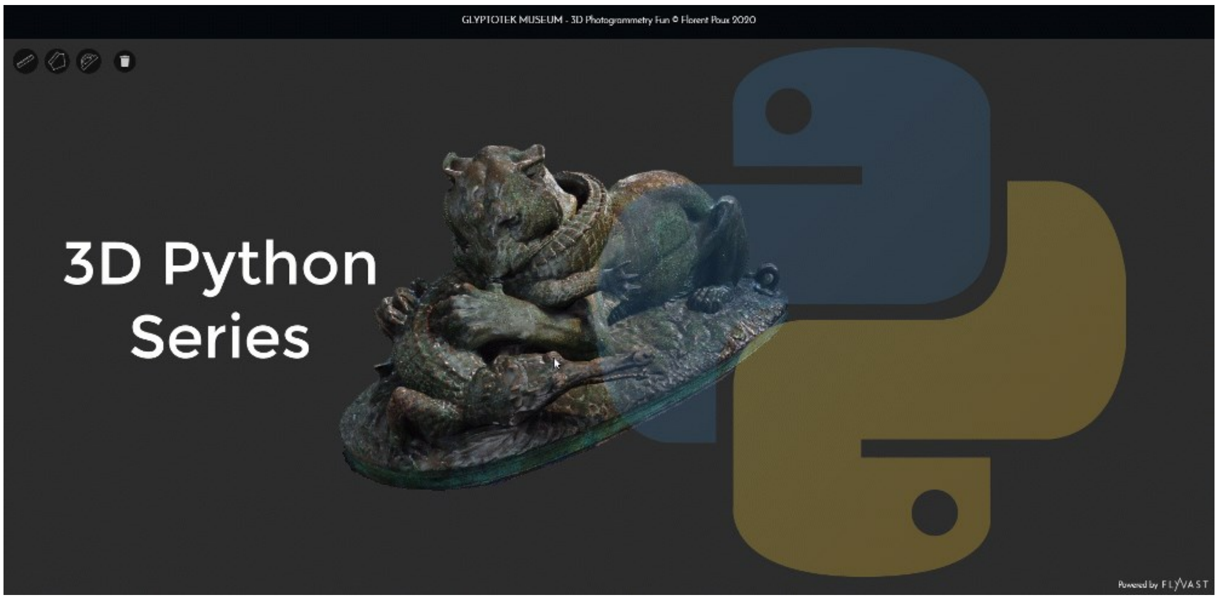
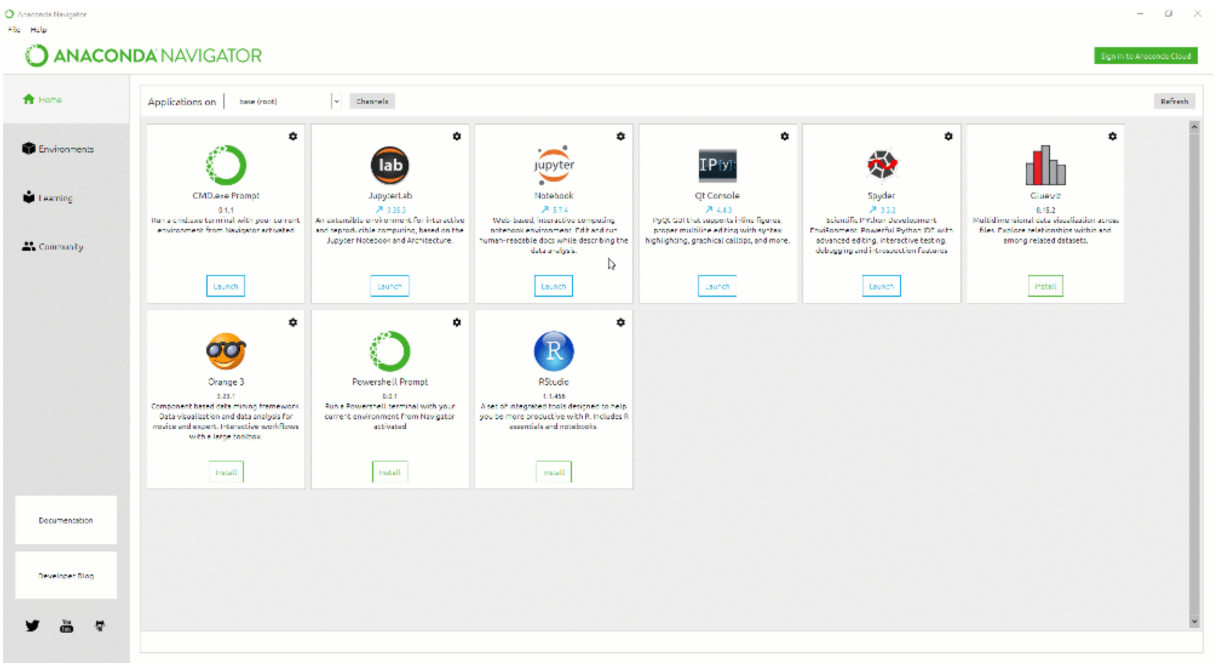
Discover 3D Point Cloud Processing with Python

Tutorial to simply set up your python environment, start processing and visualize 3D point cloud data. Highlights Anaconda, NumPy, Matplotlib and Google Colab.


GLYPTOTEK MUSEUM - 3D Photogrammetry Fun © Florent Poux 2020
3D Python
Series
Powered by FLYVAST

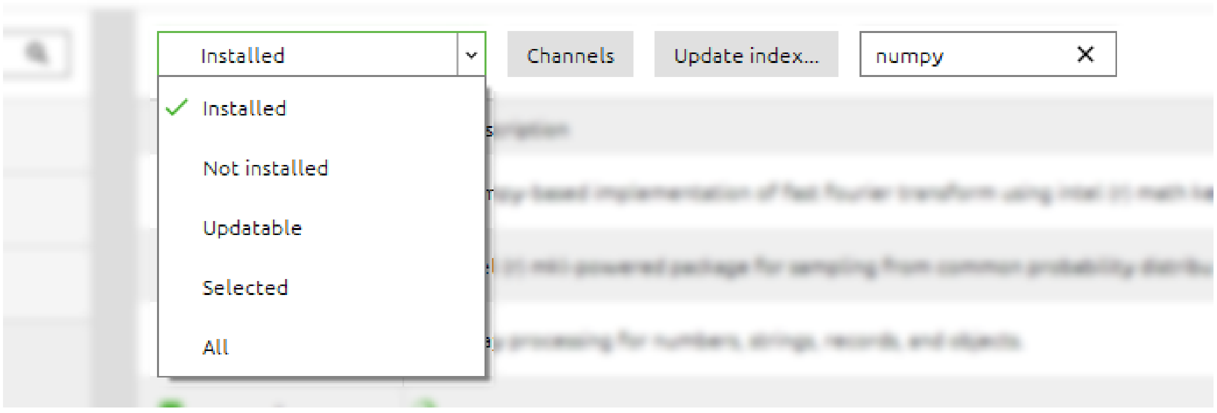
5-Step Guide to set-up your python environment

- We need to set-up our environment. I recommend to download Anaconda Navigator, which comes with an easy GUI.
- Once downloaded and installed, create an environment (2nd tab on the left > Create button), which allows you to specify a python version (the latest is fine). You know it is selected when the green arrow is next to it.



Anaconda GUI—Manage independent projects through environments with different libraries / Python versions

- Once created, you can then link wanted libraries without conflicts. Very handy! For this, just search packages in what is installed (E.g. NumPy, Matplotlib), and if it is not popping, then select Not installed, check them both and click on Apply to install them. Numpy and Matplotlib are standard libraries that will be useful for this and future projects.



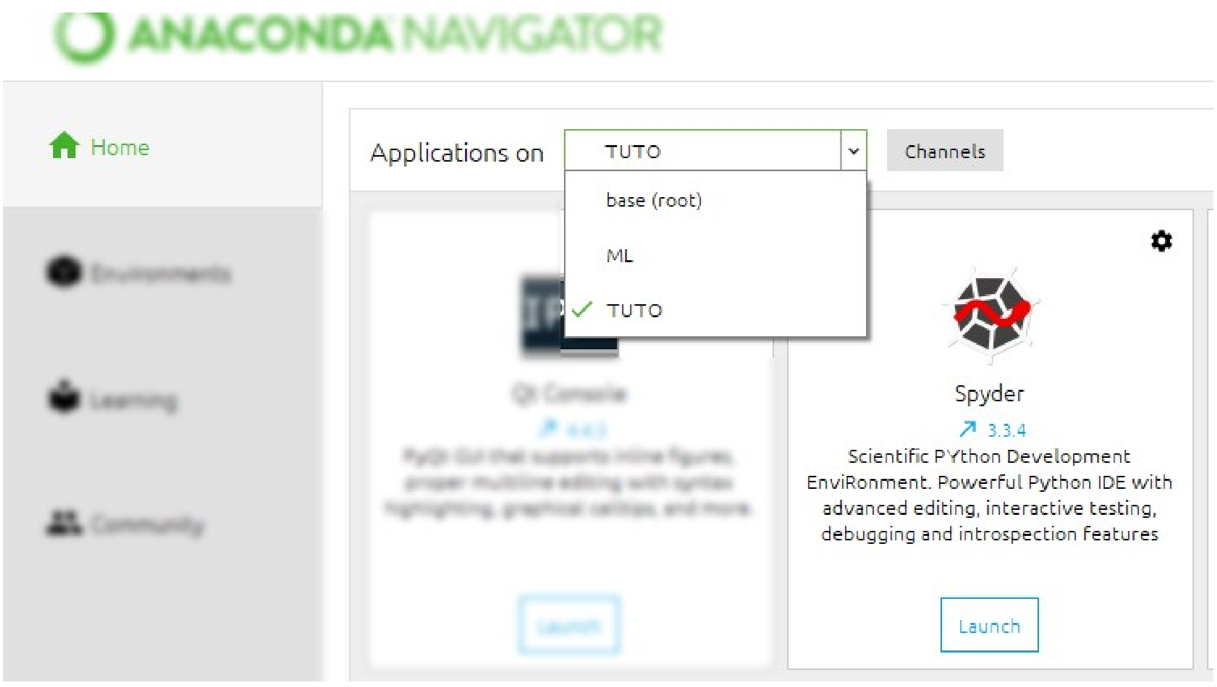
Searching for libraries in the conda library package manager

- You are almost set-up, now back to the Anaconda Home Tab, make sure you are in the right environment (Applications on XXX), then you can install Spyder as the IDE (Integrated Development Environment)to start your code project.

🤓 ***Note****: Spyder is one of the best tools for any amateur who is new to python coding. Another great tool is Jupyter, which does a great job of presenting interactive code to higher management for better*

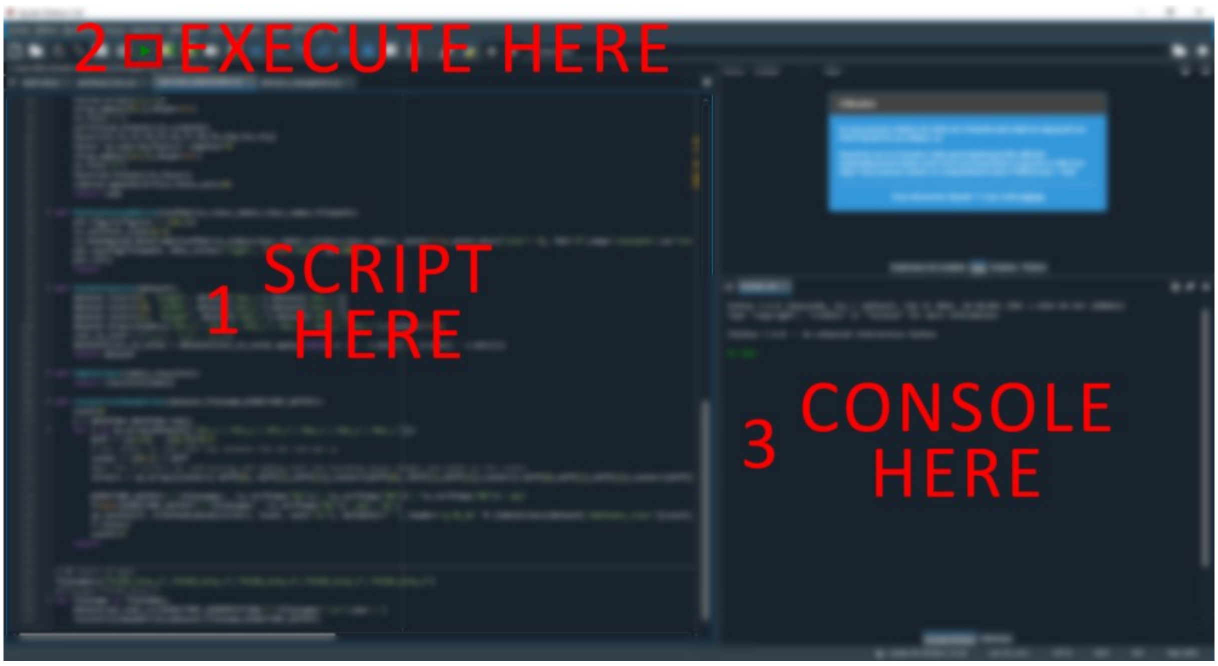
Published in Towards Data Science

visualization. We will explore this as well as Google Colab on future posts.



Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

- Once the installation progress bar is done you are ready! You can now launch Spyder. While the GUI may allow several possibilities, to directly obtain results, first write your script (1), execute your script (2) and explore + interact with results in the console (3).



The 3 basic elements of the Spyder Python interface

Download your point cloud dataset

Published in Towards Data Science

For future experiments, we will use a sampled point cloud that you can freely download from this repository. If you want to visualize it beforehand without installing anything, you can check the webGL version.

3D Point Cloud of a statue from the Glyptotek Museum in Copenhagen, reconstructed with Photogrammetry. The acquisition of the dataset extends the scope of the article and will be covered in a separate post. If you want to learn such techniques, you can visit the Geodata Academy and 3D Reconstructor.

Your first lines of code

In Sypder, let us start by using a very powerful library: NumPy. You can already write your first shortcode in the script area (left window):

```
1 import numpy as np
2 file_data_path=”E:\sample.xyz”
3 point_cloud= np.loadtxt(file_data_path, skiprows=1,
max_rows=1000000)
```

This shortcode (1) import the library NumPy for further use as a short name “np”; (2) create a variable that holds the string pointing to the file that contains the points; (3) import the point cloud as a variable named point_cloud, skipping the first row (holding, for example, the number of points), and setting a maximal number of rows to run tests without memory shortages.

You can now run your script (green arrow), and save it as a .py file on your hard-drive when the pop-up appears. You can now access the first point of the entity that holds your data (point_cloud) by directly writing in the console:

```
In: point_cloud[0]
```

You will then get an array containing the content of the first point, in this case, X, Y and Z coordinates.

```
Out: array([0.480, 1.636, 1.085])
```

These were your first steps with python and point clouds. Now that you know how to load point data, let us look at some interesting processes.

Extracting desired attributes

We have a point cloud with 6 attributes: X, Y, Z, R, G, B. It is important to note that when playing with NumPy arrays, the indexes always start at 0. So, if loaded without field names, then getting the second point is done doing:

```
In: point_cloud[1]
Out: array([0.480, 1.636, 1.085, 25, 44, 68])
```

If from there we want to obtain the Red (R) attribute (the NumPy "column" index is 3), we can do:

```
In: point_cloud[1][3]
Out: 25
```

If we want to extract the Z attribute for all the points in the point cloud:

```
In: point_cloud[:,1]
Out: array([2.703, 2.716, 2.712, …, 2.759, 2.741, 2.767])
```

If we want to extract only X, Y, Z attributes for all the points:

```
In: point_cloud[:,:3]
Out: array([[4.933, 2.703, 2.194],
[4.908, 2.716, 2.178],
[4.92 , 2.712, 2.175],
…,
[5.203, 2.759, 0.335],
[5.211, 2.741, 0.399],
[5.191, 2.767, 0.279]])
```

Congratulations, you just played around with multi-dimensional indexing 👏. Note that in the example above :3], the third column (R) is excluded from the selection. Each result can be stored in variables if they are meant to be used more than one time:

```
xyz=point_cloud[:,:3]
rgb=point_cloud[:,3:]
```

Attribute-based data analysis

Now let us look at some useful analysis. If you want to know the mean height of your point cloud, then you can easily do:

```
In: np.mean(point_cloud,axis=0)[2]
Out: 2.6785763
```

💡 ***Hint:*** *here, the axis set to 0 is asking to look at each "column" independently. If ignored, then the mean run on all the values, and if set to 1, it will average per row.*

If now you want to extract points which are within a buffer of 1 meter from the mean height (we assume the value is stored in mean_Z):

```
In: point_cloud[abs( point_cloud[:,2]-mean_Z)<1]
Out: array([…])
```

💡 ***Hint:*** *In python, and programming in general, there is more than one way to solve a problem. The provided is a very short and efficient way, which may not be the most intuitive. Trying to solve it using a for loop is a great exercise. The aim is a good balance between clarity and efficiency, see the PEP-8 guidelines.*

Now you know how to set-up your environment, use Python, Spyder GUI and NumPy for your coding endeavors. You can load point clouds and play with attributes, but you can try other scenarios such as color filtering, point proximities …

Basic 3D Visualisation

For this first 3D point clouds plotting experience, we will get our hands-on one essential library: Matplotlib. First, we will add packages in the import section of the initial script, to allow us to use them:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
```

The imported dependencies act as following:

- numpy you already know this one 😉
- matplotlib.pyplot handles the plotting
- mpl_toolkits.mplot3d allows to create a 3d projection

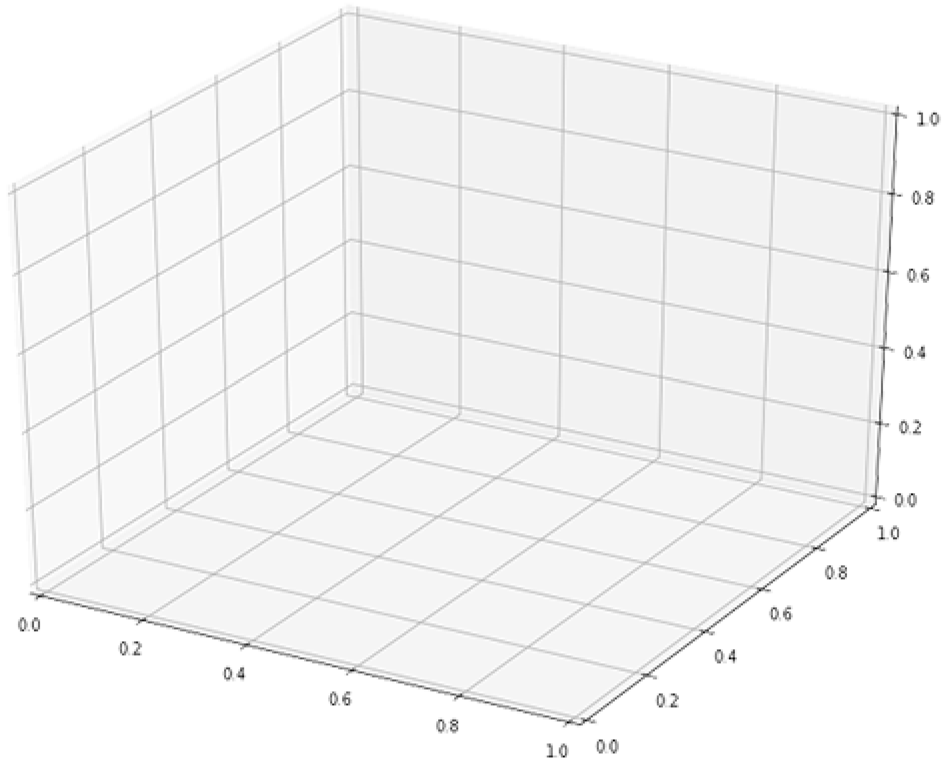
From there, we can add the code that we explored before:

```
file_data_path=”E:\sample.xyz”
point_cloud= np.loadtxt(file_data_path,skiprows=1)
mean_Z=np.mean(point_cloud,axis=0)[2]
spatial_query=point_cloud[abs( point_cloud[:,2]-mean_Z)<1]
xyz=spatial_query[:,:3]
rgb=spatial_query[:,3:]
```

🤓 ***Note****: We usually try to enhance readability when coding and the [PEP-8](#) is a great resource if you want nice guidelines. Ideally, variable names should be lowercase, with words separated by underscores.*

What you get is 4 variables (xyz and rgb), based on the spatial query filtering the initial point cloud. Let us now dive into how to simply plot the results. To begin with, let’s create a 3d axes object. For this, we pass projection='3d' to plt.axes, which returns an Axes3DSubplot object.

```
ax = plt.axes(projection='3d')
```



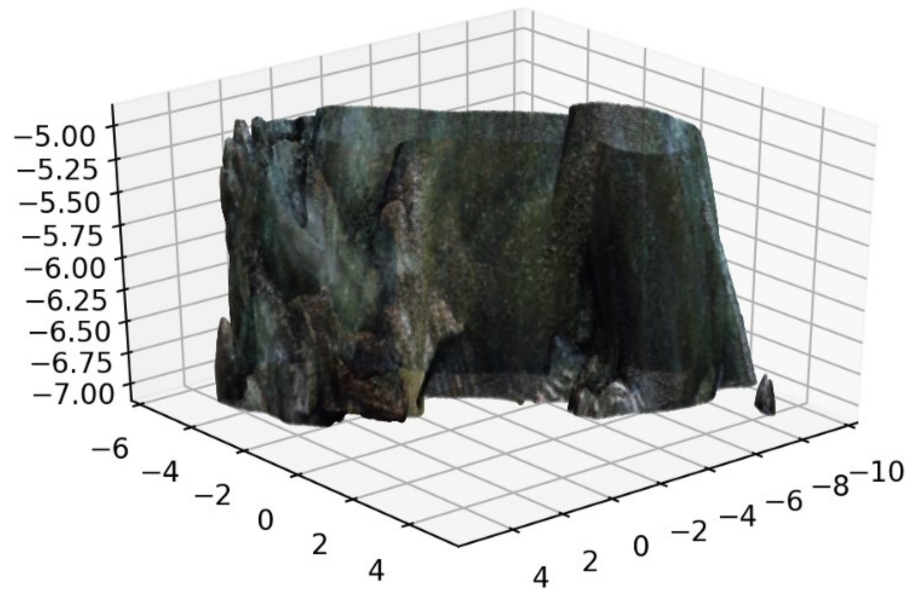
This is the empty canvas that we will be painting on. We will then draw a scatter plot and we want to assign to each point a color. This can be quite a tedious process, that is simplified by the following line of code

```
ax.scatter(xyz[:,0], xyz[:,1], xyz[:,2], c = rgb/255, s=0.01)
```

As you can see, we pass to the function the coordinates x, y, z through xyz[:,0], xyz[:,1], xyz[:,2]; the colors for each point throughc = rgb/255, by normalizing to a [0:1] interval dividing by the highest value; the on-screen size of each point through s = 0.01.

Finally, we can plot the graph with the command below and enjoy the visualization:

```
plt.show()
```



Hint*: in the Spyder 4.1.2 version and above, you can access your plots in the graph tab of the variable explorer window, which creates images by default. If you want to create an interactive visualization, before launching the script, type* %matplotlib auto *to switch to "automatic" (i.e. interactive) plots.*

The full script is available here, and can be remotely executed on Google Colab.

Published in Towards Data Science

Conclusion

You just learned how to import, process and visualize a point cloud composed of millions of points, with as little as 12 lines of code! Well done 👏. But the path does not end here, and future articles will dive deeper in point cloud spatial analysis, file formats, data structures, visualization, animation and meshing. We will especially look into how to manage big point cloud data as defined in the article below.

References

1. Poux, F. The Smart Point Cloud: Structuring 3D intelligent point data, Liège, 2019.
2. Poux, F.; Valembois, Q.; Mattes, C.; Kobbelt, L.; Billen, R. Initial User-Centered Design of a Virtual Reality Heritage System: Applications for Digital Tourism. *Remote Sens.* **2020**, *12*, 2583, doi:10.3390/rs12162583.
3. Poux, F.; Neuville, R.; Nys, G.-A.; Billen, R. 3D Point Cloud Semantic Modelling: Integrated Framework for Indoor Spaces and Furniture. *Remote Sens.* **2018**, *10*, 1412, doi:10.3390/rs10091412.
4. Billen, R.; Jonlet, B.; Luczfalvy Jancsó, A.; Neuville, R.; Nys, G.-A.; Poux, F.; Van Ruymbeke, M.; Piavaux, M.; Hallot, P. La transition numérique dans le domaine du patrimoine bâti: un retour d'expériences. *Bull. la Comm. R. des Monum. sites Fouill. 30* **2018**, 119–148.
5. Poux, F.; Billen, R. Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS Int. J. Geo-Information* **2019**, *8*, doi:10.3390/ijgi8050213.
6. Kharroubi, A.; Hajji, R.; Billen, R.; Poux, F. Classification And Integration Of Massive 3d Points Clouds In A Virtual Reality (VR) Environment. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *42*, 165–171, doi:10.5194/isprs-archives-XLII-2-W17-165-2019.
7. Bassier, M.; Vergauwen, M.; Poux, F. Point Cloud vs. Mesh Features for Building Interior Classification. *Remote Sens.* **2020**, *12*, 2224, doi:10.3390/rs12142224.
8. Poux, F.; Ponciano, J. J. Self-Learning Ontology For Instance

Segmentation Of 3d Indoor Point Cloud. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*; ISPRS, Ed.; Copernicus Publications: Nice, 2020; Vol. XLIII, pp. 309–316.

9. Poux, F.; Mattes, C.; Kobbelt, L. Unsupervised segmentation of indoor 3D point cloud: application to object-based classification. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*; 2020; Vol. XLIV–4, pp. 111–118.

10. Poux, F.; Billen, R.; Kaspryzk, J.-P.; Lefebvre, P.-H.; Hallot, P. A Built Heritage Information System Based on Point Cloud Data: HIS-PC. *ISPRS Int. J. Geo-Information* **2020**, *9*, 588, doi:10.3390/ijgi9100588.

11. Poux, F.; Billen, R. A Smart Point Cloud Infrastructure for intelligent environments. In *Laser scanning: an emerging technology in structural engineering*; Lindenbergh, R., Belen, R., Eds.; ISPRS Book Series; Taylor & Francis Group/CRC Press: London, United States, 2019; pp. 127–149 ISBN in generation.

12. Tabkha, A.; Hajji, R.; Billen, R.; Poux, F. Semantic Enrichment Of Point Cloud By Automatic Extraction And Enhancement Of 360° Panoramas. *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *XLII-2/W17*, 355–362, doi:10.5194/isprs-archives-XLII-2-W17-355-2019.

13. Poux, F.; Neuville, R.; Hallot, P.; Van Wersch, L.; Jancsó, A. L.; Billen, R. Digital investigations of an archaeological smart point cloud: A real time web-based platform to manage the visualisation of semantical queries. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch.* **2017**, *XLII-5/W1*, 581–588, doi:10.5194/isprs-Archives-XLII-5-W1-581-2017.

14. Poux, F.; Hallot, P.; Jonlet, B.; Carre, C.; Billen, R. Segmentation semi-automatique pour le traitement de données 3D denses: application au patrimoine architectural. *XYZ* **2014**, *141*, 69–75.

15. Novel, C.; Keriven, R.; Poux, F.; Graindorge, P. Comparing Aerial Photogrammetry and 3D Laser Scanning Methods for Creating 3D Models of Complex Objects. In *Capturing Reality Forum*; Bentley Systems: Salzburg, 2015; p. 15.