How to represent 3D Data?

A visual guide to help choose data representations among 3D point clouds, meshes, parametric models, depth-maps, RGB-D, multi-view images, voxels…

Different data representation of a 3D point cloud dataset

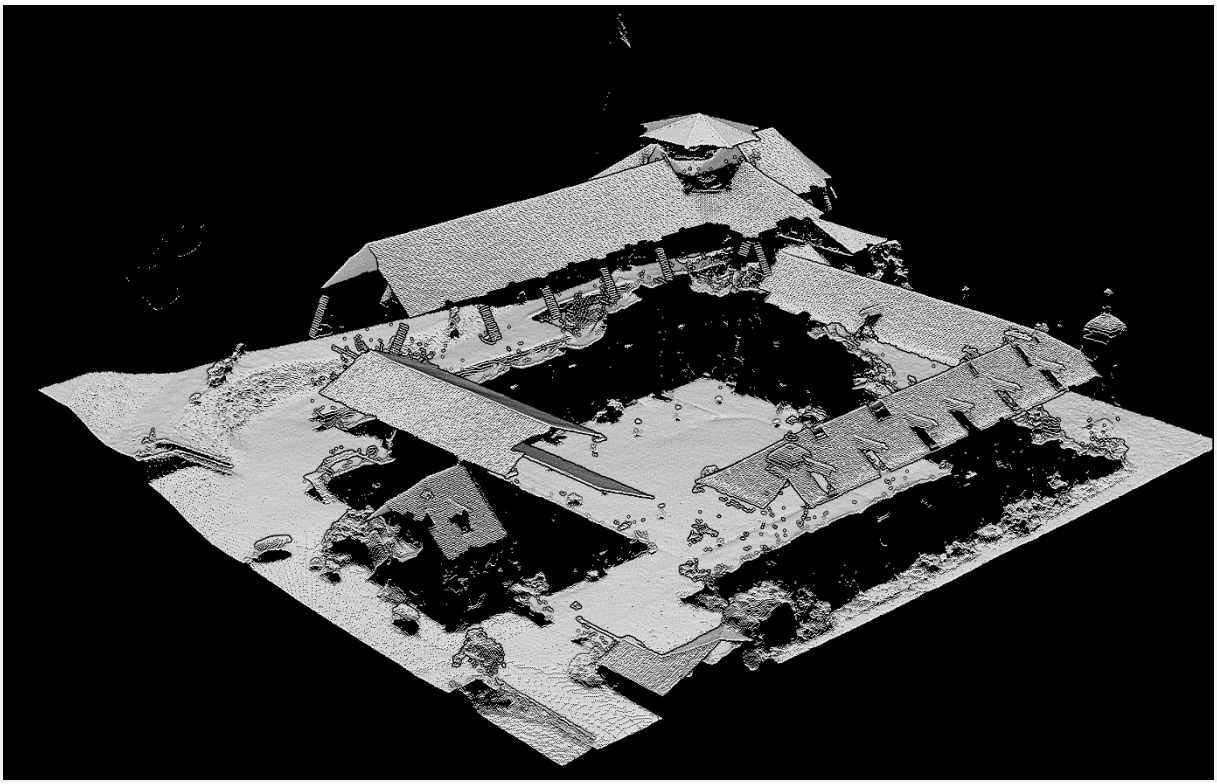
The 3D datasets in our computerized ecosystem — of which an increasing number comes directly from reality capture devices — are found in different forms that vary in both the structure and the properties. Interestingly, they can be somehow mapped with success to point clouds thanks to its canonical nature. This article gives you the main 3D data representations modes to choose from when bindings point clouds to your application.

3D Point Clouds

Published in Towards Data Science

A 3D Point Cloud of an Abbey acquired in 2014 using photogrammetry (Gerpho), next to my hometown in the South of France 🙂. The Resolution is 1 cm, expressed as the Ground Sampling Distance.

A point cloud is a set of data points in a three-dimensional coordinate system. These points are spatially defined by `X`, `Y`, `Z` coordinates and often represent the envelope of an object. Reality capture devices obtain the external surface in its three dimensions to generate the point cloud. These are commonly obtained through Photogrammetry (example above), LiDAR (Terrestrial Laser Scanning, Mobile Mapping, Aerial LiDAR as simulated below), depth sensing, and more recently deep learning through Generative Adversarial Networks.



An aerial LiDAR simulated point cloud. See how this is mostly 2.5D from top-down sensing.

Each technique holds several specificities influencing the quality and completeness of the data, and you can already see the difference between a full 360° capture vs a classical aerial LiDAR acquisition. This extends the scope of this specific article and will be covered in another issue.

Point clouds provide simple yet efficient 3D data representations, and I summarize below the main operations, benefits, and disadvantages that come with them.

Main Operations

- Transformations: You can multiply the points in the point list with linear transformation matrices.

- Combinations: “Objects” can be combined by merging points list together.
- Rendering: Projects and draws the points onto an image plane

Main Benefits

- Fast rendering
- Exact representation
- Fast transformations

Main Disadvantages

- Numerous points (obj. curve, exact representation)
- High memory consumption
- Limited combination operations

While fast rendering and transformations make a direct inspection of a point cloud handy, they often are not directly integrated into commonly used three-dimensional applications. However, recent developments show a trend for better support even within pure mesh-based rendering platforms with a recent example within the Unreal 4 game engine.

A common process is to derive a mesh using a suitable surface reconstruction technique. There are several techniques for

transforming point cloud into a three-dimensional explicit surface, some of which are covered in the article below.

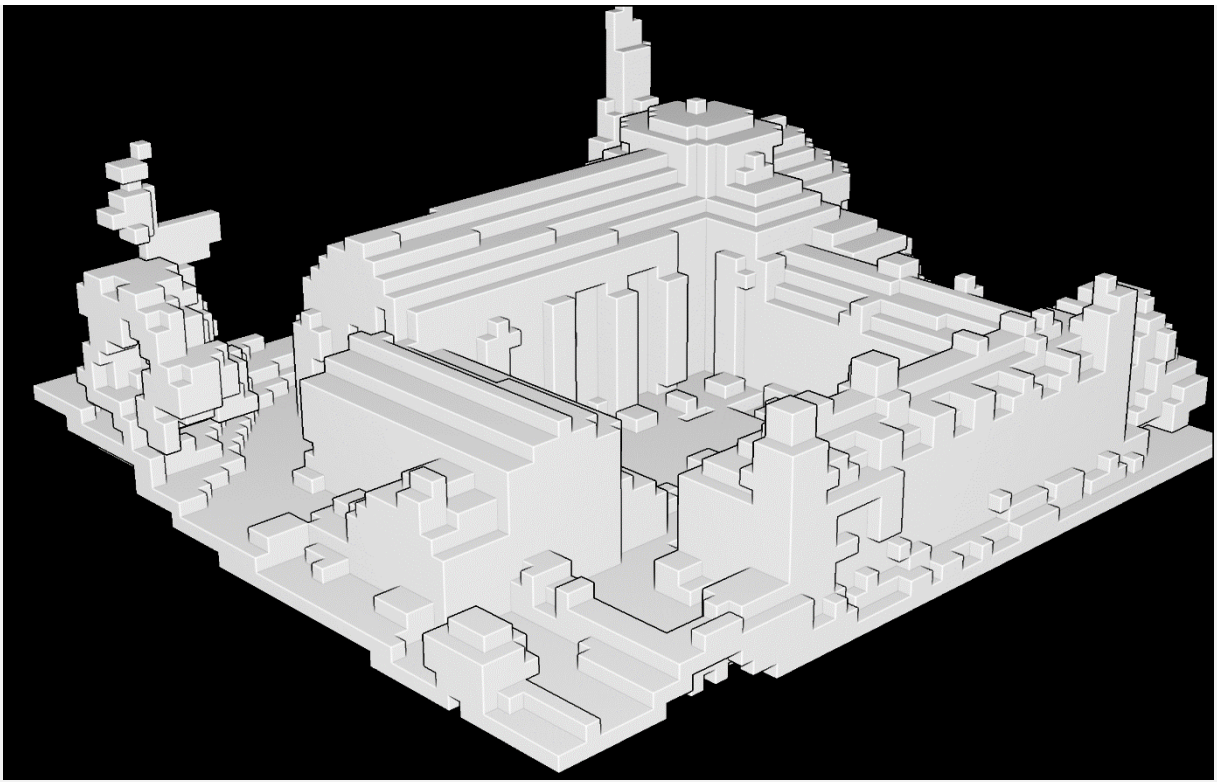
5-Step Guide to generate 3D meshes from point clouds with Python
Tutorial to generate 3D meshes (.obj, .ply, .stl, .gltf) automatically from 3D point clouds using python. (Bonus)…
towardsdatascience.com

Let us further dive into 3D models as a representation to better grasp the range of possibilities.

3D Models

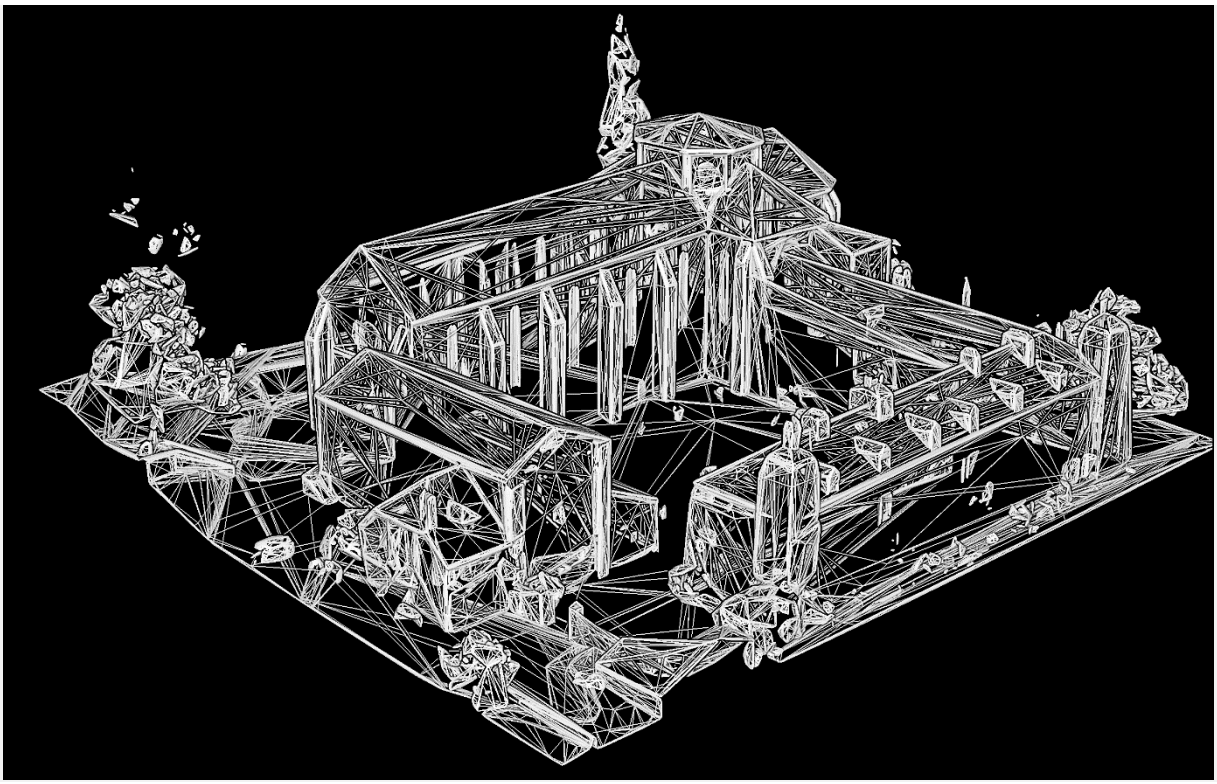
Almost all 3D models can be divided into two categories.

- **Solid**: These models define the volume of the object they represent. Solid models are mostly used for engineering and medical simulations and are usually built with Constructive Solid Geometry or voxels assemblies.



Example of a solid 3D model through voxelization.

- **Shell** or **boundary** (B-Reps): These models represent the surface, i.e. the boundary of the object, not its volume. Almost all visual models used in reality capture workflows, games and film are boundary representations.



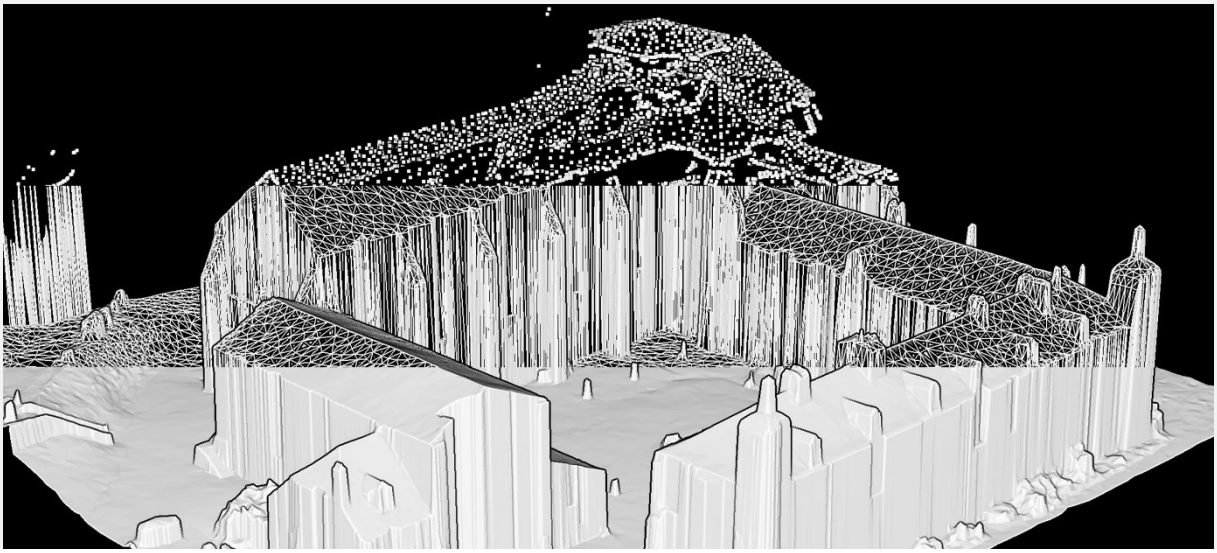
Example of a shell representation of the Abbey.

Solid and shell modeling can create functionally identical objects. Differences between them are mostly variations in the way they are created and edited and conventions of use in various fields and differences in types of approximations between the model and reality.

Three main strategies permit to describe a point cloud through a 3D models. Constructive Solid Geometry, Implicit surfaces (+Parametric modeling), and Boundary representations (B-Reps). While Constructive Solid Geometry is very interesting and will be shortly discussed, the most common 3D models are B-Reps as 3D meshes. Let us first extend on theses.

3D Mesh

A mesh is a geometric data structure that allows the representation of surface subdivisions by a set of polygons. Meshes are particularly used in computer graphics, to represent surfaces, or in modeling, to discretize a continuous or implicit surface. A mesh is made up of vertices (or vertex), connected by edges making faces (or facets) of a polygonal shape. When all faces are triangles, we speak of triangular meshing. These are the most common in Reality Capture workflows.

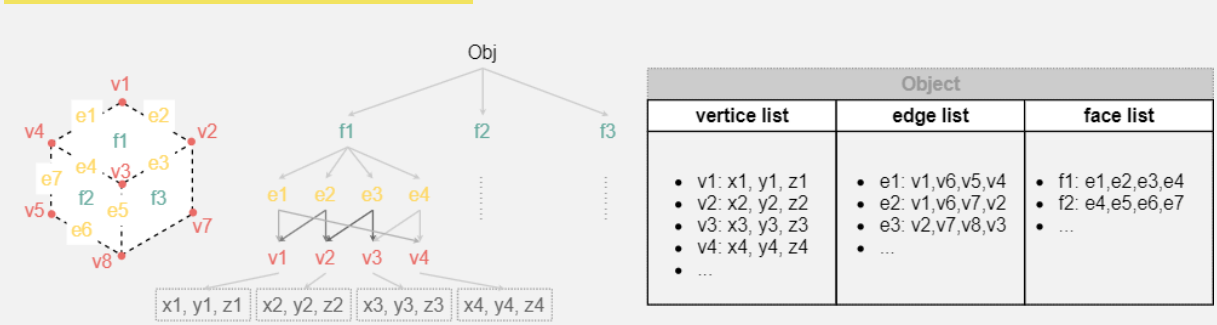


From top to bottom: The vertices of the mesh; the edges linking vertices together; the faces filling formed by vertices and edges, mostly triangular.

Quadrilateral meshes are also very interesting but often obtained through mesh optimizations techniques to get more compact representations. It is also possible to use volumetric meshes, which connect the vertices by tetrahedrons, hexahedrons (cuboids), and prisms. These so-called meshes are based on the boundary representation, which depends on the wire-frame model (The object is simplified by 3D lines, each edge of the object is represented by a line in the model). Let us extend the theory.

Boundary Representation

The Boundary representation of 3D models is mainly composed of two parts: the topology (organization of elements) and the geometry (surfaces, curves, and points). The main topological items are faces, edges, and vertices and I schematized below a simple B-Rep for a cube.



The schematization of the boundary representation underlying structure

Operations

- Transformations: All points are transformed as with the wire-frame model (Multiply the points in the point list with linear matrices), besides, the surface equations or normal vectors can be transformed.
- Combinations: Objects can be combined by grouping point lists and edges to each other; Operations on polygons (Divide based on intersections, Remove the redundant polygons, Combine them ...
- Rendering: Hidden surface or line algorithms can be used because the surfaces of the objects are known so that visibility can be calculated.

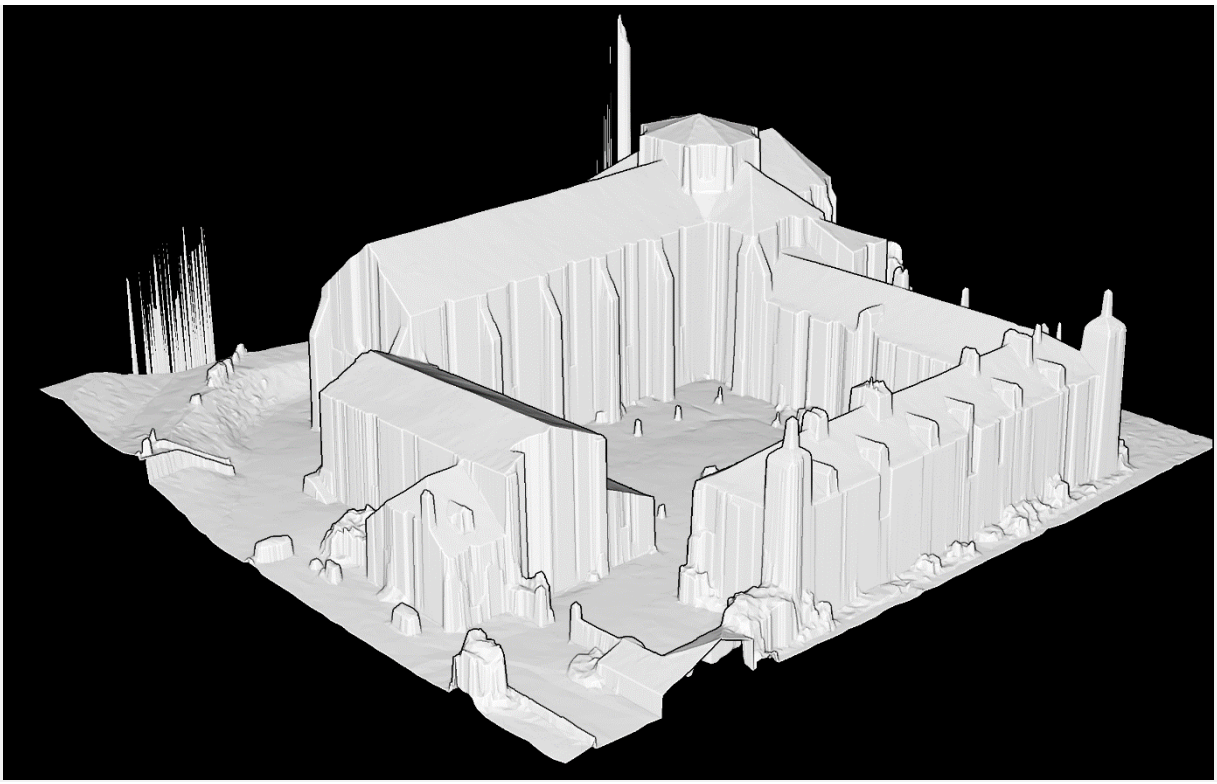
Benefits

- Well-adopted representation
- Model generation via “new-gen scanning”
- Transformations are quick and easy

Disadvantages

- High memory requirements
- Expensive combinations
- Curved objects are approximated

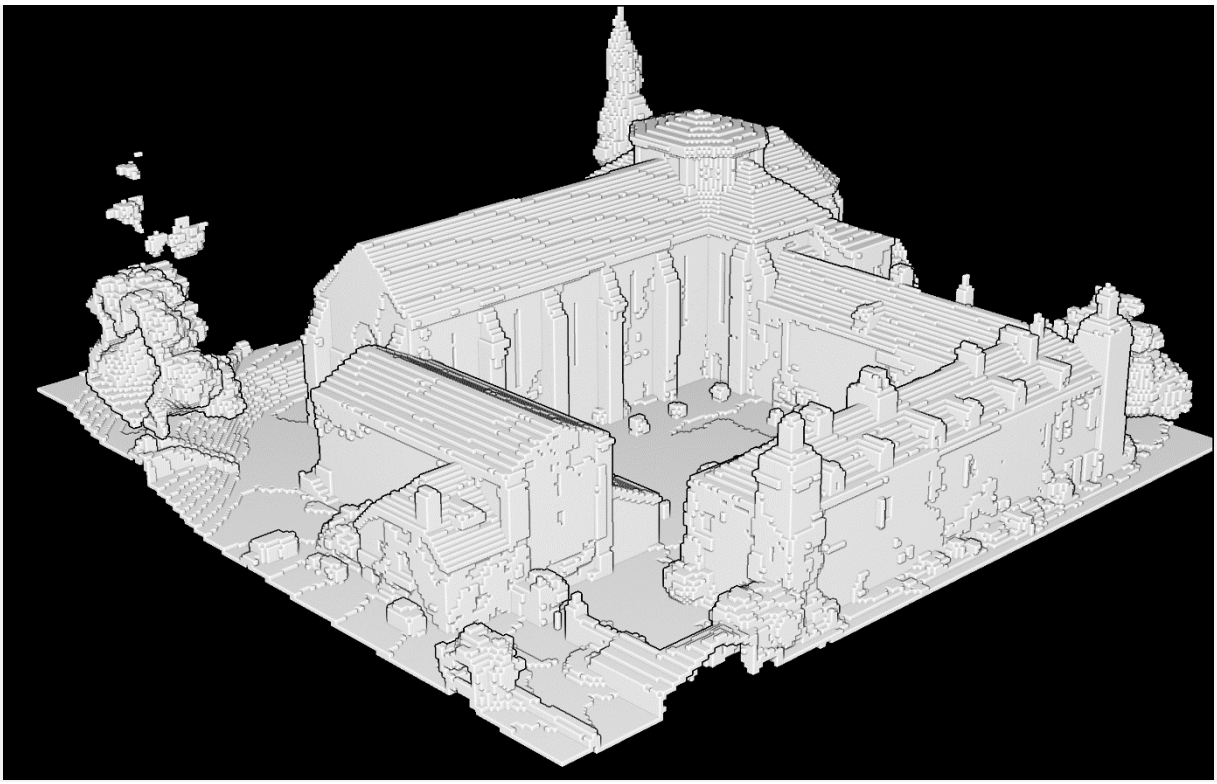
Meshes are a great way to explicit the geometry of a point cloud, and often permits to widely reduce the number of needed points as vertices. On top, it permits to get a sense of the relationship between objects through the faces connectivities. However, meshing is an interpolation of the base point cloud geometry, and can only represent the data to a certain degree, linked to the complexity of the mesh. There exist a multitude of strategies to best mesh a point cloud, but this often demands to have some theoretical background and to know which parameter’s to adjust for an optimal result.



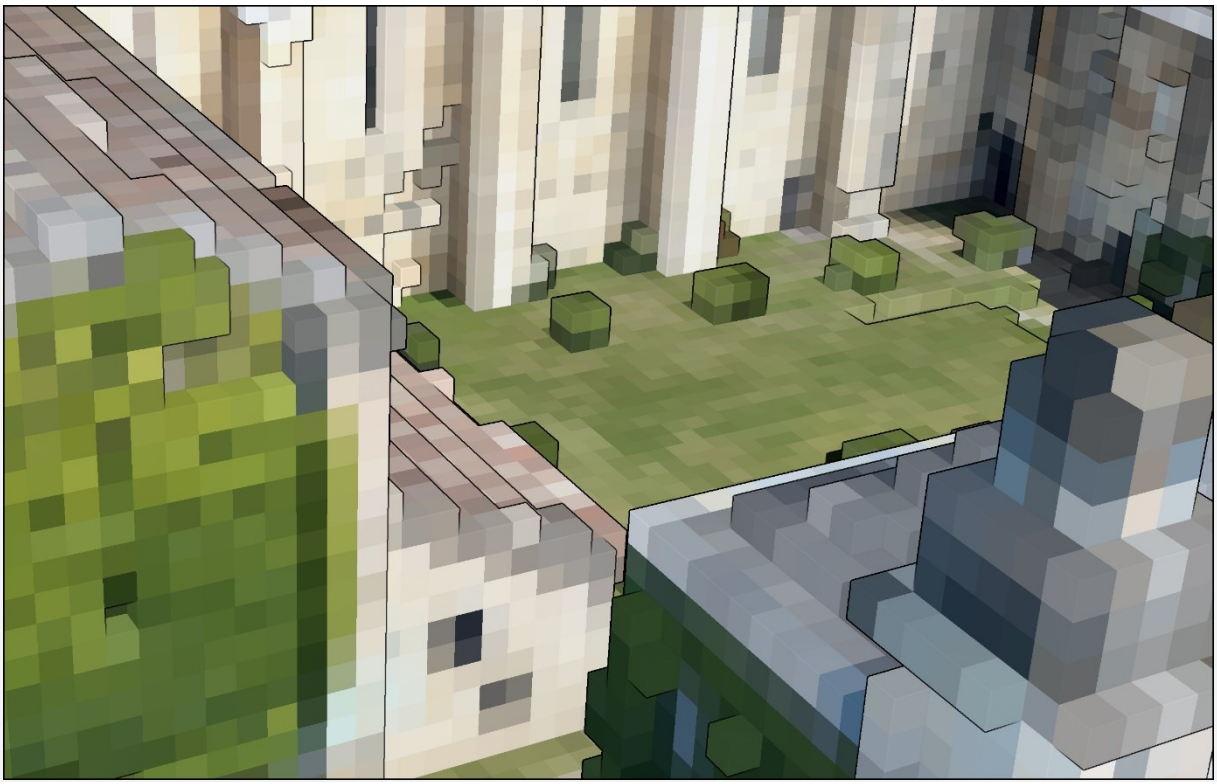
Example of a 2.5D Delaunay triangulation of the point cloud.

Voxel-based models

A voxel can be seen as a 3D base cubical unit that can be used to represent 3D models. Its 2D analogy is the pixel, the smallest raster unit. As such, a voxel-based model is a discretized assembly of “3D pixels”, and is most often associated with solid modeling.



In the case of point cloud data, one can represent each point as a voxel of size $_x$, to get a “filled” view of empty spaces between points. It is mostly associated with data structures such as octrees, and permit to average a certain amount of points per voxel unit depending on the level of refinement needed (see example in the image below). This is very interesting, and I will cover the theory as well as the implementation in another dedicated article.

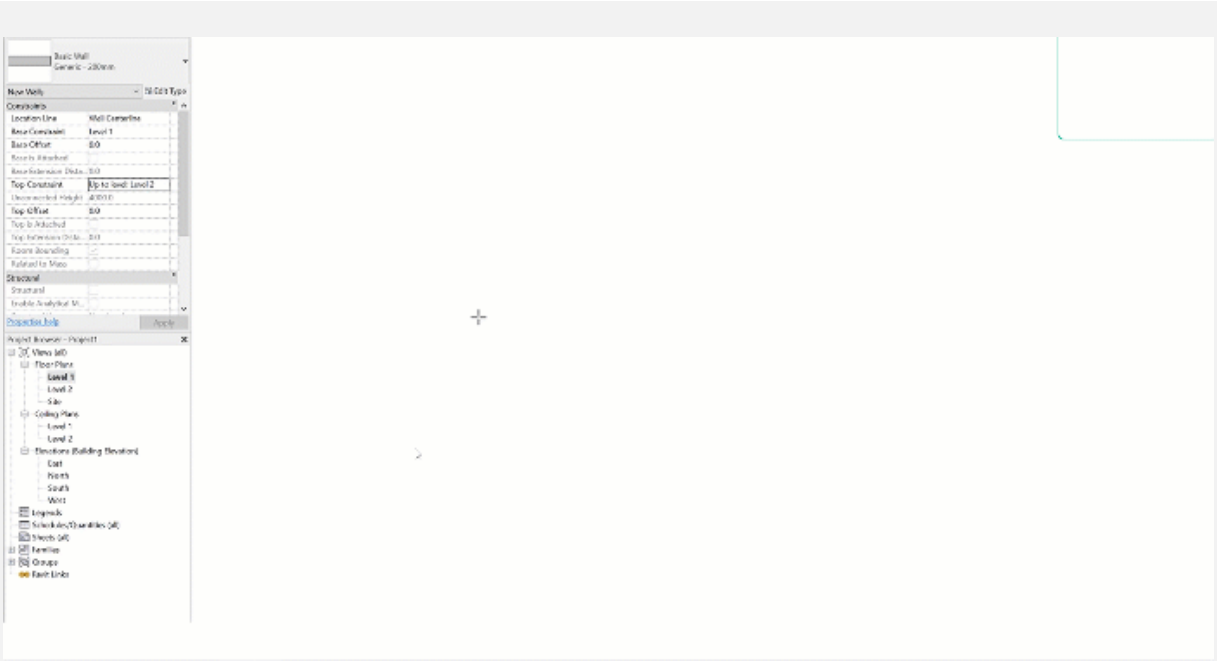


Example of voxel generalization based on an octree subdivision of the space occupied by the point cloud data.

While this is practical for rendering and smooth visualization, it comes to approximating the initial geometry coupled with aliasing artifacts and can give false information if the volume information is used unproperly. However, due to the very structured grid layout of voxel models, it can be very handy for processing tasks such as classification through 3D convolutional neural networks.

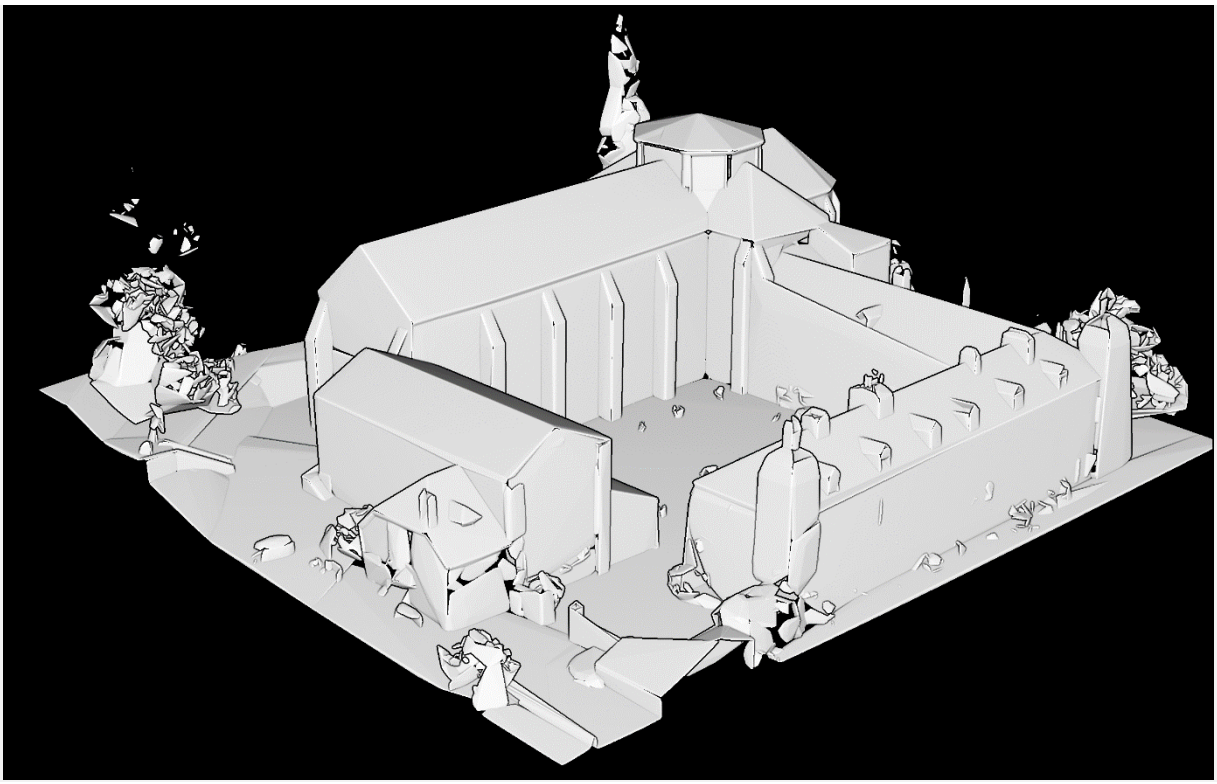
Parametric Model (CAD)

“Parametric” is used to describe a shape’s ability to change by setting a parameter to a targeted value that modifies the underlying geometry. This is e.g. very handy if you want to model “walls” by just setting up their orientation, length, width, and height.



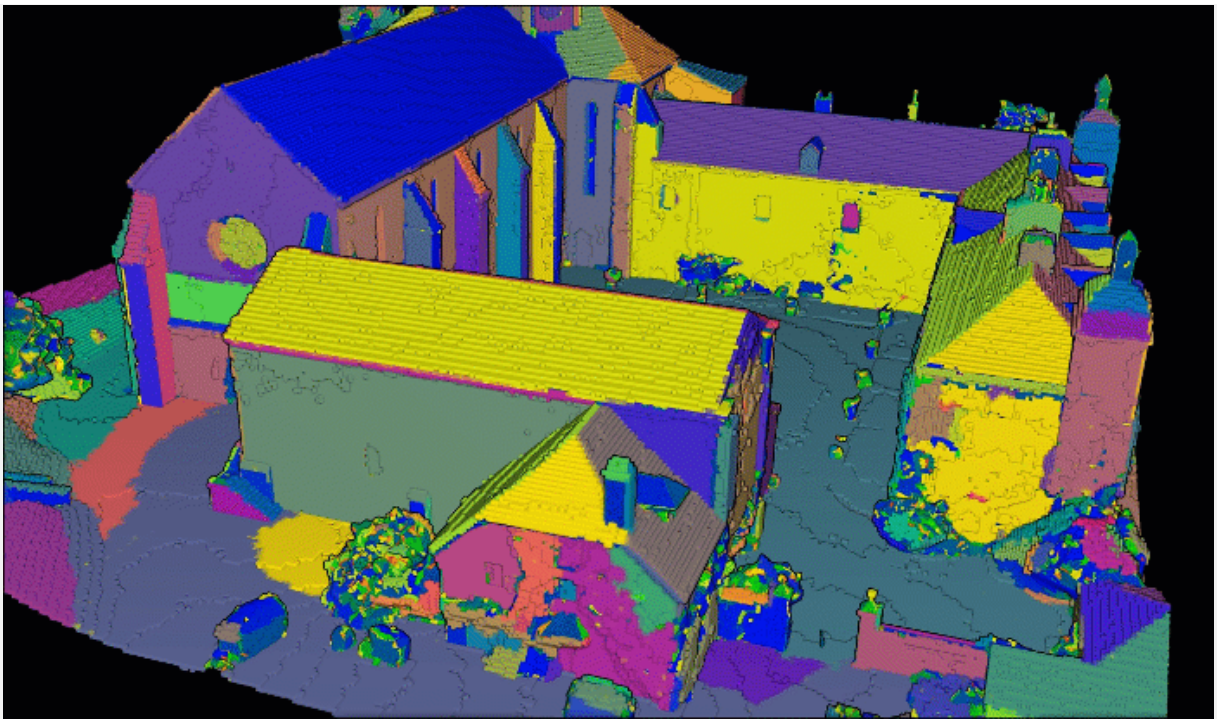
Example of modeling a wall by setting the parameter’s interactively to create a BIM model (Building Information Modelling)

Parametric modeling is then suited to using computing capabilities that can model component attributes with an aim of real-world behavior. Parametric models use a composition of feature-based (parametric, as describe in a later section), solid and surface modeling to allow the manipulation of the model’s attributes.



This is an automatically generated CAD model without a topology fix.

One of the most important features of parametric modeling is that interlinked attributes can automatically change values. In other words, parametric modeling allows defining entire “classes of shapes”, not just specific instances. This however often demands a very “smart” structuration of the underlying point cloud geometry, to decompose the model entity into sub-entities (E.g. segments) that are aggregated in classes.



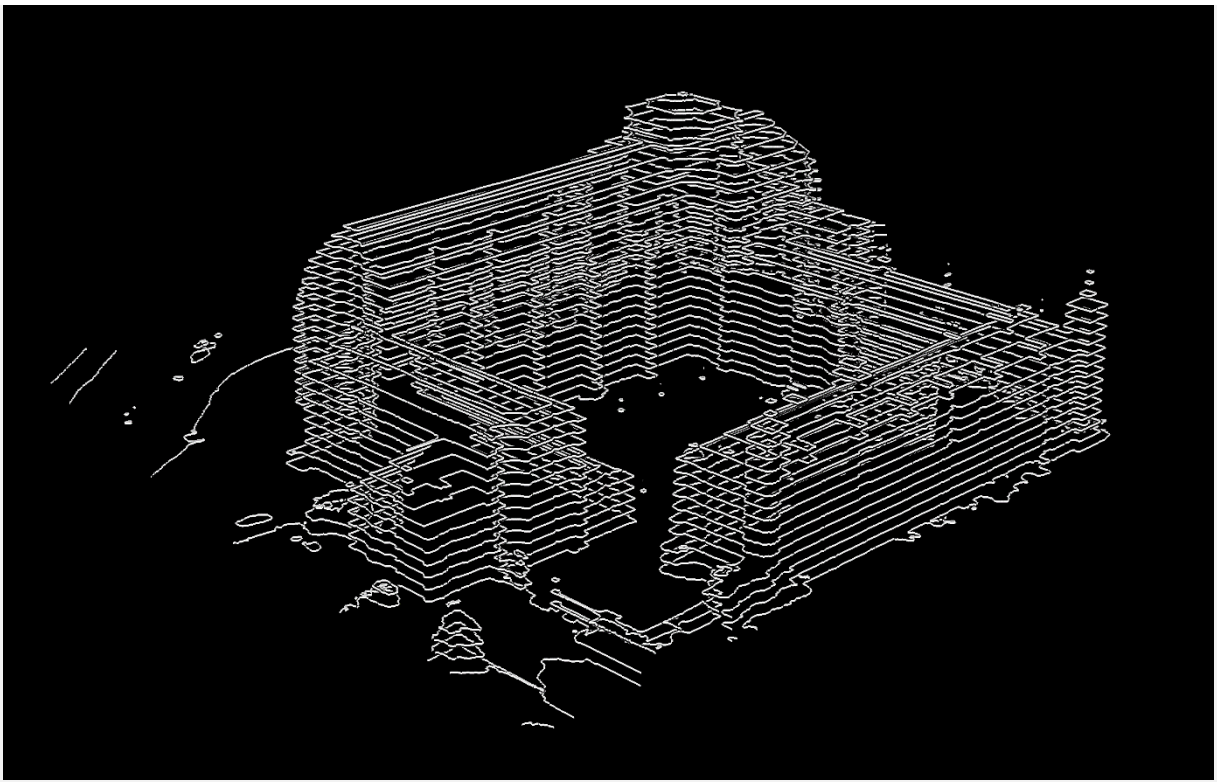
Example of automatic segmentation as described in this award-winning Open Access Article [0]

This process immensely benefits from object detection scenarios and the Smart Point Cloud Infrastructure as defined in the following article.

The Future of 3D Point Clouds: a new perspective

Discrete spatial datasets known as point clouds often lay the groundwork for decision-making applications. But can they…
towardsdatascience.com

Often, these parametric models can also be combined or extracted by combining 2D CAD drawings that interpolate the point cloud shape, and layers it depending on the class of elements.

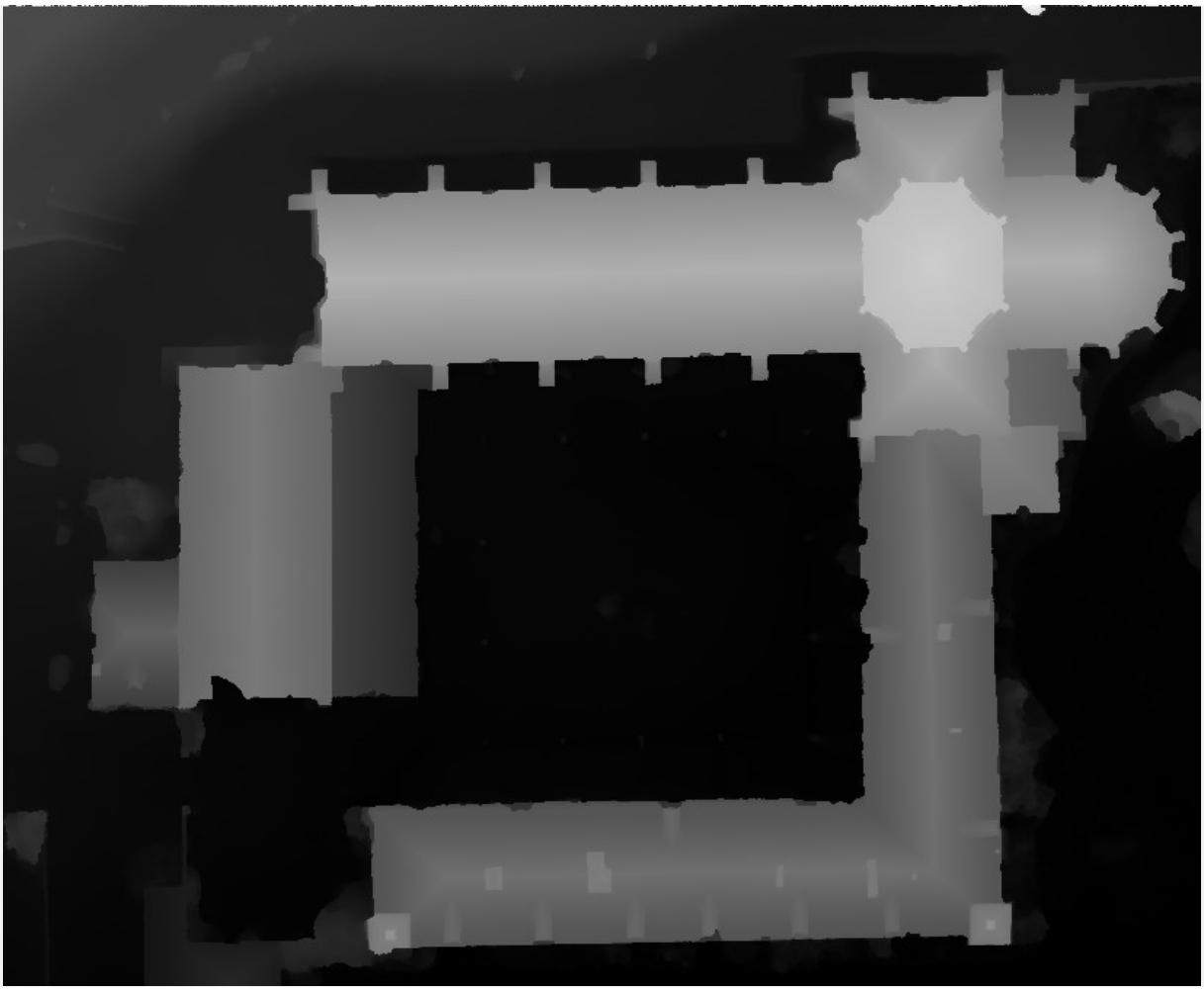


Example of several raw CAD sections from the initial point cloud before layering.

These parametric models are oftentimes consuming to create but are the ones that give the most value to the 3D point cloud data. These come through massive semantic enrichment and additional triggers on the relations between objects constituting the scene.

Depth map

Now, we jump to raster-based point cloud representation. The first one is the depth-map.



Depth-map of the point cloud based on a top-bottom view

A depth map is an image or an “image channel” that contains information relating to the distance of the points constituting the scene from a single viewpoint. While we are used to working with RGB images, the simplest form of expressing the depth is to color-code on one channel, with intensity values. Bright pixels then have the highest value and dark pixels have the lowest values. And that is it. A depth image just presents values according to how far are objects, where pixels color gives the distance from the camera.

💡 ***Hint:*** *The depth map is related to the Z-buffer, where the "Z" relates to the direction of the central axis of view of a camera and not to the absolute Z scene coordinate.*

This form of point cloud representation is fine if you just need surface information linked to a known point of view. This is the case for autonomous driving scenarios where you can very quickly map the environment at each position through a 360 projected depth map. However, the big counterpart is that you are not working with 3D data, rather 2.5D as you cannot represent 2 different values for on line sight. Here are operations, benefits, and disadvantages depth-map come with:

Operations

- Transformations: Multiply the pixels in the image with linear transformation matrices
- Combinations: Objects can be combined by merging the points lists.
- Rendering: Draws pixels on the image plane

Benefits

- Low memory requirements
- Very well know Raster format
- Transformations are quick and easy

Disadvantages

- Essentially a 2.5-D representation
- Cannot describe a full 3D scene on its own
- Weak topology

Special case: RGB–D

Third, representing 3D data as RGB-D images have become popular in recent years thanks to the popularity of RGB-D sensors. RGB-D data provides a 2,5D information about the captured 3D object by attaching the depth map along with 2D color information (RGB).

This is the RGB raster imagery

This is the depth channel associated

Besides being inexpensive, RGB-D data are simple yet effective representations for 3D objects to be used for different tasks such as identity recognition [1], pose regression [2], and correspondence [1]. The number of available RGB-D datasets is huge compared to other 3D datasets such as point clouds or 3D meshes and as such is the preferred way of training deep learning models through extensive training datasets.

Special case: Projections

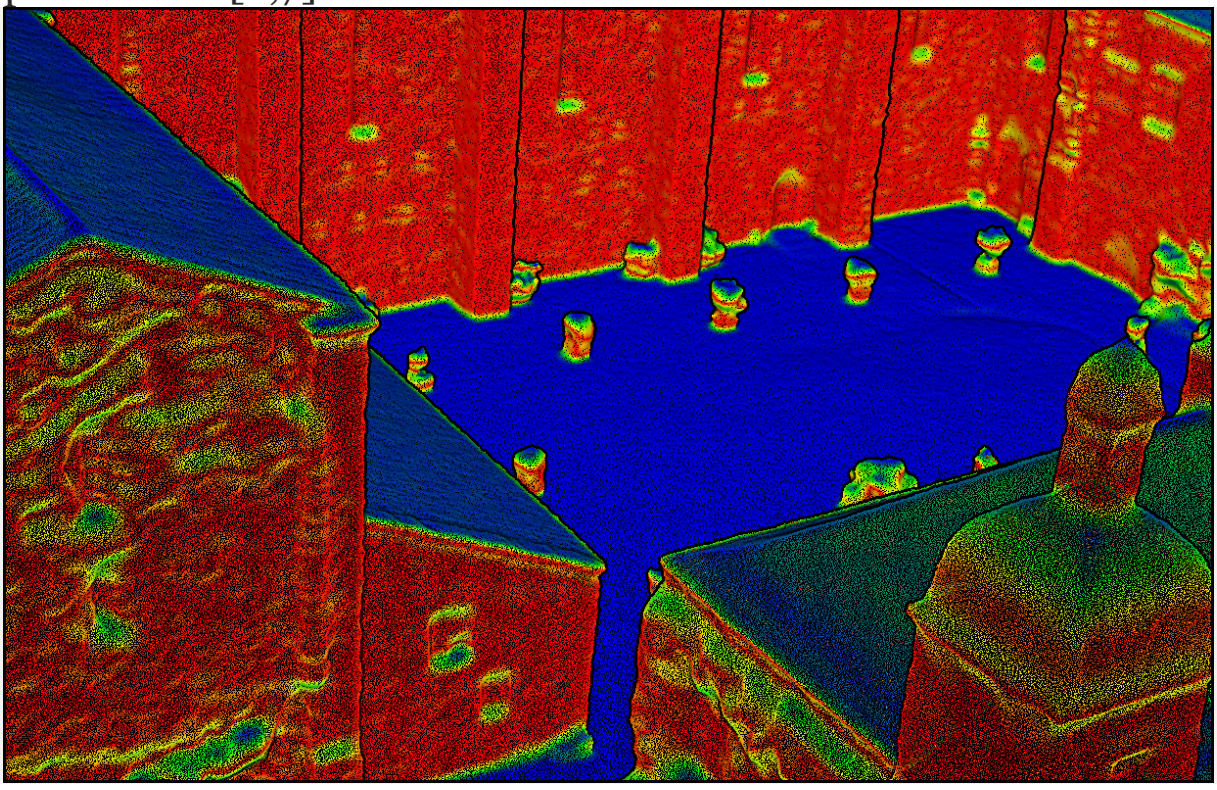
Secondly, projecting 3D data into another 2D space is another representation of raw 3D data where the projected data encapsulates some of the key properties of the original 3D shape [3].

Example of widely deformed cylindrical projection of the point cloud

Multiple projections exist where each of them converts the 3D object into a 2D grid with specific information. Projecting 3D data into the spherical and cylindrical domains (e.g. [4]) has been a common practice for representing the 3D data in such format. Such projections help the projected data to be invariant to rotations around the principal axis of the projection and ease the processing of 3D data due to the Euclidean grid structure of the resulting projections. However, such representations are not optimal for complicated 3D computer vision tasks such as dense correspondence due to the information loss in projection [5].

Implicit representation

Now, we move to what is the lesser visual component of point clouds: implicit representation. It just is a way to represent point clouds by a set of shape descriptors as described in the articles provided in [6,7].



Color-based visualization of the verticality feature extracted to characterize the point cloud

These can be seen as a signature of the 3D shape to provide a compact representation of 3D objects by capturing some key properties to ease processing and computations (E.g. expressed as a.csv file)

```
x      y      z   surface   volume   omn.  ver.
9.9   30.5   265.3   334.5   103.3   4.6   0.0
-27.0   71.6   274.2   18.2   12.5   1.3   0.4
-11.8   48.9   273.8   113.2   620.4   3.7   0.7
26.9   43.8   266.1   297.1   283.6   3.9   0.0
42.9   61.7   273.7   0.1   0.0   0.3   0.8
-23.1   36.5   263.3   26.3   14.8   1.6   0.0
-9.5   73.1   268.2   24.0   11.4   2.2   0.0
32.2   70.9   284.0   36.0   139.1   1.7   0.8
-20.5   20.7   263.2   34.0   3.4   1.8   0.8
-2.3   73.6   262.2   28.2   15.6   2.6   1.0
```

The nature and the meaning of this signature depend on the characteristic of the shape descriptor used and its definition. For example, global descriptors provide a concise yet informative description for the whole 3D shape while local descriptors provide a more localized representation for smaller patches in the shape. The work of Kazmi et al. [6], Zhang et al. [7] and more recently Rostami et al. [8] provide comprehensive surveys about such 3D shape descriptors.

Implicit representation is very handy as part of a processing pipeline, and to ease data transfer among different infrastructures. It is also very useful for advanced processes that benefit from informative features hard to visually represent.

Multi–View

Fifth, we can access 3D information from a multi-view image, which is a 2D-based 3D representation where one accesses the information by matching several 2D images for the same object from different points of view. Representing 3D data in this manner can lead to learning multiple feature sets to reduce the effect of noise, incompleteness, occlusion, and illumination problems on the captured data. However, the question of how many views are enough to model the 3D shape is still open, and linked to the acquisition methodology for photogrammetric reconstructions: a 3D object with an insufficiently small number of views might not capture the properties of the whole 3D shape (especially for 3D scenes) and might cause an over-fitting problem. Both volumetric and multi-view data are more suitable for analyzing rigid data where the deformations are minimal.

What about Machine Learning and Deep Learning?

3D Data has a tremendous potential for building Machine Learning systems, especially Deep Learning. However, currently, true 3D data representations such as 3D meshes need to be considered regarding another Deep Learning paradigm.

Published in Towards Data Science

Indeed, the vast majority of deep learning is performed on **Euclidean data.** This includes datatypes in the 1-dimensional and 2-dimensional domain. Images, text, audio, and many others are all euclidean data. Of this, particularly the RGB-D datasets are then nowadays able to build on to of massive labeled libraries if one seeks to automatically detect objects in the scene. But meshes or structured point clouds could benefit from exploiting their rich underlying relationships. This is achieved for example by embedding them in a graph structure (a data structure that consists of **nodes** (entities) that are connected with **edges** (relationships)), but this makes them Non-Euclidean (which meshes are by nature), thus non-usable by classical Machine Learning architectures.

For this, an emerging field called **Geometric Deep Learning (GDL)** aims to build neural networks that can learn from non-euclidean data.

As nicely put by a fellow scientist Flawnson Tong in this recommend article (here):

The notion of relationships, connections, and shared properties is a concept that is naturally occurring in humans and nature. Understanding and learning from these connections is something we take for granted. Geometric Deep Learning is significant because it allows us to take advantage of ***data with inherent relationships, connections, and shared properties.***

Published in Towards Data Science

Thus, every 3D Data Representation can be used within a Machine Learning project, but some will be for more experimental projects (non-euclidean representations), whereas euclidean data can directly be ingested in your application

---

Conclusion

If you read up until now, kudos to you ! To summarize, the 3D data representation world is super flexible, and you now have the knowledge to make an informed decision for choosing your data representation:

- 3D Point clouds are simple and efficient but lack connectivity;
- 3D models found as 3D meshes, Parametric models, voxel assemblies propose dedicated levels of additional information but approximate the base data;
- Depth maps are well known and compact but essentially deal with 2.5D data;
- Implicit representation encompasses all of the above but is hardly visual;
- Multi-view is complimentary and leverage Raster imagery but is prone to failure case for optimal viewpoint selection.

And as always, if you want to go beyond, you will find several references below.

References

1. Poux, F.; Neuville, R.; Hallot, P.; Billen, R. MODEL FOR SEMANTICALLY RICH POINT CLOUD DATA. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *IV-4/W5*, 107–115, doi:10.5194/isprs-annals-IV-4-W5-107-2017.

2. Poux, F. The Smart Point Cloud: Structuring 3D intelligent point data, Liège, 2019.

3. Poux, F.; Billen, R. Smart point cloud: Toward an intelligent documentation of our world. In *PCON*; Liège, 2015; p. 11.

4. 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.

5. Poux, F.; Neuville, R.; Van Wersch, L.; Nys, G.-A.; Billen, R. 3D Point Clouds in Archaeology: Advances in Acquisition, Processing and Knowledge Integration Applied to Quasi-Planar Objects. *Geosciences* **2017**, *7*, 96, doi:10.3390/geosciences7040096.

6. 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*, 213, doi:10.3390/ijgi8050213.

7. Poux, F.; Billen, R. Laser Scanning. In *Laser scanning: an emerging technology in structural engineering*; Riveiro, B., Lindenbergh, R., Eds.; ISPRS Book Series; CRC Press: London, UK, 2019; pp. 127–149 ISBN 9781351018869.

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.; 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.

10. 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.

11. 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.

12. Poux, F.; Neuville, R.; Hallot, P.; Billen, R. Point clouds as an efficient multiscale layered spatial representation. In *Eurographics Workshop on Urban Data Modelling and Visualisation*; Vincent, T., Biljecki, F., Eds.; The Eurographics Association: Liège, Belgium, 2016.