D4.2 Documenting the Changing State of Built Architecture

Software prototype v2

DURAARK

FP7 – ICT – Digital Preservation
Grant agreement No.: 600908

Date: 2015-03-31
Version 1.0
Document id. : duraark/2015/D.4.2/v1.0

Grant agreement number	:	600908
Project acronym	:	DURAARK
Project full title	:	Durable Architectural Knowledge
Project's website	:	www.duraark.eu
Partners	:	LUH – Gottfried Wilhelm Leibniz Universitaet Hannover (Coordinator) [DE] UBO – Rheinische Friedrich-Wilhelms-Universitaet Bonn [DE] FhA – Fraunhofer Austria Research GmbH [AT] TUE – Technische Universiteit Eindhoven [NL] CITA – Kunstakademiets Arkitektskole [DK] LTU – Lulea Tekniska Universitet [SE] Catenda – Catenda AS [NO]
Project instrument	:	EU FP7 Collaborative Project
Project thematic priority	:	Information and Communication Technologies (ICT) Digital Preservation
Project start date	:	2013-02-01
Project duration	:	36 months
Document number	:	duraark/2015/D.4.2/v1.0
Title of document	:	Documenting the Changing State of Built Architecture – Software prototype v2
Deliverable type	:	Software prototype
Contractual date of delivery	:	2015-03-31
Actual date of delivery	:	2015-03-31
Lead beneficiary	:	UBO
Author(s)	:	Sebastian Ochmann `<ochmann@cs.uni-bonn.de>` (UBO) Richard Vock `<vock@cs.uni-bonn.de>` (UBO) Raoul Wessel `<wesselr@cs.uni-bonn.de>` (UBO) Peter Törlind `<peter.torlind@ltu.se>` (LTU)
Responsible editor(s)	:	Sebastian Ochmann `<ochmann@cs.uni-bonn.de>` (UBO) Richard Vock `<vock@cs.uni-bonn.de>` (UBO)
Quality assessor(s)	:	Dag Fjeld Edvardsen `<dag.fjeld.edvardsen@catenda.no>` (Catenda) Martin Hecher `<martin.hecher@vc.fraunhofer.at>` (FhA)
Approval of this deliverable	:	Stefan Dietze `<dietze@L3S.de>` (LUH) – Project Coordinator
Distribution	:	Public
Keywords list	:	Curation, Documentation of Building State, Registration, Alignment, Difference Detection, BIM, Point Cloud, E57, IFC

Executive Summary

This document describes the second software prototype for documenting the changing state of built architecture which was developed in the scope of WP4. In addition to describing improvements of the Registration prototype (Task 4.1) over year 1, we tackle the problems of associating parts of low-level point cloud data with high-level entities specified in BIM (Building Information Model) datasets (Task 4.3), and determining differences between concurrent datasets of the same building (Task 4.2). Detected differences are visualized for interactive inspection by the user, e.g. in the ingest phase of the digital long-term preservation lifecycle, or when checking the integrity of datasets after data migration. Furthermore, we give an overview of the integration of our software component into the overall DURAARK system. In addition to the software prototype, we also describe the aquisition of test data at Luleå Tekniska Universitet.

Table of Contents

1 Introduction

Digital three-dimensional building models are quickly becoming the standard for planning the construction of new buildings, for planning renovation or retrofitting of existing architecture, and for documenting a building's state throughout its lifetime. Each of these tasks poses individual requirements on the data formats used. On the one hand, Building Information Models — which are often stored in the IFC (Industry Foundation Classes) format — usually represent the "as-planned" state of a building in form of parametric, interrelated elements like walls, floor slabs, stairs, doors, etc. Such models usually represent an idealized state of the building instead of its real (and possibly deviating) state. On the other hand, point cloud scans are used to capture a building's "as-built" state by performing on-site scans of the actual object. While such scans usually provide a high-resolution pointwise sampling of the building's surfaces and high spatial accuracy, these datasets are almost completely unstructured. For instance, an association of parts of the point cloud with building elements like walls or floors, which would enable e.g. a semantically high-level visualization of the point cloud by highlighting or hiding certain elements, is not available from the scanned data.

In order to bridge the gap between these seemingly orthogonal representations, methods for determining the association of (parts of) BIM models with (parts of) point cloud scans are the basis for higher-level processing and analysis of both kinds of datasets. For instance, such methods enable automatic detection of differences between the "as-planned" and "as-built" states of a building, visualizing these differences, and assisting the user to incorporate changes of the real building into the BIM model in order to keep it up-to-date.

Figure 1 shows an overview of the overall DURAARK system. The software components developed in WP4 (and WP5) are part of the Geometric Enrichment tools which work on BIM models (i.e. IFC files) and/or point cloud datasets (i.e. E57 files). In the envisioned workflow, tasks related to WP4 include the registration of datasets, e.g. spatially aligning a point cloud measurement to a corresponding Building Information Model, and the detection and visualization of possible deviations (e.g. intentional or unintentional differences between the as-planned model and the as-built measurements) between concurrent datasets of the same building. Linking these tasks to the use cases defined in D2.2.3, WP4 tackles

- UC4: Detect differences between planning and as-built state,

- UC5: Monitor the evolution of a structure over time,
- UC7: Plan, document and *verify* retrofitting/energy renovations,

and is furthermore related to the use cases

- UC1: Deposit 3D architectural objects (registration may be performed as part of ingest),
- UC6: Identify similar objects within a point cloud scan (detection of differences may also help to identify similar objects).

In this document, we first describe the year 2 developments of the Registration prototype (Task 4.1). Since development efforts regarding Task 4.1 were slightly decreased for the benefit of specific Tasks which were identified as particularly important to the stakeholder community (see Section 3.1), the developments in Task 4.1 were mainly focused on the integration into the DURAARK system prototype of Month 18. Furthermore, we present our software prototype for establishing associations between parts of point clouds with BIM entities (Task 4.3, see Section 4), as well as the detection and visualization of differences between concurrent datasets of the same building (Task 4.2, see Section 5). An overview of the integration of this software component into the overall DURAARK system is also provided. We conclude the document by describing a currently ongoing scanning effort at Luleå Tekniska Universitet that documents the renovation of a major part of the building. The collected data serves as a test and evaluation basis for the algorithms developed in the aforementioned tasks.
For details how to obtain the software prototype for the association of point clouds and BIM models, please see Appendix A.

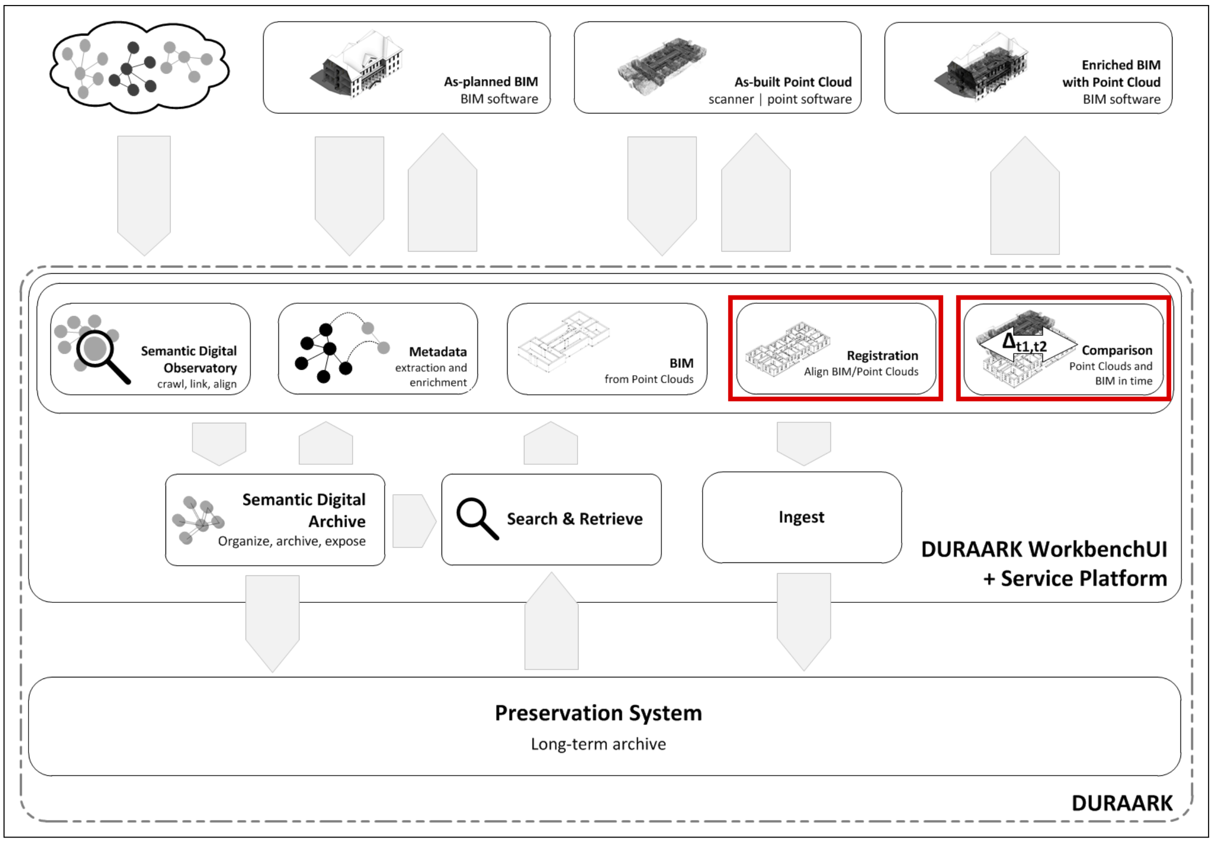


Figure 1: Overview of the DURAARK system. Components related to WP4 are highlighted. They are part of the Geometric Enrichment tools developed in WP4 and WP5.

2 Related Work

In this Section we give a brief overview of the related research work in the field specifically addressed by this deliverable as well as WP4 in general.

As becomes clear from the introduction, the problem of interlinking *as-planned* (BIM) and *as-built* (measured) geometric representations of architectural data is a reoccurring problem in several fields of building management ranging from the *building process* to *facility management* and ultimately *retrofitting/renovation.* It is therefore hardly surprising that there is a wide field of related research into methods that use contact-less methods of geometric measurement in order to facilitate and/or improve said processes.

Possible applications – to name a few – range from detection of unfinished building processes / construction process management (see [3], [4] and [1]), to structural deformation analysis [2] and ultimately very specific detection problems like e.g. automatic fall risk identification [6] or wall finish quality inspection [5].

Apart from the latter the general approach follows the same pattern: After acquisition of geometric measurements, all approaches first determine correspondences between both representations, determine an alignment and finally compute some form of correspondence in order to detect BIM entities in the measured data.

However, the type of measured data in these approaches differs significantly: Used capture devices produce measurements ranging from simple image data (i.e. photography [3]), over LIDAR/LADAR data [4] to laser scan data ([6], [2], [5] and [1]). Accordingly, the methods of registration and determination of correspondences differ as well.

In this deliverable we focus on the detection of *object-level differences* – which implies the problem of point-subset-to-object association – in laser scan data. The probably closest related research work has been done by Bosché [1] and follows a similar – albeit simpler – approach. The main difference in our work is that we restate the problem of object detection in that we consider objects "found" in the point cloud data based on *possibly observable surface* instead of the overall surface as done by Bosché. In our evaluations this proved to be a more reliable (and implementation-wise faster) approach to the overall problem.

With respect to visualization of differences we follow a similar approach to Golparvar-Fard et al. [3] in that we use basic, yet distinctive uniform colorization of object-level differences in order to provide a simple and fast overview over areas of structural deviation.

3 Registration

3.1 Introductory Notes

Concerning research and development work by UBO on Task 4.1 (Registration), it is important to note that we propose a continuation of this Task in Year 3 although the Description of Work states that the Task ends at the end of Year 2. The reason for this is that more focus had to be put on D5.3 than was anticipated. In particular, the reconstruction of parametric building models from indoor point clouds, which was developed as part of Tasks 5.1 and 5.2, was identified as a crucial building block in the course of the evaluation of the stakeholder community performed by WP7, as well as in face-to-face discussion during the stakeholder workshop in Copenhagen.

As such, we deemed it necessary and beneficial to the stakeholder community to intensify development efforts on the aforementioned WP5 tasks so that tangible results are produced well before the end of the project, even though the achieved results were initially planned for project year 3. This enables us to gather feedback on the developed methods during the third project year, which in turn allows timely exploitation.

As a consequence, UBO needed to find opportunities for compensating for the increased development effort of D5.3. We identified Task 4.1 (Registration) as the most suitable candidate without risking a negative impact on the stakeholder community since:

- the first software prototype for Task 4.1 already yields usable results, although there still exists potential for optimizations and, in particular, extensions regarding fully automatic registration, and
- the registration task requires – contrary to e.g. the reconstruction of BIM models, or analysis of differences between datasets – relatively little effort to be done manually in the worst case.

Therefore, development effort regarding Task 4.1 was decreased in project year 2 to the benefit of complex Tasks for which timely feedback and exploitation is crucial. Nevertheless, we describe Task 4.1 developments during the second project year in the following Sections, and propose the continuation of Task 4.1 in project year 3 in order to complete development work of planned extensions and optimizations.

3.2 Problem Definition / Challenges

In this Section we will briefly describe the overall registration problem and outline challenges encountered during evaluation of the Year 1 prototype. For an in-depth discussion of the related work on this problem we refer the reader to the Year 1 deliverable 4.4.1[1].

Since the overall scope of WP4 is to fill the gap between different geometric representations of architectural data (most importantly between BIM and point cloud representations) it is imperative for all tasks to be able to properly align said geometric representations. This is due to the fact that the geometry in all representations is merely positioned *relative* to a global reference system (i.e. coordinate origin and relative rotational pose) where the global reference system is not common across these representations. This leads to geometry being equally sized and structured but positioned differently in space (Figure 2, left). Registration now describes the process of finding a transformation T such that one representation transformed by T results in it being aligned to the other representation (Figure 2, right).

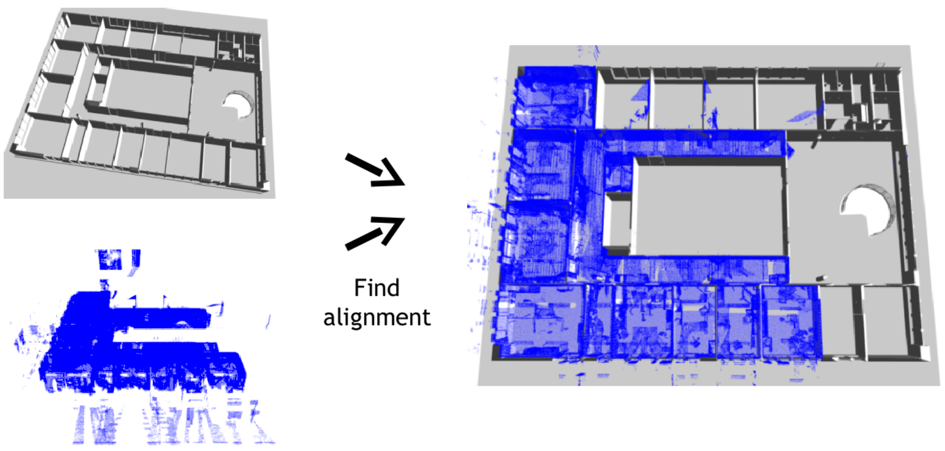


Figure 2: Schematic of the registration problem: Two different representations of the same building usually do not live in the same global coordinate system (left). The task is to find a transformation such that both representations are spatially aligned (right).

The primary difficulty (apart from performance issues) when registering geometry is the fact that both representations may be incomplete in that parts of either geometry do not have a correspondence in the other. For example point cloud scans usually contain additional geometry like scanned furniture; trying to align this geometry without

[1]`http://duraark.eu/wp-content/uploads/2014/02/duraark_d4.4.1_final.pdf`

corresponding representations in the BIM data is likely to introduce large errors in the computed transformation. For this reason the problem of registration contains the sub-problem of finding *corresponding geometry.*

The algorithm implemented for D4.1 tackled this problem by iterating a two stage process where in each iteration

1. correspondences between local geometry are searched, and then
2. the transformation which leads to the globally minimal distance between these correspondences is computed.

This approach is suitable in that it always converges to a minimum and terminates. However it tends to converge to a *local* minimum leading to wrong results after termination. Due to this, the stability of the algorithm is only given if both representations are at least coarsely pre-aligned since a good initial transformation prevents the algorithm from favoring wrong local minima.

In D4.1 this drawback was accounted for by allowing the user to manually pre-align the geometry before automatically fine-aligning using this iterative approach. This obviously poses major restrictions on software components that rely on a prior registration since it requires manual user intervention even if the subsequent computation might be fully automatic (as is the case with the association and difference detection described in this deliverable). Additionally, the time efficiency of the registration process depends substantially on the quality of the manual pre-alignment. Due to this there is a need for a fully automatic coarse as well as fine alignment.

3.3 Improvements Over D4.1 / Workflow

Our improvements over the first registration prototype during the second project year were mainly related to its integration into the M18 DURAARK software prototype. With the first DURAARK system prototype described in D2.4 and delivered in Month 18, the requirements for the integrated registration component have changed and the need for redesigning the interface led to some changes in the software.

More precisely, support for the E57 point cloud file format has been implemented in order to have a system-wide support for this file format. Additionally the need to call the software component automatically with predefined parameters has been met by providing a version of the registration prototype with command-line parameter support for the input files. The new component allows the registration prototype to be embedded in the DURAARK WorkbenchUI as an automatically callable tool. Lastly, the output format – which is a RDF/Turtle description of the alignment transformation – has been improved in order to account for the new requirements. The main improvement here is the ability to cope with arbitrary transformation types in order to account for future changes to the registration requirements (in its current form rigid transformations are supported, but the output format allows for non-rigid transformations in case the need for these arises).

In terms of user interaction workflow – apart from major improvements to the graphical output quality – the overall user experience has not changed since D4.1. The tools and user interface elements proved to be a viable way of interaction and have been kept as they were in Year 1. We therefore refer the reader to D4.1 for a description of the GUI workflow. Additionally the manual alignment is used in the software prototype for the association / difference detection and therefore part of the GUI described in Sections 4.2 and 5.2.

3.4 Future Plans / Extension

As mentioned in the beginning of this Section, we propose to extend the registration task to year 3. For most of the geometric enrichment components as well as for the overall integration into the DURAARK Service Platform there is a need for a fully automatic registration workflow. An extension to year 3 will allow us to

- exploit the specific properties of geometric building representations to find a way to quickly and automatically pre-align geometric representations of architectural data,
- substantially improve the processing time of the automatic fine-alignment,
- combine both approaches above into a merged, fully-automatic alignment workflow and
- use the resulting registration method to have all components developed in WP4 work without the requirement of manual user interaction.

4 Association of BIM Models and Point Clouds

In this Section, we present our results of Task 4.3 – Transfer of structure and semantics which deals with the association of (parts of) BIM models with (parts of) point clouds. We chose to describe Task 4.3 before Task 4.2 since the latter builds upon associations determined using the method described here.

4.1 Problem Definition

The overall goal of WP 4 is to merge the two worlds of BIM and point cloud scans. This task is far from being trivial since both representations describe completely different things on many different levels. For the sake of outlining the general problem we will briefly describe the geometric differences before addressing our solutions.

BIM (IFC) In terms of geometric representation, IFC files describe *semantic entities* like e.g. walls associated with a *parametric* description of an actual object. In the example of walls, one way of achieving this is to describe the ground silhouette of the wall together with a height (an example of a geometric IFC representation with basic structures is shown in Figure 3).

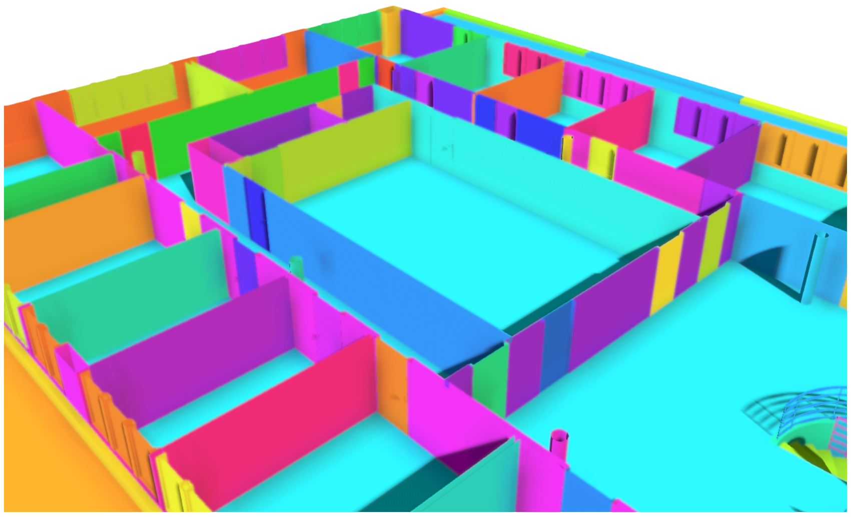


Figure 3: IFC model with entities colored separately.

It is important to stress that these descriptions model mathematical descriptions of *volumes*. This also implies that the *surface* of said representations is not explicitly defined. While this is even the case in a mathematical sense (possible surface representations would be e.g. implicit definitions, boundary representations etc.) it is especially complicated in terms of the actual representation commonly used in software (which is triangle-based boundary representations where the tesselation is arbitrary). Finally it is also important to mention that these objects may result in overlapping geometry; for example it is often arbitrary where one wall ends and the next begins (see Figure 4). This means that it is always (ambiguously) possible to map an IFC object to a volume in 3D space, but not vice-versa.

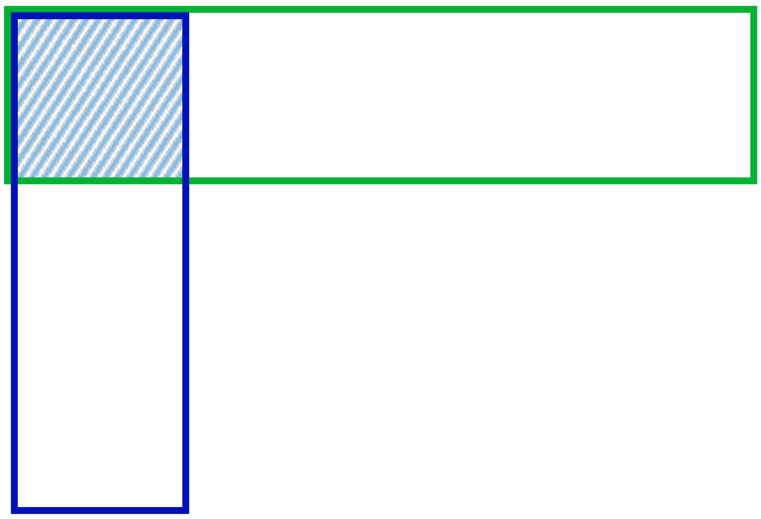


Figure 4: Overlapping wall volumes. The shaded area cannot be associated unambiguously to one of the two walls.

Point Cloud Scans On the other hand, point cloud scans are a discrete measurement of *surfaces* which is usually performed using time-of-flight or phase-shift-based scanning techniques. This also means that measurements are highly unrelated to the actual geometry whenever highly transparent or reflective materials are encountered. It is important to stress that each point is a distinct, independent measurement so the resulting sampling of surfaces has no information at all about surface connectivity/topology. It also obviously lacks any semantic value.

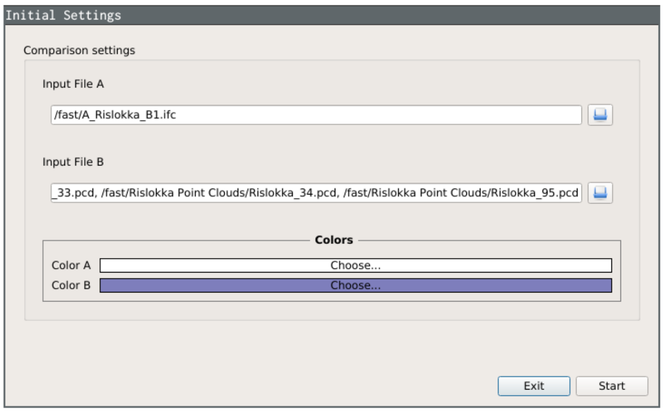
With these geometric differences highlighted, it is clear that the primary goal of this task, which is to *associate subsets of scanned points with their respective IFC entities*, is in theory ambiguous, ill-posed and therefore not solvable. However these problems do not prevent the human brain from "solving" this task in a consistent and generally agreed-upon way which brings with it the wish to find a well-working heuristic that accomplishes this automatically.
The primary use of this association is to semantically enrich the purely geometric information given in point cloud scans. This enables a broad range of applications in editing and visualization where it e.g. provides a way to hide or emphasize entire object families (think: walls or floors) in order to facilitate 3D navigation.

4.2 Workflow

In this Section we give an overview of the general workflow of our interactive comparison tool developed as the Year 2 deliverable in WP4. The general idea is to allow the user to load two representations of the same building, perform the automatic associative analysis, and finally write the results in the form of an RDF file to disk. We will now provide a step-wise description of an example usage scenario:

Step 1: Setup.
When starting the application, the user is shown a simple setup dialog (Figure 5) where

1. files for both representations (IFC and point cloud data) should be chosen, and
2. initial colors for both representations may be specified (which helps greatly when the subsequent alignment step is performed).


Initial Settings

Comparison settings

Input File A: /fast/A_Rislokka_B1.ifc

Input File B: _33.pcd, /fast/Rislokka Point Clouds/Rislokka_34.pcd, /fast/Rislokka Point Clouds/Rislokka_95.pcd

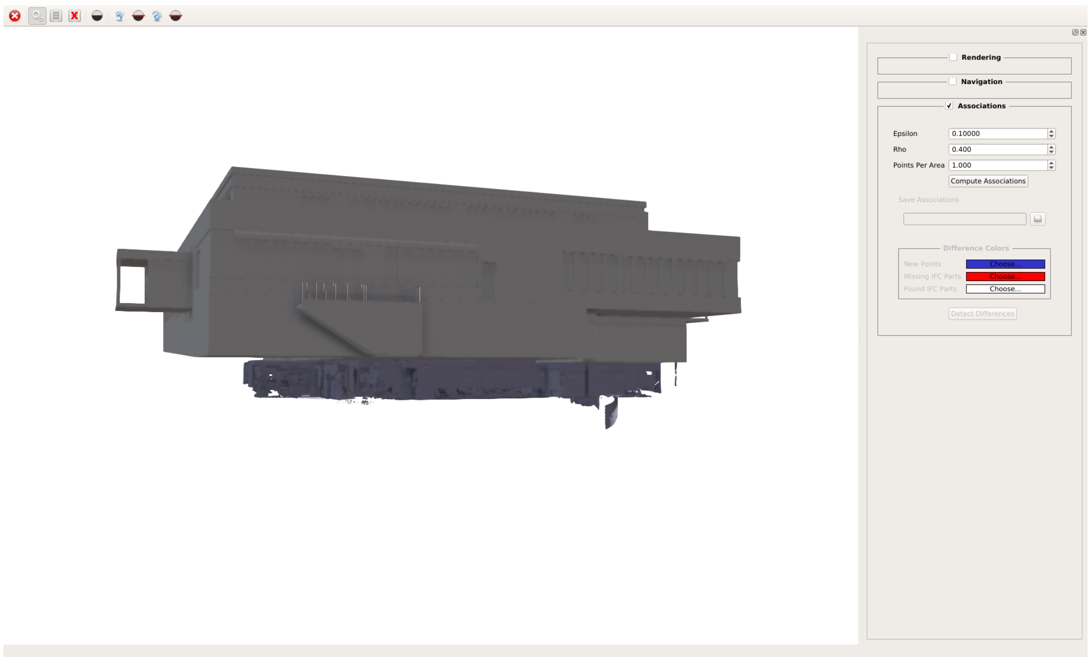
Colors

Color A: Choose...

Color B: Choose...

Figure 5: Initial settings dialog with example input.

Step 2: Navigation.
After both representations have been loaded (which may take a few minutes due to the IFC being converted to a renderable mesh representation as well as a few other preprocessing steps), they are rendered in a mouse-navigable 3D view with the previously chosen colors (Figure 6).
Since navigating the geometry with two representations can be quite cumbersome, the application includes a tool (in the upper toolbar) to partially hide geometry based on



Epsilon 0.10000
Rho 0.400
Points Per Area 1.000

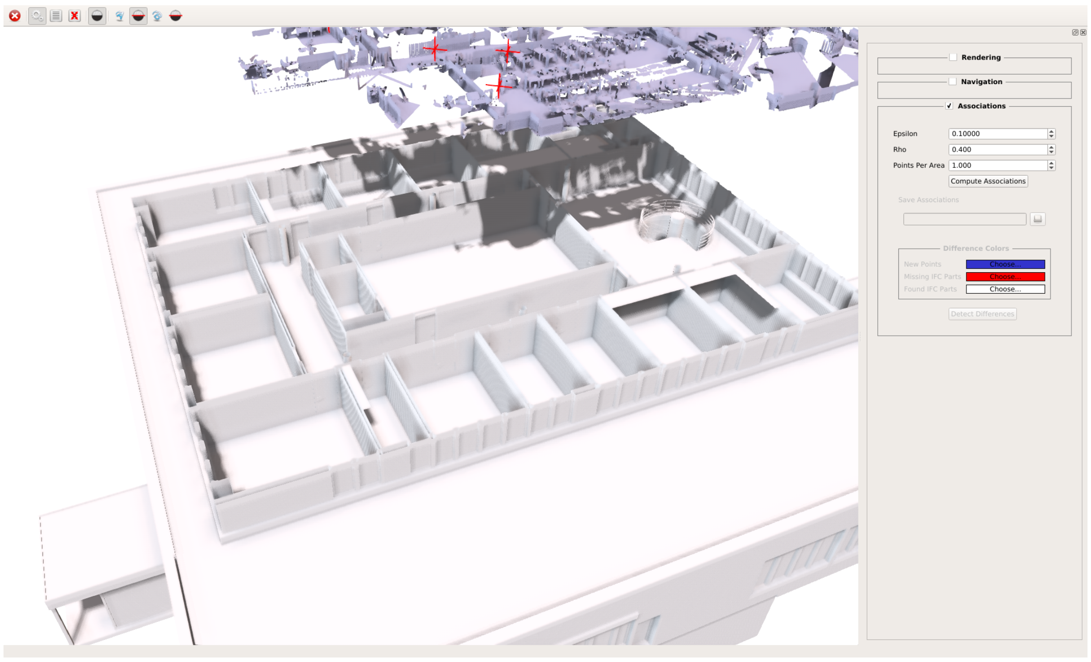
Figure 6: Prototype application after loading both representations.

height (Figure 7). This works by utilizing fast GPU-based clipping for cutting everything above an adjustable height which plays a key role in focusing on the relevant storey.

Step 3: Manual alignment.
In order to compare both representations the user must align the geometry manually[2]. Via two transformation modes reachable through the upper toolbar, the user can drag both representations separately in order to align the geometry (Figure 8).

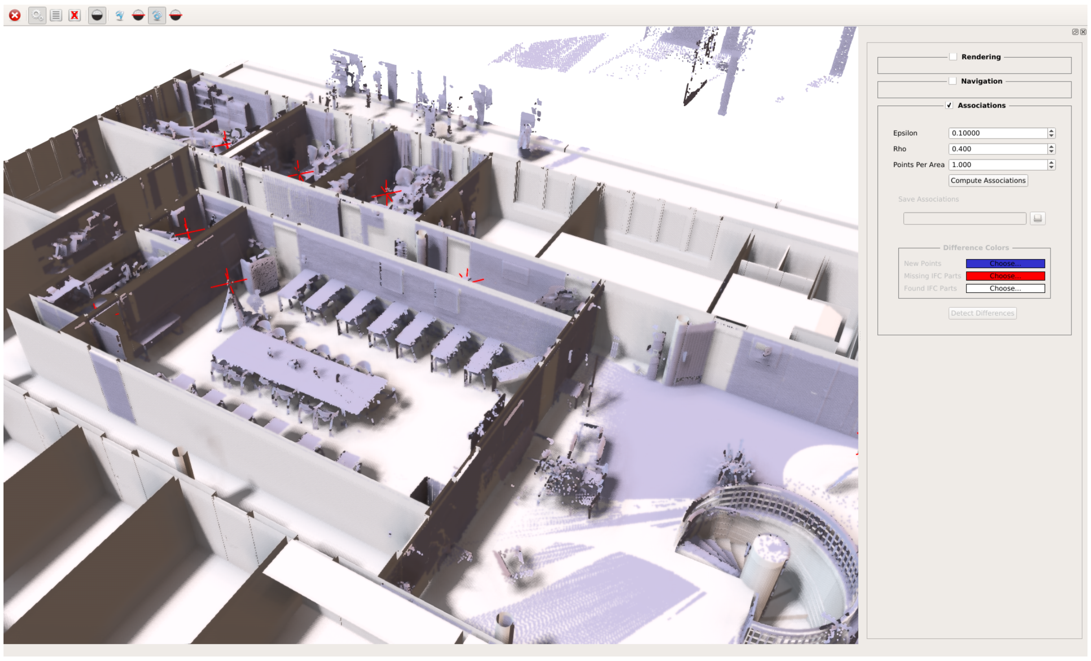
Step 4: Compare geometry.
Once both representations are aligned, the user can automatically compare both representations using the right-hand settings tab (Figure 9). The tab consists of adjustable parameters with a sensible default value set and a button to perform automatic comparison. Once the comparison is finished the user can write the output to a file using the provided file dialog.

[2]This manual alignment will be substituted by an automated registration in Year 3.



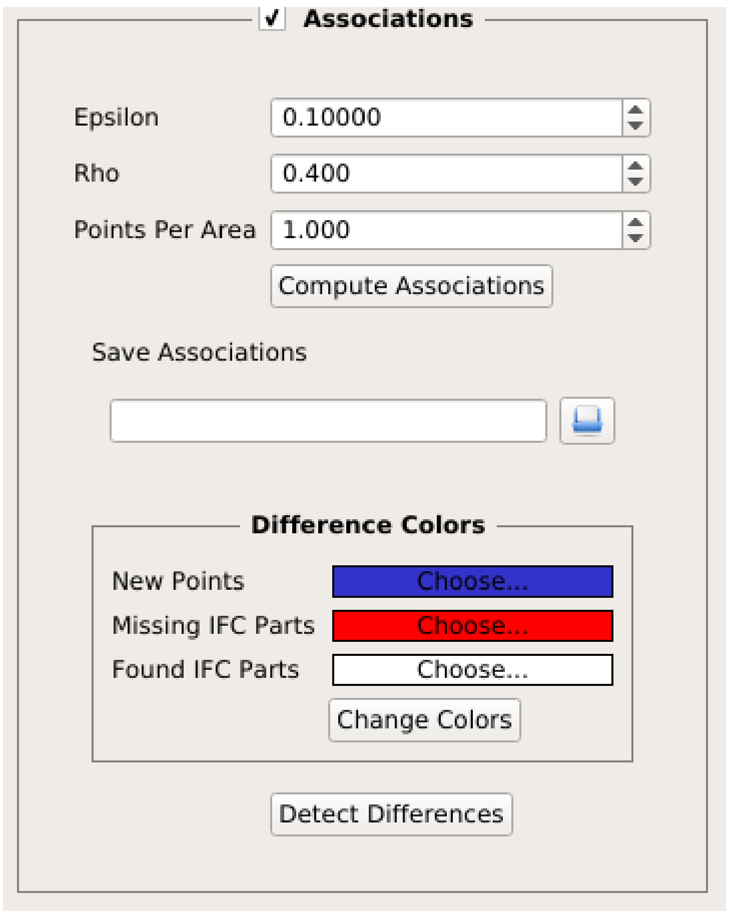
Epsilon 0.10000
Rho 0.400
Points Per Area 1.000

Figure 7: Clipping of geometry based on height.



Epsilon 0.10000
Rho 0.400
Points Per Area 1.000

Figure 8: Manually aligned geometry.



[✓] **Associations**

Epsilon	0.10000
Rho	0.400
Points Per Area	1.000

Figure 9: Adjustable properties for the comparison operation.

4.3 Method Overview, Implementation Details

Our method for the association of point subsets to BIM objects is a “phenomenological” approach which serves as a means to prune noisy data as well as resolve ambiguities. The primary idea is to *simulate* the scan process in a triangle mesh version of the input IFC file. There are two primary motivations for this approach:

First, it solves the problem of false positives when matching points with objects. One could merely look at point positions, find the nearest IFC object and if the distance between both is small enough associate the point to the IFC object. However, this assumes that all points on an object *directly* originate from scanning this object which is not the case when considering transparent or reflective materials. Second, our approach gives a good estimate of the observable surface of the object which plays a key role in the difference detection method described later in this report.

Our approach is based on a simple idea: Consider the scan origin and all “rays” (laser directions) emitted; now assume the same position in the BIM model and perform the virtual scan by casting rays in the exact same directions. By comparing both the real and virtual rays we can determine whether they hit the same object and – if this is the case – which object. This way we can safely reason if a set of points actually originates from a scan of a specific object and consider both associated.

The actual algorithm is in theory quite simple but requires some notation we will introduce here. We will denote a point cloud scan $\mathcal{P}$ as a tuple (P, p_o) with $P \subset \mathbb{R}^3$ being a set of 3D points and $p_o \in \mathbb{R}^3$ the scan origin. The latter is important, since it allows us to define for each point $p_i \in P$ a *scan depth* $t_i := \|p_i - p_o\|_2$ and a *scan direction* $d_i = \frac{p_i - p_o}{t_i}$.

For each point cloud scan $\mathcal{P}$ we now perform a virtual laser scan in the triangle mesh version of the input IFC scene. This is done by GPU-accelerated ray casting from the original scan origin p_o in the directions of the original d_i. This yields a new virtual scan $\mathcal{P}_v = (P', p_o),\ |P'| = |P|$ with the same directions d_i and new scan depths t'_i such that $p'_i = t'_i d_i$.

The method now simply compares the scan depths of both scans: We associate each point p_i with $|t_i - t'_i| < \varepsilon$ to the IFC object hit by the i'th ray in the virtual scan, where ε is a tolerance threshold chosen by the user to fit the data. For each IFC object o_j this approach yields an index set $A_j \subset \mathbb{N}$ of associated points in all performed point cloud scans which is then used for visualization and output.

4.4 Evaluation

In order to provide a scientifically meaningful evaluation of the association algorithm's output it would be necessary to have annotated groundtruth data, which was unfortunately not available for this deliverable, since generating said data would mean a lot of manual work. However, since the association algorithm's output provides the fundamental data on which the difference detection described in Section 5 operates, the visual output of the latter can provide a decent insight into the association algorithm's output quality. We therefore direct the reader to Section 5.4 where the difference detection has been evaluated visually on two larger datasets.
Additionally, Figure 10 provides a visual depiction of the associaton computed for the Risløkka dataset by differentiating associated points using colors.

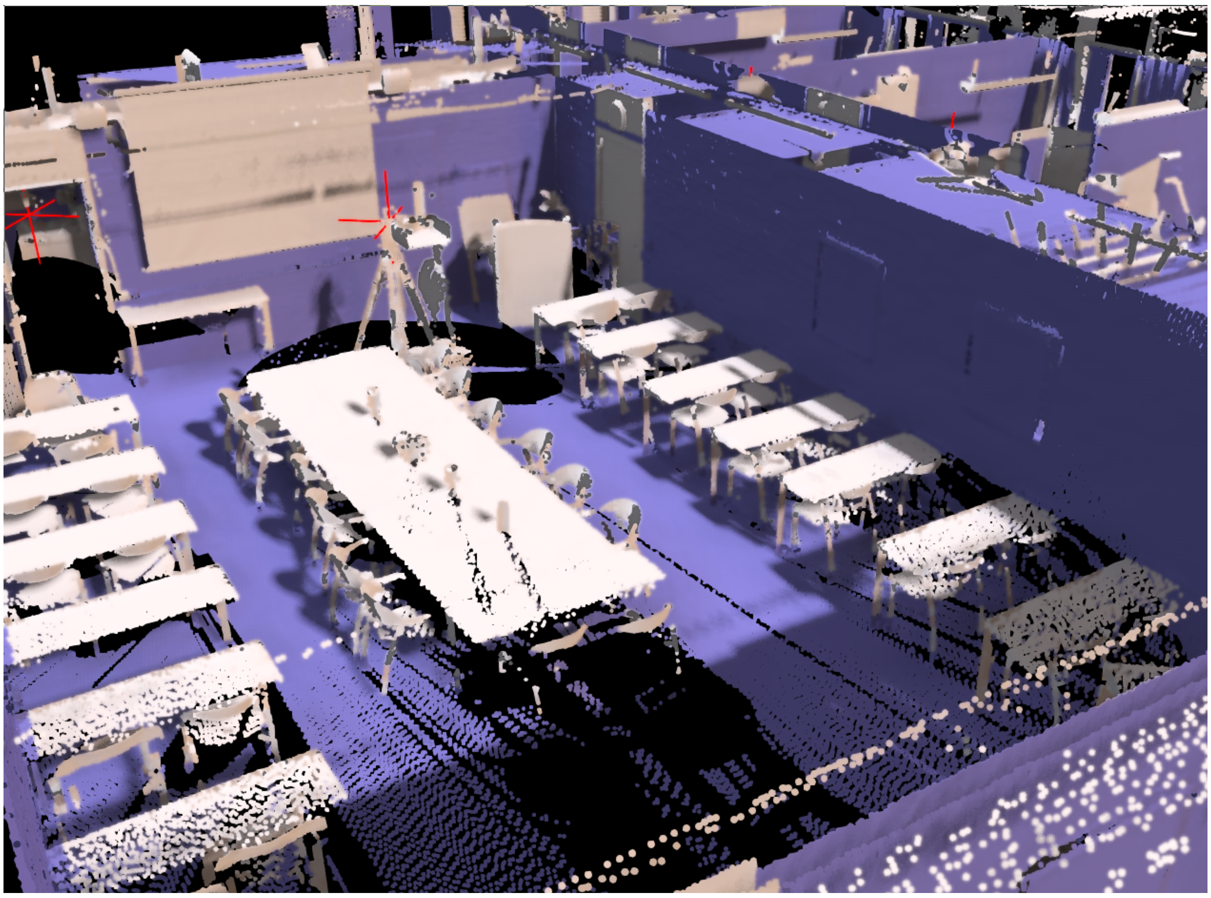


Figure 10: Association results for the Risløkka dataset. Blue points are successfully associated to recognized floor, ceiling and wall entities. The corresponding entities can be seen in Figure 3.

4.5 Integration

With respect to the DURAARK system, the prototype software for the association of building representations is supplied in the form of two software packages.
The first version is a graphical desktop application which serves as a prototypical user interface for the software and works as outlined in Section 4.2. This software is deployed separately as a Windows executable and can be accessed via the DURAARK WorkbenchUI on the client's system (Figure 11, upper-right).
The second version is a command-line application (which will be part of the "microservice-ifcaugmentation" in Figure 11) which can be used without a graphical interface by providing necessary parameters as program arguments and can therefore be tied to the Service Platform via a REST interface. The output of this tool is an RDF representation of the resulting associations. This software is deployed as a Docker [3] container (a self-contained OS with a pre-installed version of the tool) and can be used server-side without any user interaction.
Additionally, a binding for the Python programming language is supplied as a reference binding to the tools' functionality in order to be used in a plugin API context. This enables the usage in larger problem-specific applications such as architectural CAD software packages (Figure 11, bottom-right).
Figure 12 provides a workflow-centric localization of the developed components into the overall system.

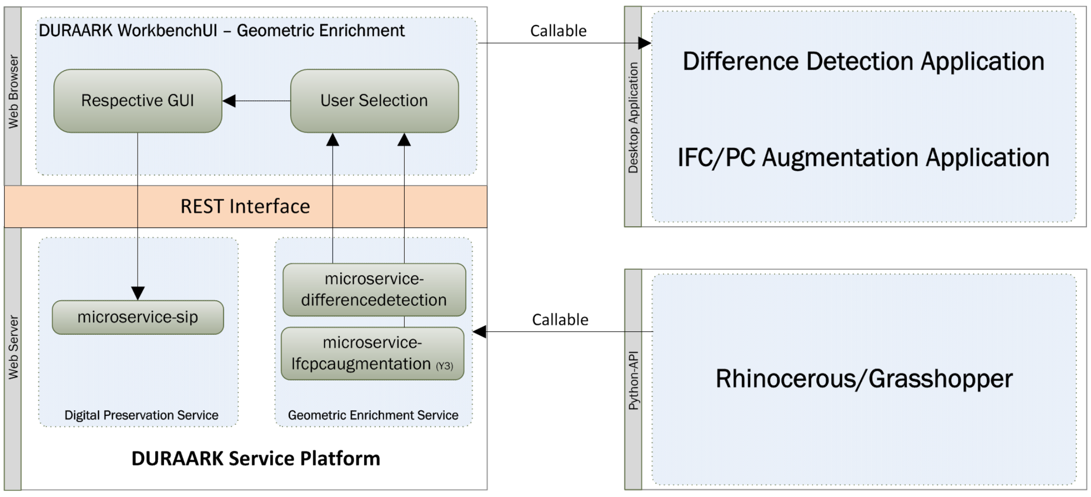

Web Browser
DURAARK WorkbenchUI – Geometric Enrichment
Respective GUI
User Selection
REST Interface
Web Server
microservice-sip
microservice-differencedetection
microservice-Ifcpcaugmentation (Y3)
Digital Preservation Service
Geometric Enrichment Service
DURAARK Service Platform
Callable
Desktop Application
Difference Detection Application
IFC/PC Augmentation Application
Callable
Python-API
Rhinocerous/Grasshopper

Figure 11: Location of the described components in the DURAARK system.

[3] https://www.docker.com/

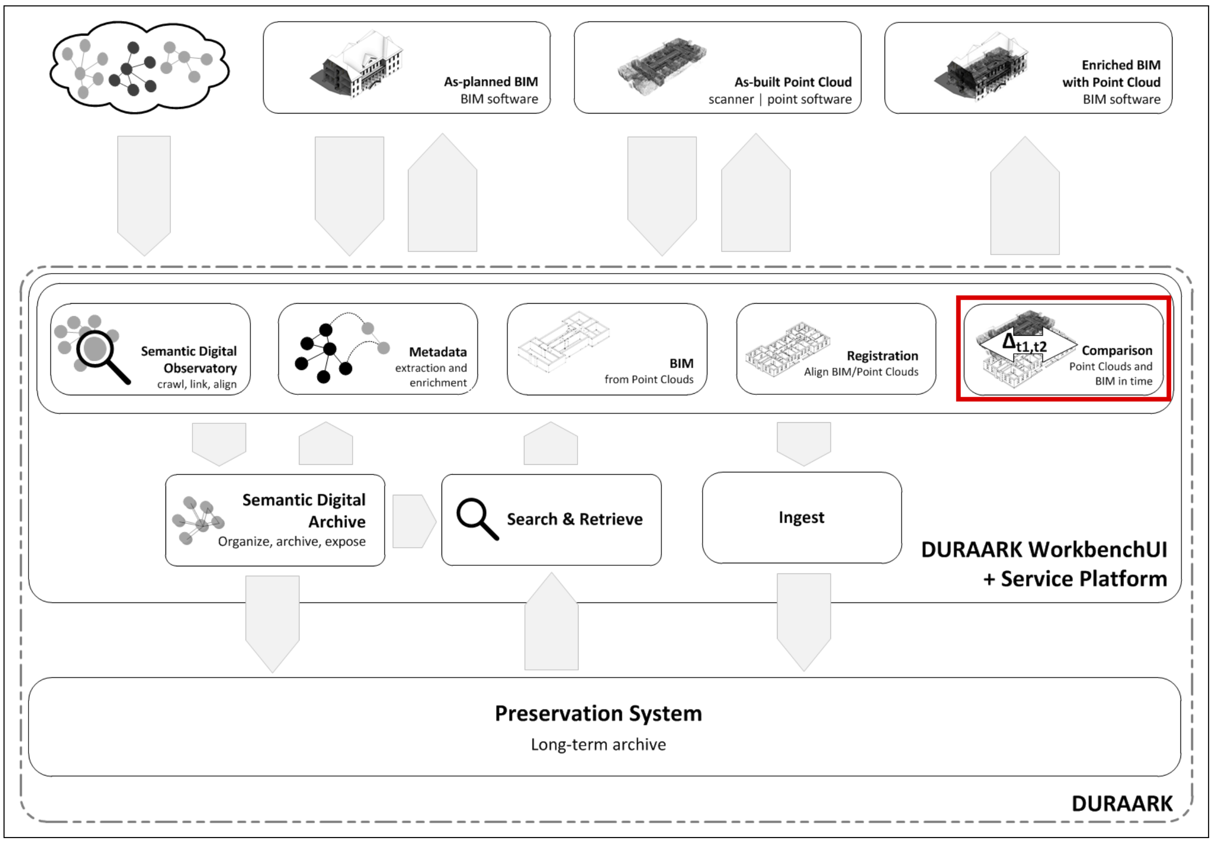


Figure 12: Location of the described components in the overall DURAARK workflow. Both the point cloud to BIM association as well as the actual difference detection are considered to be part of the "Comparison" step of the workflow.

5 Difference Detection

5.1 Problem Definition

In this Section we briefly present the motivation and challenges behind an algorithmic difference analysis tool. Once a suitable means of association between the two primary representations of architectural data in the DURAARK project – BIM models in the IFC format and point cloud scans – has been established (using e.g. the solution established in Section 4), it is only natural to also provide the complementary view of the comparison: Where associations cannot be made, the two representations possibly differ to an extent that is interesting to the user.
One obvious scenario is the automated check of the construction process itself: Once the architect designed and modeled the building, the resulting BIM representation serves as a means of communicating the *planned* state of the building to contractors (most importantly building companies). However, due to miscommunication, practical considerations, or simply matters of lack of precision, the *as-built* outcome might differ in small or even large aspects. At this point we aim for a new automated workflow of detecting said deviations and finding a quick and easy visual access to their regions of influence. The idea is to perform a laser scan to capture the as-built state of the building and have the software automatically scan for different kind of deviations.
In this context we differentiate between 3 classes of deviations:

- Systematic structural deviations. These arise when entire structures like e.g. walls are either missing, (apparently) surplus, or misplaced.
- Systematic geometric deviations, when objects are sufficiently associated in both representations but show systematic deformations, holes etc. A possible scenario here are floors which are structurally bending over time.
- Random deviations, i.e. noise introduced for example by the scanner or it's operating conditions/precision.

The latter two deviations imply the absence of the first kind, which is why we – in this deliverable – concentrate on detection and visualization of the first scenario. This brings us to the problem definition we aimed to solve for the year 2 contribution to WP4:

Given a BIM representation of a building (in IFC format) and a corresponding set of point cloud scans performed in the real building, determine which described objects/components have been observed *sufficiently well* in the point cloud measurement.

Note the emphasized part of this definition: The question whether or not we consider a scanned part of the surface of an object as a sufficiently complete representation – such that we then consider the object "existent" in the as-built state – is entirely subjective. This problem becomes more severe when realizing that the amount of (at all) *observable* surface depends on many factors (accessibility, occlusion etc.) and can be almost arbitrarily small.

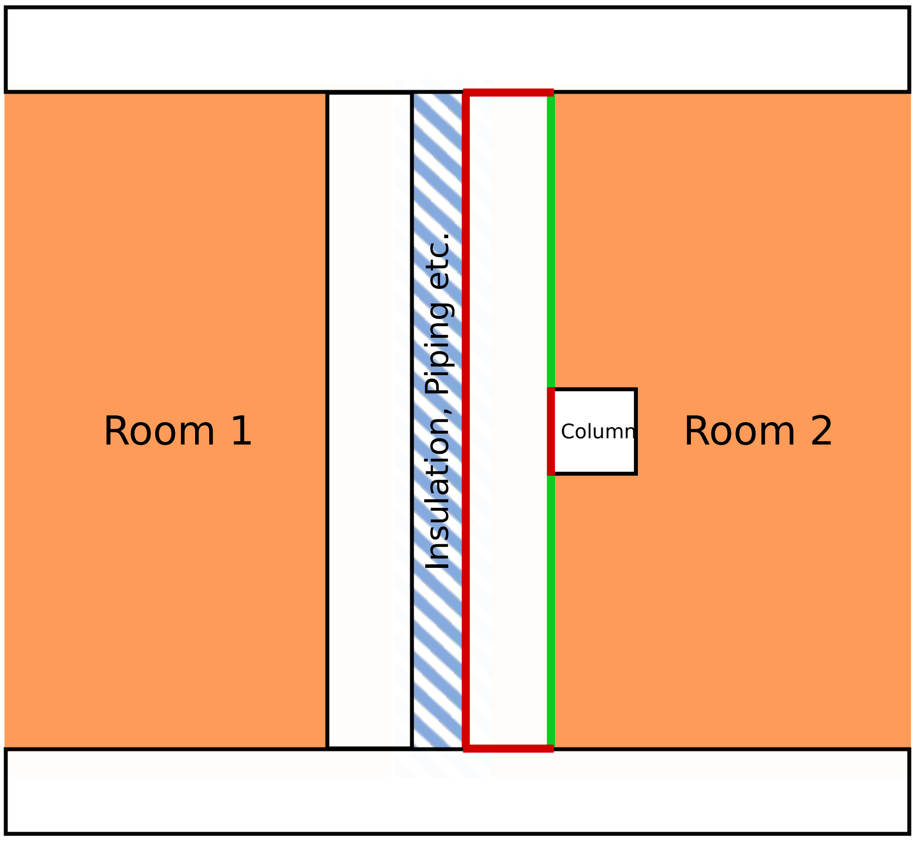


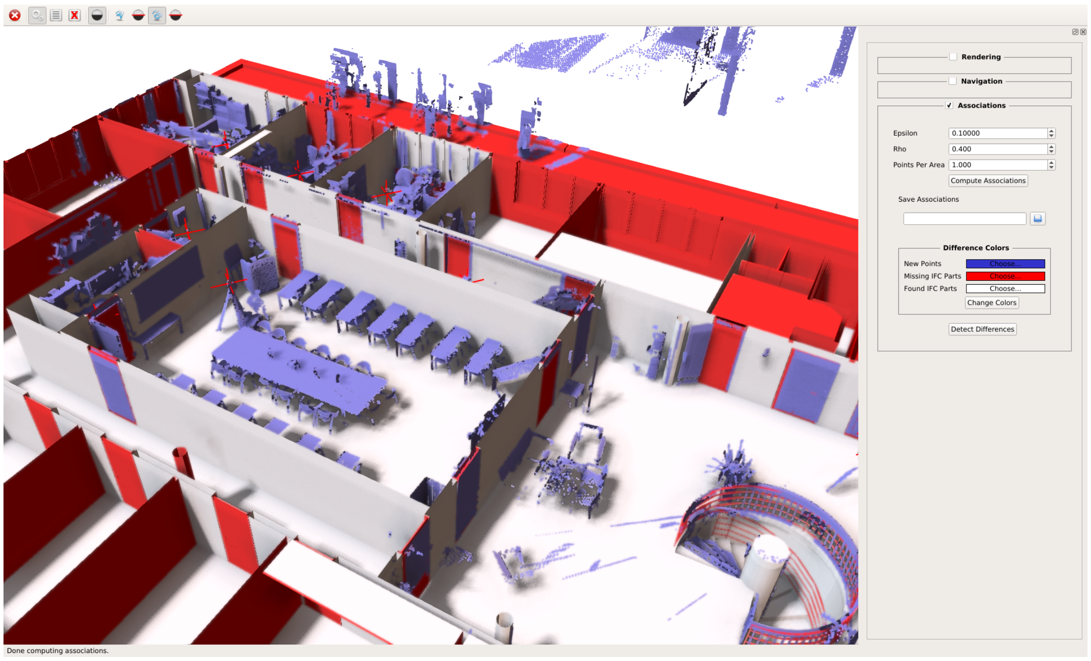
Figure 13: Partial observability. Green marks observable surface, red unobservable surface.

In order to illustrate the problem consider the situation illustrated in Figure 13. There are two walls separating two rooms with a gap modeled for insulation and piping. Additionally a support column is modeled (and hopefully built) in room 2. The interesting case now is the wall bounding room 2. The observable surface when scanning (green) is relatively small compared to the overall surface (red plus green). Note that this ratio could theoretically become arbitrarily small. The interesting question is: When do we consider this wall “observed” in the scan? It is clear from this example that observed surface area (ratio) is not a good measure. Our solution to this problem is to consider an object observed if the surface observed with respect to the surface that *could have been observed from the scanner position* is sufficiently equal. The underlying motivation is that if one measures an object from one side only, one can safely assume its back side to be present as well. Note that while this assumption is in theory obviously invalid it is an assumption made by the human brain as well and in the case of missing data by default valid enough to serve as a hint for further inspection by the user.

5.2 Workflow

In this Section we give an overview of the graphical version of our difference detection tool. Apart from exporting the results to a file, the initial workflow is the same as for the association tool described in Section 4.
Once the association is computed as previously explained, the GUI tool allows the user to compute the differences automatically using the parameters supplied via the right-hand menu.
The user is then shown a colorized version of the difference detection output (Figure 14) where

- points that are associated to an object are hidden,
- points that are not associated are shown in a selectable color and
- IFC objects are colored with chosen colors depending on whether they are considered "observed" or not.



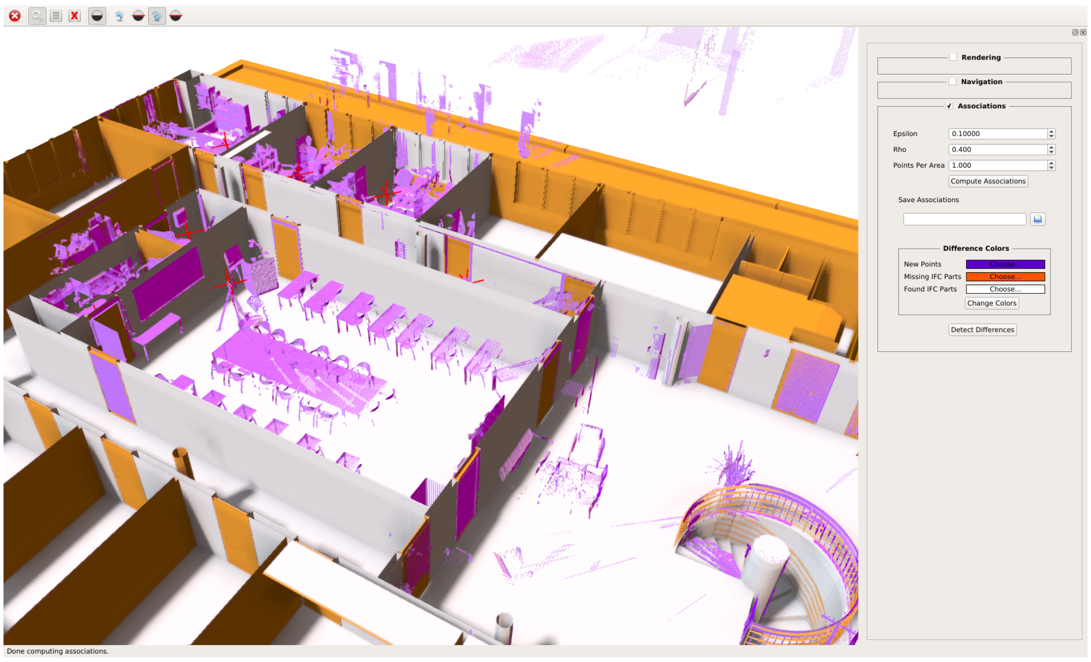
Epsilon 0.10000
Rho 0.400
Points Per Area 1.000

New Points 🟦
Missing IFC Parts 🟥
Found IFC Parts ⬜

Done computing associations.

Figure 14: Colorized output of the detected differences.

The colors used for the output visualization are selectable via a section in the right-hand side menu (see Figure 9). After the result has been computed, the user may also change the color and update the output according to the changes (transparent colors are also supported and result in transparent geometry) (Figure 15).



Epsilon 0.10000
Rho 0.400
Points Per Area 1.000

New Points 🟪
Missing IFC Parts 🟧
Found IFC Parts ⬜

Done computing associations.

Figure 15: Alternative coloring used for the difference visualization. Unassociated points are shown semi-opaque.

5.3 Method Overview, Implementation Details

In this Section the actual approach to the detection of observed and missing IFC objects is described. Our approach to a solution makes heavy use of the algorithm for automatic association outlined in Section 4; more precisely, said algorithm was specifically designed in the way described in order to also be able to efficiently detect object level deviations. From the problem definition it should be clear that for each object in the IFC representation we need two corresponding measures for the point cloud scans: The amount of points associated with an object and the amount of points *possibly observable* on the objects' surface from the scanner position. The latter measure is the precise reasoning behind the approach outlined in Section 4. The virtual laser scan performed for each respective real scan serves as a "groundtruth" to base a measure of observability on. More precisely yet simple: The amount of rays n_v in a virtual scan from the original scan origin in the directions of the original points which finally hit an object is precisely the amount of theoretically observable points. The amount of points n_p that are labeled by the aforementioned algorithm as associated to the IFC object (i.e. for which the ε threshold is not passed) serves as the measure of actually observed surface. We can now compare $\frac{n_p}{n_v}$ to a user specified threshold $\rho \in [0, 1]$ in order to decide whether or not we consider an IFC object represented in the point cloud scan.
However in some edge cases where the amount of possible as well as actually observed points are very low (although 100% coverage, we don't want to consider an object represented for which one out of one rays hit the object) we additionally check for covered area. More precisely we compare the average number of points per square surface area against a – usually low – relative minimum area $a_{min} \in [0, 1]$. This serves as a means to prune invalid false positives.
The final output of the algorithm is then a set of IFC entities considered "found" in the point cloud scan, which can then be used to highlight/navigate missing IFC objects or additional point data (e.g. furniture) missing in the BIM representation.

5.4 Evaluation

In this Section we present some images of the graphical output of the difference detection algorithm. In all images

- white geometry represents IFC objects found in the point cloud representation,
- red geometry represents IFC objects not found, and
- blue points represent subsets of scanned points not recognized in the IFC model.

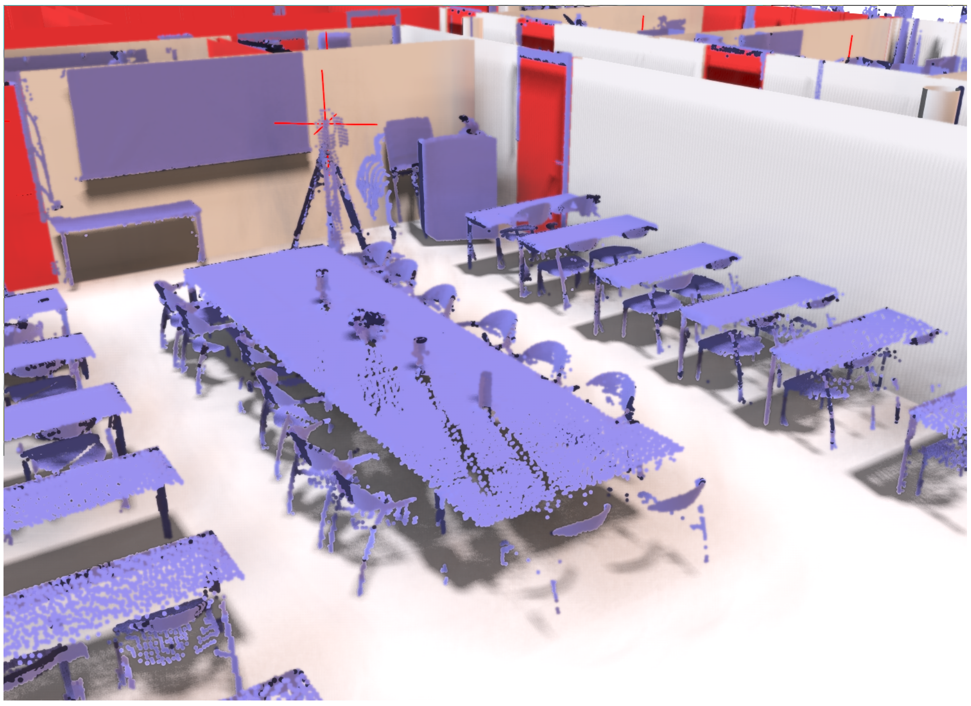


Figure 16: Risløkka datasets.

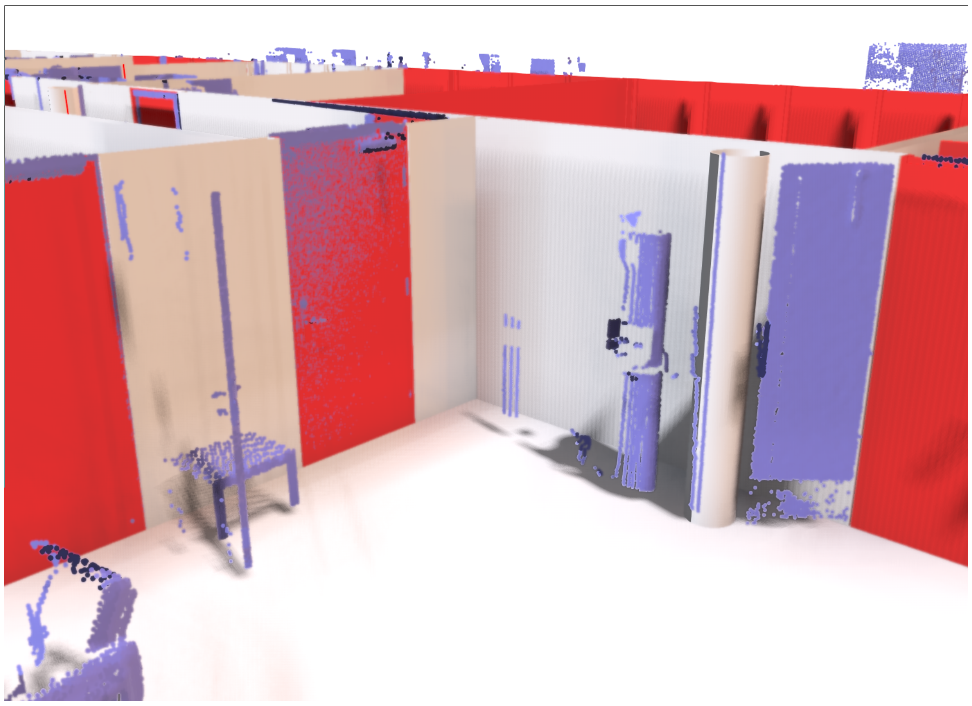


Figure 17: Risløkka datasets.

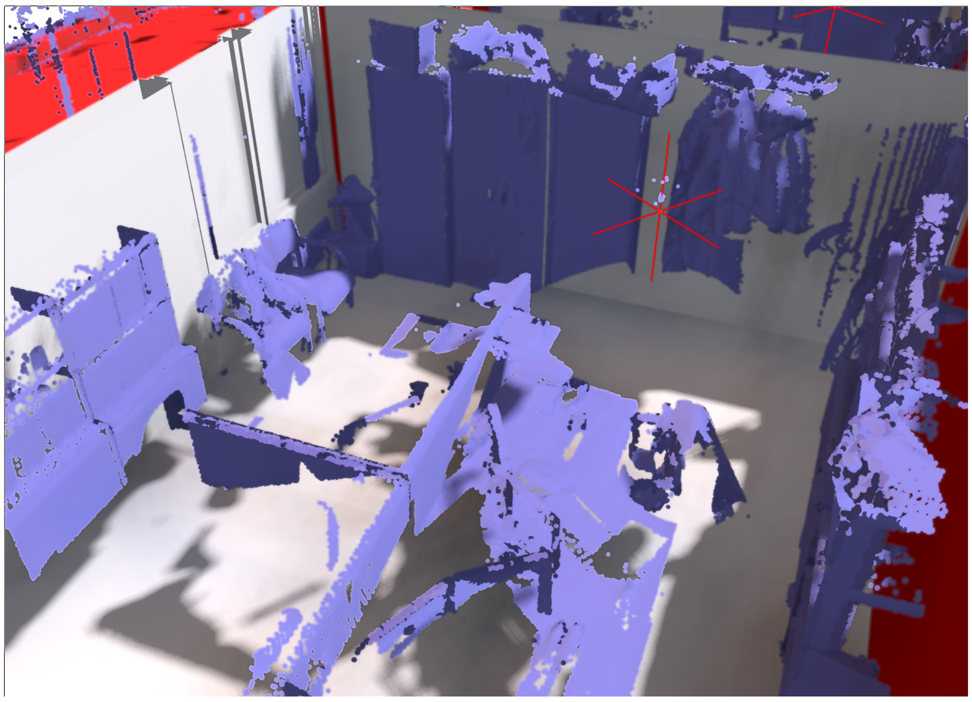


Figure 18: Risløkka datasets.

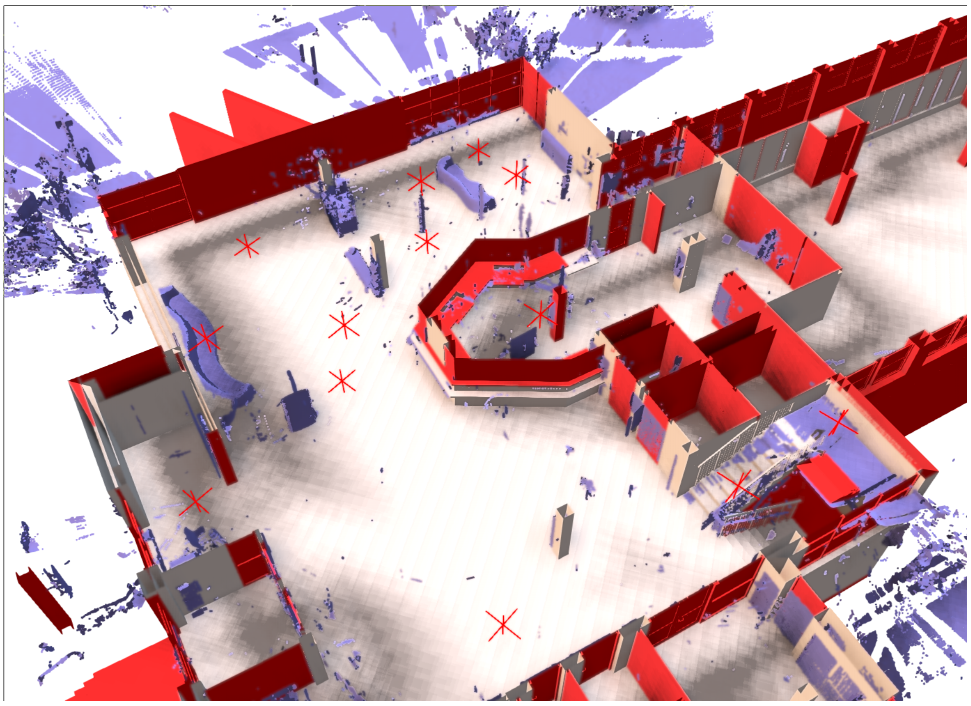


Figure 19: LTU datasets.

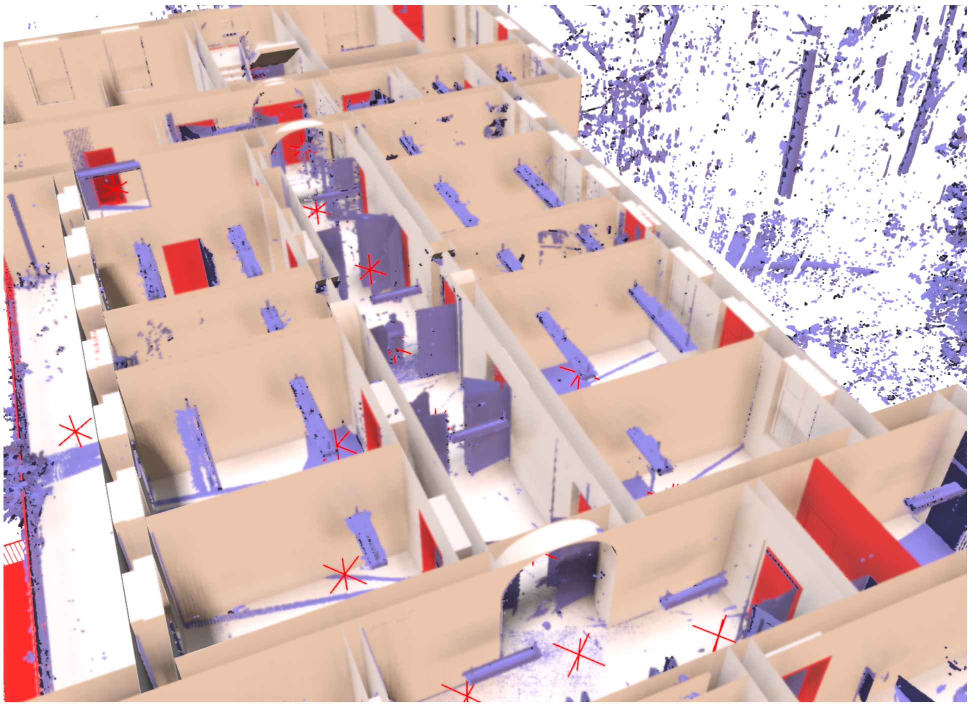


Figure 20: Plan3D (Haus 30) datasets.

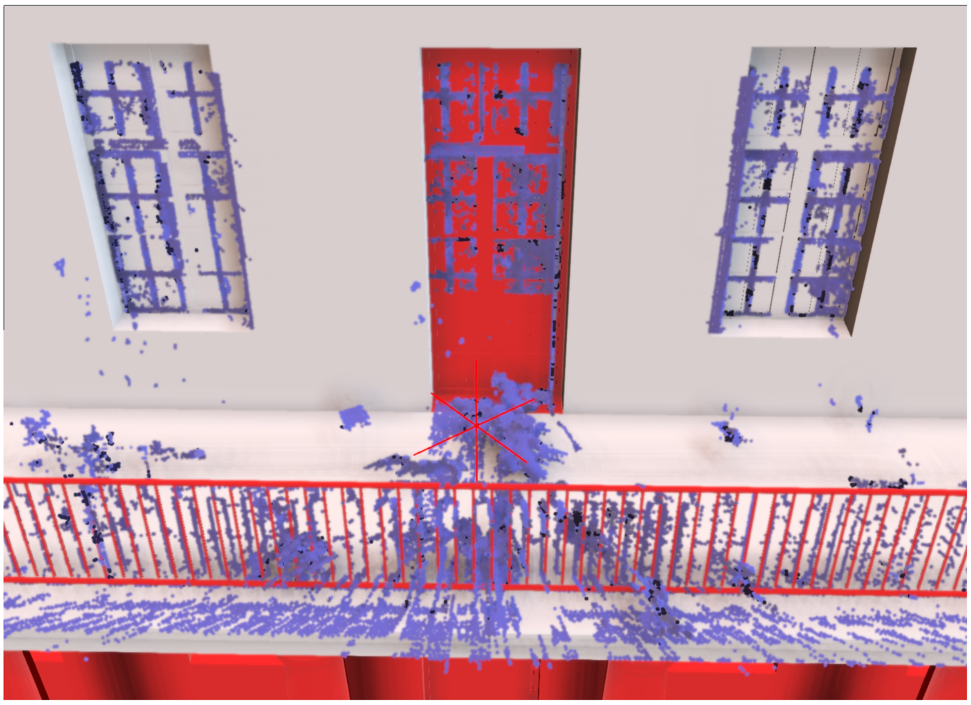


Figure 21: Plan3D (Haus 30) datasets.

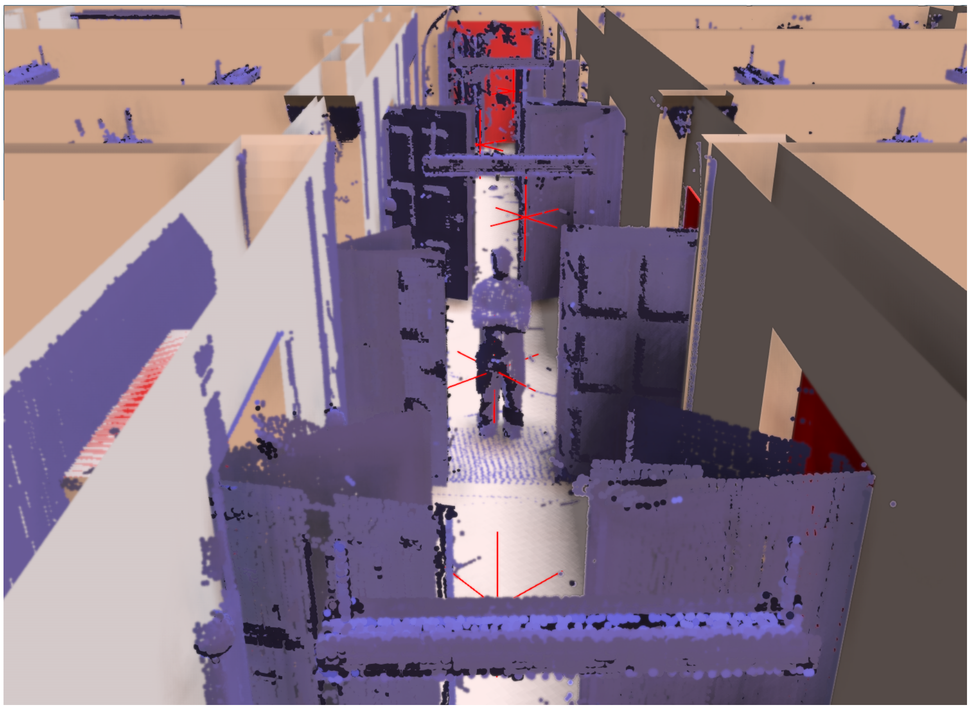


Figure 22: Plan3D (Haus 30) datasets.

5.5 Integration

Since the difference detection is primarily derived from the algorithm for the association of BIM and point cloud datasets, the prototypical implementation is supplied as an extension to the association tool.

Additionally, since the primary use case of the difference detection is an intuitive *visualization* of its results, we decided to focus on providing a standalone GUI application since this provides the greatest freedom in terms of custom visualization. In the context of the DURAARK system the aforementioned prototypical GUI application is integrated as a software module callable via the client-side WorkbenchUI interface (Figure 23, top-right). Given registered datasets and the computed associations, it is also possible to run the difference detection component in a headless manner as a microservice (Figure 24, bottom-left).

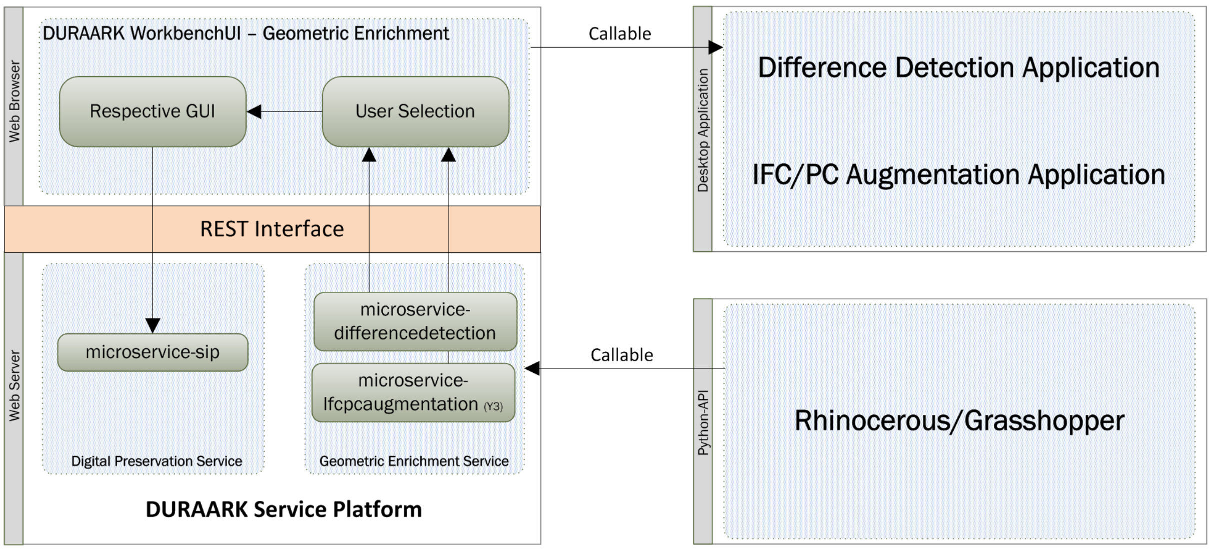

Web Browser
DURAARK WorkbenchUI – Geometric Enrichment
Respective GUI
User Selection
REST Interface
Web Server
microservice-sip
microservice-differencedetection
microservice-Ifcpcaugmentation (Y3)
Digital Preservation Service
Geometric Enrichment Service
DURAARK Service Platform
Callable
Desktop Application
Difference Detection Application
IFC/PC Augmentation Application
Callable
Python-API
Rhinocerous/Grasshopper

Figure 23: Location of the described components in the DURAARK system.

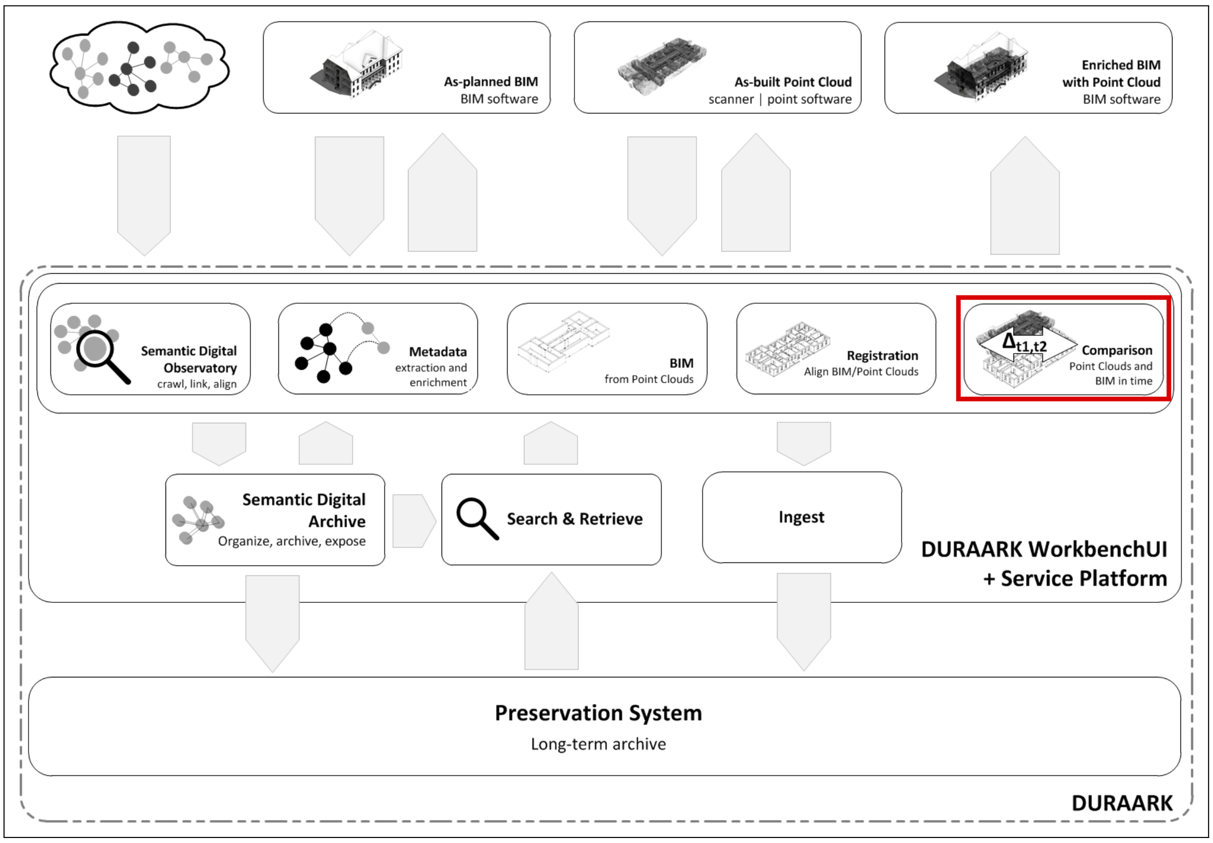


Figure 24: Location of the described components in the overall DURAARK workflow. Both the point cloud to BIM association as well as the actual difference detection are considered to be part of the "Comparison" step of the workflow.

6 Data Acquisition

One issue to some of the use cases was to have a complete documentation with IFC and point clouds before and after a rebuilding/renovation process. Therefore a current reconstruction at Luleå University of Technology was used as a demonstrator.
The data set will consist of two datasets from different parts of the building, one from the A-square which will be completely rebuilt, and one corridor which will undergo only minor renovations. Figure 25 below presents the old design and the new design.

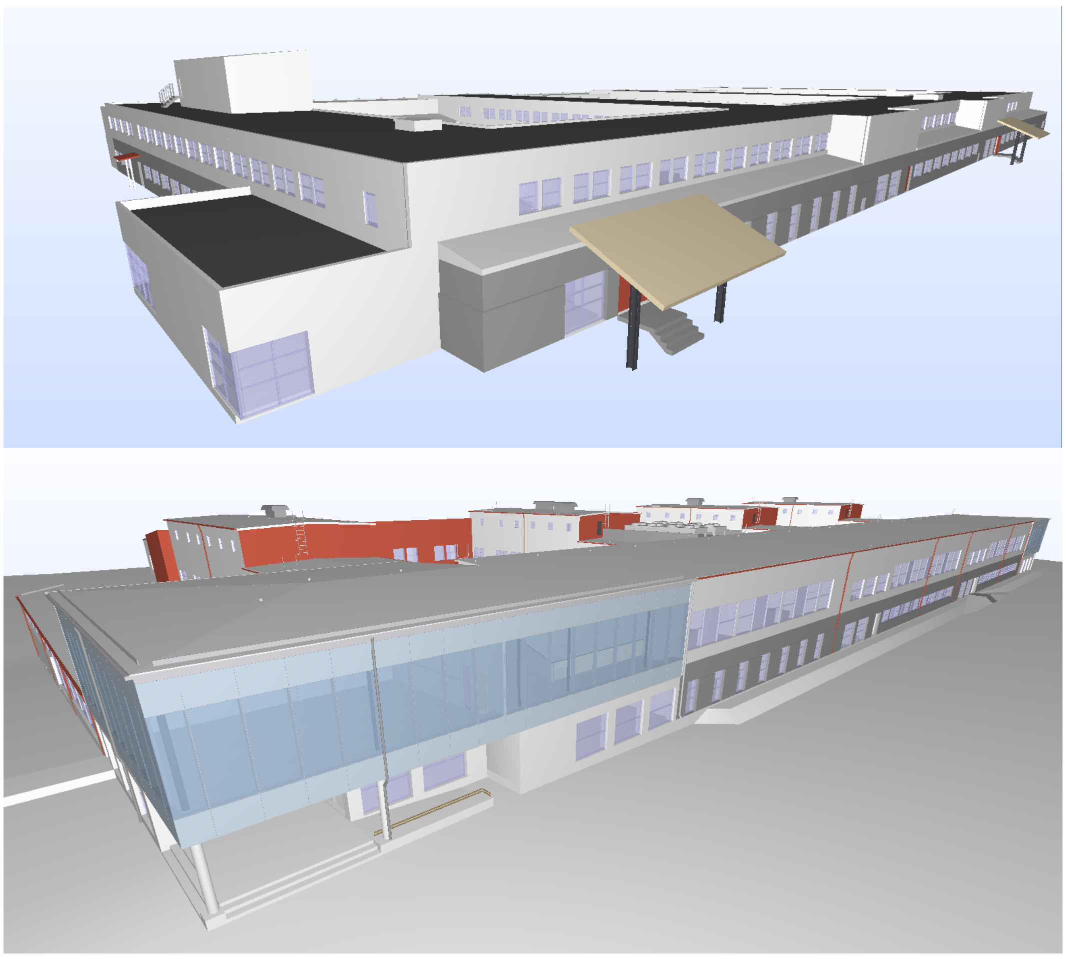


Figure 25: LTU A-house before (top) and after (bottom) the rebuilding.

6.1 Rationale

The A-house of Luleå University of Technology is undergoing a major reconstruction during December 2014 – December 2015. During this process, old documentation, 2D drawings, and measurements were used to create a model of the current building (before December 2015). This 3D model was generated in ArchiCAD and then exported to IFC.

6.2 Building model

The building model contains one IFC file for the current building (before renovation) and several files for different purposes for the new building.

IFC Filename	**Contents**
A-modell current conditions.ifc	Architectual model current conditions
A-modell redesign.ifc	Architectual model redesign
Air.ifc	Model of air ducts
Ducting.ifc	Model of ducting
Heating.ifc	Model of heating piping etc
K-model.ifc	Construction model
Plumbing.ifc	Model of plumbing
Sanitation.ifc	Model of sanitation plumbing (toilets, waste water etc.)
VOIDS.ifc	Model of empty spaces

Table 1: IFC files.

6.3 Scanning process

The scans were performed with a Faro Focus 3D S laser scanner. The settings for the scanner were set to capture 28 million coordinates for each scan.
To synchronize all scans, scan reference targets were placed at various locations in the building. Some targets were mounted on walls by magnetic precision washers, some were placed on small mini tripods on the floor, and some were mounted on clamps on pillars. Targets were placed at well-chosen positions and moved to appropriate positions during the separate scans. When moving the scanner to a new location connections were kept between the scanner and at least 3 targets from the earlier scan position and 3 targets in the new position. This is due to be able to register and merge all scans into the same coordinate system in the used point cloud software (Faro Scene).

6.4 Data set for the A-square

The A-square was chosen because this part of the building will include a large reconstruction, several outer walls and floors will be removed and rebuilt (see Figure 26).

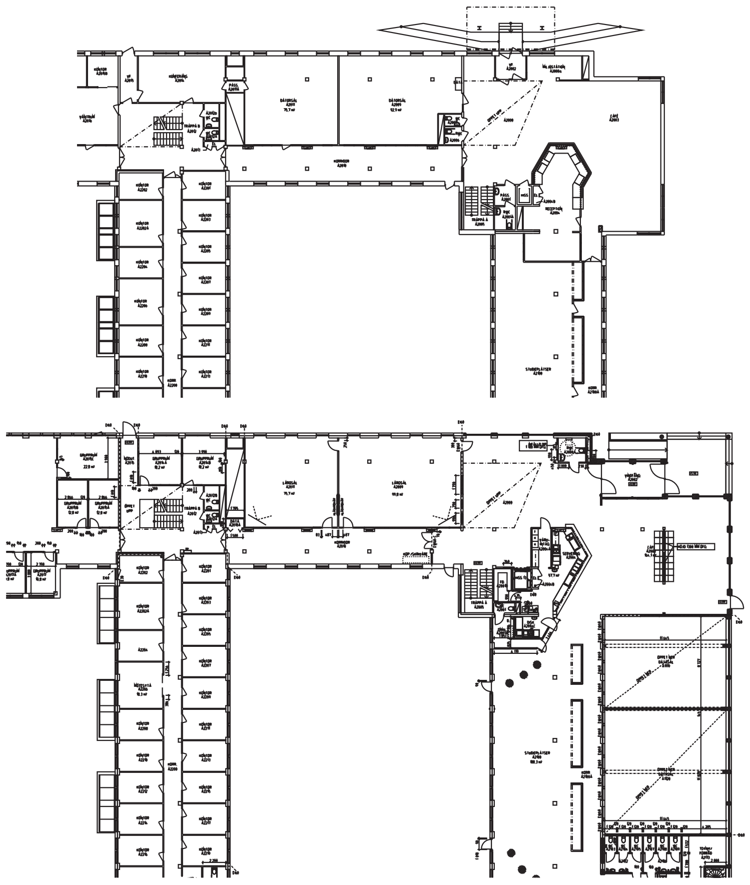


Figure 26: Floor drawing of the A-square before and after the rebuild.

The A-square consists of several separate areas on two different floors. 9 scan reference targets were used. Scans have been performed on two different occasions, on June 2014 (with some furniture) and August 2014 (without furniture). One additional scan will be performed when the building is finished (July 2015). Each scan contains about 360M points.

6.4.1 Point cloud set 1

For the first point set some furniture was still remaining in the facility, this scan has been done to enable tests with some furniture present which is a quite common issue when performing interior scans of an existing building. This of course complicates further work on the point cloud since furniture obstructs the view of the scanner. The positions of the scanners and the point cloud data is similar to the point cloud from point cloud set 2.

6.4.2 Point cloud set 2

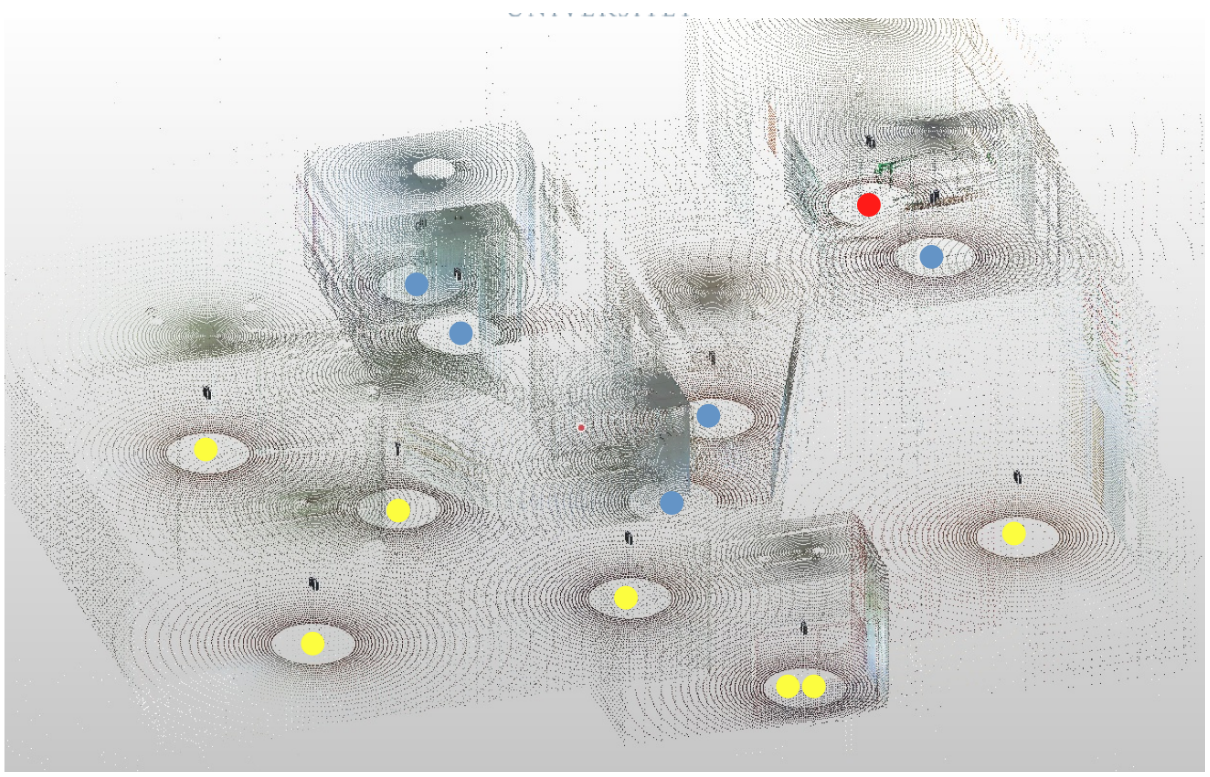


Figure 27: View of the 13 (12) different scan positions. Base floor= yellow, Stairs (between the floors) = red and Top floor = blue. The 13th scan is removed from this cloud but the position for this scan is seen (yellow mark).

The 13 scans (Figure 27) were registered and aligned in Scene software. No data has been removed, only merged into one point cloud. The merged cloud consists of about 366 million points (Figure 28). The merged cloud has then been exported to a vendor-neutral format (e57).

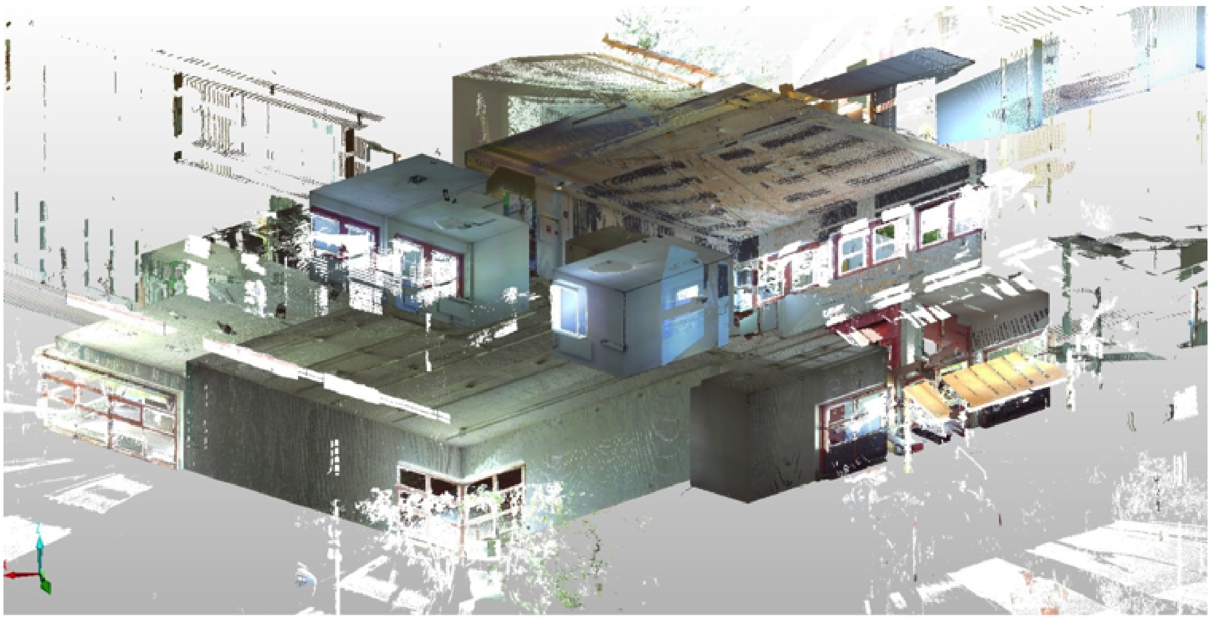


Figure 28: View of the complete point cloud which consists of about 366 million points.

6.5 Impact on Research and Development

Having the (rare) possibility to directly acquire realistic and relevant data on-site and within the DURAARK project period was – despite being an ongoing process – already very beneficial. Especially in the context of WP4, where tasks like the difference detection have very specific demands on the structure and context of measurements (buildings in different states on different "granularities"), usable and realistic datasets are very hard to come by. With the project-wide agenda to ease and increase the usage of scan measurements of different states of a building, having a large amount of suitable non-synthetic data is an important ingredient to reliable software development and we expect to profit from this and possible future acquisitions as we already do today.

7 Decisions & Risks

7.1 Technical Decisions

Choice of programming language The difference detection component is written in the C++ programming language, using features introduced in the recent C++11 and C++14 standards. The language offers a good compromise between low-level access to the hardware and reasonably easy usage. Since many libraries for e.g. geometry processing are written in C or C++, or offer interfaces for usage in C++ programs, functionality required for our tasks is directly available.

Choice of target platform/operating system Our software prototype was developed on a GNU/Linux operating system on a x86-64 platform. The GUI version has also been cross-compiled for the Windows platform using the “gcc” compiler of MinGW which demonstrates operating system independence.

Usage of (hardware) acceleration Some steps of our implementation of the reconstruction algorithm can be accelerated using parallel processing on multiple CPU cores, or computations on the graphics card (general purpose computation on graphics processing unit, GPGPU). This accelerates computations in case multiple CPU cores, and/or a suitable graphics card are available.

Usage of third-party libraries We use several third-party libraries for e.g. geometry processing of point clouds and meshes, ray casting, numerically robust data structures, and parsing of file formats since in-house development of all required functionality would not be feasible.

Choice of data formats In order to be able to communicate with the outside world through file input/output, commonly used file formats need to be agreed upon. For point cloud data input, we agreed early in the project to use the E57 file format which offers a rich feature set, good software support, integrated checksumming, and a freely available library (libE57) for parsing the format. For BIM model input, we use the IFC (Industry Foundation Classes) format which is widely supported by architectural software such as Autodesk Revit. The IFC specification is extensively documented, and a parsing library (IfcOpenShell) is freely available. As output format of our component for associating point clouds and BIM models, we use an RDF- (Resource Description Framework-)based format since other components (esp. in WP3) are also based on RDF. An open source parsing library, Raptor[4], is used.

7.2 Risk Assessment

Unavailability of hardware acceleration support (e.g. GPU)

- **Risk Description:** Certain hardware acceleration features are not available, e.g. a graphics card.
- **Risk Assessment:**
  - **Impact:** Low
  - **Probability:** High
  - **Description:** Especially in server environments, certain hardware such as a GPGPU-capable graphics card may not be available while our software prototypes offers acceleration support.
- **Contingency Solution:** Our software is written in a way such that acceleration hardware is optional. For instance, it will automatically adjust to the number of CPU cores, and will fall back to CPU-only computations in case a suitable graphics card is not available. Thus the software is usable, although speed may be decreased.

Abandonment of third-party library support

- **Risk Description:** One of the third-party libraries used is discontinued.
- **Risk Assessment:**
  - **Impact:** Medium
  - **Probability:** Medium
  - **Description:** Due to the number of third-party software libraries used, it is possible that development, support, or availability of one of the libraries is discontinued at some point. While current builds and copies of the libraries will continue to function, it may influence future development.

[4]`http://librdf.org/raptor/`

- **Contingency Solution:** Since most of the libraries used are free and open source, their source code will be available even if the original developers drop support. As could be observed in the past, open source projects are often forked or continued by other developers even if the original developers are no longer actively developing their software. In our case, NVIDIA OptiX is the only closed source library we currently use, and there exist alternatives (e.g. Intel Embree) which can be used almost as drop-in replacements.

Obsolescence of data formats used

- **Risk Description:** The file formats used (E57, IFC, RDF) are abandoned/replaced by newer alternatives.
- **Risk Assessment:**
  - **Impact:** Medium
  - **Probability:** Medium
  - **Description:** The file formats used are no longer supported, especially by third-party software products (e.g. export of E57 files from scanner software, import of IFC files in architectural software).
- **Contingency Solution:** The aforementioned file formats are widely used and well documented. This will facilitate migration of files already stored in the aforementioned formats.

8 Licenses

The IPR type "software" is implied for all IPs listed below.

IP used / generated	**Software Name**	**License**	**Information**
used	libE57	libE57 license (similar to Boost Software License)	`http://www.libe57.org/license.html`
used	Xerces	Apache License 2.0	`http://www.apache.org/licenses/`
used	ICU	ICU License	`http://source.icu-project.org/repos/icu/icu/trunk/license.html`
used	IfcOpenShell	LGPL v3	`http://ifcopenshell.org/`
used	Open CASCADE community edition	LGPL v2.1 with additional exception	`http://www.opencascade.org/getocc/license`
used	Point Cloud Library	3-clause BSD	`http://pointclouds.org/`
used	Eigen	Mozilla Public License 2.0 (except for few parts that are under LGPL)	`http://eigen.tuxfamily.org/`
used	Boost	Boost Software License	`http://www.boost.org/users/license.html`
used	Flann	2-clause BSD	`http://www.cs.ubc.ca/research/flann/`
used	OpenMesh	LGPL v3 (with exception clause that "you may use any file of this software library without restriction")	`http://www.openmesh.org/license/`
used	Qt5	Different licensing schemes available; we would suggest using LGPL 2.1	`http://doc.qt.io/qt-5/licensing.html`
used	OpenGL	Depends on implementation	`http://www.sgi.com/tech/opengl/?/license.html`

used	GLEW	Modified BSD License, Mesa 3-D License and Khronos License	`http://glew.sourceforge.net/credits.html`
used	zlib	zlib/libpng License	`http://opensource.org/licenses/zlib-license.php`
used	Graphene	CC0	`https://creativecommons.org/about/cc0`
used	NVIDIA OptiX	Inclusion in free applications allowed; commercial use requires Commercial License	`https://developer.nvidia.com/optix`
used	Raptor RDF library	Various licensing options; we propose to choose LGPL v2.1	`http://librdf.org/raptor/LICENSE.html`
generated	Difference detection shared library	2-clause BSD	`http://opensource.org/licenses/BSD-2-Clause`

9 Conclusion, Impact and Outlook

We presented the second software prototype for documenting the changing state of built architecture. In particular, Tasks 4.3 (transfer of structure and semantics), and 4.2 (identification of inconsistencies) have been tackled in the second project year. Our method for associating parts of the point cloud with corresponding BIM entities takes into account characteristic systematic differences between models and scans such as occlusions during the scanning phase by simulating the scanning process in the BIM model. Building on the determined associations, differences between concurrent representations of the same building are extracted and presented to the user in an interactive manner.

Our software prototype demonstrates the applicability of our approach and allows to export point cloud to BIM associations in form of RDF files which can be stored alongside the archived data. Our GUI tool can directly be called from the WorkbenchUI as part of the DURAARK system prototype which allows for the integration of the developed components into a coherent workflow. First evaluations performed by CITA as part of WP7 show that the implemented methods are beneficial for the target community; planned developments in the third project year (e.g. fully automatic registration) will further strengthen our geometric enrichment tools.

Additionally – in order to deal with the lack of extensive test data – a large amount of exemplary datasets have been acquired by exploiting the current rebuilding/renovation process at the Luleå University of Technology. Several point cloud scans were captured and BIM model datasets were gathered before and during the renovation; with post-renovation acquisitions planned for when the building is finished. The acquisition is planned to be finalized in the third project year after which testing of the geometric enrichment tools using the gathered data will be performed.

References

[1] Bosché, Frédéric. Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. *Advanced Engineering Informatics*, 24(1):107–118, 2010. Informatics for cognitive robots.

[2] C. Gordon and F. Boukamp and Daniel Huber and Edward Latimer and K. Park and B. Akinci. Combining reality capture technologies for construction defect detection: A case study. In *EIA9: E-Activities and Intelligent Support in Design and the Built Environment, 9th EuropIA International Conference*, pages 99–108, October 2003.

[3] Golparvar-Fard Mani and Peña-Mora, Feniosky and Savarese, Silvio. Application of D4AR – a 4-dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication. *ITcon*, 14:129–153, 2009. Special Issue Next Generation Construction IT: Technology Foresight, Future Studies, Roadmapping, and Scenario Planning.

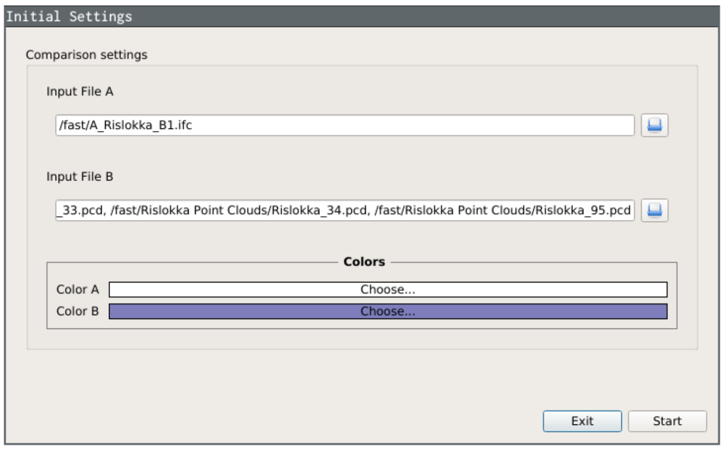
[4] C. Gordon and B. Akinci. Technology and process assessment of using ladar and embedded sensing for construction quality control. In *Construction Research Congress*, pages 5–7. ASCE Reston, Va., 2005.

[5] Shih, Naai-Jung and Wang, Pin-Hung. Using point cloud to inspect the construction quality of wall finish. In *Proceedings of the 22nd eCAADe Conference*, pages 573–578, 2004.

[6] Wang, Jun and Pradhananga, Nipesh and Teizer, Jochen. *Automatic Fall Risk Identification Using Point Cloud Data in Construction Excavation*, chapter 122, pages 981–988. American Society of Civil Engineers, 2014.

A Software Manual

This Section describes the basic usage of the GUI tool provided as the prototypical implementation of both the association as well as the difference detection tool. The software prototype is available at `ftp://ftp.cg.cs.uni-bonn.de/pub/outgoing/duraark/d4_2_prototype.zip` (anonymous login).
Note: For basic camera control/navigation we refer the reader to the manual Section in deliverable D4.1.
After starting the tool using `comparison.exe`, the application initially shows a startup settings dialog like shown in Figure 29, where the user must choose input files for a BIM, as well as a point cloud representation. For the latter, it is possible to choose multiple files in order to support file formats where each scan is represented by disjoint files. Additionally, the user may choose separate colors for the different representations which serves as a visual aid when performing manual alignment in the next step.



Input File A
/fast/A_Rislokka_B1.ifc

Input File B
_33.pcd, /fast/Rislokka Point Clouds/Rislokka_34.pcd, /fast/Rislokka Point Clouds/Rislokka_95.pcd

Color A
Color B

Figure 29: Initial settings dialog with example input.

When the settings are chosen according to preference, pressing the “Start” button will load the actual visualization interface with a central 3D view showing both representations, as well as a toolbar and a settings menu.
The first task to be performed by the user is to manually align both representations. In order to do so the software provides a few “edit modes” reachable via the upper toolbar (see Figure 30).
The “A” and “B” buttons enable the movement mode for the first and second representation respectively. The actual moving is performed by left-dragging with the mouse (while

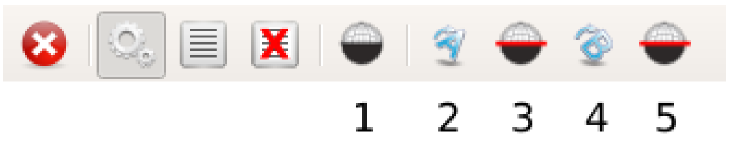
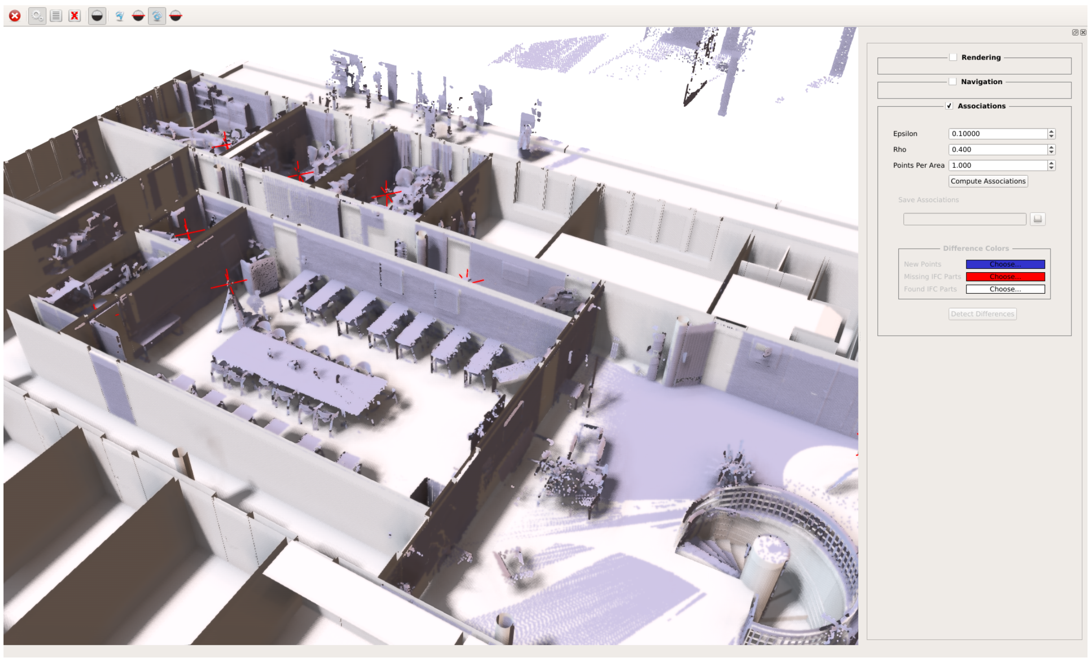


Figure 30: Edit modes reachable via the upper toolbar. 1. Toggle clipping for both representations. 2. Move first representation. 3. Edit clipping height for first representation. 4. Move second representation. 5. Edit clipping height for second representation.

simultaneously holding the Shift button allows for rotation around the view direction). The first button shown in Figure 30 toggles clipping mode which can be used to hide everything above an adjustable height in order to cut away the ceiling or entire floors of the building. The height at which clipping occurs is adjustable per representation via the 3rd (first representation) and 5th (second representation) button respectively. When the latter modes are active, the clipping height is again adjusted by left-dragging via the mouse vertically.

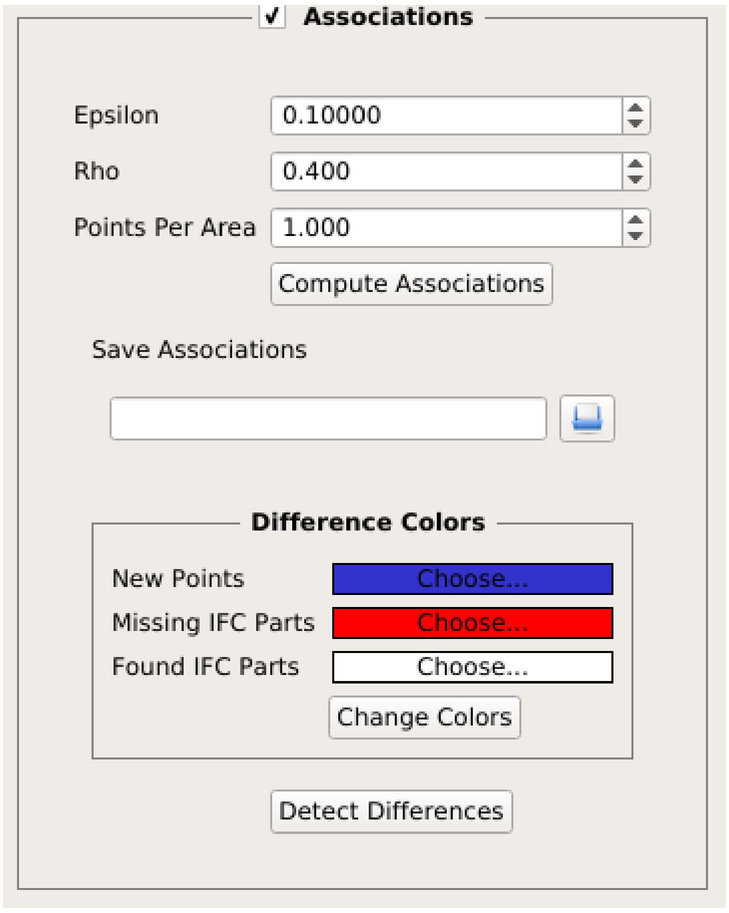
Using these tools the user can specify parts of the building to have in view and align these as good as possible. The automatic operations performed in the next step are designed to cope with larger deviation such that a perfect alignment (if even possible) is unnecessary.



Epsilon	0.10000
Rho	0.400
Points Per Area	1.000
New Points	Choose...
Missing IFC Parts	Choose...
Found IFC Parts	Choose...

Figure 31: Manually aligned representations.

Once the user has aligned the representations as exemplified in Figure 31, the automatic computation of associations can be triggered via the “Compute Associations” button in the right-hand side menu (see Figure 32). In order to provide for a way to cope with imperfect alignment, the associated tolerance threshold (“Epsilon” in meters) can be adjusted directly above the button. However the default value of 0.1m should provide a quite tolerable value for most scenarios.


✓ Associations

Epsilon 0.10000

Rho 0.400

Points Per Area 1.000

Figure 32: Right-hand menu for the algorithmic part of the tool.

Once the association has been computed, a file chooser directly beneath the aforementioned button becomes accessible, which allows the user to save the results to a file. This way the user can choose an RDF file to save the results to (the output is saved in N3/Turtle RDF format).
If the user is interested in a difference analysis between the two representations, the automatic detection of said differences is also possible after the computation of the as-

sociations. In order to perform a difference analysis, the user can start the computation via the "Detect Differences" button in the right-hand side menu. Once the results are computed, the 3D view is updated with colors indicating (un-)associated IFC and point cloud parts. The colors used for the visualization can be changed via the "Difference Colors" section in the right-hand side menu and updated after the computation by clicking the "Change Colors" button below.