FACULTY OF CIVIL ENGINEERING CTU IN PRAGUE

PointCloud2BIM library
User Manual

Authors: E. Dvořáková, B. Patzák, D. Rypl and J. Vořišek

Department of Mechanics, Faculty of Civil Engineering, Czech Technical University in Prague

Thakurova 7, 166 29 Prague, Czech Republic

Table of Contents

Copyright

Department of Mechanics, Faculty of Civil Engineering, Czech Technical University in Prague, Thákurova 7, 166 29 Prague, Czech Republic.

Obtaining PointCloud2BIM library

The library can be obtained from Department of Mechanics, Faculty of Civil Engineering, Czech Technical University. Please send request to borek.patzak@fsv.cvut.cz for licensing options.

Introduction

The 3D laser scanning is a convenient and accessible technology to document the existing infrastructure, including buildings. The output is a set of data points in space, so-called point clouds. While point clouds can be directly rendered and inspected, they need to be converted into BIM representation for efficient digital processing.

The PointCloud2BIM library provides a set of algorithms to facilitate the processing of Building Point Clouds and the identification of fundamental entities, such as floors, rooms, walls and openings.

The overall strategy of point cloud transformation into the digital BIM representation of the building is based on segmenting the point cloud individual floors. The floor segmentation is based on the histogram of vertical coordinates. From the histogram, the ceiling and floor levels are extracted, corresponding to locations with significant location frequency.

After segmenting the building into floors, the individual floors are segmented into rooms. This step is based on projecting all points associated with the floor into a horizontal plane. The individual rooms are then identified using region growing algorithm.

For each floor, the wall planes are identified using the Random sample consensus (RANSAC) algorithm. Various filtering steps are applied to detect only relevant wall planes. Later the identified wall planes are matched, segmented and paired to establish a representation of walls. Finally, the wall openings are identified and classified as doors/ windows and other openings.

The PointCloud2BIM library is written in C++. It can be integrated into any BIM software to assist the conversion of point clouds into digital BIM representation. The library API is documented in the Reference Manual [1]. The library can also be used as a standalone tool without any dependency on BIM software by offering a set of command-line tools that can perform the individual steps introduced above. The individual tools operate on top a of common database, which stores the original point cloud data as well as the results.

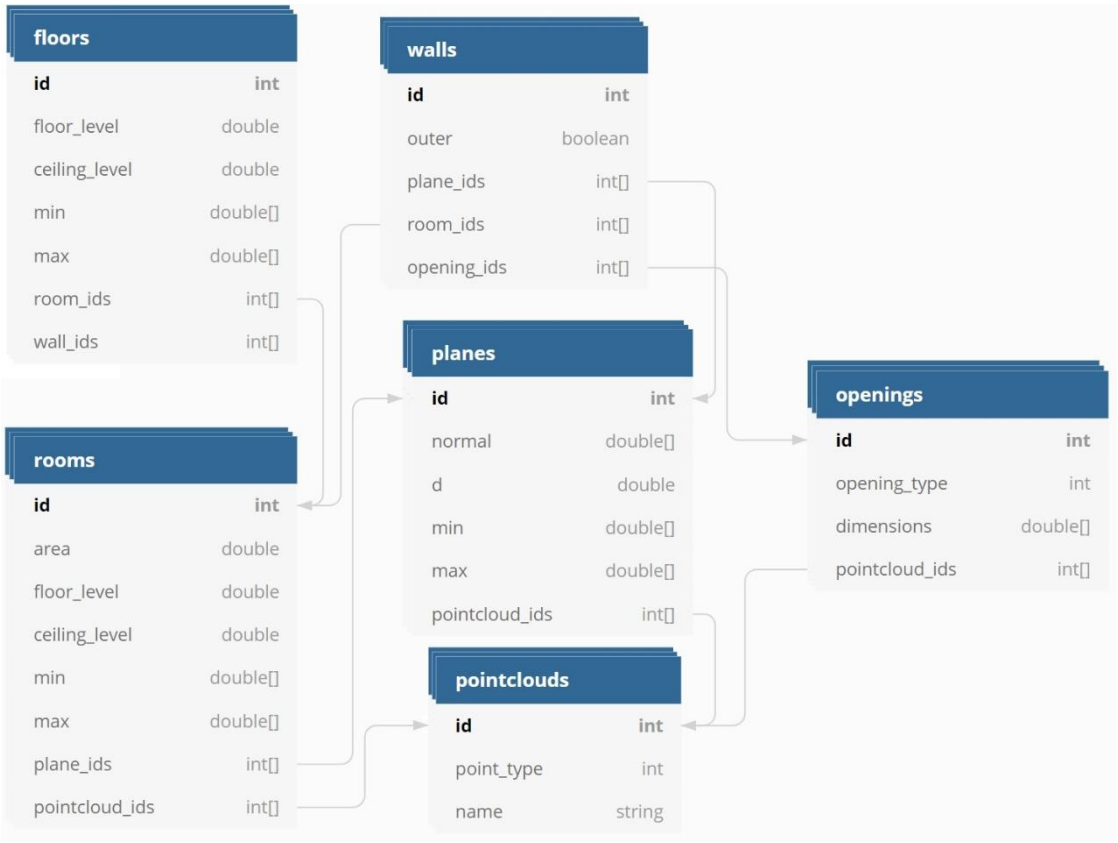


Figure 1: Schema of project database.

Requirements

- PointCloud2BIM requires a cmake and C++ compiler (supporting C++17 standard). It can be compiled on Unix/Linux systems using g++ compiler or on Windows systems using Visual C++.
- Optionally, Easy3D library (https://github.com/LiangliangNan/Easy3D) is supported for visualization

Compilation

- Linux/Unix
  - Create a build directory
  - In the build directory run cmake followed by make
    - cmake PATH_TO_PointCloud2BIM_SOURCE_DIR
    - make
- Windows
  - Run cmake
  - Open project solution file created by cmake in the build directory with MS Visual
  - Build the project

  Modern versions of Visual Studio (2019 onwards) allows directly building CMake projects.

Available commands

PointCloud2BIM is primarily a library intended to be used in different applications. The library API is described in detail in the Reference manual. In addition, a set of tools is provided in the form of executable applications, which are documented in this section.

Command name	**`Project`**	
Description	Manipulate project database. The project is created for every point-cloud.	
Parameters	--project <string PATH>	Project file to open
	--import <string PATH>	Imports point cloud file. Allowed formats: x,y,z or x,y,z,r,g,b,I
	--export<string PATH>	Export any entity/entities (for example using --floors). Creates xyz file of all points.
	--pointcloud <int ID>	Print details about given pointcloud
	--floor <int ID>	Print details about given floor
	--rooms <int ID>	Print details about given room
	--plane <int ID>	Print details about given plane
	--wall <int ID>	Print details about given wall
	--opening <int ID>	Print details about given opening
	--pointclouds	Print all point clouds in a table
	--floors	Print all floors in a table
	--rooms	Print all rooms in a table
	--planes	Print all planes in a table
	--walls	Print all walls in a table
	--openings	Print all opening in a table

Command name	**Downsampler**	
Description	Performs ownsampling in variable grid length and also calculates normal vectors for each downsampled point.	
Parameters	--project <string PATH>	Project file to open
	--pointcloud <int ID>	
	--step <double XYZ_STEP>	
	--normals	Calculate normal vector for each downsampled point (cell) using the covariance matrix.

Command name	**Rooms**	
Description	Runs room finding algorithm in the desired height range. It creates a new floor entity as a parent of found rooms. It is also possible to run plane finding algorithm in each room. SVG output of the room grid is available for debugging.	
Parameters	--project <string PATH>	Project file to open
	--floor <int ID>	Select floor ID to perform room finding
	--step <double XY_STEP>	Room finding is performed on a 2D XY (floor plane) grid with step XY_STEP
	--zlimit <double FLOOR_Z double CEILING_MIN double CEILING_MAX>	All points with Z in interval <CEILING_MIN; CEILING_MAX> are selected for room finding
	--planes	Calculates planes for each found room using RANSAC algorithm
	--backfill	Fills holes in each room's grid (floor plan)
	--svg	Exports SVG grid for all rooms and for whole floor

Command name	**Walls**	
Description	Runs wall finding algorithm in each floor. Minimal length and thickness of the wall should be specified according to building inspection while scanning the subject pointcloud. SVG output of each wall grid is available for debugging.	
Parameters	--project <string PATH>	Project file to open
	--floor <int ID>	Select floor ID to perform wall finding. Planes must exist in the floor rooms.
	--minlength <double>	Minimum wall length in its local X axis (in the wall direction)
	--minthickness <double WIDTH=0.1>	Minimal allowed wall thickness
	--maxthickness <double THICKNESS=0.5>	Maximal allowed wall thickness
	--svg	Exports floor plan of walls as a SVG image

Command name	**Openings**	
Description	Identifies openings on segmented point cloud created by **Walls** command.	
Parameters	--project <string PATH>	Project file to open
	--floor <int ID>	Select floor ID to perform openings finding
	--step <double STEP>	Openings finding is performed on a 2D (wall/plane) grid with step STEP
	--pointcloud <int ID>	Openings finding is performed on a pointcloud with id ID
	--minopeningpoints <double MIN_OPENING_POINTS=0.35>	Coefficient of minimum required number of inner opening points <0,1>
	--extwallwidth <double EXT_WALL_WIDTH=0.3>	Assumed width of external walls [m]
	--tolwallwidth <double WALL_WIDTH_TOLERANCE=0.9>	Tolerance of average wall width <0,1>

Command name	**Explorer**	
Description	Allows visualization of the project database entities using Easy3D library.	
Parameters	--pointclouds	Displays point clouds one by one
	--rooms	Displays all rooms one by one
	--allrooms	Displays all rooms in a single view
	--orooms	Displays all rooms with openings in a single view
	--planes	Displays planes one by one
	--allplanes	Displays all planes in a single view
	--oplanes	Displays all planes with openings in a single view
	--allwalls	Displays all walls in a single view
	--owalls	Displays all walls with openings in a single view
	--ogwalls	Displays all walls (in greyscale) with openings in a single view

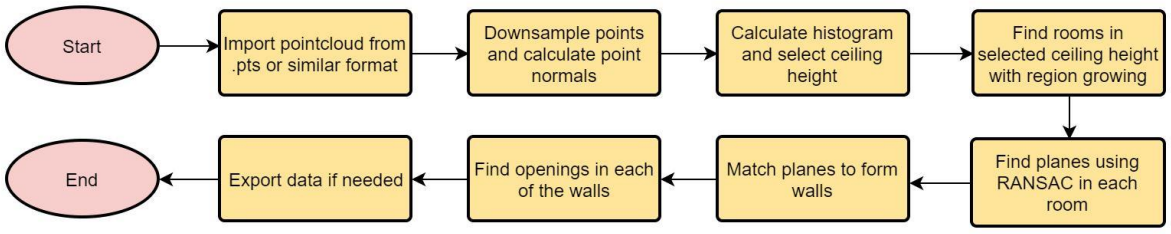


Figure 2: Typical steps involved in point cloud transformation into digital BIM model.

Examples

Simple office building, one floor

The first example consists of a relatively simple office building with one floor. There was no equipment at the time of laser scanning. The original data set consists of 85 316 410 points, the original point cloud is shown in Fig. 3.1.

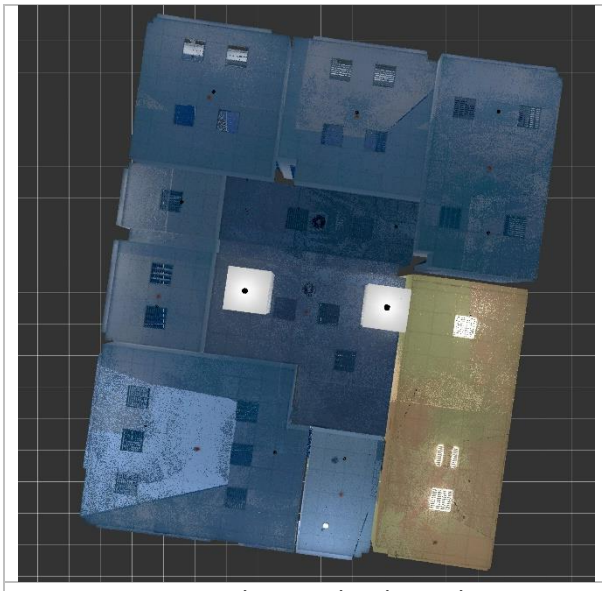


Fig. 3.1 Original point cloud visualization

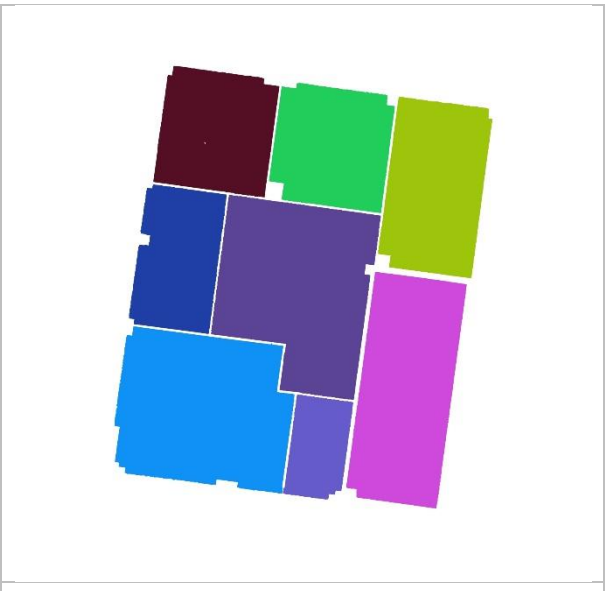


Fig. 3.2 Detected rooms

First, the new project has been created using *project* command, passing project location on input

```
./project --project C:/Pointclouds/Project.spcl
```

Then the original point cloud (stored in C:/Pointclouds/example.xyz file) was imported into the project database

```
./project --project C:/Pointclouds/Project.spcl --import C:/Pointclouds/example.xyz
```

The imported point cloud gets id 0, as it is the first cloud in database. To verify, the database can be inspected using *project* command

```
./project --project C:/Pointclouds/Project.spcl --pointclouds
```

In the next step it is often recommended to down-sample the original point cloud, as the density of the typical laser scans is high. For the vectorization, lower density is sufficient and leads also to lower processing time. In this case we have used sampling step 0.01 (1 cm^3) and during down-sampling we request computation of normals for each down sampled point

```
./downsampler --project C:/Pointclouds/Project.spcl --pointcloud 0 --step 0.01 --normals
```

The down-sampled point cloud is stored with id 1 in the database. Next, the detection of individual rooms is performed. By inspecting the point cloud, we have identified the floor and ceiling levels and use all points in between (excluding points representing floor and ceiling) for room finding. In the same step, we also identify all vertical planes within the floor

```
./rooms --project C:/Pointclouds/Project.spcl --pointcloud 1 --step 0.02 --planes --zlimit "368.0 370.45 371.40"
```

The result is illustrated in Fig. 3.2 where individual rooms are represented as regions filled with unique color and in Fig. 3.3. which shows individual (vertical) planes identified for given floor.

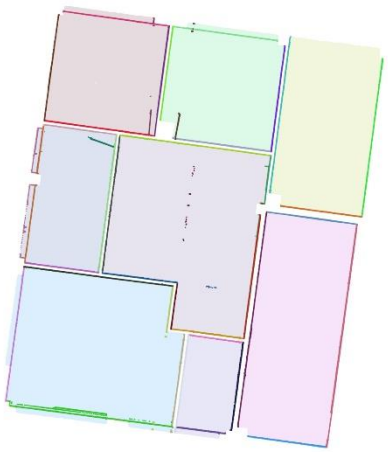


Fig. 3.3: Identified vertical planes

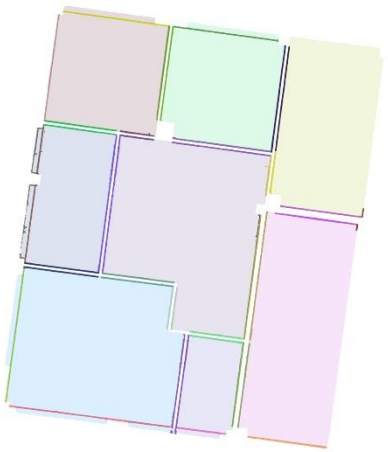


Fig. 3.4: Identified walls

In the next step, the individual walls can be detected using *walls* command. In this example, we aim to identify walls in the given floor with thickness 0.1 - 0.3 m. Length of at least 1.0 m is required in the wall direction.

```
./walls --project C:/Pointclouds/Project.spcl --floor 1 --minlegth 1.0 --minthickness 0.1 -- maxthickness 0.3
```

The result is shown in Fig. 3.4 where individual walls detected are drawn using unique colors. Note that the point cloud was made only from indoors, so there are no façade points. As a result, the outer walls could not be fully identified.

Finally, the openings (windows and doors) can be identified using *openings* command

```
./openings --project C:/Pointclouds/Project.spcl --floor 1 --step 0.02 --pointcloud 1 --extwallwidth 0.3 --tolwallwidth 0.9 --minopeningpoints 0.35
```

The results are shown in Figs 3.5 and 3.6.

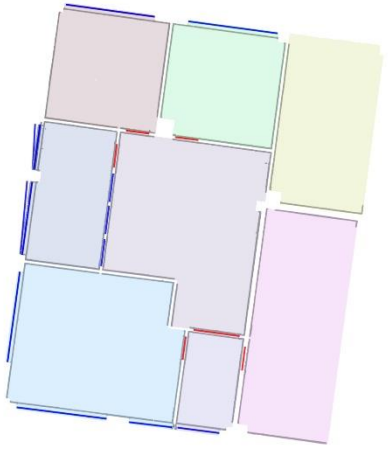


Fig. 3.5 Identified openings (top view)

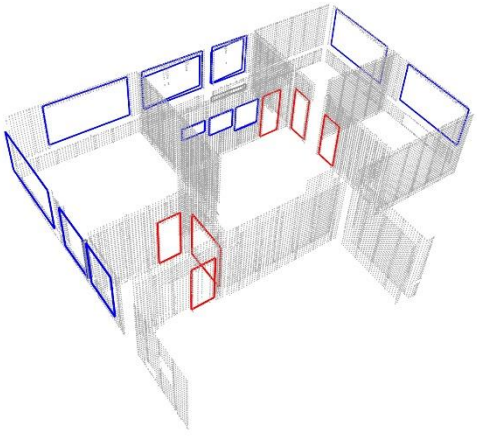


Fig. 3.6 Identified openings (perspective)

The central room (see Fig. 3.5) is visualized using original point cloud (Fig. 3.7) data to illustrate the performance of opening detection for a wall with three openings of different size and one door.

Fig. 3.7 Visualization of central room with original point cloud data set.

Some statistics of the project is summarized in Table 3.1 below.

Parameters	
Original points	85 316 410
Downsampled points (1x1x1cm grid)	12 358 178
Found rooms	8
Found vertical planes	93
Found wall segments	36
Found openings	6 doors 3 inner windows 10 outer windows 1 indeterminate opening

Administrative single floor building with equipment.

This example illustrates the results for more complex building. At the time of scanning, the building was fully equipped with furniture. Some of the parameters of this example are summarized below.

Parameters	
Original points	62 947 576
Downsampled points (1x1x1cm grid)	20 213 992
Found rooms	21
Found vertical planes	247

Found wall segments	84
Found openings	14 doors 23 windows 2 indeterminate openings

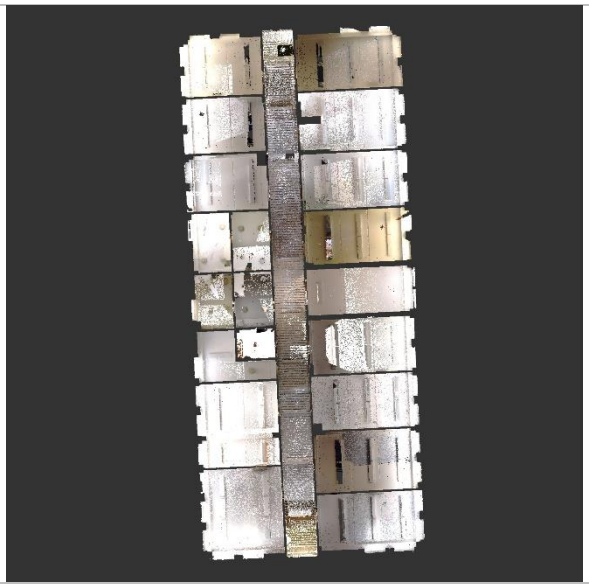


Fig. 4.1 Original point cloud visualization

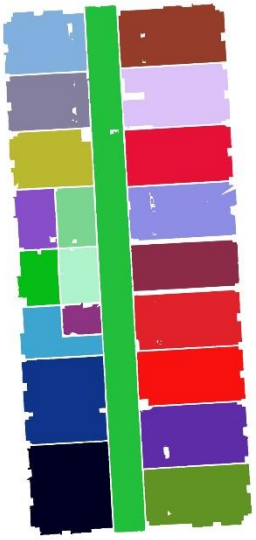


Fig. 4.2 Rooms detection

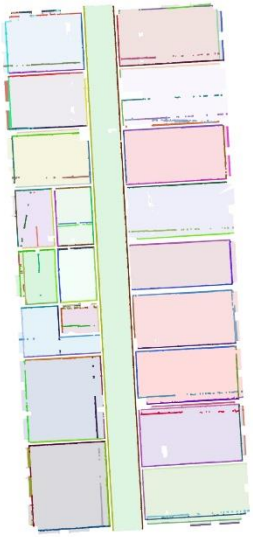


Fig. 4.3 Vertical plane detection

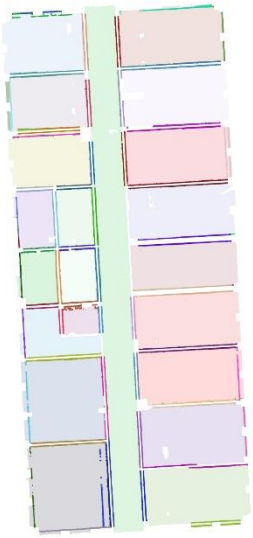


Fig. 4.4: Wall detection

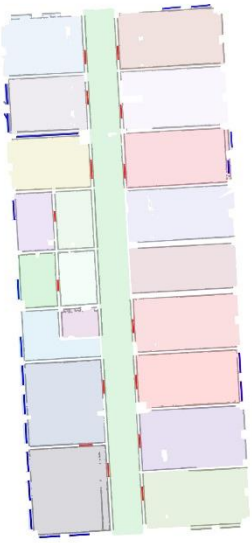


Fig. 4.5 Identified openings (top view)

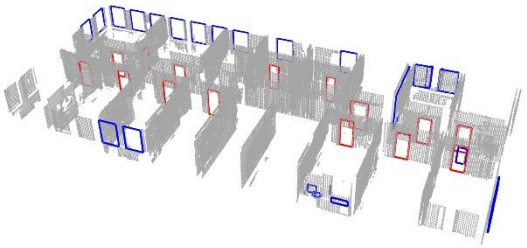
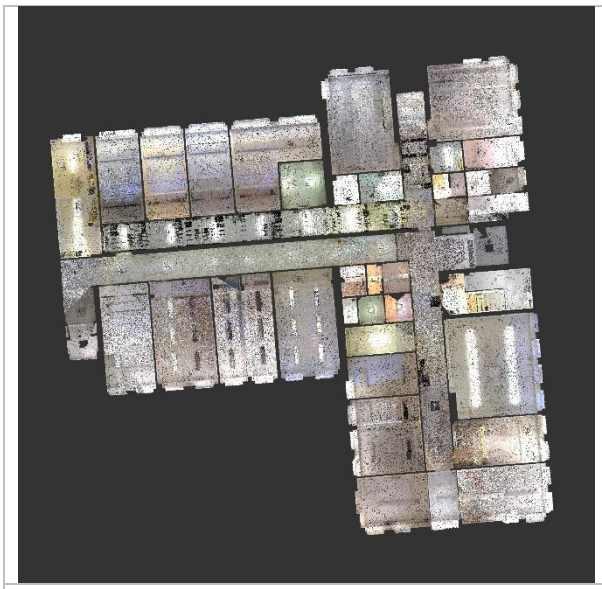


Fig. 4.6: Identified openings (perspective)

Single floor of a furnished office building – more complex floor

Parameters	
Original points	108 200 686
Downsampled points (1x1x1cm grid)	34 672 667
Found rooms	38
Found vertical planes	437
Found wall segments	146
Found openings	23 doors 38 windows 13 indeterminate opening



5.1 Original point cloud visualization

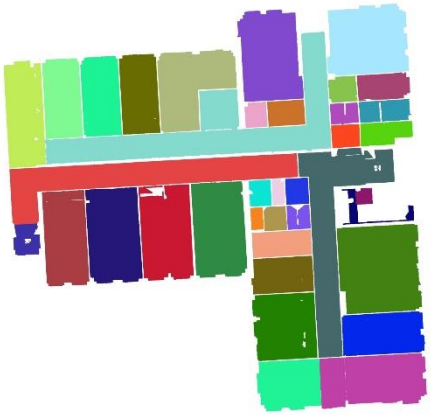


Fig. 5.2 Rooms detection

Fig. 5.3 Vertical plane detection

Fig. 5.4: Wall detection

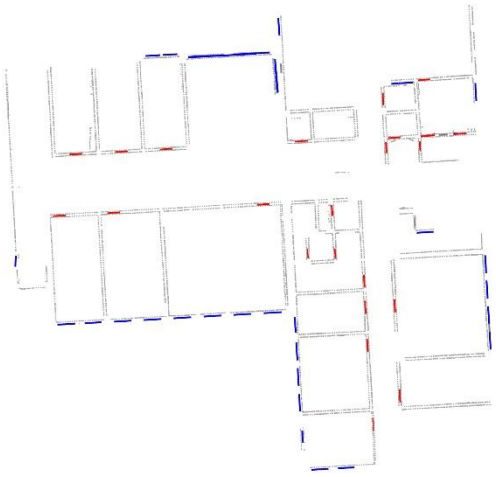


Fig. 5.5 Identified openings (top view)

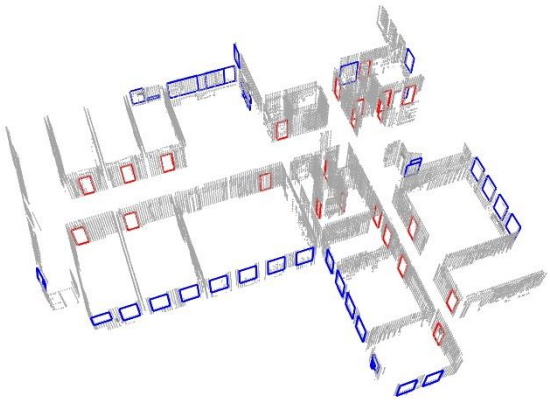


Fig. 5.6: Identified openings (perspective)

Acknowledgement

The development of this software was supported by the Technology Agency of the Czech Republic, under program of the National Competence Center 1 as project “Center for Advanced Materials and Efficient Buildings (CAMEB), project registration number: TN01000056.

References

(1) E. Dvořáková, P. Patzák, D. Rypl, J. Voříšek: PointCloud2BIM library Reference manual, available online at http://mech2018.fsv.cvut.cz/pc2bim/doku.php, 2020.

(2) E. Dvořáková, P. Patzák, D. Rypl, J. Voříšek: PointCloud2BIM library web pages, http://mech2018.fsv.cvut.cz/pc2bim/doku.php, 2020.