World Academy of Science, Engineering and Technology
International Journal of Civil and Architectural Engineering
Vol:18, No:11, 2024

Automated Transformation of 3D Point Cloud to Building Information Model: Leveraging Algorithmic Modeling for Efficient Reconstruction

Radul Shishkov, Petar Penchev

Abstract**—**The digital era has revolutionized architectural practices, with Building Information Modeling (BIM) emerging as a pivotal tool for architects, engineers, and construction professionals. However, the transition from traditional methods to BIM-centric approaches poses significant challenges, particularly in the context of existing structures. This research presents a technical approach to bridge this gap through the development of algorithms that facilitate the automated transformation of 3D point cloud data into detailed BIM models. The core of this research lies in the application of algorithmic modeling and computational design methods to interpret and reconstruct point cloud data — a collection of data points in space, typically produced by 3D scanners — into comprehensive BIM models. This process involves complex stages of data cleaning, feature extraction, and geometric reconstruction, which are traditionally time-consuming and prone to human error. By automating these stages, our approach significantly enhances the efficiency and accuracy of creating BIM models for existing buildings. The proposed algorithms are designed to identify key architectural elements within point clouds, such as walls, windows, doors, and other structural components, and to translate these elements into their corresponding BIM representations. This includes the integration of parametric modeling techniques to ensure that the generated BIM models are not only geometrically accurate but also embedded with essential architectural and structural information. This research contributes significantly to the field of architectural technology by providing a scalable and efficient solution for the integration of existing structures into the BIM framework. It paves the way for more seamless and integrated workflows in renovation and heritage conservation projects, where the accuracy of existing conditions plays a critical role. The implications of this study extend beyond architectural practices, offering potential benefits in urban planning, facility management, and historical preservation.

Keywords**—**Algorithmic modeling, Building Information Modeling, point cloud, reconstruction.

I. Introduction

THE integration of point cloud data into BIM has garnered significant attention in recent years, with several successful research efforts aimed at automating the process of converting dense point cloud data into accurate and detailed BIM models.

Although there are some successful research papers on the topic of converting point cloud data into BIM, there is currently no commercially implemented methodology that has gained widespread adoption in the industry. The existing research offers promising solutions, but these have not yet translated into practical tools that can be easily used by professionals in the architecture, engineering, and construction (AEC) sectors.

Part of the problem lies in the technical complexity of the proposed solutions. Many of these methodologies depend heavily on advanced coding programs and libraries such as Python, MATLAB, and C++. While these tools are powerful and flexible, they require a level of programming expertise that is not commonly found among architects and other AEC professionals. This reliance on complex coding environments creates a barrier to the widespread adoption of these methods, as they are not easily accessible to most potential users.

To overcome this barrier and develop a more usable method for converting point cloud data into BIM models, it is essential to employ an approach that is more user-friendly and specifically tailored to the needs of architects. Our research addresses this need by utilizing visual scripting through the Grasshopper and Rhino platforms, combined with the extensive capabilities of Revit and ArchiCAD—two of the most widely used BIM software applications among architects. This combination allows users to generate BIM models directly from point cloud data without the need for extensive coding knowledge, making the process more accessible and practical for everyday use.

Grasshopper and Rhino are particularly well-suited for this research due to their wide availability of open-source libraries, which can be used in a user-friendly manner. These platforms offer a variety of potential geometry outputs, providing the flexibility needed to handle complex architectural forms and details. Additionally, the visual scripting environment in Grasshopper enables users to build complex algorithms through a graphical interface, which simplifies the process of creating and manipulating geometry.

Our approach leverages an algorithmic methodology to extract specific data from the point cloud and use it to rebuild the model in different ways. By automating the extraction and reconstruction processes, we aim to enhance the accuracy and efficiency of the BIM model generation while maintaining a high level of flexibility in how the models are created.

This research is closely related to the work of Bassier [1], who also used the Rhino.Common API for data extraction and model reconstruction via RhinoInside Revit. However, our methodology differs in the logical operations integrated into the Grasshopper algorithm, which adds a layer of versatility to the

Radul Shishkov is Master Architect, assistant professor and PhD graduate, with the University of Architecture, Civil Engineering and Geodesy, Sofia, Bulgaria (phone: +359 878 167 475; e-mail: rshishkov_far@uacg.bg).

Petar Penchev is Master Architect and PhD candidate, with the University of Architecture, Civil Engineering and Geodesy, Sofia, Bulgaria (e-mail: ppenchev_far@uacg.bg).

process. Additionally, we propose two separate approaches for rebuilding the model: one using the Grasshopper ArchiCAD library and another using a Python script inside Revit. These dual approaches provide options for users depending on their specific needs and the software they are most comfortable with.

Our work is structured as follows: In Section II, we provide background information and review previous work in the field. Section III presents our methodology for algorithmic data extraction from point clouds. In Section IV, we describe the methodology for reconstructing the geometry in a BIM format. Finally, in Section V, we present our conclusions and suggest directions for future research and development in this area.

II. Related Work

The automatic reconstruction of 3D models has long been a challenging area of research, with numerous studies published over the years exploring a variety of reconstruction methods. Despite the development of a wide range of modeling techniques—spanning from highly manual to nearly fully automated approaches, the task of fully automating the reconstruction process remains a significant challenge that continues to attract considerable attention and effort from experts in the field.

Ochmann et al. [1] create methodology by registering multiple indoor 3D point cloud scans and then decomposing the point cloud into different rooms using a probabilistic clustering algorithm. The algorithm considers the visibility between points to determine room boundaries and assigns points to specific rooms iteratively. Once the rooms are segmented, the method detects doors between adjacent rooms and constructs a graph that encodes the building's topology, representing rooms as nodes and doors as edges.

In another step of this research [2], Ochmann et al. present a methodology for the automatic reconstruction of parametric building models from indoor point cloud data, with the primary goal of creating detailed, editable, and semantically rich BIM from point clouds captured within existing buildings. The approach addresses the challenges associated with indoor environments, such as clutter, occlusions, and the complexity of architectural elements.

The process begins with the segmentation of the point cloud into different regions corresponding to individual rooms. The point cloud is then further segmented into planar surfaces that represent walls, floors, and ceilings, which are classified based on their orientation and spatial relationships. Once these surfaces are identified, the method generates a parametric model by fitting geometric primitives, such as planes and cuboids, to the segmented data. This allows for the creation of walls, doors, windows, and other architectural elements that are essential for BIM.

Ochmann et al.'s work [1] is significant because it provides a practical approach to automating the creation of parametric BIM models from indoor point clouds. The research contributes to the field by offering a method that balances automation with the need for detailed, semantically rich models, making it particularly relevant for applications in renovation, facility management, and architectural analysis, where accurate and editable BIM models are essential.

In Ochmann's latest research [3], key innovation is formulating the reconstruction task as an integer linear programming problem. This allows for an exact solution that enforces consistency and connectivity between volumetric wall entities, unlike previous methods that often resulted in disconnected or paper-thin surfaces.

Román et al. [4] also do not primarily focus on translating point cloud data into BIM models; instead, their approach begins by rationalizing curved walls through the application of RANSAC clustering techniques. Xiong et al. [5] focus on reconstructing the semantics in the 3D model, but also do not tackle BIM transformations.

Some research [6], [7] emphasize on generating 2D plans from 3D point cloud data, a crucial step in the process of creating a comprehensive BIM model, other studies [8]-[10] concentrate on specific BIM elements, such as walls, doors, and ceiling.

The papers by Bassier et al. [11], [12] present an advanced approach for the automated procedural modeling of wall geometry from point clouds. The authors use a combination of machine learning models and software tools like Rhino.Inside and the Revit API to automate the process. The methodology begins with preprocessing the point cloud data by representing it as a voxel octree, which simplifies the data and prepares it for further processing. Planar patches representing walls, floors, and other structural elements are then extracted from the vowelized data.

These planar patches are classified using a Random Forest model, which segments the data into groups that correspond to individual walls. Once the walls are identified, the authors of [12] use the Rhino.Inside and Revit API to create native BIM objects directly from the classified data.

Bassier et al.'s work [12] is particularly relevant to our research, since Rhino the platform chosen by us to create most of the algorithms, but aim to circumvent the Rhino.Inside API, in order to achieve more user-friendly method for BIM modeling in the last steps of recreating the scan in Revit.

The extensive research on previous work shows significant advancements in the topic, but the developed techniques have not been widely adapted by professionals yet. Our proposed methodology aims to leverage more user-friendly algorithmic modeling techniques, that depend on a more intuitive, geometrically driven approach, that is not only easier to reproduce but also more accessible for professionals who may not have extensive technical expertise. In the following section, we will detail our methodology, which builds on these principles to create a more practical and widely applicable process.

III. Methodology

A. Overview

Summary of our proposed process is shown in Fig. 1. The point cloud is imported in Rhino3D and Grasshopper, where it is used as input data for algorithms, which extract the data needed for the reconstructing the model with BIM primitives.

The reconstruction is also done algorithmically in two separate ways, using Grasshopper AchiCAD live connection and pyRevit. Thus, the whole process of transforming the point cloud into BIM model is automated. The presented methodology focuses on reconstructing walls and openings in them.

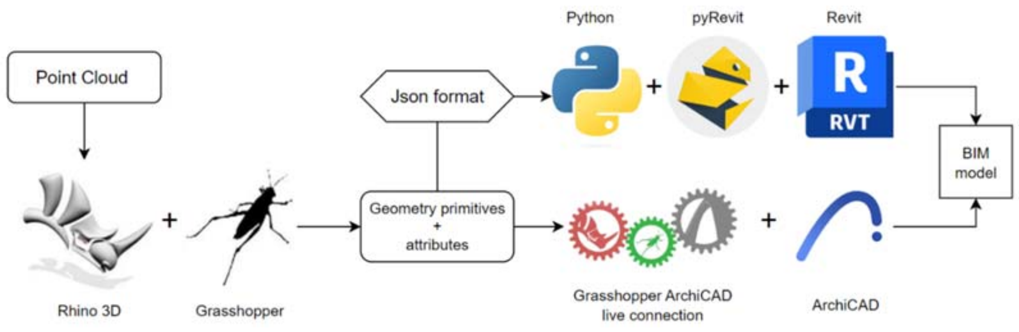


Fig. 1 Process map

For importing and handling the 3D point cloud data, we utilized the “Cockroach” library [13] within Grasshopper, which provides a comprehensive set of tools for processing point clouds. This library is based on the “Open3D” library [14], a widely recognized open-source library for 3D data processing, and it integrates features from the Computational Geometry Algorithms Library [15] and the “Cilantro” library [16]. The combination of these libraries within the Cockroach framework allows for efficient and flexible manipulation of large point cloud datasets, enabling us to perform complex operations such as segmentation, plane detection, and data extraction with high accuracy and speed.

B. Data Processing

The algorithm we developed begins by preprocessing the point cloud thro ugh a manual step that involves cropping the cloud with a predefined geometric volume, isolating a single floor of the scanned building. This manual intervention is necessary to focus the analysis on one floor at a time, ensuring that the subsequent steps are accurate and manageable. Once this is done, the algorithm proceeds automatically, divided into several key parts: extracting data for external walls, extracting data for interior walls, identifying window openings, determining levels, and finally, exporting the data into other formats suitable for further BIM modeling. This structured approach ensures that all critical architectural elements are accurately captured and prepared for integration into the BIM workflow.

After cropping and cleaning the point cloud, the next step involves down sampling the data. Down sampling is crucial for optimizing the running time of the algorithm, especially when dealing with large and dense point clouds. By reducing the number of points, we can significantly decrease the computational load, making the algorithm more efficient while still retaining the essential geometric details needed for accurate modeling. This optimization step is a critical part of the workflow, as it balances the need for detail with the practical limitations of computational resources.

C. Walls Data Extraction

The process of analyzing and extracting information for both external and internal walls begins with clustering the points into planes. This is a foundational step in identifying the flat surfaces that constitute the walls of the building. To achieve this, we first compute the normal vector for each point in the cloud, which provides information about the orientation of the surface at that point. The computed normals are then fed into the Cilantro Cloud clustering algorithm, which groups the points based on their alignment into planes. This algorithm has proven to be faster and more effective than the traditional RANSAC algorithm, which we previously ran in Python for comparison.

The improved performance of the Cilantro algorithm is particularly important because the calculation of normals and the clustering of points into planes account for approximately one-third of the total computational time of the algorithm. By optimizing this step, we were able to achieve cleaner and more accurate results in a shorter amount of time, which is crucial for the overall efficiency of the point cloud processing workflow.

The next step in our methodology involves cleaning the clusters obtained from the point cloud data by introducing geometrical thresholds based on size and spatial positioning. This cleaning process is essential for ensuring that only the relevant points associated with walls and window planes are retained for further analysis. By applying these thresholds, we effectively cull out horizontal clusters and other sets of points that do not contribute to the vertical planes of interest and are irrelevant to our focus on walls and windows

Following the cleaning process, the algorithm proceeds with a series of steps designed to project and interpolate the points associated with the vertical planes. Fig. 3 shows these steps. Initially, the points from each vertical plane are projected onto a horizontal plane (a). This projection allows us to work with a simplified 2D representation of the vertical structures. The algorithm then interpolates a line through the projected points, intersecting them with a defined threshold (b). The outcome of this process is a rough 2D representation of the walls and windows (c), which forms closed regions that define interior spaces and the external boundary of the building (d). Valero et al. [17] also solve intersections to create an enclosed area, but they do it for planes, for which they need to solve topological connectivity first.

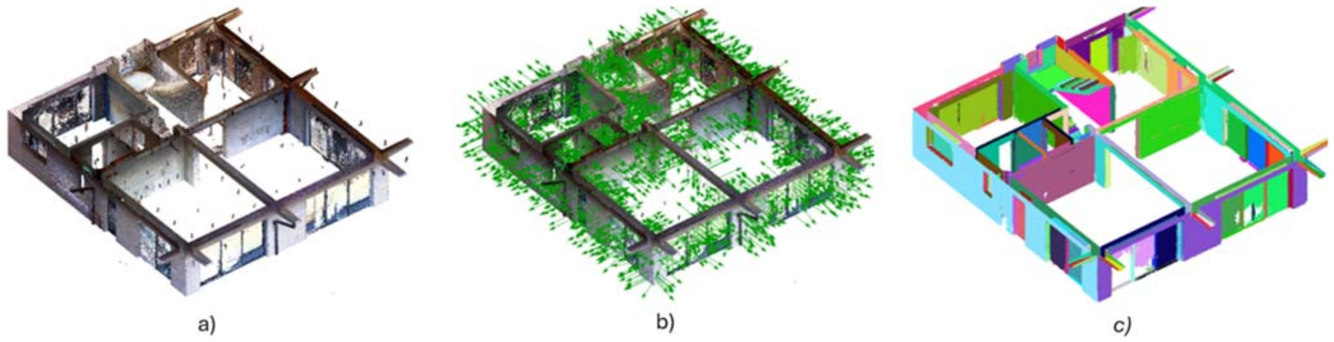


Fig. 2 (a) Cropped point cloud, (b) Point cloud with calculated normal, (c) Clustered point cloud by planes

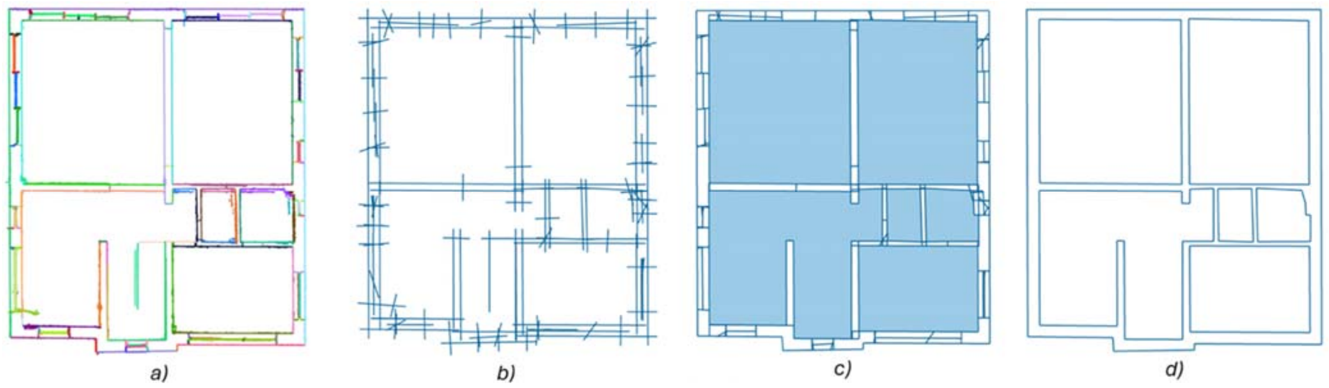


Fig. 3 Extracting 2D walls data from the point cloud; (a) projecting points from planes; (b) interpolate lines and extend; (c) create regions; (d) extract boundaries

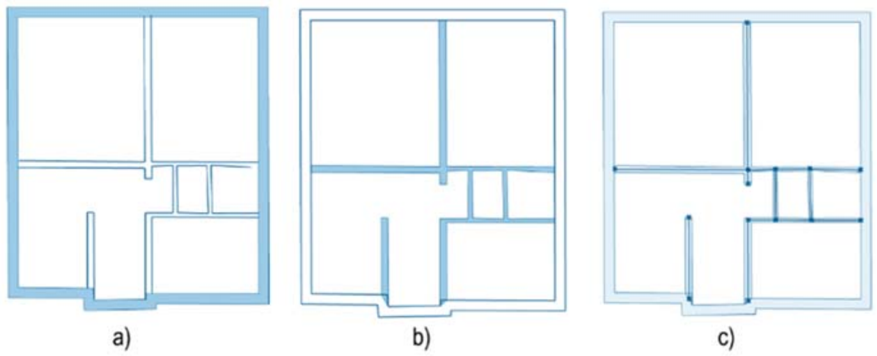


Fig. 4.Walls data extraction: (a) Exterior walls boundary; (b) Region Boolean difference; (c) Axis extraction

Once the external 2D wall contours and their corresponding widths have been determined, these contours serve as the basis for extracting the boundaries of the internal walls (Fig. 4 (a)). This is achieved through a Boolean difference operation between the regions defined by the external walls and the remaining unprocessed internal wall regions. By subtracting the external wall regions, the algorithm isolates the projected regions corresponding to the internal walls (Fig. 4 (b)). These projected internal wall regions are then utilized as input data for the "extract central line from buffer" algorithm, which is a component of the "Metacity" library [18]. This algorithm processes the internal wall regions by identifying and extracting the central axis of each wall. To ensure that the central axes are not overly complex and remain manageable for further processing, they are simplified within a predefined threshold value. This simplification process reduces the number of vertices along the axes, while still preserving the overall geometry and alignment of the internal walls. The result is a set of central axes that accurately represent the internal walls' layout, ready for integration into the overall BIM model (Fig. 4 (c)).

The widths are again calculated, as averaged, sampled distances from the axis to the corresponding region boundary. This part of the algorithm ends with merging the two lists of axes (exterior and interior walls) and their corresponding attributes (width and height), thus preparing the data as input data for further algorithms that would reconstruct the walls as BIM objects. The subsequent steps in the algorithm involve extracting the necessary data for the reconstruction of façade windows and internal walls.

D. Windows and Doors Data Extraction

To reconstruct the windows, the first step is to identify and extract the point clusters that represent the surfaces of the façades. This is accomplished by selecting the clusters that have the closest projection to the previously determined 2D boundaries of the exterior walls. By focusing on these clusters, the algorithm effectively isolates the relevant points that are associated with the façade surfaces.

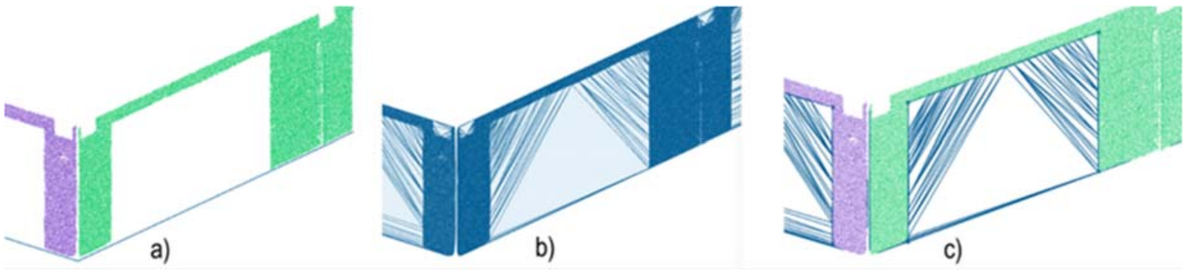


Fig. 5 Windows data extraction: (a) extract façade plane clusters; (b) Delaunay triangulation mesh; (c) Extract windows faces and simplify boundary

For each identified cluster of façade points, a Delaunay triangulation is then performed [19]. This triangulation creates a dense mesh over the solid portions of the walls, while areas where the windows are located result in disproportionately distorted mesh faces. These distortions serve as key indicators, providing a basis for extracting the mesh faces that correspond to window openings. The faces belonging to the wall opening have significant bigger perimeter compared to the rest of the faces in the triangulated wall meshes and this is used in the algorithm to dispatch them. The algorithm then reconstructs the window boundaries by applying certain thresholds and constructing a bounding rectangle on the adjacent plane. The dimensions of the rectangles, along with their distances to the ground plane, are prepared as attribute data for windows size and seal. These are attributed to their positions, represented by a projected center point. To achieve a successful reconstruction of the windows in the BIM model, it is essential to also extract information about the adjoining wall axis for each window. These data are identified and incorporated as an ID attribute, encoding the relationship between the windows and their corresponding walls. An analogous approach to that used for windows is employed to retrieve and organize the data required for the reconstruction of the doors in the model. The algorithm is applied to the list of extracted interior wall axes, where a wall ID attribute is added to each, along with the corresponding center points and dimensions for the doors. This ensures that all necessary information is accurately captured to facilitate the precise placement and sizing of doors within the BIM model. The information, retrieved with the algorithm (Fig. 7) at this point, is sufficient to also reconstruct algorithmically the walls and the openings as b-rep geometry (Fig. 6). To facilitate BIM reconstruction, the primitive geometries, including lines and points, along with their associated attribute data, are exported in JSON format using the BearGIS library [20]. The data are organized into three distinct files corresponding to walls, windows, and doors, respectively. These files serve as input for the subsequent algorithm, which is executed in Revit using pyRevit [21].

IV. Reconstructing the Geometry as BIM Objects

A. Revit Workflow Overview

Autodesk Revit serves as the primary BIM authoring tool. It is one of the most widely used BIM software applications globally, recognized for its robust capabilities in architectural, structural, and MEP design. Autodesk Revit offers a comprehensive API that enables power users and external application developers to create custom tools and integrate them seamlessly into the Revit environment. This flexibility, coupled with a large developer community and extensive resources, facilitates the relatively straightforward creation of add-ons and extensions tailored to specific project needs.

pyRevit functions as a Rapid Application Development (RAD) environment for Autodesk Revit. It enables the rapid development and integration of custom tools directly into the Revit user interface, enhancing user experience and workflow efficiency. pyRevit provides a library of pre-built input and output UI components and other functionalities that streamline the tool creation process, making it easier for developers to build, test, and deploy new tools within Revit quickly.

Python is utilized as the primary programming language due to its widespread popularity, flexibility, and ease of use. Known for its simple and readable syntax, Python supports both object-oriented and structured programming paradigms, making it suitable for quick prototyping and iteration. Python's versatility and rapid development capabilities make it particularly well-suited for developing tools and scripts that interface with Revit's API and pyRevit.

To ensure seamless data exchange between the earlier steps and the BIM authoring workflow, the language-independent format JSON (JavaScript Object Notation) is used. JSON is an open standard file format that employs human-readable text to store and transmit data objects consisting of attribute-value pairs and arrays. It effectively facilitates the transfer of complex geometric and attribute data from the processing environment to the BIM software.

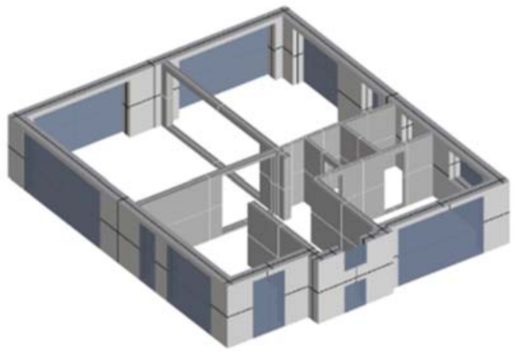


Fig. 6 Results from the algorithm: b-rep geometry

B. Scope of Work

The scope of this work encompasses the creation of several key building elements within the Revit environment, including walls, doors and windows.

Autodesk Revit employs a structured data organization and classification system that is essential to understand for effective model creation. In Revit, all elements are divided into various categories, and within each category are "Families" that further contain specific "Types." At the lowest level of this hierarchy are the individual elements, also known as instances. Each instance represents a unique occurrence of an element within the model.

The creation process within Revit must adhere to this hierarchical structure. It is also important to consider specific requirements inherent to Revit's data model, particularly the dependency relationships between different elements. For example, both doors and windows are classified as hosted elements, meaning they must be placed within a wall and cannot exist independently. This dependency dictates that the workflow sequence begins with the creation of walls, which must precede the placement of doors and windows.

To achieve the desired outcomes, the creation of each building element follows a structured sequence of steps:

1) Browse for a JSON File: It begins by selecting the appropriate JSON file that contains the geometry and attribute data for the specific building element to be created.
2) Read and Process JSON Data: The JSON data are then read and processed to extract all relevant information, such as dimensions, positions, and associated parameters.
3) Create Element Based on JSON Data: Using the extracted data, the corresponding element—be it a wall, door, or window—is created within the Revit environment, adhering to Revit's data hierarchy and organizational rules.
4) Assign Parameter Values: Finally, additional parameter values, such as material properties, levels, and other attributes, are assigned to the created element based on the information contained in the JSON file.

C. Walls' Reconstruction

The process of wall reconstruction begins with an investigation of the Revit API methods available for generating the specific building element [22]. This identifies the necessary information required for wall creation, which, in turn, defines the structure of the JSON file used to input data. There are several methods for creating walls, each of which requires different arguments. The chosen method for this workflow utilizes the following parameters: the current Revit document (Document), the wall's location curve (Curve), the associated level (Level), and a Boolean for structural use (Structural).

To effectively use the Revit API for wall creation, the JSON file must contain both initial and additional data. Initial Data (Minimum Required for Wall Creation):

1) Document: refers to the currently opened Revit document where the wall will be created.
2) Curve: represents the location curve for each wall, defined by the centerline. The curve is determined by its start and end point coordinates.
3) ElementId: represents the ID of the level on which the wall is placed; typically, the standard ground level of the Revit document is used.
4) Bool (Structural use): indicates whether the wall is structural or non-structural; in this case, non-structural is used.

Additional Data:

1) Wall Width: is determined by the Revit Wall Family Type. For this tool, single-layer Basic Wall Types are used. If a Wall Type with the required width does not exist in the document, a new Wall Type is created.
2) Wall Height: is assigned once the wall has been created to match the desired specifications.
3) Wall ID: is a unique identifier assigned to each created wall to track the host walls for windows and doors.

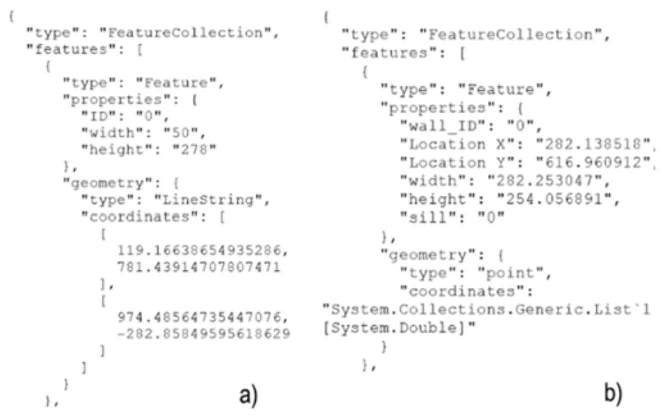
The tool for wall creation follows a systematic sequence of operations:

1) Browse JSON File: utilizes the built-in functionality of pyRevit to select the appropriate JSON file containing the wall data.
2) Read JSON Data: uses Python's JSON module to read and parse the data from the selected JSON file.
3) Check Current Wall Types: verifies the existing wall types in the current Revit model. If a wall type with the required specifications does not exist, it creates a new wall type as needed.
4) Create Walls and Set Parameters: executes the wall creation process in Revit and sets the necessary parameter values to align with the data extracted from the JSON file.

The data in the JSON file follow the structure shown in Fig. 7.

The algorithm goes through the following steps:

1) Browse JSON file using built-in pyRevit functionality.
2) Read JSON data using Python JSON module.
3) Check current Revit model Wall types and create new if needed.
4) Creation of the walls and set desired parameter values to match data.



```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "ID": "0",
        "width": "50",
        "height": "278"
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            119.16638654935286,
            781.43914707807471
          ],
          [
            974.48564735447076,
            -282.85849595618629
          ]
        ]
      }
    },
```

a)

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "wall_ID": "0",
        "Location X": "282.138518",
        "Location Y": "616.960912",
        "width": "282.253047",
        "height": "254.056891",
        "sill": "0"
      },
      "geometry": {
        "type": "point",
        "coordinates":
"System.Collections.Generic.List`1
[System.Double]"
      }
    },
```

b)

Fig. 7 JSON data structure: (a) walls; (b) windows and doors

D. Windows and Doors Reconstruction

The creation process for doors and windows in Revit follows a similar logic, but multiple options exist depending on the arguments used. The method selected for this purpose is "NewFamilyInstance," which is versatile and applicable to various element types. To ensure accurate placement and compatibility, the method chosen uses specific parameters tailored to the nature of door and window elements, including Element Location, Element Family Type, Host Element, and Structural Type

To create doors and windows effectively, the JSON file must contain the following initial data:

1) XYZ: The precise location of the door or window, defined by XYZ coordinates.
2) FamilySymbol: The specific Family Type to be used for the door or window, as per Revit's organizational hierarchy.
3) Element: The host wall where the door or window will be placed.
4) Structural Type: Indication of whether the element is for structural use, based on Revit's information hierarchy.

Additional data, such as width and height, are typically governed by the Family Type in Revit. However, this tool uses a workflow that assigns these dimensions at the instance level rather than the Family Type level. This allows for more flexibility and customization in the placement of each element. Additional parameters include:

1) Width and Height: is assigned at the instance level to allow customization.
2) Wall_ID: identifies the host wall for each door or window.
3) Sill: is the height of the window sill from the finished floor (applicable to windows only).

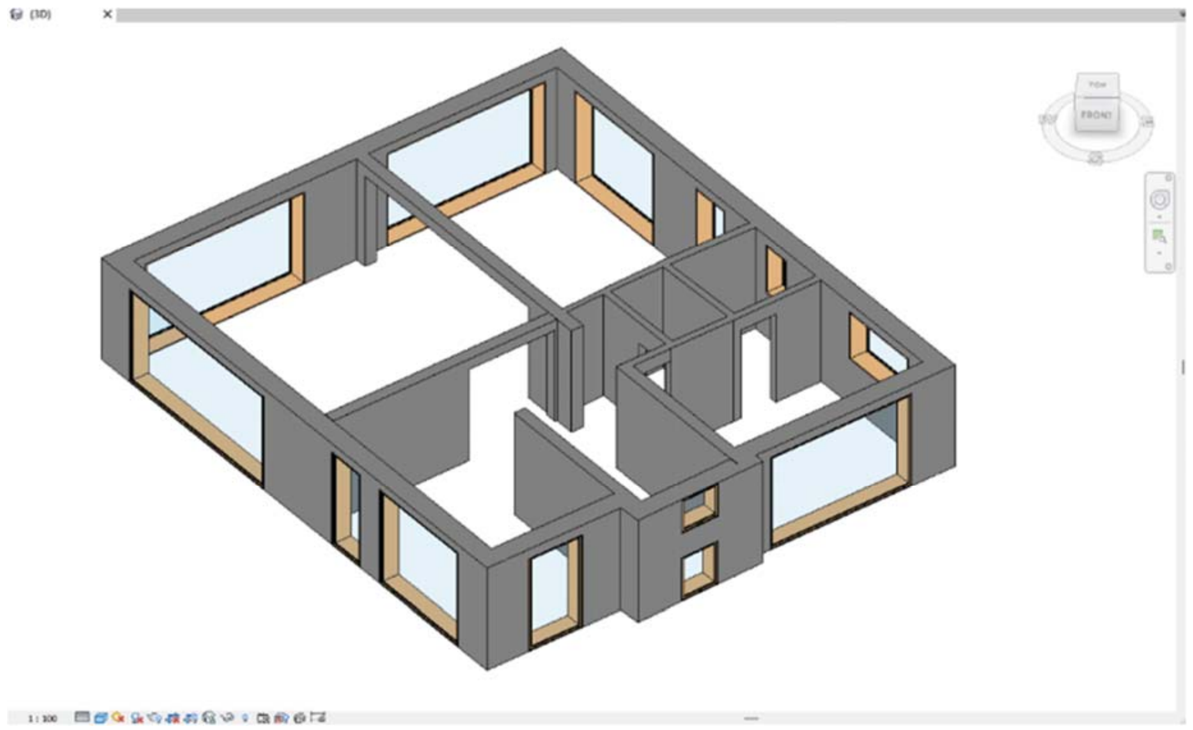


Fig. 8 Reconstructed model in Revit

The algorithm goes through the following steps:

1) Get all walls,
2) Get pre-loaded window/door Family Type,
3) Browse JSON file using built-in pyRevit functionality,
4) Read JSON data using Python JSON module,
5) Creation of the window/door and set desired parameter values to match data.

The tree scripts are run consecutively, thus creating the final result – reconstructed walls, windows and doors as BIM elements. Fig. 8 shows the generated model in Revit.

Additionally, the model is reconstructed in ArchiCAD (Fig. 9) to provide a comparative analysis of the results and explore an alternative method for BIM generation. This process utilizes the Grasshopper-ArchiCAD live connection, which allows for real-time data exchange between the two software platforms. By employing ArchiCAD's native components within the Grasshopper environment, the algorithm directly uses the originally extracted primitives (lines and points) as input data, with the associated attribute data serving as parameters for the ArchiCAD components.

This approach leverages the powerful algorithmic capabilities of Grasshopper for model creation, enabling a streamlined workflow that does not require data export or conversion into other formats, unlike the Revit-based method. However, one significant limitation of this method is that it necessitates running both Grasshopper and ArchiCAD simultaneously, which can increase computational demand and complicate the modeling process compared to the Revit approach. Despite this disadvantage, the integration provides a flexible and dynamic modeling environment that can be beneficial in scenarios where real-time adjustments and iterations are needed.

V. Conclusion and Future Work

The proposed methodology demonstrates a streamlined approach for automating the process of converting point cloud data into BIM. Through a series of well-defined steps and tools, the methodology simplifies complex tasks, reducing manual effort and improving overall workflow efficiency. The results of this research include a set of ready-to-use algorithms that can significantly enhance the task of generating BIM models from laser-scanned point clouds. These algorithms offer a practical solution by automating, which can save considerable time and resources in real-world applications.

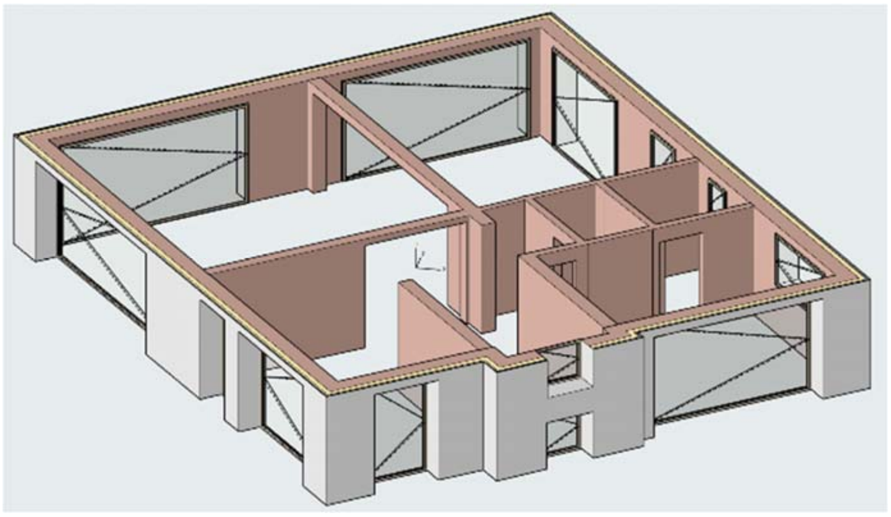


Fig. 9 Reconstructed model in ArchiCAD

Moreover, the research provides solutions tailored to the two most widely used BIM software platforms, Autodesk Revit and ArchiCAD, ensuring that the developed methodology is versatile and accessible to a broad range of users. By offering an approach compatible with both software environments, the methodology allows for flexibility and choice, making it applicable to various project types and user preferences.

Future work in this area could involve more extensive testing of the algorithms on multiple diverse point cloud datasets to validate their robustness and adaptability across different building types and scanning conditions. Additionally, expanding the methodology to include the analysis and reconstruction of additional BIM elements, such as slabs and roofs, would further enhance its applicability and value. The algorithms developed through this research could also serve as a foundation for generating training data for neural networks, potentially advancing the research.

Acknowledgment

This research was conducted as part of the scientific research program at the "University of Architecture, Civil Engineering and Geodesy," Sofia. The point cloud was provided by Mladen Tsenov.

References

[1] Ochmann, Sebastian & Vock, R. & Wessel, R. & Tamke, Martin & Klein, Reinhard. (2014). Automatic generation of structural building descriptions from 3D point cloud scans. GRAPP 2014 - Proceedings of the 9th International Conference on Computer Graphics Theory and Applications. 120-127.
[2] Ochmann, Sebastian & Vock, Richard & Wessel, Raoul & Klein, Reinhard. (2015). Automatic Reconstruction of Parametric Building Models from Indoor Point Clouds. Computers & Graphics. 54. 10.1016/j.cag.2015.07.008.
[3] Ochmann, Sebastian & Vock, Richard & Klein, Reinhard. (2019). Automatic reconstruction of fully volumetric 3D building models from oriented point clouds. ISPRS Journal of Photogrammetry and Remote Sensing. 151. 251-262. 10.1016/j.isprsjprs.2019.03.017.
[4] Román, J. & Lerones, Pedro & Llamas, Jose & Zalama, Eduardo & Gómez-García-Bermejo, Jaime. (2019). Towards the Automatic 3D Parametrization of Non-Planar Surfaces from Point Clouds in HBIM Applications. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLII-2/W15. 1023-1030. 10.5194/isprs-archives-XLII-2-W15-1023-2019.
[5] Xiong, Xuehan & Adan, Antonio & Akinci, Burcu & Huber, Daniel. (2013). Automatic Creation of Semantically Rich 3D Building Models from Laser Scanner Data. Automation in Construction. 31. 325–337.

10.1016/j.autcon.2012.10.006
[6] Okorn, B., Xiong, X., & Akinci, B. (2010). Toward Automated Modeling of Floor Plans
[7] Stojanovic, Vladeta & Trapp, Matthias & Richter, Rico & Döllner, Jürgen. (2018). Generation of Approximate 2D and 3D Floor Plans from 3D Point Clouds. 10.5220/0007247601770184.
[8] Anagnostopoulos, Ioannis & Pătrăucean, Viorica & Brilakis, Ioannis & Vela, Patricio. (2016). Detection of Walls, Floors, and Ceilings in Point Cloud Data. 2302-2311. 10.1061/9780784479827.229.
[9] Macher, Hélène, Tania Landes, and Pierre Grussenmeyer. 2017. "From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings" *Applied Sciences* 7, no. 10: 1030. https://doi.org/10.3390/app7101030
[10] Díaz Vilariño, Lucia & Verbree, Edward & Zlatanova, Sisi & Diakité, Abdoulaye. (2017). Indoor Modelling from SLAM-Based Laser Scanner: Door Detection to Envelope Reconstruction. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLII-2/W7. 345-352. 10.5194/isprs-archives-XLII-2-W7-345-2017.
[11] Bassier, Maarten & Mattheuwsen, Lukas & Vergauwen, Maarten. (2019). BIM Reconstruction: Automated Procedural Modeling from Point Cloud Data. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLII-2/W17. 53-60. 10.5194/isprs-archives-XLII-2-W17-53-2019.
[12] Bassier, Maarten & Vergauwen, Maarten. (2020). Topology Reconstruction of BIM Wall Objects from Point Cloud Data. Remote Sensing. 12. 1800. 10.3390/rs12111800.
[13] Petras Vestartas and Andrea Settimi, Cockroach: A Plug-in for Point Cloud Post-Processing and Meshing in Rhino Environment, EPFL ENAC ICC IBOIS, 2020, https://github.com/9and3/Cockroach.
[14] Qian-Yi Zhou and Jaesik Park and Vladlen Koltun, Modern Library for {3D} Data Processing, arXiv:1801.09847, 2018 , https://github.com/isl-org/Open3D/blob/main/README.md
[15] CGAL, Computational Geometry Algorithms Library, https://www.cgal.org
[16] Zampogiannis, Konstantinos and Fermuller, Cornelia and Aloimonos, Yiannis, cilantro: A Lean, Versatile, and Efficient Library for Point Cloud Data Processing, Proceedings of the 26th ACM International Conference on Multimedia, 2018
[17] Valero, Enrique & Adan, Antonio & Cerrada, Carlos. (2012). Automatic Method for Building Indoor Boundary Models from Dense Point Clouds Collected by Laser Scanners. Sensors (Basel, Switzerland). 12. 16099-115. 10.3390/s121216099.
[18] https://github.com/MetaCityGenerator/MetaCityGenerator_GHComponent
[19] Delaunay, B. (1934). Sur la sphère vide. *Bulletin of the Academy of Sciences of the USSR: Classe des sciences mathématiques et naturelles*, *6*, 793–800.
[20] https://github.com/nicoazel/BearGIS
[21] https://github.com/pyrevitlabs/pyRevit
[22] https://www.revitapidocs.com/2024/3ef7e31c-b41b-c8cc-2713-8f098954613d.htm

Radul Shishkov has PhD degree in computational and parametric design (2013-2018) in the department of “Interior and Architectural Design”, faculty of “Architecture”, “University of Architecture, Civil Engineering and Geodesy”, Sofia, Bulgaria. Master’s degree (2007-2012) in architecture from “University of Architecture, Civil Engineering and Geodesy”, Sofia, Bulgaria.

He is currently Assistant Professor in the “University of Architecture, Civil Engineering and Geodesy”, Sofia, Bulgaria and Managing Partner in “CAST Studio”, Sofia, Bulgaria.

Asst.prof. dr. arch. Radul Shishkov is member of the “Chamber of Architets in Bulgaria”