Acta Polytechnica CTU Proceedings **34**:133–137, 2022
https://doi.org/10.14311/APP.2022.34.0133

Published by the Czech Technical University in Prague

AUTOMATIC EXTRACTION OF 3D BUILDING MESHES FROM LIDAR DATA

Jan Voříšek*, Bořek Patzák, Daniel Rypl

Czech Technical University in Prague, Faculty of Civil Engineering, Thákurova 7, 166 29 Prague 6, Czech Republic

* corresponding author: jan.vorisek@fsv.cvut.cz

Abstract. Terrestrial laser scanning and drone-based scanning are often combined to gain a complete picture of the external envelopes of buildings. While the resulting data sets are easy to understand for humans, extracting any semantical information from the data is a challenging task. Creating a BIM model for large cities is desirable, especially for planning and inventory purposes. Nowadays, engineers manually separate buildings and draw corresponding floor plans or any other required entities, making creating such a model nearly impossible for the whole city.

In this contribution, we present the design and concept of a C++ library that provides fundamental algorithmic tools for automated detection of the ground points, separation of buildings into individual point clouds, and segmentation of walls and roofs. The buildings' 3D model (both volumetric and surface mesh) can be constructed using the data. Furthermore, the building area and volume calculation can be performed on such a model, which can then be compared with the existing city plan.

Keywords: Laser scanning, BIM, point cloud, city planning.

1. Introduction

Building Information Modelling (BIM) brings excellent opportunities for the efficient and more automated design process of building structures and maintenance and reconstruction planning of existing city structures. However, obtaining a digital BIM representation of the latter is often challenging when no up-to-date city ground plans are available in digital form. The plans are usually just a set of polylines with no relation to individual buildings or other entities.

Laser scanning is a convenient and accessible technology to document the existing infrastructure. For cities, the combination of the terrestrial laser scanner and drone-based scanning is often used to capture the external envelope of the buildings. The output is a dense set of data points in space, the so-called point cloud. See Figure 1 for an example of a laser-scanned part of Dublin. The dataset was obtained from [1]. While point clouds can be directly rendered and inspected, they carry no additional information about the scanned objects. Therefore, it is desirable to convert the raw point data into a BIM representation. Civil engineers and architects must identify and separate buildings manually, draw the corresponding floor plans, and measure any desired quantity (i.e., building area and volume).

Our goal is to provide automated extraction of the individual buildings from the city point cloud and construction of the corresponding building meshes. The idea is based on the segmentation of the ground-level points. Then, those points are used as a filter to obtain all the points above a certain height. Segmentation of the individual buildings is then performed on the filtered point cloud. Furthermore, the outer surface planes are found, and a voxel-based 3D model is built for each building. The calculation of the build-

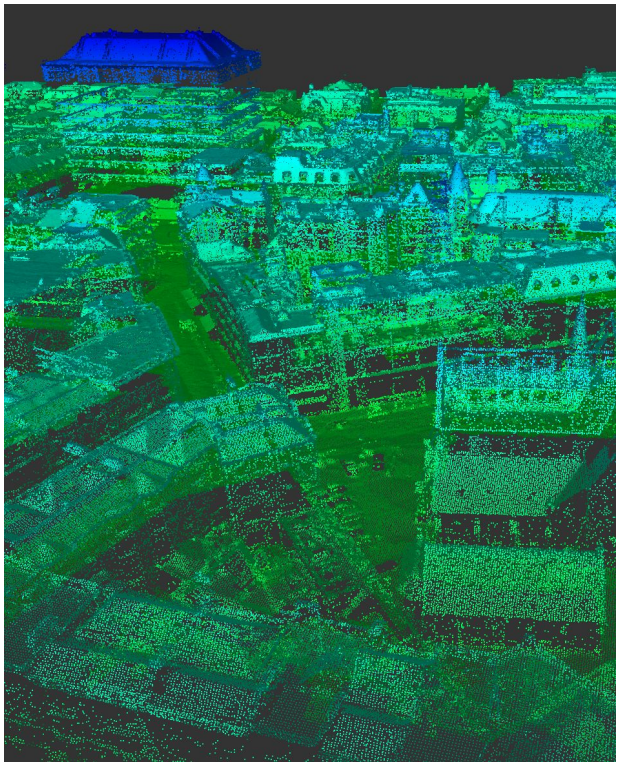

Figure 1. Laser scan of Dublin consisting of 226 million points before downsampling. The buildings and streets are clearly visible because the points were obtained from an airplane, so the point density is much higher than on the walls. Unfortunately, no RGB color information is available. Source: [1]

ing volume is straightforward. By extraction of the surface of the voxel mesh, the building area can be approximated. Such a surface model may be suitable for texture mapping and city visualization in planning and architectural applications.

We have used a 2×2 km area of Dublin for the implementation and verification of our implemented algorithms. The raw dataset is described in [1].

2. Pre-processing

Laser scanners produce very dense point clouds containing hundreds of millions of points. The distance between the scanned points often ranges in millimeters. Therefore the point cloud data processing must be handled carefully in order to achieve competitive computation times and memory efficiency.

2.1. Downsampling

While storing the points as binary data offers high reading speeds, with hundreds of millions of points, reading all the values into memory might still not be possible on a standard computer with limited memory. For example, storing 100 million XYZ coordinates (3x64 bit float) occupies 2.4 GB of memory, more than half of a commonly available 4 GB.

Reducing the number of points to work with is inevitable. In our case, the points were downsampled in a 3D voxel grid using the efficient implementation of a C++ hash map [2]. Reasonable selection of a grid step of say $1 \times 1 \times 1$ cm reduces all the points within the 1 cm^3 cube to a single point with weighted coordinates.

As we have discussed in [3], one can see that just by downsampling to a 1 cm^3 grid, the size of points goes down to at least 30%. In the case of city scanning, the grid step length may be much longer. For example, we have used a downsampling step of 5 cm for the Dublin point cloud, and even a larger step of 10 cm still produced satisfying results.

2.2. The normal vector of downsampled points

Further processing can take advantage of determining the normal vector at each downsampled point. This is done in the following steps.

For each point, its ten closest points are found. This is done by implementing and using a K-d tree data structure. The total number of points n for the normal evaluation is at most $n = 10$.

The 3×3 covariance matrix of the point X, Y, Z coordinates relative to the point is calculated using the formula for the covariance of A and B below.

$$\mathrm{cov}(A, B) = \frac{\sum_{i=1}^{n}(a_i - \mu_a)(b_i - \mu_b)}{n - 1}, \qquad (1)$$

where μ_x, μ_y and μ_z are the coordinates of the subject point.

The smallest Eigenvector of the covariance matrix is the plane normal [4]. The direct solver of eigenvalues was implemented for the 3×3 matrix using the arccos function, as described in [5].

Figure 2. **The results of a ground segmentation**: The streets are dominating the picture. It can be seen that most of the inner areas of the building were captured as well. The large flat park area in the top right is also notable.

3. Ground-level segmentation

There are numerous algorithms for ground segmentation. For example, in [6], they use a ground plane fitting algorithm that is suitable for small areas of the point cloud. As we have wanted to obtain ground points for the whole dataset (2 km^2 Dublin area), we propose a different approach suitable for cities with mostly vertical building walls. We claim that this is a reasonable approach for most cities. However, the algorithm may not perform well for areas where no vertical building walls are present.

The first step is to activate a two-dimensional grid in the horizontal plane. Grid coordinates are calculated for each point, and the point is added to a list at a corresponding grid index.

In the next step, we iterate over all the active grid cells. Cells without a sufficient vertical coordinate difference Δ_z will be removed from the grid. The minimum vertical coordinate difference equals the minimum expected building height. We have chosen $\Delta_z = 3$ m because we do not expect any building to be smaller than such a height.

Most of such selected grid cells will contain building wall points. However, lamps, some higher trees, or noise will be captured as well. A minimum number of points n_{min} in the cell is introduced to limit the number of false positives. The average point density ρ is calculated for the point cloud. For the Dublin data set, the value is around 1 point per 3 cm^3. The average point density ρ_{cell} is calculated for the grid cell and compared to the ρ. Only cells with $\rho_{cell} \,/\, \rho > 0.25$ are accepted to accommodate walls with large openings (i.e., windows, balconies).

Finally, any high walls present on the buildings and not reaching the ground need to be removed. Another horizontal grid with a much higher step length

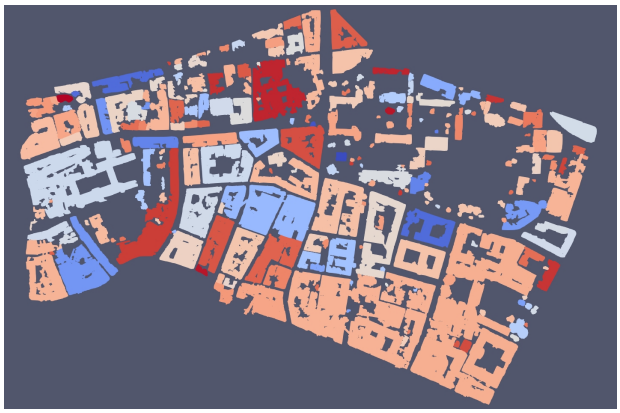

FIGURE 3. **The results of building segmentation**: Each of the buildings are marked in a single color. It can be seen that noise is present in the results as small entities (primarily trees). Multiple buildings may be identified as one when some joining element is present (i.e., bridge or a tree in between).

is introduced (25 m in our case). Points from the previous grid (z-coordinate is equal to the minimum vertical coordinate in the cell) are activated within the larger grid. We can then calculate the median of the z-coordinate per the larger grid's steps and filter out the smaller grid by not taking into account the points above the median.

Finally, we can calculate the middle point (in terms of X, Y coordinates) for each active cell, and the point's Z coordinate is considered the minimal Z coordinate within the cell. Those calculated points are considered to be the ground-level approximation.

In order to separate the road surface and sidewalks, the Z coordinate of the ground points should be increased by some small tolerance. We have used values around 0.5 m. It is then possible to iterate over all of the initial points and find its closest ground level point. If the point lies below the ground points, then it can be considered as a surface point.

The results of a ground segmentation can be seen in Figure 2.

4. Building segmentation

The ground level was estimated to perform building segmentation similarly as we have performed rooms segmentation in our previous work [3].

The closest ground point is found for each point. Each point that is at least 3 meters higher than the ground point is projected onto a 2D grid resulting in a pixel-like image. The step of the grid should be at least twice the step used for the initial downsampling, or the grid will contain missing points as some points may not be appropriately captured.

Region growing [7] is applied to join adjacent pixels into separate regions. Each region corresponds to a single building. Backfilling of each of the found regions is performed to include missing areas (i.e., chimneys). Knowing the pixel count and pixel area (step × step), the area of the regions can be estimated.

Finally, the original point cloud is iterated, and points inside of the found regions (in terms of X and Y coordinates) and above the ground are selected.

5. Plane segmentation

The surface planes are identified for each building using the modified Random Sample Consensus [8] with minimum area and continuity control. Three separate sets of trial points for outer walls, horizontal roofs, and pitched roofs are constructed. Running the RANSAC directly on the point cloud has a high chance of finding a generally oriented plane across the building. The three separated sets of points are defined as follows.

Outer walls Only points with horizontal normal are considered for plane finding, a slight deviation of 15 degrees is allowed to account for deviations in construction.

Horizontal roofs Only points with vertical normal are considered for plane finding, a slight deviation of 15 degrees is allowed to account for deviations in construction.

Pitched roofs Only points not contained in the previous two datasets are considered for plane finding.

The results of a plane segmentation for one of the building can be seen in Figure 4.

6. Building mesh

Each building point cloud is iterated, and its points are projected in a horizontal 2D grid. The single region is then backfilled to avoid losing volume in places with no points captured (i.e., shadows or reflective surfaces). Finally, the grid is deactivated at places where any points exist below the ground level.

The simplest mesh consists of blocks, each defined by the grid cells (in the XY plane), the ground level, and the highest point within the cell. Such a model allows for a simple building volume calculation by summing up the volumes of the blocks. The area is the product of the sum of the active cells in the grid and the grid step squared. Such a mesh can be seen in Figure 5.

Calculating the building volume, including antennas or chimneys, may not be desirable. Therefore, the previously found roof planes can be used to clip the blocks so that no Z coordinate is above the subject plane.

Voxel-based mesh does not precisely represent the building near the surface if the planes are not parallel to the voxel grid. The previously found planes could be used to straighten up the voxel model by moving its vertices onto the planes within a small distance.

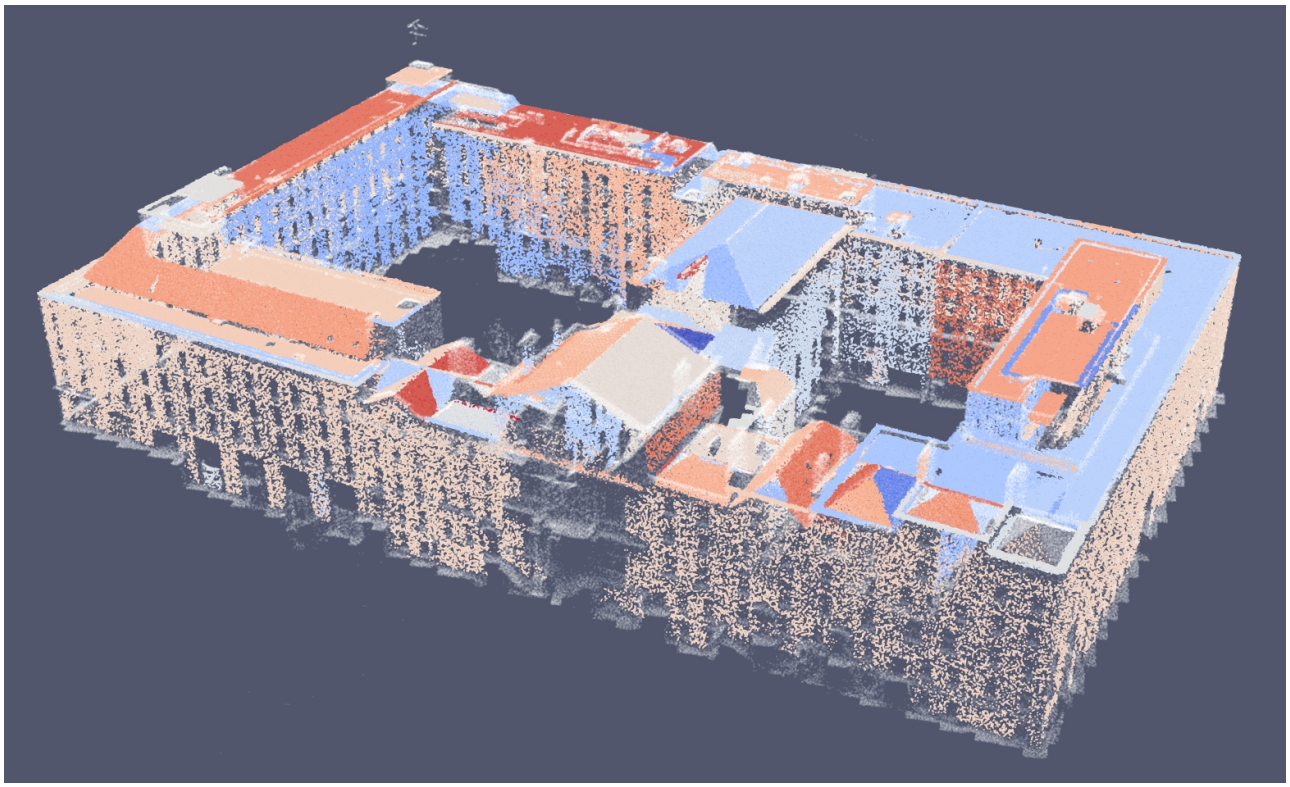

Figure 4. **The results planes of plane segmentation for one building**: Each of the planes is marked with a unique color. It can be seen that most of the surface planes were found with some smaller pieces missing.

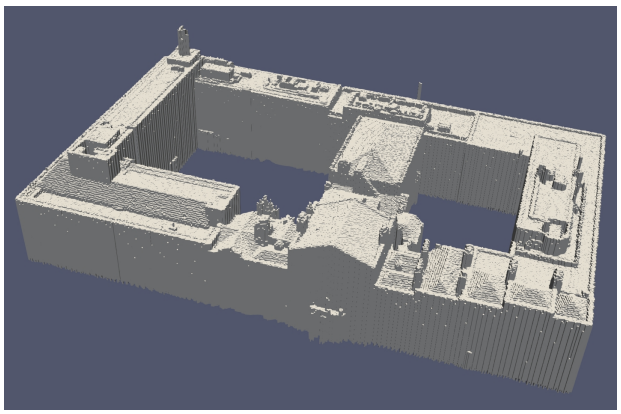

Figure 5. **The resulting voxel model of a single building.**

7. Conclusions

This paper analyzed the methods of segmentation of the BIM entities from the city point clouds. The primary motivation was to separate individual building point clouds from the large city point cloud and obtain water-tight mesh of such buildings. The meshes are suitable for approximation of the building volume and area. The proposed algorithms were tested on a 2015 aerial laser and photogrammetry survey of Dublin city.

The findings can be summarized as follows.

- Downsampling with a step of 0.05 m dramatically reduces the number of points to around 20% of the original dataset. Such precision is still sufficient for the segmentation of the buildings and their features.
- The ground segmentation procedure works well for cities with mostly vertical building walls. More complex buildings can still be identified if there are enough nearby ground points.
- The building segmentation algorithm works well for buildings with a sufficiently large area. However, when using a small minimum area of around $A_{b,min} < 10$ m^2, it is not easily possible to distinguish between a small building (i.e., a garage) and a tree. In order to eliminate the trees, buildings with no found planes can be excluded.
- Modified RANSAC algorithm with minimum area and continuity checking is used to find the building surface planes. Pre-filtering of the trial point clouds is done using the point normal vectors, which dramatically increases performance. The results are promising for vertical (outer walls) and horizontal (roof) planes. However, some skewed roof planes are often missing due to noise or additional structures (i.e., large chimneys).
- 3D voxel mesh can be constructed for each building. The building area and volume can be easily estimated from the voxel mesh. The voxel mesh can be clipped by the found planes, removing small features (i.e., chimneys and antennas). The voxel points can also be projected onto the found planes to straighten out the surface.

Acknowledgements

The development of this software was supported by the Technology Agency of the Czech Republic under the program of the National Competence Center 1 as a project "Center for Advanced Materials and Efficient Buildings" (CAMEB) - project registration No. TN01000056 and by the Grant Agency of the Czech Technical University in Prague (SGS project No. SGS21/037/OHK1/1T/11), both gratefully acknowledged.

References

[1] D. F. Laefer, S. Abuwarda, Anh-Vu Vo, et al. 2015 aerial laser and photogrammetry datasets for dublin, ireland's city center, 2017. https://doi.org/10.17609/N8MQ0N.

[2] G. Popovitch. The parallel hashmap or abseiling from the shoulders of giants. https://greg7mdp.github.io/parallel-hashmap/. Accessed: 2020-04-14.

[3] J. Voříšek, B. Patzák, E. Dvořáková, D. Rypl. Automated BIM Entity Reconstruction from Unstructured 3D Pointclouds. *Acta Polytechnica CTU Proceedings* **30**:126–130, 2021. https://doi.org/10.14311/app.2021.30.0126.

[4] E. Ernerfeldt. Fitting a plane to noisy points in 3d. https://www.ilikebigbits.com/2017_09_25_plane_from_points_2.html, 2017. Accessed: 2020-04-14.

[5] S. Hartmann. Computational aspects of the symmetric eigenvalue problem of second order tensors. *Technische Mechanik* **23**:283–294, 2003.

[6] D. Zermas, I. Izzat, N. Papanikolopoulos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5067–5073. 2017. https://doi.org/10.1109/ICRA.2017.7989591.

[7] M. Montoya, C. Gil, I. García Fernandez. Implementation of a region growing algorithm on multicomputers: Analysis of the work load balance., 2000.

[8] M. A. Fischler, R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* **24**(6):381–395, 1981. https://doi.org/10.1145/358669.358692.