

In the previous lesson we discussed how to model 3D objects using the boundary representation. You learned about data structures to represent the geometry and topology of a 3D object's surface in a very structured and organised way. In this lesson we will look at how you could create such a structured representation from a much less structured form of 3D geoinformation, namely a point cloud.

In automatic building reconstruction we aim to construct 3D mesh models for individual buildings from some form of elevation measurements, ie a raster-based DSM or a point cloud, without any manual interventions (see Figure 1.1). It can be considered as one step in the geoinformation chain, since we essentially transform 'raw' and unorganised point measurements into more structured and semantically rich 3D models. Compared to a point cloud, such models are much more useful for applications such as environmental simulations of wind, air pollution, and noise propagation, but also building energy demand estimation and urban planning in general. Many of these applications require knowledge about the volume or surface area of a building, or the distinction between the interior and exterior of a building, which is evidently much easier to derive from a mesh with a clearly defined boundary than from a point cloud. In addition, meshes are typically more compact which makes them more efficient to store and process.

This is not to say that meshes are always superior to point clouds. For example some of the finer details that may be present in a point cloud could be lost in the mesh representation. Furthermore, there is always the risk of introducing new errors and deviations from the original measurement in the building reconstruction process. But ultimately, the many benefits of representing a building as a mesh outweigh these disadvantages for many applications.

In this lesson we will first list common challenges and requirements for the building models that are to be constructed. Second, we will look at the important engineering choices in designing a building reconstruction algorithm. And finally, we will discuss one particular approach that was designed to work on Dutch open data in more detail.

1.1 Building model requirements and reconstruction challenges

When designing a building reconstruction method, it is important to carefully consider both the *model requirements* and the *reconstruction challenges*. The model requirements specify in detail what properties a reconstructed building model should have. Model requirements are mostly application dependent. For example, an application that performs heavy geometric processing on the building models has stricter geometry

- 1.1 Building model requirements and reconstruction challenges . . . 1
- 1.2 Data driven versus model driven building reconstruction 3
- 1.3 Automatic LoD2 reconstruction for the Netherlands 4
- 1.4 Notes and comments 8
- 1.5 Exercises 8

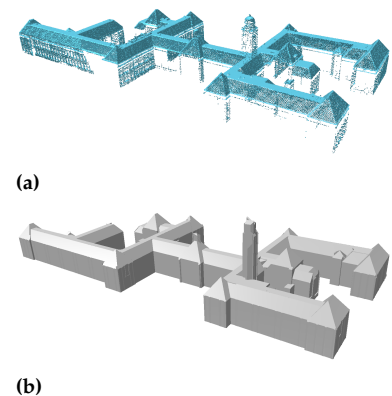


Figure 1.1: Building reconstruction transforms a point cloud (a) into a mesh model (b).

and topology requirements than an application that merely visualises the building models. Reconstruction challenges, on the other hand, are mostly input dependent. It is about the characteristics of the input data and the typical shape of the buildings that are present in the input data. It is relatively easy to design a reconstruction approach for a very high quality point cloud that contains only very simple building shapes, whereas reconstructing complex building shapes from a very sparse and low quality input point cloud is substantially more difficult.

1.1.1 Building model requirements

Following are commonly encountered building model requirements. Notice that the exact requirements will depend on the application.

Low complexity Means that the building model ought to have as few vertices, edges, and faces as possible. Building models with a low complexity are faster to process and take up less storage space.

High accuracy The surfaces of the building model should have the lowest possible error with respect to the input point cloud. This error can be measured as the root mean square of all the distances from each input point to the model surface.

Geometrically valid This means among other things that the mesh is 2-manifold, has consistent faces orientation, no duplicated vertices, and no self intersecting geometries. This makes the model generally easier to process since many assumptions can be made about the structure of the mesh. Lesson 3.2 discusses this (and the relevant ISO19107 standard) in more detail.

Level of Detail (LoD) Specifies the degree of generalisation in the roof structure of the reconstructed building model when compared to how the actual building is built. An LoD1 model for example only allows horizontal flat roof surfaces (even if the actual building roof looks different), whereas an LoD2 model also allows for more detailed multi-pitched roof shapes. For the remainder of this lesson we will focus on the more detailed LoD2 models. Lesson 4.1 discusses the possible LoDs for building models in more detail.

1.1.2 Reconstruction challenges

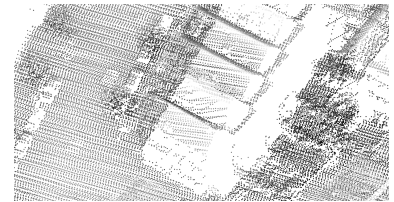
Why can it be hard to satisfy the model requirements? This depends on the reconstruction challenges. We distinguish between two main categories.

Firstly, there are variations in architectural style. Urban environments can be complex and organised with a high degree of randomness due to their anarchical creation over time. This makes it difficult to design a reconstruction algorithm that is able to model 100% of the buildings on earth. It is probable that there are always a few cases that violate some of the assumptions made in the building reconstruction method. For example, to simplify a reconstruction approach, it may seem reasonable to assume that buildings are not built on top of each other without touching each other. And while this assumption is valid in more than 99% of the cases, in practice there are some violations of this assumption (see Figure 1.2).



Figure 1.2: A complex urban environment.

Secondly, we need to consider the quality and completeness of the input data. This mostly relates to how the input data, ie the point cloud, was acquired (compare eg Figure 1.3a to 1.3b). Most building reconstruction methods work with point clouds that are captured from an airplane. This is the most efficient way to cover large areas, but it also means that not all the exterior surfaces of a building are captured due to occlusion. In particular facades and the underside of overhanging structures may be missing in such datasets. If a surface is missing in the point cloud we need to compensate for that with assumptions on what we expect the building to look like. For instance, we could assume that facades are always vertical so that we can simply model a vertical plane from the roofline to the ground. However, while this is reasonable for the majority of buildings there are bound to be some exceptions. Other point cloud properties are also important. For example the point density is indicative for the smallest details that we can reliably detect in the point cloud. Consequently we can not reasonably expect to see smaller details in the reconstructed building model unless very strong assumptions are taken on the type of building shape that is modelled. Some surface materials can also lead to problems. Glass surfaces for example are notoriously difficult to measure with airborne acquisition techniques, leading eg to holes in the roof surface which can lead to problems in a building reconstruction method.



(a)



(b)

Figure 1.3: Varying point cloud qualities. a) low point density with missing facades, b) high point density and points on facades.

1.2 Data driven versus model driven building reconstruction

Building reconstruction has been a popular topic among researcher over the last few decades. Many approaches exist that vary in the expected type and resolution or density of input data, the precise model requirements, and in how restricted they are to a particular architectural style. One could classify these methods on a linear scale with on one extreme the purely so-called *data driven* approaches and on the other extreme the purely so-called *model driven* approaches.

The data driven approach strongly relies on the quality and completeness of the input data. The resulting building models have a good data fit, but a high complexity (high number of faces). Defects in the input data

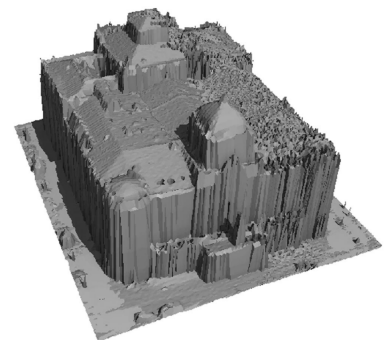


Figure 1.4: Data driven reconstruction based on a triangulation of the input points (Axelsson, 1999)

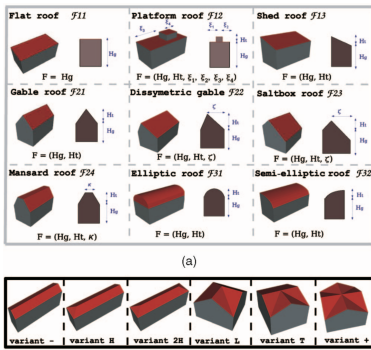


Figure 1.5: Model driven reconstruction by fitting parametrised roof models (Lafarge et al., 2010)

are likely to cause problems in the building model, such as holes and non-2-manifoldness. Examples of the data driven approach are methods that triangulate directly the input point cloud (see Figure 1.4).

The model driven approach, on the other hand, relies on strong modelling assumptions about the building shape. This typically results in models with a low complexity, but a poorer fit with the input points when compared to a purely data driven approach. Because the model driven approach does not rely so heavily on the quality of the input, defects in the input point cloud are less likely to lead to problems in the building model. Examples of the model driven approach are methods that fit pre-defined roof shapes such as a simple gable roof to a point cloud (see Figure 1.4). Such a method will only work for buildings for which a pre-defined roof shape is available. Yet, if this the case it can already work reliably for a very sparse point cloud.

Clearly both approaches have limitations. The most advanced building reconstruction methods, including the one discussed below, try to combine the best of both to come to an optimal compromise, eg a method that has both a good datafit and a high degree of flexibility in building shapes but also a low complexity and perfect geometric validity. However, be aware that such a mixed approach combines not only the advantages, but likely also the disadvantages of both approaches to some degree.

1.3 Automatic LoD2 reconstruction for the Netherlands

In this section we will discuss an automatic LoD2 reconstruction method that I developed to work with Dutch open data*. The output of this method should have both a good data fit and a low model complexity and is aimed to have completely valid geometry output. This means the resulting models are suitable for various kinds of environmental simulation applications.

1.3.1 Modelling assumptions

The following assumptions are taken in the reconstruction method. They are deemed reasonable for the Dutch input datasets that the method was designed on, and with these assumption the reconstruction problem is somewhat simplified.

piecewise planar The shape of a building can be adequately approximated using planar faces that are detectable from the point cloud.

2.5D with vertical walls The roof of the building is 2.5D and all walls are vertical. This implies the 3D building model can be extruded from a 2D planar partition of the roof. The 2.5D assumption is quite reasonable for airborne point cloud, because each building is only scanned from above anyhow.

classified point cloud A reliable classification of the input point cloud is expected, ie at least a building and a terrain (ground) class must be present. This is the case for the AHN3 dataset that is used.

* the AHN3 point cloud and the BAG building footprints

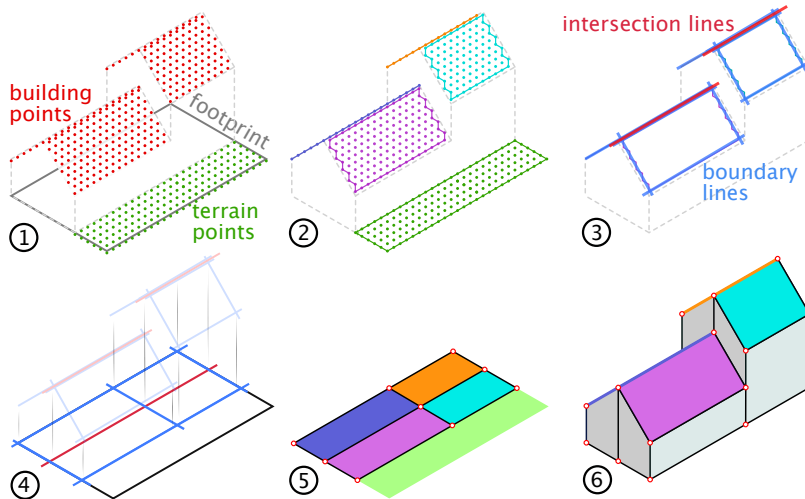


Figure 1.6: The main steps in the reconstruction algorithm. 1) the classified (aerial) point cloud is cropped on the 2D footprint, 2) planes and their boundaries are detected in the point cloud, 3) from the roof planes the intersection lines and boundary lines are extracted, 4) the lines are regularised and projected onto the 2D footprint, 5) the roof-partition is created. This is a DCEL where each face is labeled with the corresponding plane (from 2, compare colors). 6) the roof-partition is extruded into a 3D mesh.

If a terrain plane is assigned to a face from the roof-partition, that face is removed (2 and 5).

footprints are available Apart from a point cloud the method also takes 2D building footprints as input. These are used to crop the point cloud for each building. It is assumed that the footprints are up-to-date and well aligned with the point cloud.

The method can be classified as a mix between the purely data and model driven approaches as discussed in the previous section. Consequently it also mixes the benefits and trade-offs of both extremes. For example, instead of forcing complete roof shapes on a point cloud, it is only assumed a building is composed of planar surfaces. This makes the method more flexible compared to a purely model driven approach that fits a pre-defined roof shape, since it should be able to handle any possible roof shape that can reasonably be approximated with (large) planar surfaces while still maintaining a low model complexity. However, if a plane cannot be fitted to a part of the roof due to defects in the point cloud, that part may lead to errors in the resulting building model.

1.3.2 Method overview

Figure 1.6 illustrates the six main steps of the algorithm. The main idea is to compute a so-called *roof-partition*; a planar partition of the footprint where each face corresponds to a planar piece of the roof and is labeled with a roof plane. Prior to creating the roof-partition the roofplane and line features must be extracted from the point cloud (Figure 1.6 step 2 and 3). And once the roof-partition is available, the 3D building model can be generated through extrusion (Figure 1.6 step 6).

1.3.2.1 Feature extraction

The roof-partition is made using lines that are derived from roof planes that are extracted from the building point cloud. The roof planes are detected using a region-growing algorithm and then two type of lines are derived from the planes: boundary lines and intersection lines (see Figure 1.6 step 3). The boundary lines are created by detecting lines in the boundary of the α -shape of each detected roof plane. The intersection

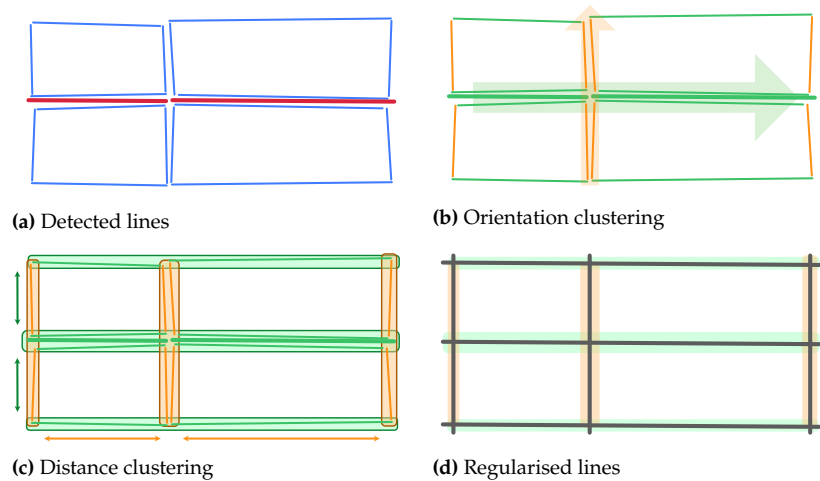


Figure 1.7: Line regularisation through clustering.

lines are created where adjacent planes intersect, such as on the top of a gable roof.

Before the boundary and intersection lines are used to partition the footprint, they are regularised. The goal of line regularisation is to remove duplicate lines and thereby reduce the complexity of the roof-partition. For example, the line on top of the gable roof in Figure 1.6 is detected three times: once as an intersection line and twice as a boundary line (once for each incident roof plane). After line regularisation only a single line remains. After projecting the detected lines to 2D, line regularisation is done in two steps: orientation clustering and distance clustering (see Figure 1.7).

Orientation clustering is performed first, and in this step lines that have approximately the same orientation in the 2D plane are put in the same cluster. For example in Figure 1.7b, there are two dominant orientations that each form a cluster of lines. Within each orientation cluster the angle between the lines is relatively small, whereas the angle between lines in different clusters is large.

Next, distance clustering is performed. This divides each orientation cluster into one or more distance clusters. This is done by computing for each orientation cluster the distances between the lines it contains. Groups of lines with a small distance with respect to each other are put in their own distance cluster, whereas the distance between different distance clusters is large (see Figure 1.7c).

Finally, one average line is computed for each distance cluster (Figure 1.7d).

1.3.2.2 Construction of the roof-partition

After the lines are detected and regularised they are used to subdivide the footprint into a planar partition called the roof-partition. A doubly connected edge list (DCEL) is used to represent the full topology of the planar partition of the footprint, that is referred to as the *initial roof-partition*. This means that each intersection is explicitly represented with a vertex. In addition there are no dangling edges. The use of a DCEL

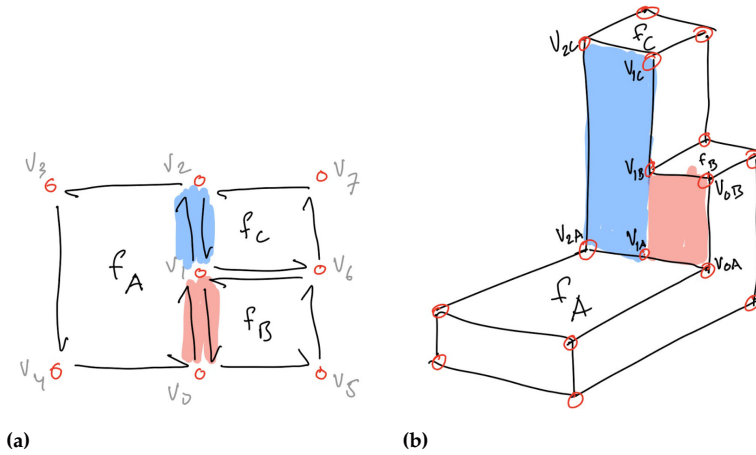


Figure 1.8: The roof-partition is represented as a DCEL (a). When extruding to a 3D mesh (b), each edge in the roof-partition becomes a wall face in the 3D Mesh. Each vertex in the roof-partition (eg v_1) needs to be replicated for each incident face: eg v_{1A}, v_{1B}, v_{1C} in the 3D Mesh.

allows for easy traversal and manipulation of the roof-partition, eg for the extrusion to a 3D mesh in the last step.

Depending on the number of lines that remain after regularisation, the initial roof-partition may still have a high complexity; it may contain many small faces. To further reduce the complexity of the roof-partition and to simultaneously assign an optimal roofplane to each face, an optimisation step is performed[†]. In this step a roof plane is assigned to each face in the roof-partition (see Figure 1.6 step 5). This is done in such a way that 1) the total error with the input point cloud is minimised and 2) the total length of the edges between faces of a different roof plane is minimised. This optimisation thus seeks an optimal balance between respectively a good data fit and a low complexity of the roof-partition. After the optimisation is complete, the edges for which the two incident faces are assigned to the same roof plane are removed from the partition. The faces in the resulting *final roof-partition* are referred to as *roof-parts*.

1.3.2.3 Extrusion

The final roof-partition is transformed into a 3D building mesh using extrusion. This is done by exploiting the topological information that is available in the the DCEL of the roof-partition, as illustrated in Figure 1.8. Notice that the building mesh consists of three types of faces, ie the floor, the roof and the wall faces. These are generated from the roof-partition in separate procedures.

floor face The geometry of the floor face consists of the edges in the roof-partition that are incident to the exterior to the footprint. The elevation of the floor face can either be set to the lowest ground point around the building, or if a terrain mesh is available it can be made exactly fitting with the terrain by computing the intersection with that terrain mesh and setting the vertex elevations accordingly.

wall faces These are vertical faces that connect the floor face with the roof faces. They are extruded from the edges in the roof-partition that have one or two incident roof parts. Depending on the plane

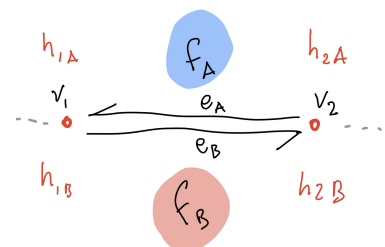


Figure 1.9: The edge e (comprising of two halfedges e_A and e_B) is incident to two faces (f_A and f_B) and two vertices (v_1 and v_2). In case of a roof-partition, the height at v_1 on face f_A is denoted as h_{1A} .

[†] Graph-cut optimisation is used. The details on how graph-cut optimisation works are outside the scope of this course.

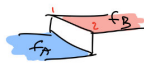
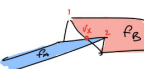
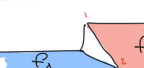
Case	condition	vertex order
	$h_{1,A} < h_{1,B}$ AND $h_{2,A} < h_{2,B}$	1. $v_{1B}, v_{1A}, v_{2A}, v_{2B}$
	$h_{1,A} < h_{1,B}$ AND $h_{2,A} > h_{2,B}$	1. v_{1B}, v_{1A}, v_{1X} 2. v_{2A}, v_{2B}, v_{1X} } 2 Faces!
	$h_{1,A} < h_{1,B}$ AND $h_{2,A} = h_{2,B}$	1. v_{1B}, v_{1A}, v_{2A}

Figure 1.10: Determining wall face geometry and vertex order. h_{1A} denotes the elevation at vertex v_1 on face f_A (see Figure 1.9).

configuration of the incident roof parts an edge is extruded differently. Figure 1.10 shows a few possible cases (there are more). Notice that an edge in the roof-partition can generate 0 (if the incident planes intersect exactly at the edge), 1 or 2 wall faces. Also notice that the order of the vertices of a wall face (so that they are oriented counter-clockwise around the face normal that points to the exterior of the mesh) is completely determined by the plane configuration case at the edge.

Special attention needs to be paid to vertices that are extruded to more than two elevations such as v_1 in Figure 1.8a. To get a topologically correct building mesh, the extruded vertices should become part of all their incident wall faces. Vertex v_{1B} should thus also be inserted in the boundary ring of the blue face in Figure 1.8b, despite the fact it is co-linear with v_{1A} and v_{1C} .

roof faces Each roof part in the interior of the roof-partition will generate a roof face in the building mesh. The planimetric geometry of the roof faces is identical to the faces in the roof-partition. The vertex elevations are found by projecting the 2D vertices to the plane of the roof-part.

1.4 Notes and comments

Rottensteiner et al. (2014) gives an overview of building reconstruction methods.

If you want to know more about the graph-cut optimisation method to optimise the roof-partition have a look at the paper from Zebedin et al. (2008).

A good example of a true 3D building reconstruction method (no 2.5D assumption) is the work of Nan and Wonka (2017)

1.5 Exercises

1. Explain the advantages of 2-manifoldness in a building model
2. Complete the table of possible plane configurations in Figure 1.10.
3. Could a non-manifold edge be created in the extrusion that is described in Section 1.3.2.3? If so, describe how that could happen.

Bibliography

- Axelsson, P. (1999). Processing of laser scanner data—algorithms and applications. *ISPRS Journal of Photogrammetry and Remote Sensing* 54.2, pp. 138–147.
- Lafarge, F., X. Descombes, J. Zerubia, and M. Pierrot Deseilligny (2010). Structural approach for building reconstruction from a single DSM. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 32.1, pp. 135–147.
- Nan, L. and P. Wonka (2017). PolyFit: Polygonal Surface Reconstruction from Point Clouds.
- Rottensteiner, F., G. Sohn, M. Gerke, J. D. Wegner, U. Breitkopf, and J. Jung (2014). Results of the ISPRS benchmark on urban object detection and 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing* 93, pp. 256–271.
- Zebedin, L., J. Bauer, K. Karner, and H. Bischof (2008). Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. *European conference on computer vision*. Springer, pp. 873–886.