open_AR_Sandbox

Release 0.0.1

Daniel

Oct 19, 2021

GETTING STARTED

1 About **1**

2 Features **3**

3 Modules **5**
- 3.1 Implemented modules . . . 5
- 3.2 Modules in implementation process . . . 6

4 License, use and attributions **7**

5 Requirements **9**

6 Installation **11**
- 6.1 open_AR_Sandbox package . . . 11
- 6.2 Standard packages . . . 11
- 6.3 Kinect . . . 12
  - 6.3.1 For Windows . . . 12
  - 6.3.2 For Linux . . . 12
- 6.4 LiDAR L515 . . . 14
  - 6.4.1 For Windows . . . 14
  - 6.4.2 For Linux . . . 14
  - 6.4.3 Running with python . . . 15

7 Download sample data **17**

8 External packages **19**
- 8.1 GemPy . . . 19
- 8.2 Devito . . . 19
- 8.3 PyGimli . . . 19

9 Project development **21**
- 9.1 Project Lead . . . 21
- 9.2 Maintainers (also external to CGRE) . . . 21

10 Obtaining a full system **23**

CHAPTER

ONE

ABOUT

Welcome to the open_AR_Sandbox repository. If you do not know what this is all about, have a look at this video:

Warning! It is unfortunate that we have to state this here, but:

Downloading the software and presenting it somewhere as your own work is serious scientific fraud! And if you develop content further, then please push these developments back to this repostory - in the very interest of scientific development (and also a requirement of the license). For more details, please consult the information below and the license.

Augmented Reality - Geology Sandbox 2

Lunch Lehre am 5. Februar 2020 (unfortunaltely only in German)

Augmented Reality (AR) Sandboxes are a great tool for science outreach and teaching due to their intuitive and interaction-enhancing operation. Recently AR Sandboxes are becoming increasingly popular as interactive exhibition pieces, teaching aids and toys.

AR Sandboxes consist of a box of sand that can be freely sculpted by hand. The topography of the sand is constantly scanned with a depth camera and a computed image is projected back onto the sand surface, augmenting the sandbox with digital information.

However, most of these common AR Sandboxes are limited to the visualization of topography with contour lines and colors, as well as water simulations on the digital terrain surface. The potential for AR Sandboxes for geoscience education, and especially for teaching structural geology, remains largely untapped.

For this reason, we have developed open_AR_Sandbox, an augmented reality sandbox designed specifically for the use in geoscience education. In addition to the visualization of topography it can display geologic subsurface information such as the outcropping lithology, creating a dynamic and interactive geological map. The relations of subsurface structures, topography and outcrop can be explored in a playful and comprehensible way.

Deployment	**|pypi|**
GitHub	
Binder	
License	
Documentation	
GitHub workflow	
Issue tracking	
Pull requests	

CHAPTER

TWO

FEATURES

- Compatible with most AR Sandbox builds
- Subroutine for calibration and alignment of depth image, sand surface and projection
- Versatile model creation with the powerful GemPy library
- Open-source under GNU Lesser General Public License v3.0
- Fully customizable color map, contours and fault line visualization
- Computer vision algorithms that have been added recently to the sandbox open up a whole new field of possibilities! By placing printed markers into the sandbox, thew user can trigger actions or define points, lines and areas in the sandbox without using the computer

CHAPTER

THREE

MODULES

The open_AR_Sandbox as well as GemPy are under continuous development and including more modules for major outreach.

3.1 Implemented modules

- MarkerDetection: Place virtual boreholes in the model, define a cross section with multiple markers, set the start position for simulations (landslides, earthquakes, etc.). For more information check ArUco's marker detection
- TopoModule: Normalize the depth image to display a topography map with fully customizable contour lines and variable heights
- SearchMethodsModule: Takes the depth image and performs Monte-Carlo simulation algorithms to construct the probability distribution based on the structure of the current DEM in an interactive way (Hamiltonian Monte Carlo demo)
- GemPyModule: Use the full advantage of the powerful GemPy package to construct geological models and visualize them on the sandbox in real-time
- GradientModule: Takes the gradient information from the depth image and highlight slopes in x and y direction, calculation of laplacian, interactive hill shading, visualization of a vector field, and streamline plot
- LoadSaveTopoModule: Takes the depth image and allows it to be saved as a DEM to reconstruct topographies previously constructed
- LandslideSimulation: With precomputed landslides simulations, recreate a topography and trigger a landslide to visualize its flow, direction, and velocity in real-time, or frame by frame
- PrototypingModule: Create your own module with the help of this module to link the live threading of the sandbox with your ideas
- LandscapeModule: Landscape generations using machine learning codes powered by CycleGAN
- SeismicModule: Module for seismic wave modelling in the sandbox. This uses the power of Devito
- GeoelectricsModule: Module for visualization of geoelectrical fields using ArUco markers as electrodes. This use power of PyGimli

Check the video below for some of the features in action:

3.2 Modules in implementation process

- More Tutorials, examples, tests and documentation to help you develop your own modules
- GemPy optimization for (much!) higher frame-rates
- On-the-fly modification of the geological model (layer dip, thickness fault throw, etc.)
- Integration of more depth sensors (support to all kinect sensors)
- Improve compatibility with Linux and MacOS
- …

CHAPTER

FOUR

LICENSE, USE AND ATTRIBUTIONS

Feel free to download and use the open_AR_Sandbox software! We do not provide any warranty and any guarantee for the use. We also do not provide professional support, but we aim to answer questions posted as Issues on the GitHub page as quickly as possible.

Open_AR_Sandbox is published under an GNU Lesser General Public License v3.0, which means that you are free to use it, if you do not do any modifications, in a wide variety of ways (even commercially). However, if you plan to modify and redistribute the code, you also have to make it available under the same license!

Also, if you do any modifications, especially for scientific and educational use, then please provide them back to the main project in the form of a pull request, as common practice in the open-source community. If you have questions on the procedure, feel free to contact us about it.

These are the main conditions for using this library:

- License and copyright notice
- Disclose source
- State changes
- Same license (library)
- For more details on the licsense, please see provided license file.

If you use open_AR_Sandbox in a scientific abstract or publication, please include appropriate recognition of the original work. For the time being, please cite:

```
@INPROCEEDINGS{2019EGUGA..2114925V,
    author = {{Virgo}, Simon and {De La Varga Hormazabal}, Miguel and {Wellmann},␣
↪Florian},
     title = "{Open-AR-Sandbox: An open-source Augmented Reality platform for geoscience}
↪",
 booktitle = {EGU General Assembly Conference Abstracts},
      year = 2019,
    series = {EGU General Assembly Conference Abstracts},
     month = apr,
       eid = {14925},
     pages = {14925},
    adsurl = {https://ui.adsabs.harvard.edu/abs/2019EGUGA..2114925V},
   adsnote = {Provided by the SAO/NASA Astrophysics Data System}
   }
```

CHAPTER

FIVE

REQUIREMENTS

You will need:

- Microsoft Kinect (we tested the first and second generation kinect with a usb adapter, but every kinect compatible with the pyKinect drivers will likely work).
- Projector
- A box of Sand

Mount the Kinect and projector facing down vertically in the center above of the box. The optimal distance will depend on the size of your sandbox and the optics of the projector, from our experience a distance of 150 cm is well suited for a 80 cm x 100 cm box. More details on how to set up the Kinect and projector can be found in the 1_calib_projector.ipynb and 2_calib_sensor.ipynb notebooks, and if you want to use the ArUco markers 3_calib_arucos.ipynb.

CHAPTER

SIX

INSTALLATION

First of all you will need a healthy Python 3 environment. We recommend using Anaconda. In addition to some standard Python 3 packages, you will need a specific setup dependent on the Kinect version you are using. In the following we provide detailed installation instructions.

6.1 open_AR_Sandbox package

Download or clone this repository open_AR_Sandbox from GitHub.

First: Clone the repository:

```
git clone https://github.com/cgre-aachen/open_AR_Sandbox.git
```

Second: Create a new anaconda environment:

```
conda create -n sandbox-env python
```

Third: When you want to use the sandbox and the packages we are about to install you will have to activate the environment before starting anything:

```
conda activate sandbox-env
```

6.2 Standard packages

To install all the standard packages please use the requirements.txt file:

```
pip install -r requirements.txt
```

You can also have a local installation of the sandbox by using the File "setup.py" by doing:

```
pip install -e
```

6.3 Kinect

6.3.1 For Windows

- Kinect v1 - Future

There is still no support for Kinect v1!

- Kinect v2 - PyKinect2 (tested on Windows 10)

Install the current Kinect SDK including drivers. You can use the software bundle to test the connection to your kinect, before you continue.

To make Python and the Kinect SDK communicate, install the related PyKinect2 wrappers which can be easily installed via:

```
pip install pykinect2
```

Unfortunately, the configuration of PyKinect2 needs to be adjusted to work on a 64 bit System. Therefore, edit the Lib/site-packages/pykinect2/PyKinectV2.py file, go to line 2216 and comment it:

```
# assert sizeof(tagSTATSTG) == 72, sizeof(tagSTATSTG)
```

Add the following lines below:

```
import numpy.distutils.system_info as sysinfo
required_size = 64 + sysinfo.platform_bits / 4
assert sizeof(tagSTATSTG) == required_size, sizeof(tagSTATSTG)
```

6.3.2 For Linux

- Kinect v1 - libfreenect

To make open_AR_Sandbox talk to the first generation kinect you will need the Libfreenect Drivers with Python Wrappers. The installation is kind of straight forward for Linux and MacOS but challenging for Microsoft (in fact: if you pull it off, let us know how you did it!) The steps can be summarized as follows (refer to any problems regarding installation in to link) To build libfreenect, you'll need

-> libusb >= 1.0.18 (Windows needs >= 1.0.22) -> CMake >= 3.12.4 (you can visit this page for detailed instructions for the installation)

Once these are installed we can follow the next commands:

```
sudo apt-get install git cmake build-essential libusb-1.0-0-dev
sudo apt-get install freeglut3-dev libxmu-dev libxi-dev
git clone https://github.com/OpenKinect/libfreenect
cd libfreenect
mkdir build
cd build
cmake -L .. # -L lists all the project options
cmake .. -DBUILD_PYTHON3=ON
make


cd ../wrappers/python
```

(continues on next page)

(continued from previous page)

```
python setup.py install
# now you can see if the installation worked running an example
python demo_cv2_async.py
```

- Kinect v2 - freenect2 or pylibfreenect2

For this we are going to use a python interface for the library libfreenect2 called freenect2.

First we need to install the freenect2 as described in the installation guide. The steps can be summarized as follows (refer to any problems regarding installation in to link):

```
git clone https://github.com/OpenKinect/libfreenect2.git
cd libfreenect2

sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libusb-1.0-0-dev libturbojpeg0-dev libglfw3-dev
```

With all the dependencies installed now we can make and install:

```
mkdir build && cd build
cmake .. -DENABLE_CXX11=ON -DENABLE_OPENCL=ON -DENABLE_OPENGL=ON -DBUILD_OPENNI2_
↪DRIVER=ON -DCMAKE_INSTALL_PREFIX=$HOME/freenect2 -DCMAKE_VERBOSE_MAKEFILE=ON
make
make install
```

Set up udev rules for device access:

```
sudo cp ../platform/linux/udev/90-kinect2.rules /etc/udev/rules.d/
```

Now unplug and replug the Kinect sensor.

Test if the kinect is correctly installed, by running:

```
./bin/Protonect
```

You should be able to see the kinect image working. If not, check libfreenect2 installation guide for more detailed instructions of installation.

If everything is working until now, we can install the python wrapper. For this first we need to indicate where the freenect2 folder can be found:

```
export PKG_CONFIG_PATH=$HOME/freenect2/lib/pkgconfig
```

NOTE: If you installed the freenect2 in other location, specify variables with the corresponding path

Now we can use pip install, or any other method described in the freenect2 installation guide:

```
pip install freenect2
```

IMPORTANT: To this point will work in any python that starts with the terminal. Nevertheless, if we start python from another source, the error ImportError: libfreenect2.so.0.2: cannot open shared object file: No such file or directory will appear every time we import the package. To fix this problem we will need to export the variables again or if you want a more permanent solution, open the .bashrc file and paste the following at the end of the file:

```
# set PATH to freenect2 to be imported in python
export PKG_CONFIG_PATH=$HOME/freenect2/lib/pkgconfig
```

With this it will always work for any python open from the terminal. Including jupyter notebooks

But now if we want to run this package in Pycharm or symilar, we can directly copy the 3 files (libfreenect2.2.s0...) from the freenect2/lib folder into the lib folder of your environment. For instance, if you are using an anaconda environment, open the folder:

```
<your_path>/anaconda3/envs/<sandbox-env>/lib
```

In this folder paste the previous copied files (3 files!!!). Keep in mind that you need to replace the <...> with your specific path. If you dont want the manual work then run directly (remember to change the paths according to your needs):

```
sudo cp $HOME/freenect2/lib/libfreenect2{.so,.so.0.2,.so.0.2.0} $HOME/anaconda3/envs/
↪sandbox-env/lib/
```

6.4 LiDAR L515

6.4.1 For Windows

First, go to the latest release page on GitHub and download and execute the file:

```
Intel.RealSense.Viewer.exe
```

Follow the instructions for the installation and update the firmware of your sensor. You should be able to use and see the depth and RGB image.

6.4.2 For Linux

Detailed installation steps can be found in the linux installation guide. The steps are as follows:

Register the server's public key:

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-key␣
↪F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE || sudo apt-key adv --keyserver hkp://
↪keyserver.ubuntu.com:80 --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE
```

In case the public key still cannot be retrieved, check and specify proxy settings:

```
export http_proxy="http://<proxy>:<port>"
```

and rerun the command. See additional methods in the following link.

Add the server to the list of repositories:

Ubuntu 16 LTS:

```
sudo add-apt-repository "deb https://librealsense.intel.com/Debian/apt-repo xenial main"␣
↪-u
```

Ubuntu 18 LTS:

```
sudo add-apt-repository "deb https://librealsense.intel.com/Debian/apt-repo bionic main"␣
↪-u
```

Ubuntu 20 LTS:

```
sudo add-apt-repository "deb https://librealsense.intel.com/Debian/apt-repo focal main" -
↪u
```

Install the libraries:

```
sudo apt-get install librealsense2-dkms
sudo apt-get install librealsense2-utils
```

Reconnect the Intel RealSense depth camera and run:

```
realsense-viewer
```

to verify the installation.

6.4.3 Running with python

After the sensor is installed on your pltaform, the Python wrapper can be easily installed via:

```
pip install pyrealsense2
```

If any problems with the installation reference to Intel RealSense Python Installation

CHAPTER

SEVEN

DOWNLOAD SAMPLE DATA

You have the option to download some publicly shared files from our open_AR_Sandbox shared folder. You will need to do this if you want to run the tests, use the landslides simulations and/or get the trained models for the the use of the Landscape generation module.

In the terminal type:

```
python3 sandbox/utils/download_sample_datasets.py
```

and follow the instruction on the terminal to download the specific files you need. We use Pooch to help us fetch our data files and store them locally in your computer to their respective folders. Running this code a second time will not trigger a download since the file already exists.

CHAPTER

EIGHT

EXTERNAL PACKAGES

8.1 GemPy

To use implicit geological models inside the sandbox, go to GemPy, clone or download the repository and follow the GemPy Installation instructions. With GemPy installed you can follow the tutorial GempyModule:

```
pip install gempy
```

If using windows you will need to install Theano separately as instructed in here:

```
conda install mingw libpython m2w64-toolchain
conda install theano
pip install theano --force-reinstall
```

8.2 Devito

This package uses the power of Devito to run wave proppagation simmulations. More about this can be found in notebooks/tutorials/10_SeismicModule/. Follow the Devito installation instructions. This module so far have only support in Linux:

```
pip install --user git+https://github.com/devitocodes/devito.git
```

8.3 PyGimli

This library is a powerful tool for geophysical inversion and modelling. Some examples can be found in notebooks/tutorials/11_Geophysics/. PyGimli can be installed following the installation intructions here. We recomend creating a new environment where PyGimli is already installed and over that one install the sandbox dependencies:

```
conda create -n sandbox-env -c gimli -c conda-forge pygimli=1.1.0
```

And now go back to installation and follow all over again the instruction but skipping step 2:

```
PyTorch
```

To use the LandscapeGeneration module we need to install PyTorch. This module use the power of CycleGAN to take a topography from the sandbox, translate this as a DEM and then display it again on the sandbox as a Landscape image. To install the dependencies for this module do:

- For Windows:

```
pip install torch===1.6.0 torchvision===0.7.0 -f https://download.pytorch.org/whl/
↪torch_stable.html
```

- For Linux:

```
pip install torch torchvision
git clone https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
cd pytorch-CycleGAN-and-pix2pix
pip install -r requirements.txt
```

Once this is installed, copy the trained model in /notebooks/tutorials/09_LandscapeGeneration/checkpoints folder, and then follow the notebook. Get in contact with us to provide you with the train model for this module.

CHAPTER

NINE

PROJECT DEVELOPMENT

Open_AR_Sandbox is being developed at the Department for Computational Geoscience and Reservoir Engineering (CGRE) at the RWTH Aachen University.

9.1 Project Lead

Prof. Florian Wellmann, Ph. D. (@flohorovic)

9.2 Maintainers (also external to CGRE)

- Daniel Escallón Botero, M. Sc. (@danielsk78)
- Dr. rer. nat. Simon Virgo (@SimonVirgo)
- Miguel de la Varga Hormazabal, M. Sc. (@leguark)
- Lasse Klein, B. Sc. (@devlk2)

CHAPTER

TEN

OBTAINING A FULL SYSTEM

If you are interested in buying a fully operating set-up including appropriate hardware, pre-installed software, and set-up and maintenance, please contact Terranigma Solutions GmbH.