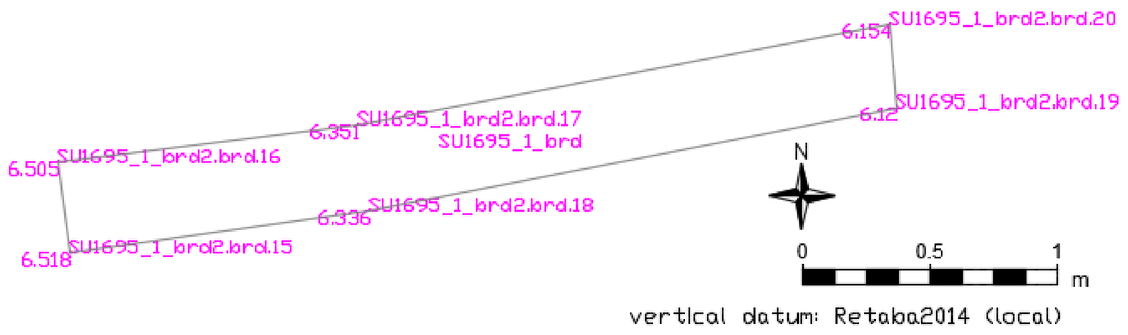

Figure 5: Export to DXF compared with original documentation.
Trench supervisor: K. Smoláriková, 2015.

North arrow, scale bar and information about vertical datum were added later (these items have not been exported by the script). Height of the text is general for the whole drawing and sometimes does not fit. Then it must be modified later manually as well.

Future work

For successful script performance, it is substantial to name the points in respect of quite simple, but also rigid system of rules that have been designed for this task. Although we believe that this system may cover needs of various human activities that require surveying measurements, for other datasets might be more effective any different system of point naming. The same can be said of exporting the data into CAD according to a template that requires creation of particular text layers connected to particular geometry entities.

Thus, in the future it might be useful to modify the script to distribute objects into the layers according to any rule given by the user. And maybe it would be useful to unify all the possibilities of DXF export that are provided in *GRASS GIS* [14] to an unique tool that would respect a variable template (i.e. content, attribute export etc.) given by the user according to the specific need of the data or of the task.

Other possibilities for further development consist of adding different input types (e.g. text attributes as well) and adding user-defined possibilities of automatic update of geometry properties in the attribute table.

Conclusion

The module supports geometry and text import into multiple GIS or CAD layers without any manual setup of their structure and properties. This may spare user's time extremely – in the case studies mentioned above, concurrent creation of CAD drawing and GIS project took minutes. Even using such useful features as *description keys* in AutoCAD [1], CAD drawing of the same datasets took days depending on the data complexity. However, point names should reflect desired structure of the layers and the data should be acquired according to topological rules (simply said, as if they were being "drawn" on the ground). Otherwise revision of the input data can take longer time than casual CAD drawing creation or import into GIS layers, especially in smaller datasets.

Acknowledgements

I thank to my colleagues from the archaeological mission, especially the trench supervisors, who contributed to acquisition of the test data with their archaeological knowledge. Special thanks goes to two anonymous reviewers whose constructive suggestions helped to improve the manuscript and enclosed script.

Archaeological mission at Tell el-Retaba has been supported by the Slovak Research and Development Agency (project no. APVV-0579-12).

References

[1] Autodesk, Inc. *AutoCAD*. URL: http://www.autodesk.com/products/autocad/overview (visited on 14/12/2015).

[2] Autodesk, Inc. *Description Keys*. URL: http://docs.autodesk.com/CIV3D/2013/ENU/index.html?url=filesCUG/GUID-DB075E8C-5E3D-4A0C-B379-90404AD181F2.htm,topicNumber=CUGd30e107714 (visited on 14/12/2015).

[3] Autodesk, Inc. *Linework Code Sets*. URL: http://docs.autodesk.com/CIV3D/2013/ENU/index.html?url=filesCUG/GUID-8810116C-BB09-46FC-B058-1B975E1B9C18.htm,topicNumber=CUGd30e52665 (visited on 14/12/2015).

[4] Bentley Systems. *MicroStation*. URL: https://www.bentley.com/en/perspectives-and-viewpoints/topics/viewpoint/connect-edition-perspective?skid=CT_PPC_GO_CNTP_EME_EN_T&gclid=CN__xYHd28kCFSgGwwodVFACZA (visited on 14/12/2015).

[5] Radim Blazek and Martin Landa. *v.category*. URL: https://grass.osgeo.org/grass70/manuals/v.category.html (visited on 11/12/2015).

[6] Hamish M. Bowman and Trevor Wiens. *v.centroids*. URL: https://grass.osgeo.org/grass70/manuals/v.centroids.html (visited on 11/12/2015).

[7] Mike Caruso. *Drawing Lines automatically by Point NAME range*. URL: https://forums.autodesk.com/t5/autocad-civil-3d-general/drawing-lines-automatically-by-point-name-range/td-p/5551586 (visited on 14/12/2015).

[8] Charles Ehlschlaeger and Radim Blazek. *v.out.dxf*. URL: https://grass.osgeo.org/grass70/manuals/v.out.dxf.html (visited on 11/12/2015).

[9] José Fonseca. *gprof2dot*. URL: https://github.com/jrfonseca/gprof2dot (visited on 06/06/2016).

[10] GEPRO s.r.o. *Kokeš*. URL: http://www.gepro.cz/produkty/kokes/ (visited on 14/12/2015).

[11] Dave Gerdes and Radim Blazek. *v.patch*. URL: https://grass.osgeo.org/grass70/manuals/v.patch.html (visited on 11/12/2015).

[12] David Gerdes, Radim Blazek, and Martin Landa. *v.clean*. URL: https://grass.osgeo.org/grass70/manuals/v.clean.html (visited on 11/12/2015).

[13] GISoft. *MGeo*. URL: http://www.gisoft.cz/MGEO/MGEO (visited on 14/12/2015).

[14] GRASS Development Team. *Geographic Resources Analysis Support System (GRASS) Software, Version 7.1. Open Source Geospatial Foundation*. URL: https://grass.osgeo.org/ (visited on 14/12/2015).

[15] GRASS Development Team. *Tools for Python programming*. URL: https://grasswiki.osgeo.org/wiki/Tools_for_Python_programming#cProfile_profiling_tool (visited on 06/06/2016).

[16] GRASS Development Team. *Vector add-ons*. URL: https://svn.osgeo.org/grass/grass-addons/grass7/vector/ (visited on 14/12/2015).

[17] Michael Higgins, James Westervelt, and Radim Blazek. *v.in.ascii*. URL: https://grass.osgeo.org/grass70/manuals/v.in.ascii.html (visited on 11/12/2015).

[18] Jozef Hudec, Emil Fulajtár, and Eva Stopková. "Historical and Environmental Determinations of the Ancient Egyptian Fortresses in Tell el-Retaba". In: *Asian and African Studies* 24.2 (Dec. 2015), pp. 247–283. URL: https://www.sav.sk/journals/uploads/120813537_Hudec.pdf.

[19] iNGs. *iNGs_Geo*. URL: http://www.ings.sk/?IDe=93899 (visited on 14/12/2015).

[20] José Fonseca et al. *Interpreting gprof's Output*. URL: http://sourceware.org/binutils/docs-2.18/gprof/Call-Graph.html#Call-Graph (visited on 07/26/2016).

[21] Moritz Lennert. *v.db.update*. URL: https://grass.osgeo.org/grass71/manuals/v.db.update.html (visited on 11/12/2015).

[22] Markus Neteler. *v.db.addtable*. URL: https://grass.osgeo.org/grass70/manuals/v.db.addtable.html (visited on 11/12/2015).

[23] Markus Neteler and Helena Mitasova. *North Carolina data set.* URL: https://grass.osgeo.org/download/sample-data/ (visited on 24/11/2015).

[24] Python Software Foundation. *The Python Profilers.* URL: https://docs.python.org/2/library/profile.html (visited on 06/06/2016).

[25] Quantum GIS Development Team. *QGIS Geographic Information System. Open Source Geospatial Foundation.* URL: http://www.qgis.org/en/site/ (visited on 05/01/2016).

[26] Eva Stopková. *Survey Control Network Retaba2014.* Tech. rep. Revision 2. 2016.