COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

D5.2 – Innovative Scan-vs-BIM-based Geometric QC component v2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310

D5.2 – Innovative Scan-vs-BIM- based Geometric QC component v2

Dissemination Level:	Public
Deliverable Type:	Demonstrator
Lead Partner:	UEDIN
Contributing Partners:	UEDIN, UCL
Due date:	31-10-2022
Actual submission date:	27-10-2022

Authors

Name	Beneficiary	Email
Martín Bueno	UEDIN	martin.bueno@ed.ac.uk
Frédéric Bosché	UEDIN	f.bosche@ed.ac.uk

Reviewers

Name	Beneficiary	Email
Apostolos Papafragkakis	Hypertech	a.papafragkakis@hypertech.gr
Panagiotis Moraitis	QUE	p.moraitis@que-tech.com

Version History

Version	Editors	Date	Comment
0.1	Martín Bueno	08.08.2022	ToC
0.2	Martín Bueno	16.09.2022	Initial draft
0.3	Frédéric Bosché	22.09.2022	Second draft for internal review
0.4	Panagiotis Moraitis	27.09.2022	Internal review
0.5	Martín Bueno	06.10.2022	Review comments addressed
0.6	Apostolos Papafragkakis	11.10.2022	Internal review
0.7	Martín Bueno	12.10.2022	Review comments addressed
0.8	Martín Bueno, Frédéric Bosché	25.10.2022	Final version
1.0	UEDIN, Hypertech	27.10.2022	Submission to the EC portal

Disclaimer

Executive Summary

The COGITO Deliverable D5.2 “Innovative Scan-vs-BIM-based Geometric QC component v2” documents the COGITO Scan-vs-BIM solution and presents the second iteration of the development activities in the context of task T5.1 “Scan-vs-BIM Geometric Quality Control”. Overall, the Scan-vs-BIM solution aims to obtain the matching between the as-built data (laser scanned point clouds) and the as-planned data (objects in the project BIM model). The output, i.e., the matched data, is stored to be used as input for the subsequent processes of the GeometricQC tool developed in task T5.3 “BIM-based Standard Test Methods for Geometric Quality Control”. As such, the Scan-vs-BIM solution lies at the start of the overall COGITO geometric quality control process.

The Scan-vs-BIM solution is a set of modular open-source tools and components designed as part of the GeometricQC tool; nevertheless, it can be included in any relevant application or project. The Scan-vs-BIM solution is composed of two sub-components, the main one being the *Point Cloud Matching and Segmentation* and the second one the *BIM Element Manager*. The *Point Cloud Matching and Segmentation* sub-component aims to perform the point cloud matching process of the laser scanned geometric data acquired at the construction site (which represents the actual geometry of the constructed building/asset components) with the as-planned geometry data. The *BIM Element Manager* sub-component aims to be the interpreter and manager of the BIM model elements (e.g., walls, slabs, columns, beams, etc.) stored in the IFC files and the as-built poses to facilitate the subsequent data processing and GeometricQC tool execution.

This deliverable aims to document the functionalities the COGITO Scan-vs-BIM solution along with its sub-components broadly deliver, the technology stacks they build upon, the inputs, outputs and APIs they expose, the installation instructions, the assumptions and restrictions, the applications examples, the development and integration status, and the requirements coverage. In this second and final release, the COGITO Scan-vs-BIM solution implements all but one of the envisaged functionalities (i.e., the IFC file interpreter, the point cloud to mesh matching, the as-built pose calculation) and its usage is illustrated and evaluated with examples obtained during the development of the sub-components using the same artificial examples adopted across all other developed tools/components throughout COGITO. Although a lot of effort has been put into the development of the tool and its sub-components, there is one main feature that still remains under development and as such cannot be reported or demonstrated yet: the machine learning enhanced segmentation and matching functionality. This feature is not critical to the functioning of the tool; but, its need will be revisited later in the project following the pre-validation activities.

Table of contents

List of Figures

List of Tables

List of Acronyms

Term	Description
API	Application Programming Interface
BIM	Building Information Modelling
B-Rep	Boundary Representation
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
COGITO	Construction Phase diGItal Twin mOdel
IFC	Industry Foundation Classes
OBJ	Object file
QC	Quality control
STL	Standard Triangle/Tessellation Language
TLS	Terrestrial Laser Scanning
UID	Unique Identifier

1 Introduction

1.1 Scope and Objectives of the Deliverable

This deliverable reports on the work conducted between M16 and M24 on the Scan-vs-BIM solution being developed as part of T5.1 "Scan-vs-BIM". It contributes an important step in the geometric Quality Control (QC) process proposed by COGITO. The scope of the Scan-vs-BIM solution is to enable the GeometricQC tool to perform automatic QC compliance and report its results back to the relevant stakeholders, namely the quality managers and the project managers. Having the QC reports at hand, these stakeholders can decide whether the already built components satisfy the standard specifications or require any remedial works.

More specifically, this deliverable reports on the development of the second release of the Scan-vs-BIM solution, focusing on its two main sub-components listed below:

- **Point Cloud Matching and Segmentation sub-component**: This sub-component takes as input the acquired and registered laser scanned data (point cloud) as well as the geometries from the as-planned data (triangle mesh), obtained from the COGITO Digital Twin Platform, and outputs which sub-set (if any) of the point cloud corresponds to each of the as-planned elements. The tool employs automated processes only.
- **BIM Element Manager sub-component**: the BIM model data pre-processing, interpreter and manager tool. This sub-component takes as input the as-planned data (using the IFC schema file format) and pre-processes the geometric information of the relevant elements (such as walls, columns, slabs, beams, etc) to be later stored in an internal structure, facilitating the subsequent data queries and management for the overall QC process. The tool employs automated processes only.

These two sub-components provide the core functionalities of the Scan-vs-BIM solution which in turn is integrated with the GeometricQC tool.

1.2 Relation to other Tasks and Deliverables

Table 1 depicts the relations of this document to other deliverables within the COGITO project; the latter should be considered along with this document for further understanding of its contents. This deliverable reports the second version (v2) of the Scan-vs-BIM solution. The Scan-vs-BIM solution is principally used to support the Use Case 2.1 (UC2.1).

Table 1: Relation to other Tasks and Deliverables

Del. Number	Deliverable Title	Relations and Contribution
D2.1	Stakeholder requirements for the COGITO system	Analysis of the end-user requirements in order to create the necessary inputs for defining the COGITO GeometricQC tool and its interactions with other components, along with a thorough description of the business scenarios, use cases and system requirements tailored to the project's goals and therefore setting the skeleton for the geometricQC tool development.
D2.5	COGITO system architecture v2	The second version of the COGITO architecture report that includes updates on the specifications of the GeometricQC tool (hardware/software requirements, programming language(s), development status, functional/non-functional requirements, inputs/outputs, along with the format, method, endpoint and protocol for each data type and interface).
D5.1	Innovative Scan-vs-BIM-based Geometric QC component v1	First version of the Scan-vs-BIM solution. For a detailed list of updates from this deliverable, please see section 1.4.
D5.5	BIM-based Standard Test Methods for	First version of the GeometricQC tool. This deliverable reports the tool used to perform automatic geometric QC and the description of its sub-components and modes, including the technology stack,

	Geometric Quality Control v1	input, outputs, integration status and usage examples. The Scan-vs-BIM solution presented in this deliverable is strongly connected with the GeometricQC tool as all of its sub-components were developed and are used by the later.
D5.6	BIM-based Standard Test Methods for Geometric Quality Control v2	Second version of the GeometricQC tool. This deliverable reports the tool used to perform automatic geometric QC and the description of its sub-components and modes, including the technology stack, input, outputs, integration status and usage examples. The Scan-vs-BIM solution presented in this deliverable is strongly connected with the GeometricQC tool as all of its sub-components were developed and are used by the later.

1.2.1 Relation to other COGITO tools and components

The Scan-vs-BIM solution serves as a data provider to the GeometricQC tool. In alignment with COGITO's system architecture, it does not interact with other components outside the GeometricQC tool.

1.3 Structure of the Deliverable

The deliverable is organised as follows: Section 2 presents an overview of the Scan-vs-BIM solution, placing the sub-components in the overall context of the solution and the GeometricQC tool processing pipeline. Sections 3 and 4 describe the *BIM Element Manager* and *Point Cloud Matching and Segmentation* sub-components, including the technology stack, implementation tools, the input/output data, and API documentation, application examples, licensing, installation instructions, development and integration status, requirements coverage, and assumption and restrictions of each sub-component. Section 5 concludes the deliverable.

1.4 Updates to the first version of the Deliverable

The second version of the Scan-vs-BIM solution presents a few new features in comparison with its first version as presented in the first version of this deliverable, namely D5.1.

First of all, two improved versions of the matching and segmentation algorithms have been introduced to the Scan-vs-BIM solution. The first one involves a more robust voxel space point matching, while the second one is a new version of the plane-point matching using voxel space plane segmentation and matching as-built planar patches against the as-planned object planes.

The second new feature is the capability to operate with the E57 file format for the input point clouds. Additionally, the component is now able to execute the point matching in multiple input point clouds, either stored in different files or in a single E57 one.

Next, the Scan-vs-BIM solution computes the as-built position of each matched element and stores the as-built pose (4.1.2) in the BIM Element Manager so that it is readily available to the subsequent geometric quality control processes.

Finally, all the examples and illustrative figures have been streamlined with the use of a COGITO project benchmark dataset, which will be used by all the different tools and components for testing and demonstration purposes during the development phase.

2 Overview of the Scan-vs-BIM solution

The aim of the proposed Scan-vs-BIM solution is to automatically match the as-built data acquired during the project's construction phase against the as-planned data. The Scan-vs-BIM solution is designed to use open formats (i.e. OBJ, STL, PLY, E57, etc), thus increasing its interoperability, accessibility, and data sharing among third parties. Concerning the as-planned data, in the context of COGITO's GeometricQC tool, the input geometric information is extracted from the as-designed BIM model of the construction project (IFC format). The geometry open formats supported by the Scan-vs-BIM solution are OBJ and STL. The as-built data are point cloud data obtained using Terrestrial Laser Scanning (TLS) devices, already registered (by the surveyor) in the project's coordinate system. The point cloud open formats supported by the Scan-vs-BIM solution are E57 and PLY.

The Scan-vs-BIM solution is part of the GeometricQC tool, defined in "*D2.1: Stakeholder requirements for the COGITO system"* and "*D2.5: COGITO system architecture v2"*, and as such, it was designed to be a back-end set of classes, part of the GeometricQC tool pipeline (see Figure 1), which is explained in detail in *"D5.6: BIM-based Standard Test Methods for Geometric Quality Control v2"*. The Scan-vs-BIM solution consists of two (2) main sub-components:

- **BIM Element Manager sub-component**: this sub-component takes the as-designed data in IFC format and generates a structure containing all the elements that require geometric quality control and all the information that will be needed for the GeometricQC tool execution. The different information stored in this structure includes: the original as-designed geometry, location, unique identifiers (UID), the associated as-built segmented point cloud after matching is completed, and the as-built pose of the element.
- **Point Cloud Matching and Segmentation sub-component**: this sub-component takes the scanned as-built data in its entirety and matches all the points to their corresponding BIM element as extracted by the BIM Element Manager, and compute the as-built pose to be used in subsequent steps.

As explained above, the Scan-vs-BIM solution is part of an overall GeometricQC tool, more specifically the *GeometricQC tool active geometric quality control analysis* component, hence some of its sub-components were designed to be utilised by other sub-components of the GeometricQC tool. A high-level diagram of the GeometricQC tool is illustrated in Figure 1, highlighting (green dashed lines) the positioning of the Scan-vs-BIM sub-components in the overall tool's pipeline.

The above two sub-components are detailed in Sections 3, and 4, respectively.

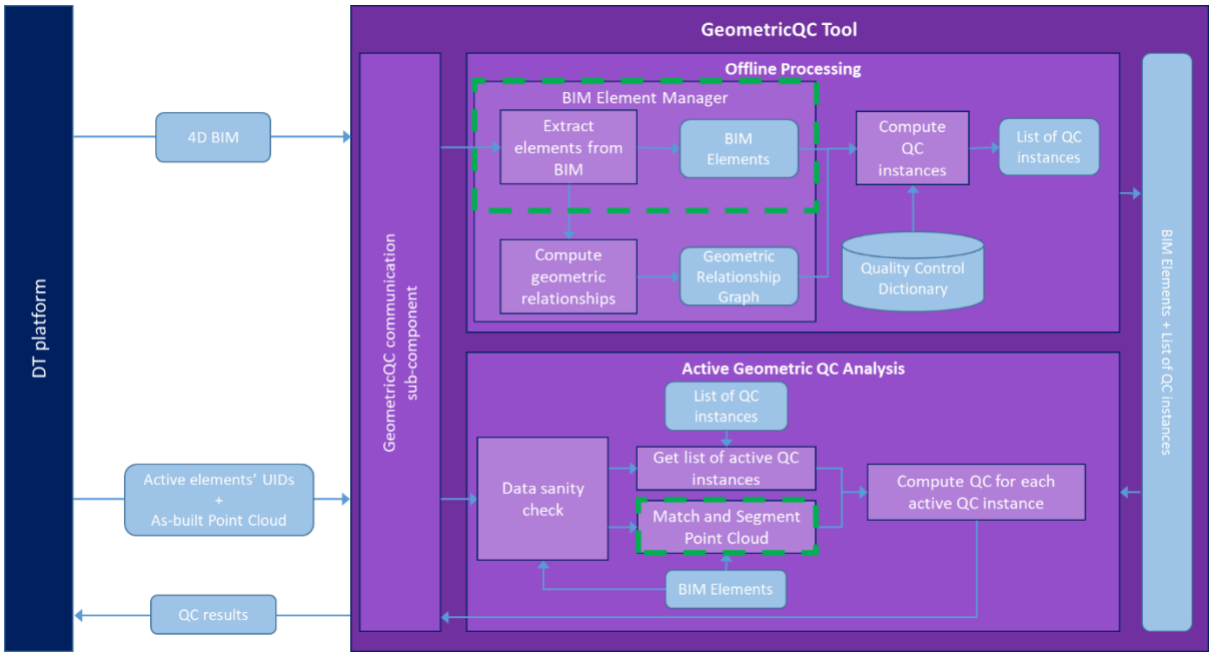


Figure 1: GeometricQC tool pipeline overview

3 BIM Element Manager sub-component

In this section, we describe the BIM Element Manager sub-component, a key component for the operation of the GeometricQC tool. The BIM Element Manager is responsible for pre-processing the IFC data and storing the resulting data in the desired format, eliminating the need for multiple queries and re-processing of the BIM information (IFC). Hence, the BIM Element Manager mainly serves as the BIM data management layer of the GeometricQC tool providing methods to obtain and store BIM data in order to be used by the rest of the GeometricQC tool. The following sub-sections describe the BIM Element Manager sub-component in more detail, including its technical aspects and requirements.

3.1 Prototype Overview

The objective of the BIM Element Manager sub-component is to allow the GeometricQC tool to load, interpret, and manage the construction project's as-planned data. The BIM Element Manager processes the as-planned BIM data in an IFC format, obtaining the relevant geometric information, as well as its identification labels, and store it, so the GeometricQC tool can be later executed without requiring re-processing of the same information on every call.

The BIM Element Manager sub-component also bears the functionality to store relevant information regarding the as-built data previously matched against the as-planned elements. The stored data involves the file path to the most recently matched as-built point cloud as well as the as-built pose, both to be later on utilised in by the GeometricQC tool.

The BIM Element Manager sub-component is quite simple to use and include/integrate with any other relevant application project. It merely requires the file path of the IFC file as input parameter, or the path to a BIM Manager file storing information previously extracted from an IFC file.

As can be seen in Figure 2, the BIM Element Manager sub-component delivers five object classes that are interconnected:

- **BIMElement class**: this class is used to describe a BIMElement object populated with all the relevant information extracted from the IFC file. It records all the relevant BIM identifiers, such as the UID, the IFC entity ID, and the human interpretable label. It also contains the element type and the material type of the component, as well as the as-planned mesh, the segmented as-built point cloud file paths, and the as-built pose transformation matrix. Additionally, it contains geometric information extracted from the IFC file, such as the global space location, and the global space vertices coordinates from the B-rep (which may need to be reinterpreted from other geometric representations). The class is also prepared with the *get* and *set* methods to modify and query information relative to the BIM component. This class is designed to be easily extended in order to facilitate the inclusion of further information potentially required to conduct the whole geometric QC process.
- **BIMElementManager class**: the main class of the sub-component to access the entire geometric information stored in the BIM model. In a nutshell, this class is the heart of the GeometricQC tool, storing all the relevant geometric information. The object contains the IFC file path where the data comes from, and a mapped list of all BIM elements of interest, indexed by their UID to facilitate the access, search and queries by the GeometricQC tool and its sub-components. Additionally, it contains the necessary methods to add elements, either as a single input or as a set of multiple inputs, to save the information to a file and load it, and finally to obtain and process the queried data.
- **ElementType class**: this class contains a list of possible BIM elements types (such as walls, columns, slabs, etc) that are of interest in the scope of the GeometricQC tool. It also contains parser methods allowing to read and save that information into the internal project files and data models. This class is likely to be extended to include more element types as the development of the tools progresses with the pre-validation and validation activities (e.g. managing railway-related elements).

- **MaterialType class**: this class, similarly to the previous one, contains the material types of the BIM components. Since the work in WP5 focuses on the standard specifications of concrete [1] and steel [2], this class only contains these main materials. However, it could naturally also be extended if needed.
- **IFCManager class**: this class is an IfcOpenShell library wrapper, created to ease the information retrieval from the IFC file and its complex structure. The only required parameter to use this object is to set the IFC file path upon object instantiation. The class contains the necessary methods to retrieve the BIM elements along with their geometric information from the IFC file. The output of the *getElement* and *getElemensList* methods serve as input for the **BIMElementManager** class. This class can also be extended (as it is likely to) to include additional functionalities that would be useful for the GeometricQC tool.

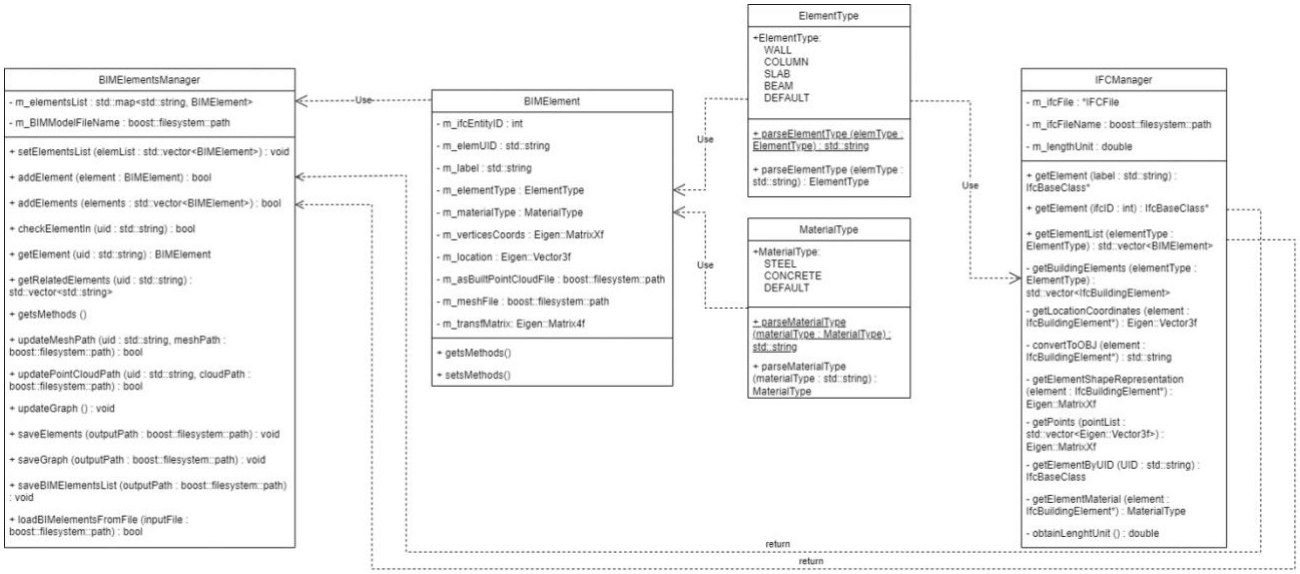


Figure 2: BIM Element Manager sub-component classes and its relationships

3.2 Technology Stack and Implementation Tools

The BIM Element Manager sub-component is a set of headers-only libraries that can be included in any project as any other library would. The classes are written in C++ v14 for compatibility purposes. In order to facilitate the development of the sub-component and in its extension the GeometricQC tool, the following open source C++ libraries were utilised:

- **Eigen**[1]: high-level C++ library of template headers for linear algebra, matrix and vector operations, geometrical transformations, numerical solvers and related algorithms. It is used to handle the matrices and vectors of point coordinates and as an algebra toolbox.
- **Boost**[2]: set of libraries for the C++ programming language that provides support for tasks and structures such as multithreading, regular expressions, unit testing, etc., and improves the capabilities of the C++ language. Boost contains over 150 individual libraries. Boost is used in a number of places within the GeometricQC tool, mainly to deal with JSON formats, system files and directories management and speed up processes.
- **IfcOpenShell**[3]: open-source library designed to enable software developers to work with the IFC file format. It is the main library to read and pre-process BIM IFC files and extract all the relevant information of each element.

[1] https://eigen.tuxfamily.org/index.php?title=Main_Page
[2] https://www.boost.org/
[3] http://ifcopenshell.org/

- **OpenCascade**[4]: open-source library designed to help software developers to work with 3D CAD, CAM, CAE, B-Rep, etc. It is used by IfcOpenShell to extract and work with the geometry of each BIM component from the IFC file.

Table 2: Libraries and Technologies used in the BIM Element Manager sub-component

Library/Technology Name	Version	License
Eigen	3.3.9	MPL2
Boost	1.76.0	Boost Software License 1.0
IfcOpenShell	0.6.0	LGPL-3.0 License
OpenCascade	7.5.0	LGPL-2.1 License

3.3 Input, Output and API Documentation

The BIM Element Manager sub-component can be demonstrated following 2 different options: (1) as a standalone object class that can be included in any project, or (2) as part of the GeometricQC tool.

The first option requires the following input parameters:

- **IFC file path**: the BIM file path (in IFC file format) of the project at object creation. The class will be able to load the IFC file and extract all the BIM components from it and store them in the manager.

For the second option, the GeometricQC tool only has to generate and populate the BIM Element Manager during the ***offline processing***. For this, it requires the following input data:

- **IFC file path**: BIM file in IFC file format of the project. The GeometricQC tool will create a BIM Element Manager object using the IFC file path as input parameter as in the standalone case.
- **Schedule file path**: detailed construction schedule containing the same elements UIDs in the IFC file and the timestamp when the construction of each element is planned to be completed and ready to be quality controlled. This schedule is only used in the context of the GeometricQC tool to support broader functionalities, such as the generation of a detailed schedule of when all QC instances are to occur. This file is required as input for the offline pre-construction processing only.

In the case that the BIM Element Manager sub-component is used as part of the GeometricQC tool, the entire solution can run from the command prompt or as part of a of the COGITO solution as a service triggered by a signalling method integrated between the GeometricQC tool communication sub-component and the Digital Twin Platform.

3.4 Application Example

The BIM Element Manager sub-component is essentially used to extract the geometric information from the BIM file. For this, it only requires as input the IFC file path. However, if the sub-component is executed as part of the GeometricQC tool offline processing, the second option of section 3.3 is employed, where the schedule file path should also be included as input parameters. In this section, we will show the outputs generated using two different example IFC files.

3.4.1 Revit Technical School model

The Revit Sample Project Technical School is a structural sample model provided by Autodesk [3]. It contains hundreds of different structural elements of concrete (588) and steel (24) material and is easy to export from its proprietary format into the IFC open format used by the COGITO solution. At this stage of development, this model has been considered as an ideal sample for testing and validating the initial version of the tools developed throughout COGITO and their integration within the final COGITO solution. Figure 3 shows the 3D model representation of the file.

[4] https://www.opencascade.com/

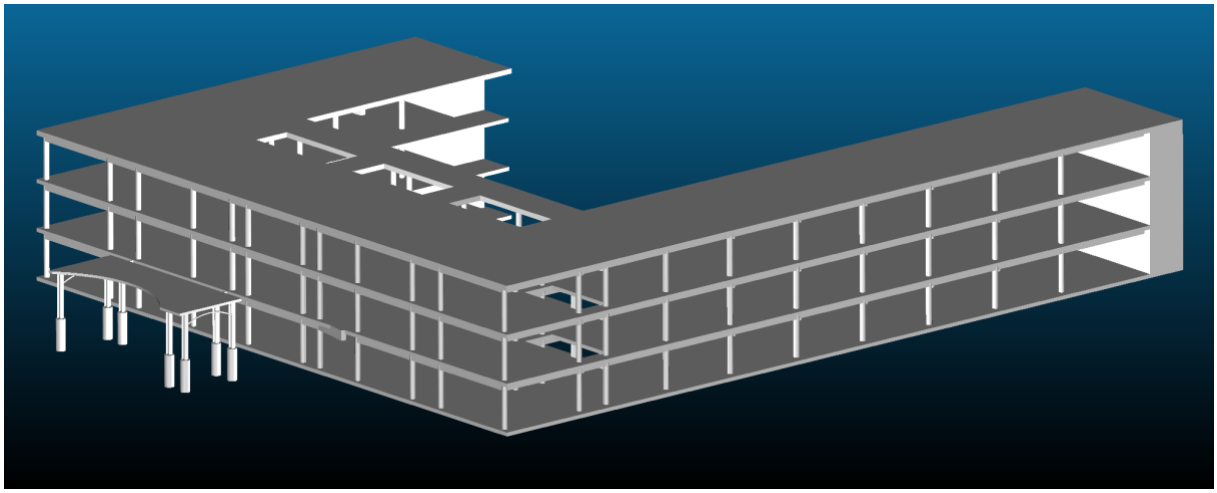


Figure 3: Revit School Sample Project 3D overview

Once the input file has been set, the BIM Element Manager uses the **IFCManager** class to query the elements from the IFC file using the appropriate *get()* method, and adds them into the mapped structure for storage using the appropriate *add()* method. Figure 4 depicts the logs of the BIM Element Manager after loading the IFC data. It can be seen that in this case a) there is a set of elements loaded by type and b) the BIM Element Manager JSON file has been saved for its reuse in future calls of the ***geometric QC active analysis*** in the same project. Figure 5 illustrates the content of this JSON file, namely the input file used, the list of elements and all their fields from the **BIMElement** class.

In subsequent calls to the GeometricQC tool under the same construction project, the tool executes the ***geometric QC active analysis***, hence the BIM Element Manager sub-component will load the BIM Element Manager JSON file instead of having to read and pre-process all the information from the IFC file, saving time, and resources.

```
************************************************
COGITO_GeomQC tool application v0.71.5
************************************************

GeomQC Offline processing mode was activated
################################################
Loading quality control rules dictionary
Saving Quality Control Dictionary info at C:\COGITO_Repos\COGITO_GeomQC\build\Output\RevitSchoolUCL\UC2.1_05_GQC_to_DTP.json
################################################
Processing IFC file
Loading information from IFC file C:\COGITO_Repos\COGITO_GeomQC\build\Output\RevitSchoolUCL\Input\rst_advanced_sample_project_ucl_v3.ifc
Loaded 18 walls from the IFC file
Loaded 16 slabs from the IFC file
Loaded 203 columns from the IFC file
Loaded 375 beams from the IFC file
Loaded 0 sleepers from the IFC file
Loaded 0 rails from the IFC file
################################################
Removing mesh vertices duplicates
################################################
Preparing Geometric Relationships Graph
Saved graph's nodes csv file C:\COGITO_Repos\COGITO_GeomQC\build\Output\RevitSchoolUCL\graphNodes.csv
Saved graph's edges csv file C:\COGITO_Repos\COGITO_GeomQC\build\Output\RevitSchoolUCL\graphEdges.csv
Saving BIM elements list info at C:\COGITO_Repos\COGITO_GeomQC\build\Output\RevitSchoolUCL\BIMElementsList.json
################################################
```

Figure 4: Revit School Sample Project file loaded and processed by the BIM Element Manager sub-component

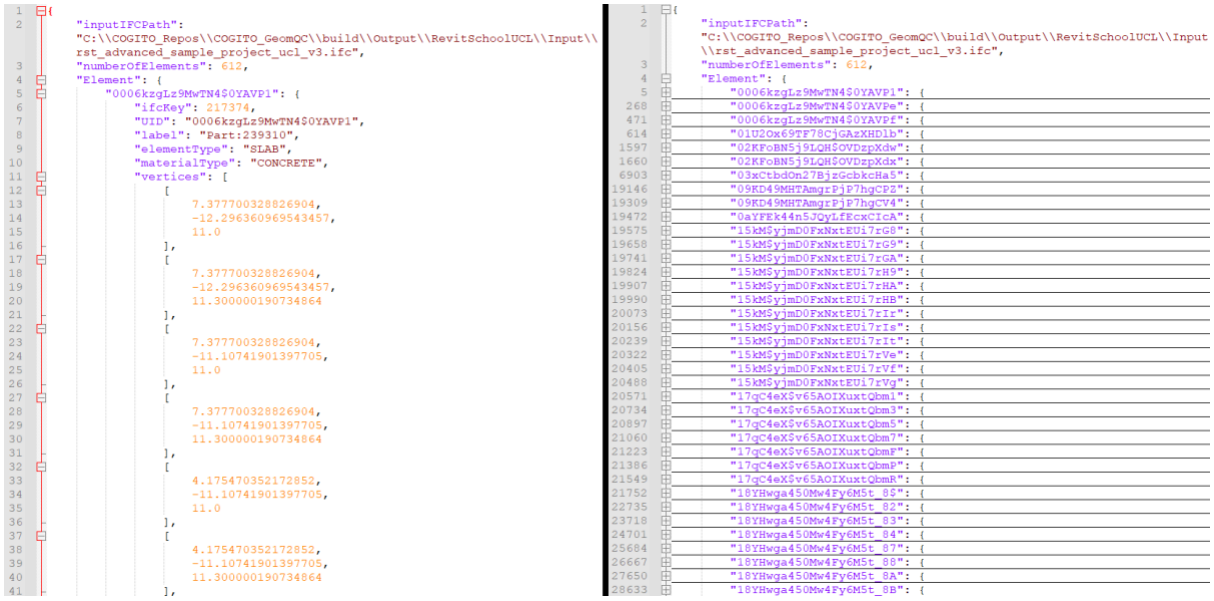


Figure 5: Saved BIM Element Manager JSON file example using the Revit School Sample Project file

3.5 Licensing

The Scan-vs-BIM solution is free software; you can redistribute it and/or modify it under the terms of the **GNU Affero General Public License** as published by the Free Software Foundation (https://www.gnu.org/licenses/agpl-3.0.en.html).

3.6 Installation Instructions

No complex installation is required. The code will be provided in the Cyberbuild Datashare collections and GitHub. BIM Element Manager sub-component is a set of header-only libraries that can be included in any project just by adding the header files to the includes list.

3.7 Development and integration status

The BIM Element Manager sub-component is considered to be finished. Extra functionalities may nonetheless be found to be missing during the pre-validation and validation phase, at which point they will be added. Regarding the integration status, the BIM Element Manager sub-component is only accessible through the GeometricQC tool, it does not interact with any external component and all the interactions are within the GeometricQC tool. A detailed integration of the GeometricQC tool is documented in *"D5.6: BIM-based Standard Test Methods for Geometric Quality Control v2"*.

3.8 Requirements Coverage

Despite the fact that the BIM Element Manager sub-component is just a back-end part of the GeometricQC tool, it already covers a number of the requirements defined in D2.5 and D2.1.

The functional and non-functional requirements that have introduced in the COGITO System Architecture and that have already been covered by this version of the BIM Element manager are presented in Table 3. Functional requirement Req-1.1 is covered using this sub-component for the 4D BIM data. Another part of the GeometricQC tool has the charge of loading the point cloud data. Thanks to the C++ feature allowing the use of different objects in header only classes, Req-2.1, Req-2.2, and Req-2.3 are well covered within this sub-component. As previously mentioned, this set of classes (as part of the full GeometricQC tool library) can be used autonomously in any other project, or as part of the full GeometricQC tool. In addition, its scalability is well handled using object-oriented programming, which enables straightforward addition of new functionalities and properties to each of the components. Additionally, the sub-component uses libraries that are prepared to handle large amounts of data, multithreading and processing of several objects. They are only be limited by the system where it is executed.

Table 3: BIM Element Manager sub-component requirements coverage from D2.4

ID	Description	Type	Status
Req-1.1	Loading as planned 4D BIM and point cloud data	Functional	Achieved
Req-2.1	Scalability	Non-Functional	Achieved
Req-2.2	Reusability	Non-Functional	Achieved
Req-2.3	Interoperability	Non-Functional	Achieved

Table 4 presents the stakeholders requirements that have been documented in D2.1 and are relevant to this component. COGI-CS-1 and COGI-CS-4 are covered simply by the programming language that has been selected for the development of the GeometricQC tool. With regards to COGI-QC-11, the BIM Element Manager allows to load and interpret the as-planned data from the BIM model, thus enabling the interconnection with other sub-components to automate the foreseen quality control activities.

Table 4: BIM Element Manager sub-component stakeholders requirements coverage from D2.1

ID	Description	Type	Priority	Status
COGI-CS-1	Runs on desktop or laptop PC	• Operational	Must	Achieved
COGI-CS-4	Runs on Windows	• Operational	Must	Achieved
COGI-CS-5	Runs on Mac	• Operational	Could	Achieved
COGI-QC-11	Automates QC-related activities	• Functional • Operational	Must	Achieved

3.9 Assumptions and Restrictions

The second version of the BIM Element Manager sub-component has been delivered under certain assumptions and restrictions, listed below:

- It has been developed to be part of the GeometricQC tool, so no standalone programme is supposed to be executed for this sub-component.
- It currently supports IFC files of version 4.0. Support for IFC version 4.3 remains under development driven by the emerging needs of the pre-validation and validation activities and the capability of the Digital Twin Platform.
- It requires the input IFC file to be syntactically correct (i.e. consistent with the IFC 4.0 schema) and free of geometric errors.
- BIM elements are expected to have the vertical along the +Z axis using the right-hand side coordinate frame.
- The elements' reference coordinate frame within the entire GeometricQC tool uses project global coordinates; hence the generated output files contain coordinates information in the same global space.
- The measures are stored using the international metric system, and the distance measurement unit used in the sub-component is the metre [m].

4 Point Cloud Matching and Segmentation sub-component

In this section, we describe the *Point Cloud Matching and Segmentation* sub-component. This sub-component provides the core functionality of the Scan-vs-BIM solution; it is responsible for performing the matching of the as-built point cloud data to the as-planned BIM elements and storing the result in the *BIM Element Manager* sub-component for further processing within the *GeometricQC* tool. The following sub-sections describe in more detail the *Point Cloud Matching and Segmentation* sub-component along with its technical aspects and requisites.

4.1 Prototype Overview

The objective of the *Point Cloud Matching and Segmentation* sub-component is to geometrically match the as-built point cloud data to the as-planned BIM elements; being able to accurately segment the as-built data of each BIM element is critical in order to obtain reliable and accurate geometric control results. The as-built data comes in the form of a point cloud (as the result of a survey work order) already registered in the project's coordinate system (i.e., the same coordinate system as the as-planned BIM model), while the as-planned data is obtained by reinterpreting the geometry of each BIM element into a triangular mesh. The Scan-vs-BIM component (like the rest of the GeometricQC tool) utilises well-known open formats, namely PLY and E57 for the point clouds and OBJ or STL for the meshes.

As with the majority of the sub-components developed in the context of the GeometricQC tool, the *Point Cloud Matching and Segmentation* sub-component is a modular component, quite simple to use and include in any other project. As a standalone object class, it only requires the input point cloud file path(s) and a list of file paths pointing to the meshes of the BIM elements of interest.

As illustrated in Figure 6, the *Point Cloud Matching and Segmentation* sub-component comprises of six object classes that are interconnected:

- **PointCloudMatching**: the main class of the Scan-vs-BIM solution. This class allows to pass all input parameters necessary to perform the matching and segmentation of the point cloud(s) to the as-planned meshes. It only contains three different fields, the file path to the input point cloud(s), the list of file paths to the BIM element meshes, and the output path where the matched and segmented point clouds will be exported. The matching process has an input allowing the selection of the point matching algorithms to be used. Its generated output is then passed to the main GeomericQC tool for further processing.
- **Open3DMatching**: this class is in charge of the heavy computational work by calling other classes and methods to obtain and process the relevant data. As its name indicates, the point cloud matching is based on point cloud and triangle mesh structures of Open3D (see Section 4.2). All the relevant fields of this class are filled automatically when one of the matching methods is called.
- **VoxelGrid**: this class is in charge of processing the point cloud into a voxel grid. Anything required from the voxel grid can be processed and queried using this class. The input point cloud and the voxel size are the only parameter required as input to this class. As most of the other classes, it can be used independently of the project and be included in any other C++ project.
- **PointCloudUtils**: this class is designed as a modular point cloud utilities toolbox. Inside, one can find all point cloud processing utilities used by the GeometricQC tool. By doing so, it is guaranteed that this toolbox can be used by any other project involving point clouds.
- **PointCloudSegmentation**: this class contains all point cloud segmentation methods, complementing the PointCloudUtils toolbox class.
- **MathUtils**: this class is another handy toolbox containing arithmetic operations and checks using mostly (but not only) vectors and matrices.

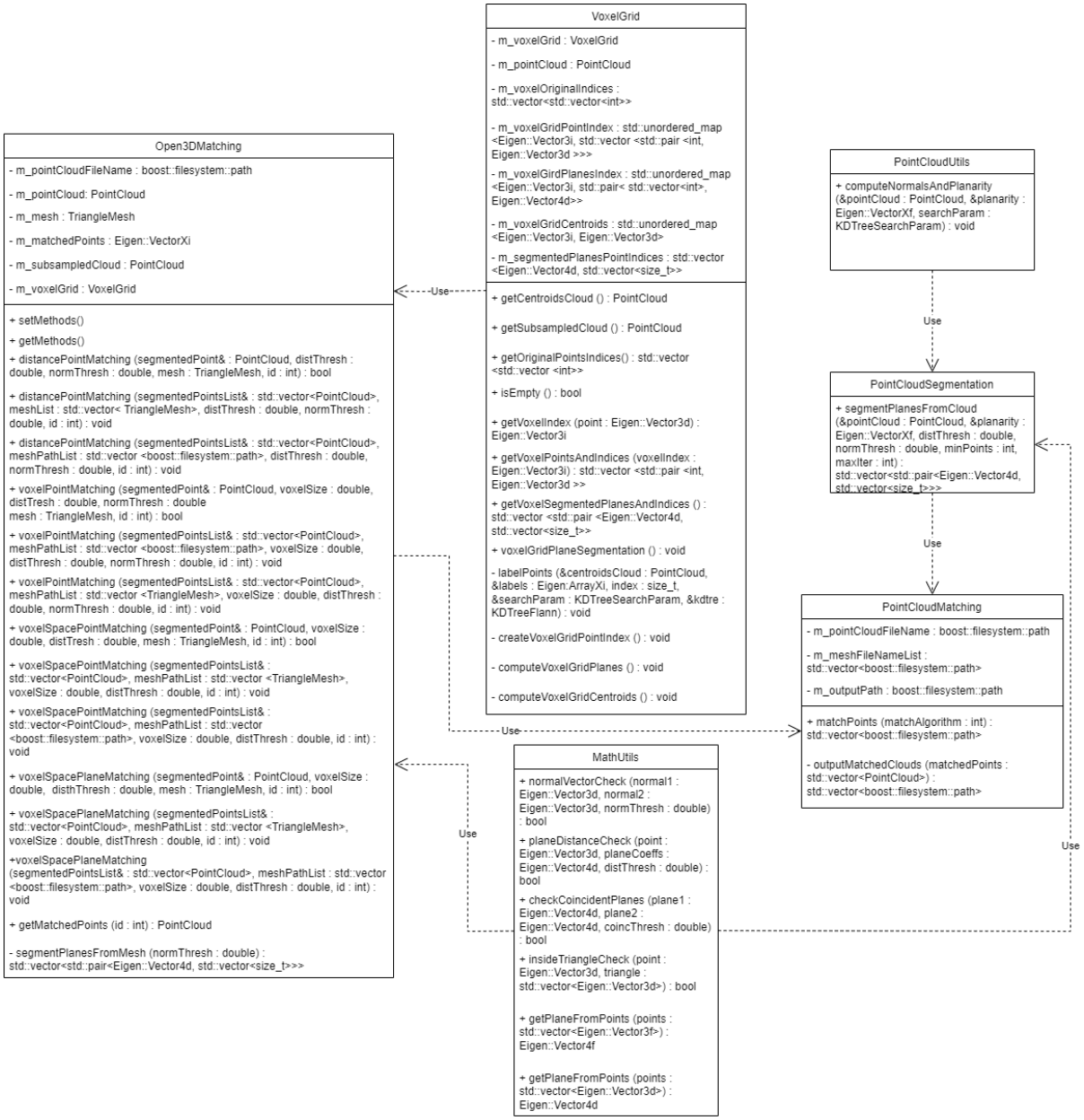


Figure 6: Point Cloud Matching and Segmentation sub-component classes and its relationships

4.1.1 Point cloud matching methods

The Point Cloud Matching and Segmentation sub-component contains four alternative matching methods implemented and tested in this second release. As detailed in Section 4.4, the majority of these methods provide segmentation results of sufficient quality (point distribution, density, completeness, etc.) for the rest of geometric quality control process. Overall, the best method appears to be the "voxel space point matching" method. These methods are briefly described below.

- **Distance and normal vector similarity point matching**: this is the most basic, "brute force" method. It basically iterates through every unmatched point in the point cloud and checks if (1) its normal vector (pre-computed if not provided) is similar to a given mesh's triangle normal vector, (2) its orthogonal distance to the plane of the triangle is within a given threshold, and (3) its orthogonal projection on the plane of the triangle falls within the triangle. If all these checks are satisfactory, the point is assigned a label as matched to that mesh (that triangle), and the algorithm continues with the rest of the cloud points. This method is referred to as the "brute force" method,

as it iterates through all the points in the point cloud and triangles in the mesh. Consequently, it is not the most resource efficient.

- **Distance and normal vector similarity using a voxel subsample point matching**: this method is very similar to the previous one, but instead of iterating through the entire point cloud, the point cloud is first subsampled using a voxel grid, reducing the number of points to be initially checked. The iteration is performed using the centroid of each voxel first, and if the three checks defined in the "brute-force" method are satisfactory, then the algorithm recovers all the points from the original point cloud that lie inside that matched voxel and checks them one by one. This method is much faster than the previous one, as it reduces the number of iterations and checks, with only minor reduction in the quality of the matching expected.
- **Voxel space point matching**: this method relies on checking if the mesh triangles lie inside an occupied voxel from the voxel space and perform the same checks as described above for the points inside that occupied voxel. As we will see later, this method produces high quality results and is fast, sometimes faster than the previous method.
- **Voxel space plane point matching**: this method's original idea is to segment the point cloud into planar surfaces and then match those planar surfaces at once. By doing so, the amount of data to be checked is significantly reduced. The algorithm segments planes from the voxel space by first using the RANSAC algorithm [4] to obtain a planar patch inside the voxel, and then grouping all the planar patches by region-growing based on proximity and normal vector similarity. Once all the planes are segmented, and the original point indices are mapped to them, the method tries to find for each segmented plane a matching plane in the mesh. This is done by checking whether the mesh's plane equation is coincident, within a threshold, with the segmented plane equation. If a coincident plane is found, the algorithm still must check if the points that belong to that plane also are inside the boundaries of the triangles defining the mesh plane, to reduce false positives matches. It must be noted that this method requires a pre-processing step, computing the normal vectors and the segmented planes, but it was hoped that the overall results would be more robust to deviations from the as-planned positions and poses. Its effective application requires the input point cloud contains a low amount of noise.

4.1.2 As-built pose computation

After obtaining all the matched and segmented point clouds for the BIM elements of interest, the Point Cloud Matching and Segmentation sub-component performs an additional step to compute what is called the as-built pose of each of the elements. The as-built pose corresponds to the actual position and orientation of each of the BIM elements obtained from the as-built data provided by the surveyor. The as-built pose is obtained by applying a 4x4 transformation matrix to the as-design (or as-planned) pose. This pose is necessary for the subsequent geometric quality control process steps that involve positions and inclinations evaluations.

The as-built pose computation is done using the state-of-the-art Iterative Closest Point (ICP) algorithm [5]. The source element for the ICP algorithm is the segmented point cloud, while the target (the aiming object) is the as-planned mesh model. Since both elements should be relative closer, the ICP parameters are reduced to a few iterations and a low correspondence distance.

Each of the as-built poses is then stored in the BIM Element Manager for later use during the Geometric QC Active Analysis phase.

4.2 Technology Stack and Implementation Tools

The Point Cloud Matching and Segmentation sub-component is a set of headers-only libraries that can be included in any project as any other library would. The classes are written in C++ v14 for compatibility purposes. In order to facilitate the development of the sub-component, the Point Cloud Matching and Segmentation sub-component, and in its extension the Scan-vs-BIM solution and the GeometricQC tool, the following open-source C++ libraries were utilised:

- **Eigen**[1]: high-level open-source C++ library of template headers for linear algebra, matrix and vector operations, geometrical transformations, numerical solvers and related algorithms. It is used to handle the matrices and vectors of point coordinates as well as an algebra toolbox.
- **Boost**[2]: set of libraries for the C++ programming language that provides support for tasks and structures such as multithreading, regular expressions, unit testing, etc., and improves the capabilities of the C++ language. Boost contains over 150 individual libraries. Boost is used in many places within the GeometricQC tool, mainly to deal with JSON formats, system files and directories management and speed up processes.
- **Open3D**[5]: open-source library designed to help the software developers and users work with three-dimensional data, such as point clouds and meshes. It is the main library used to read and process the point cloud and mesh data.

Table 5: Libraries and Technologies used in the Point Cloud Matching and Segmentation sub-component

Library/ Technology Name	Version	License
Eigen	3.3.9	MPL2
Boost	1.76.0	Boost Software License 1.0
Open3D	0.13	MPL2
LibE57	1.1.312	MIT License

4.3 Input, Output and API Documentation

Similar to the BIM Element manager, the Point Cloud Matching and Segmentation sub-component provides two run-time options: (1) running as a standalone object class that can be included in any project, or (2) running as an object part of the GeometricQC tool.

The first option requires the following input parameters:

- **pointCloudFileName**: file path to the input as-built point cloud(s) to be matched to the input object meshes.
- **meshFileNameList/meshList**: list of the file paths pointing to the mesh geometries against which the input point cloud is to be matched. The list uses the C++ standard library format. The mesh can be of OBJ or STL type. Within the GeometricQC tool, the mesh file path is stored and handled by the BIM Element Manager, so there is no need to keep elements in memory, but the option to use a list of mesh geometries allocated in memory is available.
- **outputPath**: (optional) output path where to store the segmented point clouds. Each point cloud would be named using the same UID as the input BIM element UID.

For the second option, the Point Cloud Matching and Segmentation sub-component is executed as part of the ***active geometric QC analysis*** mode in the GeometricQC tool, as so it would require as an input the following parameters:

- **As-built point cloud data file name(s)**: file path(s) to the as-built data input point cloud(s) to be matched to the input object meshes.
- **As-built point cloud file(s) timestamp(s):** the timestamp(s) for each of the input point cloud(s) representing the date when the as-built data were captured.
- **File with the list of elements to be checked**: text file containing the UIDs of the elements (as defined in the as-planned 3D model, e.g., the IFC file) that the input point cloud is to be matched with. The GeometricQC tool is in charge of reading this information, obtaining, and preparing the necessary data in the appropriate data format for the Point Cloud Matching and Segmentation sub-component class.

[5] http://www.open3d.org/

In case the Point Cloud Matching and Segmentation sub-component is used as part of the GeometricQC tool, the entire solution can run from the command prompt or by the DTP and GeometricQC tool communication sub-component signalling system.

4.4 Application Example

The Point Cloud Matching and Segmentation sub-component includes all of the main methods required to perform Scan-vs-BIM. It is used to load the point cloud data and run the matching algorithms to link the as-built and the as-planned BIM elements data, serving as input for the geometric quality control. To provide its functionality, the Point Cloud Matching and Segmentation sub-component requires only the point cloud file path(s) and the list of the triangle meshes of the as-planed BIM components (this input can come by connecting and using the BIM Element Manager sub-component). In this section, the different outputs obtained with the implemented matching algorithms for COGITO project's common benchmark dataset are presented.

4.4.1 Revit Technical School model and point cloud

As already explained in Section 3.4.1, a Revit Technical School model is a structural sample model provided by Autodesk [3] is used by the different COGITO partners to develop and test their different applications. As discussed below, it also allows to create different unsatisfactory conditions, where the GeometricQC tool can detect discrepancies between the as-planned and the as-built data.

During the development test, a synthetic point cloud has been created from the mesh model, containing around 10 million points (Figure 7) with a 2.5 mm standard deviation noise applied to them (to generating a more challenging and realistic scenario for the matching algorithms). The point cloud also contains a variety of defects in some of the structural elements (misplacements, centres deviations, inclinations, etc.).

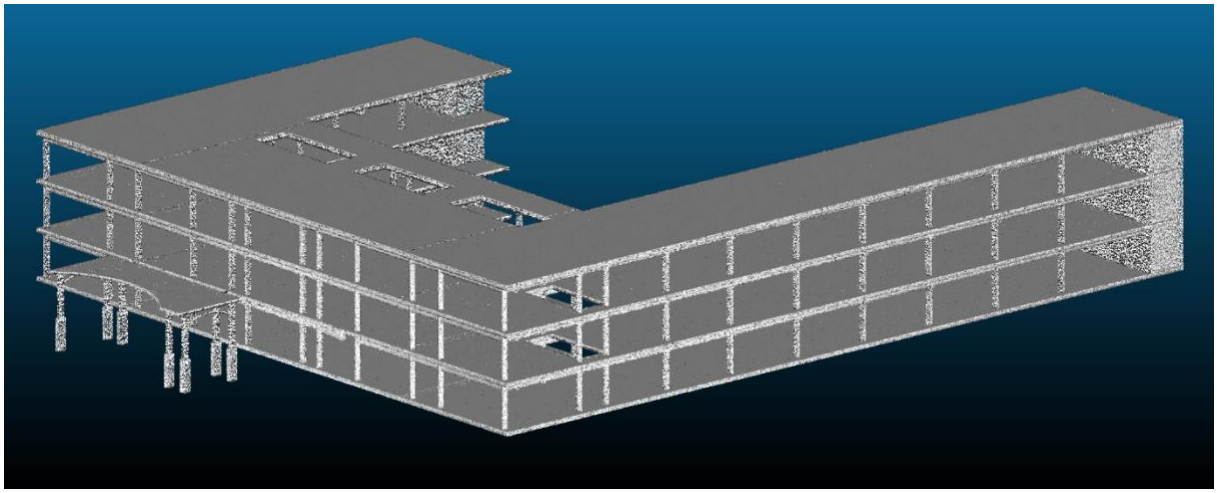


Figure 7: Revit Technical School Sample Project point cloud generated from the as-designed triangle mesh with additional 2.5 mm standard deviation noise and added geometric defects.

Two different tests have been conducted. The first test assesses and compares the processing time required by each algorithm to match and segment 28 BIM elements from the IFC file, including columns, slabs and beams; the results are shown in Table 6. The distance and normal vector similarity point matching algorithm (Brute-Force) is a slow matching algorithm, yet the simplest one both implementation- and logic-wise, gives good results, allowing it to be used as a benchmark method. The results indicate that the fastest approaches are the ones employing a voxel structure. A more detailed analysis can be performed in the plane matching algorithm. The algorithm is divided in 2 steps, the first one pre-processes the input point cloud, computing the points' normal vectors (although this step is also performed in all the methods and is not the main contributor to the computational time) and segmenting the planes, while the second one performs the actual matching between the as-planned and the as-built data. Arguably, the point cloud would be pre-processed only once, and the extra features obtained from it could be useful to other sub-

components. However, as can be seen in the second test below, the quality of the results is not as good as expected.

Table 6: Comparison of the processing time (in seconds) for each of the matching algorithms used with the Revit Technical School Sample Project

	Distance + normal vector similarity	Voxel subsample	Voxel Space	Voxel Space Plane Matching
Revit Technical School Sample Project	26007.3	17970.6	1014.5	Plane seg: 4496.2 Matching: 886.7

For the second test, the quality of the segmented objects was compared. A total of 28 BIM components were matched and segmented, and the total number of segmented points have been collected in Table 7. As can be seen in Figure 8 to Figure 11, the first 3 algorithms produce high numbers of matching points when there are small deviations between the as-built and as-planned data. However, when there are some significant deviations (see the column in the centre of the images which is a few centimetres moved to the side), only the voxel space method is capable of dealing with them. In this last method, the output data quality is good enough for the sub-sequent geometry quality control processing. Taking a closer look at the images, it can be seen that using all of the matching algorithms the segmented elements also contain points that belong to different elements, mainly those really close to the ones of interest. The main reason for these misclassified points is due to the relaxation of the different thresholds used in the algorithms logics. Thanks to the multiple possible deviations of the as-built elements, the algorithms must be ready to adapt and extract the relevant points to be matched with the as-planned data. However, this causes multiple misclassified points in each of the elements. Nevertheless, the as-built pose computation is not terribly affected by these misclassified points; the quality of the results appears accurate enough and thus appropriate to serve as input to the rest of the GeometricQC tool. The voxel space method is not only more robust regarding deviations, but also the fastest of all methods. The voxel space plane point matching method is the only one not generating a favourable outcome. The different kinds of geometries, such as cylinders or curved shapes and the various sizes of the elements makes it too difficult for the algorithm to obtain relevant planar surfaces that can be used to match the as-planned meshes. Hence it is not recommended to use this method at this stage.

Table 7: Number of segmented points from the Revit Technical School Sample Project dataset with 2.5 mm std noise

	Distance + normal vector similarity	Voxel subsample	Voxel Space	Voxel Space Plane Matching
1WrzGm1SD2ev45B_OWQ3E2	8110	7599	8907	7622
1WrzGm1SD2ev45B_OWQ3EP	6609	6214	7606	6365
02KFoBN5j9LQH$OVDzpXdx	1307	876	1654	149
2Ci2k7uxXCqAqqsxOmESO_	3323	3104	3675	22
2Ci2k7uxXCqAqqsxOmEVak	3345	3048	3758	0
2Ci2k7uxXCqAqqsxOmEVd5	3341	3078	3723	0
2UD3D7uxP8kecbbBCRtz8d	3469	3361	3964	2712
2UD3D7uxP8kecbbBCRtz8q	7580	7364	8127	5539
2UD3D7uxP8kecbbBCRtz9i	2312	2188	2466	1384
2UD3D7uxP8kecbbBCRtz9L	5739	5337	6213	4240
2UD3D7uxP8kecbbBCRtz9U	6515	6087	6853	4218
2UD3D7uxP8kecbbBCRtz9V	5999	5563	6915	4417
2UD3D7uxP8kecbbBCRtz83	2738	2632	3669	933
2UD3D7uxP8kecbbBCRtzB_	435	398	2637	423
2UD3D7uxP8kecbbBCRtzBi	2237	2174	2856	723
2UD3D7uxP8kecbbBCRtzBm	2202	2106	2657	627
2UD3D7uxP8kecbbBCRtzBn	2183	2067	2650	535
2UD3D7uxP8kecbbBCRtzBO	2261	2167	2667	669
2UD3D7uxP8kecbbBCRtzBP	441	404	2588	412

2UD3D7uxP8kecbbBCRtzBy	2203	2093	2695	557
2UD3D7uxP8kecbbBCRtzCg	2573	2403	3062	918
2UD3D7uxP8kecbbBCRtzCL	2583	2451	3060	1146
2UD3D7uxP8kecbbBCRtzEv	2645	2520	3096	865
3KQkoT3ZD758EdANtsCisP	1170	761	1786	23
3KQkoT3ZD758EdANtsCisU	1333	934	1387	69
18YHwga450Mw4Fy6M5t_8F	2645	2484	3095	819
18YHwga450Mw4Fy6M5t_8L	2694	2520	3186	904
18YHwga450Mw4Fy6M5t_8v	2732	2588	3200	943

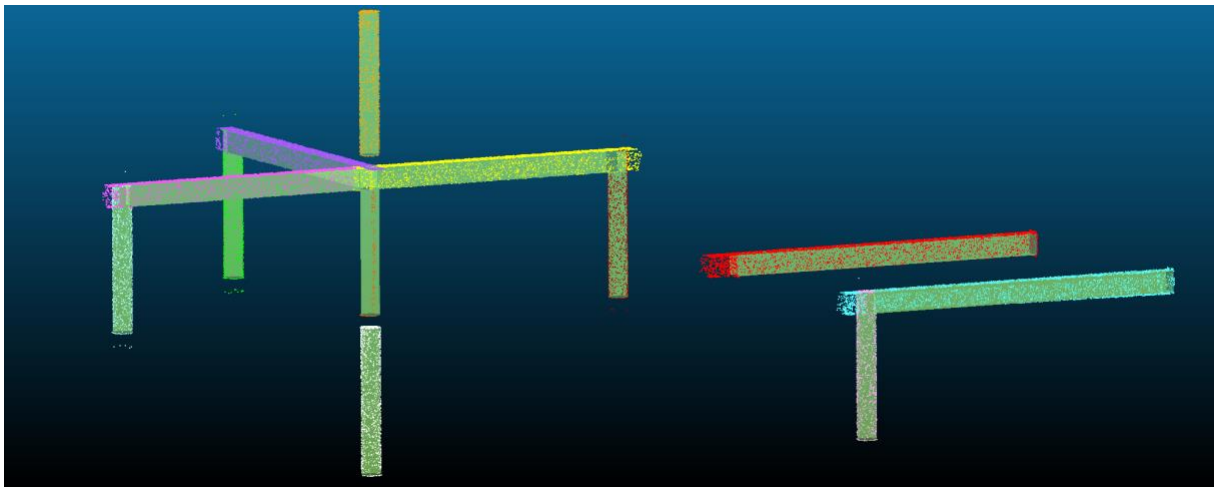


Figure 8: Distance and normal vector similarity matching output examples

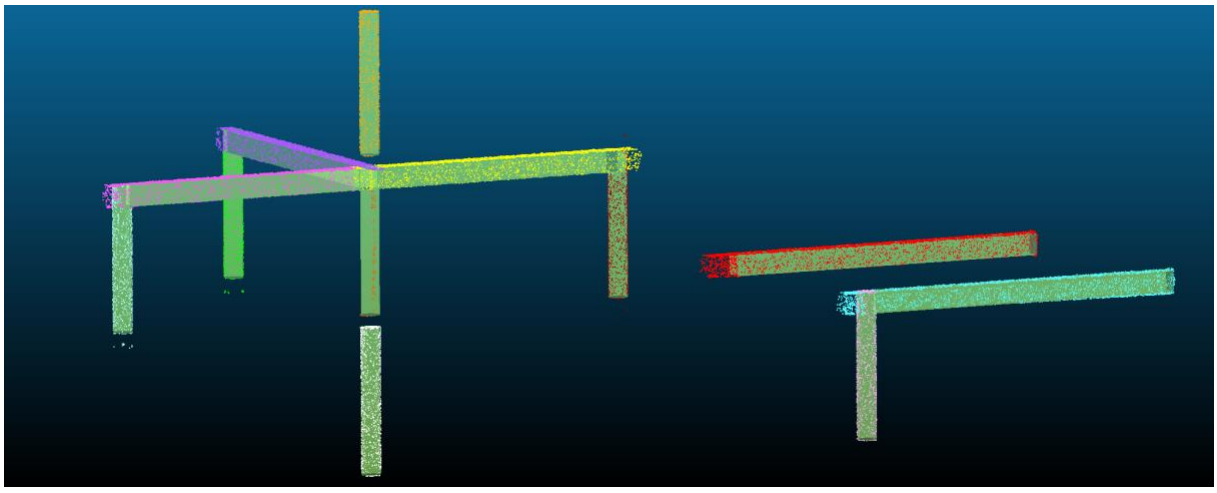


Figure 9: Voxel subsample matching output examples

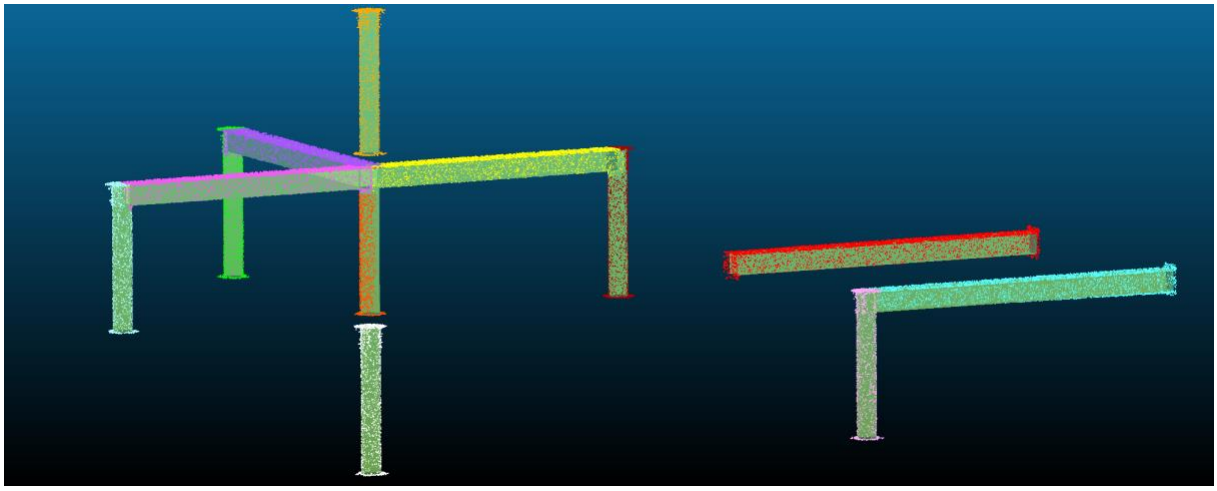


Figure 10: Voxel space matching output examples

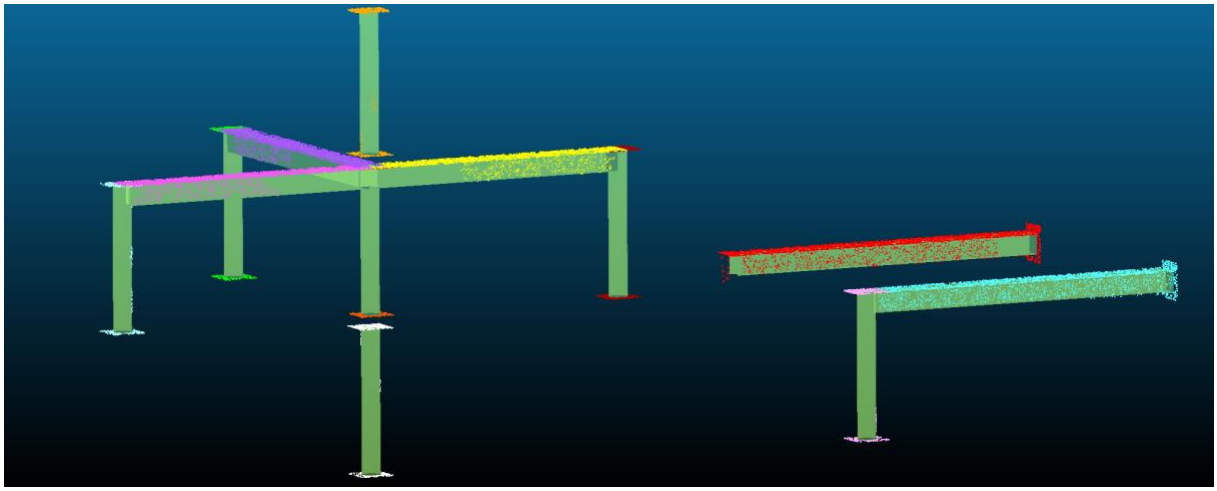


Figure 11: Voxel Space Plane matching output examples

Comparing the quality of the segmented points and the processing time, the best matching and segmentation method appears to be the voxel space one. Thus, this method is chosen for producing point-mesh matches used for developing the QC-focused sub-components of the GeometricQC tool in T5.3.

4.5 Licensing

The Scan-vs-BIM solution is free software; you can redistribute it and/or modify it under the terms of the **GNU Affero General Public License** as published by the Free Software Foundation (https://www.gnu.org/licenses/agpl-3.0.en.html).

4.6 Installation Instructions

No complex installation is required. Binaries will be provided in the Cyberbuild Datashare collections. The Point Cloud Matching and Segmentation sub-component is a set of header-only libraries that can be included in any project.

4.7 Development and integration status

The Point Cloud Matching and Segmentation sub-component can be considered complete and functional. But, one of the sought enhancements has not been completed yet and remains under exploration: the deep learning-enhanced segmentation and matching functionality. The criticality of its need will be further assessed following the pre-validation activities. Besides, any extra functionalities requirement raised

during the pre-validation and validation phases could be added as part of WP8. The implemented matching algorithms, especially the voxel space one, already produces accurate results.

As part of the integration status, the Point Cloud Matching and Segmentation sub-component (and similarly the Scan-vs-BIM component in its whole), is only accessible through the GeometricQC tool, it does not interact with any external component and all its interaction are contained within the GeometricQC tool. A detailed integration of the GeometricQC tool is documented in *"D5.6: BIM-based Standard Test Methods for Geometric Quality Control v2"*.

4.8 Requirements Coverage

Although the Point Cloud Matching and Segmentation sub-component is delivered as a back-end solution of the Scan-vs-BIM and the GeometricQC tool, it already covers several of the requirements defined in D2.5 and D2.1.

The functional and non-functional requirements are presented in Table 8. Req-1.1 is covered using this sub-component for the point cloud data thanks to the point cloud loader and interpreter. Another part of the GeometricQC tool has the charge of interpreting the 4D BIM data. Req-1.2 has not been achieved yet. This will be part of additional work in the remaining of the COGITO project. Thanks to the C++ feature allowing the use of different objects in header only classes, Req-2.1, Req-2.2, and Req-2.3 are well covered by the component. As stated above, this set of classes (as part of the full GeometricQC tool library) can be used autonomously in any other project, or as part of the full GeometricQC tool. In addition, the scalability requirement is met due to the object-oriented programming, streamlining the addition of new functionalities and properties to each of the components. Additionally, the sub-component uses libraries that are prepared to handle large amounts of data, employing multithreading and processing of several objects. There are only limited by the system where it is executed.

Table 8: Point Cloud Matching and Segmentation sub-component's coverage of the requirements reported in D2.5

ID	Description	Type	Status
Req-1.1	Loading as planned 4D BIM and point cloud data	Functional	Achieved
Req-1.2	Object detection in point cloud	Functional	Not yet supported
Req-2.1	Scalability	Non-Functional	Achieved
Req-2.2	Reusability	Non-Functional	Achieved
Req-2.3	Interoperability	Non-Functional	Achieved

Table 9 presents the stakeholder requirements which are relevant to this component as documented in D2.1. COGI-CS-1 and COGI-CS-4 are covered by the programming language selected for development itself. COGI-QC-8 is partially achieved by matching the as-built data to the as-planned data of concrete and steel works, allowing to process these matched data in the subsequent GeometricQC tool procedures. While substructure and earthworks were identified as types of elements that should ideally be covered by the developed solution, these are currently not covered. It must however be highlighted that the Scan-vs-BIM sub-component can in fact already be used to the control such elements, but at the level of the overall GeometricQC solution, supporting the control of such element would require developing relevant QC Rules (see D5.6 for details of what QC Rules are). With regard to COGI-QC-11, the Point Cloud Matching and Segmentation sub-component allows to load and interpret the as-built data from the acquired point clouds, allowing the interconnection with other sub-components to automate the subsequent QC processes. Finally COGI-QC-16 is partially achieved by using the Open3D library, which is able to handle the point clouds in PLY and E57 formats.

Table 9: Point Cloud Matching and Segmentation sub-component's coverage of the stakeholder requirements reported in D2.1

ID	Description	Type	Priority	Status

COGI-CS-1	Runs on desktop or laptop PC	• Operational	Must	Achieved
COGI-CS-4	Runs on Windows	• Operational	Must	Achieved
COGI-CS-5	Runs on Mac	• Operational	Could	Achieved
COGI-QC-8	Supports systematic quality control on earthworks, substructure, concrete works	• Performance	Should	Partially achieved
COGI-QC-11	Automates QC-related activities	• Functional • Operational	Must	Achieved
COGI-QC-16	Handles point clouds standard formats (E57, PLY)	• Functional	Must	Achieved.

4.9 Assumptions and Restrictions

The Point Cloud Matching and Segmentation sub-component has a number of assumptions and restrictions, which are presented in the following:

- BIM elements and point clouds are expected to have the vertical along the +Z axis using the right-hand side coordinate frame.
- The coordinate reference frame within the entire GeometricQC tool uses global coordinates, hence, the generated output files contain coordinates information in the same global space.
- The measures are stored using the international metric system, and the distance measurement unit used in the sub-component is the metre [m].
- The input as-built data has to be pre-registered in the same global coordinate system as the BIM file, minimising the registration error to ensure an accurate point cloud matching and segmentation.

5 Conclusions

The Scan-vs-BIM solution is to be employed to generate the input data for the geometric QC components. The solution uses as input the as-planned data in the form of a BIM model (digital semantically-rich 3D model of the building/asset created by the design team) as well as the as-built point cloud data obtained through laser scanning surveying during construction. The BIM model data is interpreted from an IFC file, while the point cloud data must be pre-registered in the coordinate system of the project (i.e., the same coordinate system as the BIM model).

As documented in this deliverable, the core functionalities of the Scan-vs-BIM solution are delivered by two sub-components. The secondary sub-component, the *BIM Element Manager* is the middleware tool to read, interpret, and manage information about the objects in the BIM model (e.g. mesh geometry, component type, etc.). The main sub-component, namely the *Point Cloud Matching and Segmentation sub-component*, performs the matching of the as-built point cloud(s) data to the as-planned 3D meshes of the elements in the BIM model. This results in a segmentation of the input point cloud(s); each segment contains the 3D cloud points matched to a given 3D element in the BIM model (and an additional segment containing non-matched points).

The sub-components (and the Scan-vs-BIM component overall) have been designed to be used by the GeometricQC tool, which in turn will ultimately automate geometric QC and export the results to the Digital Twin Platform. The sub-components have been designed to be modular and scalable, allowing the creation of a toolbox which could be used in any other relevant project. To check the validity and performance of the implementation, the Scan-vs-BIM solution has been tested with a common benchmark synthetic dataset commonly used across the various developed COGITO tools developers. The sub-components presented in this deliverable include all the relevant functionalities and produce outputs with good quality that can be used by the GeometricQC tool. They are integrated within the GeometricQC tool, and further details on this as well as the integration of the GeometricQC tool with the rest of the COGITO solution are described in D5.6. During pre-validation and validation works, new functional and non-functional requirements may arise that will be addressed in WP8.

References

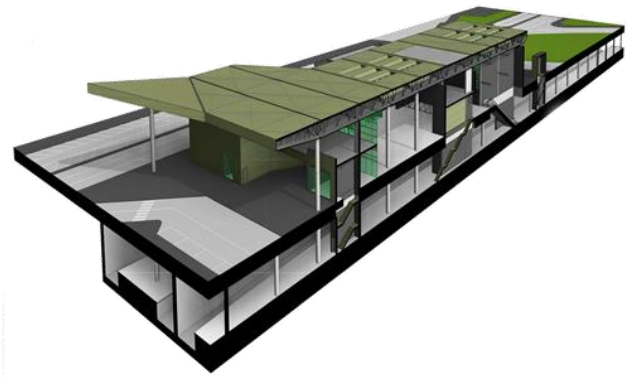
[1] BS EN 13670:2009, “Execution of concrete structures,” 2010.

[2] BS EN 1090-2, “Execution if steel structures and aluminion structures. Technical requirements for steel structure,” 2018.

[3] Autodesk, “Revit Sample Project Files,” Autodesk, [Online]. Available: https://knowledge.autodesk.com/support/revit/getting-started/caas/CloudHelp/cloudhelp/2022/ENU/Revit-GetStarted/files/GUID-61EF2F22-3A1F-4317-B925-1E85F138BE88-htm.html. [Accessed 29 04 2022].

[4] M. Fischler and R. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Communications of the ACM,* vol. 24, no. 6, pp. 381-395, 1982.

[5] P. Besl and N. D. McKay, “A method for registration of 3-D shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 239-256, 1992.



COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310