REM Installation instructions

Michael Haas - mhaas@gfz-potsdam.de

1 General information

This installation guide is for an Ubuntu 16.04 operating system. The instructions assume a fresh install of Ubuntu 16.04 and go through all steps necessary to install all software used in the training course. There are two parts the first part (Part A) describes the installation for all components used for data preparation and analysis. The second part (Part B) is an instruction on how to set up a server for the RRVS analysis. Although the target system is Ubuntu 16.04, all steps described here should be similar on other Linux distributions. Be aware that some of the dependencies might not be packed the same way as in Ubuntu and might be installed additionally.
Ubuntu 16.04 and all components (except software related to the Ladybug 3 camera) are open source and can be used free of charge. All software developed by the GFZ-Center for Early Warning Systems can be found on the online repository service github at:
`https://github.com/GFZ-Centre-for-Early-Warning`
Most components are also available for Windows environments. A good starting point for installing most tools discussed here on Windows is OSGeo4W. `https://trac.osgeo.org/osgeo4w/`

2 PART A. Software used for Pre- and Postprocessing

2.1 Before the installation

We assume a basic experience in linux environments, i.e., familiarity with a terminal and how to use a package manager furthermore root privileges are required. Make sure your repository list and all installed packages are up-to-date, run in a terminal:

```
sudo apt-get update
sudo apt-get upgrade
```

2.2 Tools

The code developed by GFZ is hosted on github, thus the first requirement is git and we will install most python packages using pip. Install them from the shell using apt-get and upgrade pip using pip:

```
sudo apt-get install git
sudo apt-get install python-pip
sudo pip install --upgrade pip
```

NOTE: We currently support only Python 2.7

2.3 Satellite Analysis

For the satellite analysis we will require the OrfeoToolbox (OTB) we get the binary from the OTB webpage and install it to the home directory:

```
wget https://www.orfeo-toolbox.org/packages/OTB-5.10.1-Linux64.run
chmod +x OTB-5.10.1-Linux64.run
./OTB-5.10.1-Linux64.run
```

This should run without errors and install all dependencies and OTB with the necessary python wrappers.
For python to find the wrappers and the wrappers to find the functions we need to add an environment variable to our profile:

```
gedit ~/.profile
```

add at the end of the file:

```
export ITK_AUTOLOAD_PATH=$HOME/OTB-5.10.1-Linux64/lib/otb/applications
export PYTHONPATH=$HOME/OTB-5.10.1-Linux64/lib/python
```

and save it. Source the file in the terminal:

```
source ~/.profile
```

This should load the newly set environmental variables. In future they should be set when you log in to the system. Try to import OTB in python you maybe have to install numpy:

```
sudo pip install numpy
sudo apt-get install ipython
ipython
```

From within ipython:

```
import otbApplication
otbApplication.Registry.GetAvailableApplications()
```

This should print a list of available functions of OTB.

2.4 GIS

Since most of the information we use during the training course is spatial information our primary visualisation and data exploration tool is a Geographic Information System (GIS). We use the free and open source QGIS. A recent version can be installed adding the QGIS repository and installing the package from there:

```
sudo sh -c 'echo "deb http://qgis.org/debian xenial main"\
            >> /etc/apt/sources.list'
sudo sh -c 'echo "deb-src http://qgis.org/debian xenial main "\
```

```
            >> /etc/apt/sources.list'
wget -O - http://qgis.org/downloads/qgis-2016.gpg.key | gpg --import
gpg --fingerprint 073D307A618E5811
gpg --export --armor 073D307A618E5811 | sudo apt-key add -
sudo apt-get update && sudo apt-get install qgis python-qgis
```

QGIS functionality can be extended by installing plugins. We will install a couple of plugins for the purpose of this workshop. Start qgis either from the dashboard (hit windows key and type "qgis") or by typing "qgis" in a terminal (better you will see in case there is a problem starting qgis).
When qgis has started move the mousepointer to the top of the window a couple of options should appear above the symbols, select:

```
Plugins -> Manage and Install Plugins
```

A new window will open type "OSM" in the "Search" field in the top of the window. It will show a list of plugins with OSM in the name. Select the "OSMDownloader". And click on Install plugin in the lower right corner of the window. This gives you a handy tool to download OSM data from within QGIS. Install the same way also the plugin "Rectangles ovals digitizing" a nice tool to digitize regular shapes, e.g., for defining rectangular regions of interest.
For the satellite analysis we will use a plugin developed at GFZ called `REM_SatEx` it is available from our github repository. Open a terminal, clone the plugin, install python sphinx and pyqt4 developer tools and deploy the plugin:

```
cd ~
git clone https://github.com/GFZ-Centre-for-Early-Warning/REM_satex_plugin
sudo apt-get install pyqt4-dev-tools
sudo pip install sphinx
cd REM_satex_plugin
make compile
make deploy
```

The core dependency of the SatEx plugin is the previously installed OTB. Open qgis and make sure the OTB installation worked and includes the necessary python wrappers. From the top menu in QGIS select

```
Plugins -> Python Console
```

and type:

```
import otbApplication
otbApplication.Registry.GetAvailableApplications()
```

This should print a list of available functions of OTB.
If this returns an error:
If the import fails make sure the pythonpath is set correctly:

```
import os
os.environ['PYTHONPATH']
```

this should contain the "PYTHONPATH" directory previously specified in ".profile". In case the list of functions is empty qgis cannot find the `"ITK_AUTOLOAD_PATH"` in the same file. Also try if it works from the python interpreter in a terminal.
For the Orfeo Toolbox to work in the processing plugin shipped with QGIS check the paths defined here point to the right directory: *Processing* → *Options* → *Providers* "OTB applications folder" should be similar to "/home/user/OTB-5.10.1-Linux64/lib/python" and "OTB command line tools folder" similar to "/home/sysop/OTB-5.10.1-Linux64/bin" if not click on the right side of the window next to the variable name and adjust it accordingly.
In the worst case there is a python script called `"run_as_script.py"` which allows to run the classification using the parameters specified in the "config.ini" in the repository.
Finally activate the plugin by opening again the "Manage and Install Plugins" from the "Plugins" dropdown menu and typing "GFZ". Tick the checkbox in front of "GFZ SatEx" to activate it. Among the symbols at the top of the QGIS Interface you should see two new buttons with "SatEx" written on them.
The SatEx repository contains a demo dataset copy it to the Desktop and try to apply the plugin using the input data provided in the directory to make sure the plugin works as intended. If you click on the help button within one of the dialogs you will find instructions on how to use the plugin.

2.5 Routing

For the routing we use PGRouting it builds upon a PostgreSQL and PostGIS installation. At the time of writing this document the most recent packages available in the official Ubuntu16.04 repository is PostgreSQL 9.5 and PostGIS 2.2. **Be aware that upgrading PostgreSQL or PostGIS can make the datacluster of a previous installation unusable! Before you upgrade PostgreSQL or PostGIS (be aware of apt-get upgrade) create a backup of the database cluster (see section ??).**

```
sudo apt-get install postgresql-9.5-postgis-2.2 postgresql-server-dev-9.5
```

Set up a password for the default PostgreSQL user "postgres" using psql , the terminal interpreter tool for PostgreSQL

```
sudo -u postgres psql postgres
```

The psql version should be printed in the shell and the prompt should change to "postgres=#", indicating that you are connected to the default database "postgres".
Execute the following command to set a password for the postgres user and leave psql afterwards:

```
\password
\q
```

Open as root the postgres configuration file:

```
sudo gedit /etc/postgresql/9.5/main/pg_hba.conf
```

Change "peer" to "md5" in the line (around 85):

```
old:local   all             postgres                                peer
new:local   all             postgres                                md5
```

and restart the postgres server:

```
sudo /etc/init.d/postgresql restart
```

Verify where your PostgreSQL datacluster resides on your harddrive using psql:

```
psql --u postgres
```

execute the following command within psql:

```
SHOW data_directory;
\q
```

to get the location of the datacluster. **Memorize or write down the location in case the server crashes at some point in the future and you have to restart it manually.** On Ubuntu the default should be "/var/lib/postgresql/9.5/main". Alongside the command line tool psql there is also a tool with a graphical user interface which can be handy for quick checks or a more comfortable look at data. It is called "pgadmin3" and can be installed using:

```
sudo apt-get install pgadmin3
```

Now to install PGRouting add a repository to aptituted:

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main'
sudo apt install wget ca-certificates
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
sudo apt update
sudo apt-get install libboost-graph-dev
sudo apt-get install libcgal*
sudo apt-get install postgresql-9.5-pgrouting
```

For the database you want to use for routing, remember to activate the PostGIS and PGRouting Extensions, i.e. within psql:

```
psql --u postgres
```

create a new database e.g. "routing", connect to it and create the extensions:

```
CREATE DATABASE routing;
\c routing
CREATE EXTENSION postgis;
CREATE EXTENSION pgrouting;
\q
```

This gives you the functions you need for routing on the database.
Hint: Routing is based on directed graphs. For this to work properly the topology of the network has to be clean. The GRASS GIS functions (e.g. v.clean) are very helpful in cleaning streetnetworks used for routing. They can be installed alongside QGIS and called from within QGIS.

2.6 Statistical Analysis

One of the best free tools for statistical analysis is the R project. We can install it using:

```
sudo sh -c 'echo "deb http://cran.rstudio.com/bin/linux/ubuntu xenial/" >> /etc/apt/sources.
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 51716619E084DAB9
sudo apt-get update
sudo apt-get install r-base
```

this gives you the baisc features of R, but its true power lies in the numerous packages. Packages can be installed easily from within R. Start it up

```
R
```

and install some packages we will need:

```
install.packages("RPostgreSQL")
install.packages("rgeos")
install.packages("rgdal")
install.packages("sp")
```

the first of this command will ask you to select a mirror from where to dowload the packages. Select one close to your location.
Also python can be used for data analysis some good libraries to start with are installed using pip:

```
sudo pip install numpy
sudo pip install scipy
sudo pip install pandas
sudo pip install lxme
sudo pip install sqlalchemy
sudo pip install matplotlib
```

3 PART B. Installation of an RRVS server

This part describes the installation of a RRVS server hosting the `REM_RRVS` tool and its Google powered pendant GRVS.

3.1 Before the installation

We assume a basic experience in linux environments, i.e., familiarity with a terminal and how to use a package manager furthermore root privileges are required. Make sure your repository list and all installed packages are up-to-date, run in a terminal:

```
sudo apt-get update
sudo apt-get upgrade
```

3.2 Tools

The server can be installed on any Ubuntu 16.04 (a display server, X-server is not necessary.) The code developed by GFZ is hosted on github, thus the first requirement is git and we will install most python packages using pip. Install them from the shell using apt-get and upgrade pip using pip:

```
sudo apt-get install git
sudo apt-get install python-pip
sudo pip install --upgrade pip
```

NOTE: We currently support only Python 2.7

3.3 REM Database

The core of the RRVS server is the REM Database. The database is a PostgreSQL database with PostGIS extension. Follow the instructions in the Routing section in Part A of this installation manual to obtain a PostgreSQL and PostGIS installation on your machine.
Get the schema for the database from github and clone it to your home directory:

```
git clone git://github.com/GFZ-Centre-for-Early-Warning/REM_DBschema ~/REM_DBschema
```

Go to the directory and connect to the database cluster using psql (providing the new password):

```
cd ~/REM_DBschema
psql --u postgres
```

Then, create a new database called "rem", connect to it, the prompt should change to "rem=#" and start creating the schemas of the REM database from the scripts in the cloned directory:

```
CREATE DATABASE rem;
\c rem
\i create_templatedb.sql
```

Monitor the output printed in your shell there should be no errors (only short commands) printed while the script is running.

3.4 RRVS

The RRVS tool is using the flask webframework for Python and we recommend using an Apache2 wsgi server for hosting the application, thus install apache2 with mod-wsgi on your system, activate wsgi and create a virtual host for the app:

```
sudo apt-get install apache2 libapache2-mod-wsgi
sudo a2enmod wsgi
sudo gedit /etc/apache2/sites-available/rrvs.conf
```

3.4.1 Apache2 settings

In the file create a virtual host by adding the following content:

```
Listen 80
<VirtualHost *:80>
        ServerName localhost
        WSGIScriptAlias / /var/www/html/rrvs/rrvs.wsgi
        <Directory /var/www/html/rrvs/webapp>
            Order allow,deny
            Allow from all
        </Directory>

        Alias /static /var/www/html/rrvs/webapp/static
        <Directory /var/www/html/rrvs/webapp/static/>
            Order allow,deny
            Allow from all
        </Directory>
        ErrorLog ${APACHE_LOG_DIR}/error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        Alias /pano /data/pano
        <Directory /data/pano/>
             Options Indexes FollowSymLinks MultiViews
                Require all granted
        </Directory>

</VirtualHost>
```

The last part sets up a virtual directory from which the panoramic images will be served we created a directory "/data/pano" but this can be any location on the system just adjust the path accordingly:

```
sudo mkdir /data/pano
sudo chmod -R 755 /data
```

Make sure the permissions are set to 755 for all directories of this path. To secure it from unwanted direct access put a file index.html in it with a redirect to the main page of the server:

```
<html class="wf-fontawesome-n4-active wf-active">
<head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>RRVS</title>
</head>
<body>
<meta http-equiv="refresh" content="0; url=http://your.server.com" />
</body>
</html>
```

After saving the virtual host file activate the site

```
sudo a2ensite rrvs
```

3.4.2 The webapplication

The RRVS tool is located in another git repository. Go to the html directory created by apache on your computer and clone the repository there in a directory called rrvs (note that only root has write permissions there):

```
cd /var/www/html
sudo git clone git://github.com/GFZ-Centre-for-Early-Warning/REM_RRVS ./rrvs
cd rrvs
```

Install a redis database which is used for server side caching and the python requirements:

```
sudo apt-get install redis-server
sudo pip install -r python_requirements.txt
```

Now we have to adjust the configuration of the application:

```
sudo cp rrvs_config.py.example rrvs_config.py
sudo gedit rrvs_config.py
```

In the configuration file adjust the following lines:

```
old: SQLALCHEMY_DATABASE_URI = 'postgresql://user:pw@localhost:5432/rem'
new: put the postgres username instead of "user" probably "postgres"
     and for "pw" the "password" you set before
old: SECRET_KEY = '42'
new: Set a strong key here,
     best is to generate a random alphanumeric string
old: DEBUG = True
new: DEBUG = False
```

Open the wsgi file

```
sudo gedit rrvs.wsgi
```

and verify that "sys.path.append("/var/www/html/rrvs") in line 4 points to the right directory.
Now restart the apache2 server

```
sudo service apache2 restart
```

3.4.3 Expanding the REM database for RRVS

In order to use the perviously created rem database we have to extend it using an sql script provided with the rrvs repository go to the scripts directory and start psql:

```
cd /var/www/html/rrvs/scripts
psql --u postgres
```

From within psql, connect to the rem database and run the extension script:

```
\c rem;
\i extend_rem_db.sql;
\q
```

Again monitor the output for error messages.

3.5 Panoramic images

In order for the webapplication to find the images put them in the folder previously defined as the /pano directory in the Apache2 server configuration. Our suggestion is to make a separate directory for each survey in the pano directory.

3.6 Populating the Database for RRVS

The population of the database is done after a survey is completed. As the first step we need to define a new survey in the table survey.survey of the rem database and populate the image schema. We assume that the GFZ Permanent tool has been used to process the data collected with the GFZ MoMa system. In this case the survey information is created alongside the image resampling and metadata creation. We can then simply connect to the database and import the sql file created by Permanent.

```
psql --u postgres
\c rem;
\i metadata.sql
\q
```

This creates the survey and writes all information about the images to the database.
In the next step we populate the asset schema in the database with building footprints. Footprints can be obtained from sources as OpenStreetMap (OSM) directly (via the QGIS plugin) or can be digitized from satellite/aerial imagery or using the Java application of OSM (JOSM) with the building tool.
The last missing information before we can generate tasks for the analysis is to set up users an example can be found

```
sudo gedit users.sql
```

Adjust the file accordingly, i.e., remove or add lines or change the name of the user in the VALUES:

```
--add test users, roles and tasks
INSERT INTO users.users(id, authenticated, name) VALUES (1, TRUE, 'Test1');
INSERT INTO users.users(id, authenticated, name) VALUES (2, TRUE, 'Test2');

INSERT INTO users.roles(id, name) VALUES (1, 'public');
INSERT INTO users.roles_users(user_id, role_id) VALUES (1, 1);
INSERT INTO users.roles_users(user_id, role_id) VALUES (2, 1);
```

After users are set up you should be able to reach the login page.
If you are prompted with an error check the error file of the apache server for error messages:

```
sudo gedit /var/log/apache2/error.log
```

3.6.1 Add a translation to another language

In order to translate the RRVS system to another language there are two things to consider. The first is adding a description in the `taxonomy.dic_attribute_value` table in the database. There is a script `add_language_tax_.py` given a csv with the attribute code and the translation (see example translation.csv in the scripts directory) this script adds a column with the description to the table in the database. In order for the app to find the translation several changes have to be made in the code.

- in `webapp/__init__.py` add the iso code for the language to the list vor the variable "lang" (around line 39)
- in webapp/model.py add the new column in the database table (around line 93)
- in webapp/forms.py add a RRVSForm class for the language (NOTE: this and the following points will be fixed in a later version of the RRVS)
- in webapp/views.py add the new RRVSForm class to the import statement (line 14)
- in webapp/views.py add the new RRVSForm class to the import statement (line 14)
- in webapp/views.py add an "elif" for the new language and the RRVSForm class

The second part is to add a translation for the html textblocks. We use Flask-Babel to render the html text in the preferred browser language. First open the `rrvs_config.py` file and add the new language to the LANGUAGES dictionary. Then run pybabel to add the new language:

```
pybabel extract -F babel.cfg -k lazy_gettext -o messages.pot webapp
pybabel update -i messages.pot -d webapp/translations
```

This adds a new messages.po file containing the translations for the language. Using for example poedit you can provide the translations (here an example for the already added Spanish 'es')

```
poedit webapp/translations/es/LC_MESSAGES/messages.po
```

3.7 GRVS

There is a version of the RRVS which does not require the collection of omni-directional images but uses Google Streetview within the app. In order for this to work properly there are some deviations from the standard RRVS installation. First of all a Google key is required. Create a google account and obtain a Google Maps API key following the instructions here: `https://developers.google.com/maps/documentation/javascript/get-api-key?hl%3C0en`
Once you obtained a key we can start installing the GRVS. The code is in the same repository as the RRVS. Get the RRVS code repository and checkout the "grvs" branch

```
cd /var/www/html
sudo git clone git://github.com/GFZ-Centre-for-Early-Warning/REM_RRVS ./grvs
cd grvs
sudo git checkout grvs
```

The rest of the installation procedure is similar to the RRVS installation.