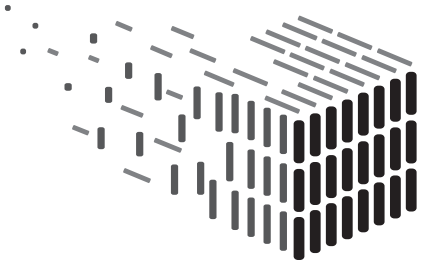

D4.3 Documenting the Changing State of Built Architecture

Software prototype v3

DURAARK

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

Date: 2016-01-31
Version 1.0
Document id. : duraark/2016/D.4.3/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/2016/D.4.3/v1.0
Title of document	:	Documenting the Changing State of Built Architecture – Software prototype v3
Deliverable type	:	Software prototype
Contractual date of delivery	:	2016-01-31
Actual date of delivery	:	2016-01-31
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)
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 report presents the third version of the software prototype for registration, identification of inconsistencies, and transfer of structure and semantics between different digital representations of a building. It provides an overview of the improvements and extensions since the last version of the prototypes, implementation details, and a description of the integration into the overall DURAARK system prototype.

Table of Contents

1 Introduction

Different kinds of digital representations are used to describe the "as-planned" and the "as-built" states of buildings. Building Information Modeling (BIM) is quickly replacing traditional means to plan, design, redesign and maintain buildings in the architectural domain while point cloud data is increasingly used to quickly capture the current state of existing or newly constructed buildings. In a Long-term Digital Preservation (LDP) setting, the availability of both modeled as well as measured data enables various use-cases such as the comparison of the planned and built state of a building, detection and verification of (un)intended changes over time, and documentation of any detected deviations.

The goal of Work Package 4 is to provide software tools for enabling or simplifying the comparison, verification and association of different concurrent BIM and point cloud datasets of the same physical building. As a prerequisite for further processing of the datasets, a software component for registration (i.e. spatial alignment) of datasets is provided since different representations of the same building do not usually reside in the same global coordinate system (Task 4.1). Building upon this alignment of different datasets, software components for associating corresponding parts of the datasets are provided (Task 4.2) which can be used for e.g. detection and visualization of any deviations or inconsistencies. Furthermore, if both a BIM model and a point cloud of the same building are available, the high-level semantic information inherent to the BIM model can be transferred to low-level point-cloud data (Task 4.3) in order to e.g. perform a meaningful annotation and segmentation of the point cloud on the level of rooms or walls.

In the third version of the WP4 software prototype, we focused on the integration of all developed software prototypes as Docker containers in order to provide seamless integration into the DURAARK Service Platform. The association component for finding corresponding parts of different datasets is now also able to detect certain movable objects in point cloud data which was not possible with the previous version. Using the high-level information of room layouts which is obtained using the reconstruction component of WP5, we exemplify how transferred semantic information from BIM to point clouds can be used to enable more comprehensible visualization.

In addition to the original planning stated in the DoW, the registration component has been further developed to provide *fully automated* spatial alignment of datasets which enables to run microservices dependent on aligned datasets to also run in an automated

manner. Also, the association component has been extended to support the determination of correspondences not only between BIM models and point clouds but also between two point cloud measurements. Since the difference detection component uses the determined correspondences of the association component, this enables the comparison of point cloud measurements taken at different points in time. This approach was tested and evaluated on real-world point cloud datasets monitoring the changing state of a building during renovation over the course of several months (see also deliverable D7.4 for evaluation results).

In the following we first give an overview of the integration of all components into the overall DURAARK Service Platform and their usage in the WorkbenchUI before describing the components in detail.

2 Integration and Software Manual

While the previous versions of the software prototypes were provided as stand-alone applications for usage on end-user systems, one of the main goals of the third and final prototypes was to provide a consistent, platform-independent deployment method for incorporating the tools as microservices into the DURAARK system. To this end, the Docker[1] containerization framework was chosen during the development of the DURAARK Service Platform in Work Package 2. Docker provides means to deploy software packages as self-contained images which allow the user to easily install and run applications on a variety of platforms. Consequently, the software prototypes developed in WP4 and WP5 are provided as Docker images which are available for installation through the publicly available Docker Hub[2]. Figure 1 shows an overview of the file formats, WP4/WP5 Docker images, and the data flow between them. A detailed description and usage guide for the images specific to this deliverable is given below.

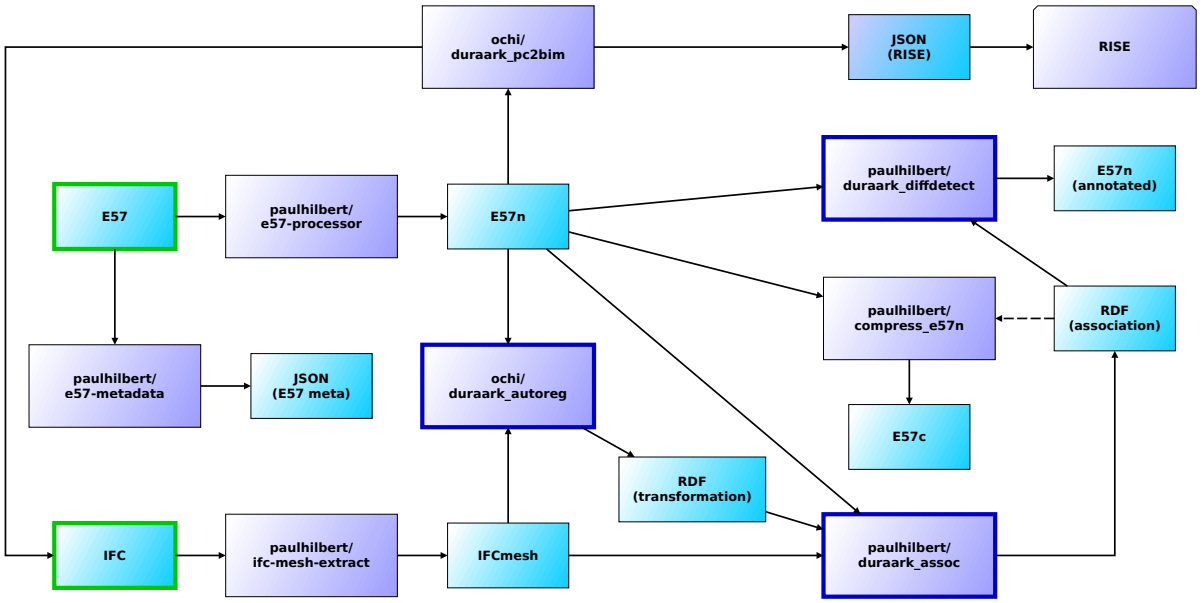

Figure 1: Overview of the Docker images provided by WP4 and WP5. The smaller, blue boxes represent files, purple boxes represent Docker images. The captions are file formats and names of the Docker images on Docker Hub, respectively. Processing starts with E57 and/or IFC files (boxes with green frames). The Docker images specific to WP4 are highlighted by blue frames.

[1] https://www.docker.com/
[2] https://hub.docker.com/

2.1 Component Installation

The software components are usually installed as part of the DURAARK Service Platform; a detailed guide can be found on GitHub at `https://github.com/DURAARK/duraark-installer/wiki`. However, it is also possible to install and use individual images manually. Assuming a working Docker installation, an image `example/image` (the real image names can be found in Figure 1) can be installed or updated to the latest version from the Docker Hub using the simple command in a Docker shell:

```
docker pull example/image
```

2.2 Images Shared Between WP4 And WP5

This Section gives an overview of Docker images which perform tasks relevant to both WP4 and WP5. These components mainly perform data preprocessing (subsampling, normal estimation, geometry generation from IFC files) resulting in intermediate representations which can subsequently be used by multiple components during a pre-ingest session.

2.2.1 E57 Preprocessor

Docker image: `paulhilbert/e57-processor`
Description: The E57 preprocessor component is used as an internal data preprocessing tool which performs a subsampling of point clouds and an estimation of normals for each point. The resulting point cloud is then used by several components such as the IFC reconstruction. Since this processing takes some time, it is meaningful to perform this preprocessing once and reuse the generated data within a pre-ingest session.
Input: E57 point cloud
Output: E57n modified point cloud
Example:

```
docker run --rm -v /home/user/work:/work paulhilbert/e57-processor
   --input /work/a.e57 --output /work/a.e57n -l 0.02
```

2.2.2 IFC Preprocessor

Docker image: `paulhilbert/ifc-mesh-extract`
Description: The IFC preprocessor component is used internally to generate geometric data (i.e. 3D meshes) from the IFC files. The rationale is that many of the developed components require this geometric information but generating it from an IFC file takes some time, depending on its complexity. It is thus meaningful to perform this conversion once and reuse the generated data within a pre-ingest session.
Input: IFC file
Output: Mesh data (OBJ) and metadata (JSON) internally referred to as "ifcmesh"
Example:

```
docker run --rm -v /home/user/work:/work paulhilbert/ifc-mesh-extract
   --input /work/b.ifc --output /work/objs --json /work/b.ifcmesh -s
```

2.2.3 E57 Metadata Extractor

Docker image: `paulhilbert/e57-metadata`
Description: The E57 metadata extractor is a utility which is used for extracting descriptive metadata (e.g. number of scans, number of points, GUID, timestamp) from E57 point cloud files.
Input: E57 point cloud file
Output: Metadata structured as XML or JSON
Example:

```
docker run --rm -v /home/user/work:/work paulhilbert/e57-metadata
   --input /work/a.e57 --output /work/a.json --format JSON
```

2.3 Images Specific to WP4

2.3.1 Auto-Registration

Docker image: `ochi/duraark_autoreg`
Description: The auto registration component performs a spatial alignment of two datasets (BIM model or point cloud) and writes out the resulting transformation encoded in an Resource Description Format (RDF) file.

Input: Two (preprocessed) datasets in either E57n or ifcmesh format
Output: RDF file containing the registration transformation
Example:

```
docker run --rm -v /home/user/work:/work ochi/duraark_autoreg --repra
   /work/a.e57n --reprb /work/b.ifcmesh --output /work/registration.rdf
```

2.3.2 Association

Docker image: `paulhilbert/duraark_assoc`
Description: The association component finds correspondences between (registered) BIM and point cloud datasets, or two point cloud datasets, and writes out found correspondences encoded in a RDF file.
Input: Two (preprocessed) datasets (either an ifcmesh and an E57n file, or two E57n files), and an optional registration RDF
Output: RDF file containing the found correspondences (association of BIM entities and point cloud subsets in case of BIM against point cloud, or the point cloud subset with at least one corresponding point in case of point cloud against point cloud)
Example:

```
docker run --rm -v /home/user/work:/work paulhilbert/duraark_assoc
   --rep-a /work/a.e57n --rep-b /work/b.ifcmesh --registration
   /work/registration.rdf --output-file /work/association.rdf
   --epsilon 0.1
```

2.3.3 Difference Detection

Docker image: `paulhilbert/duraark_diffdetect`
Description: The difference detection component takes as its input an E57n file and a RDF file from the association component and annotates measured points which do not have correspondences. The result is a colorized E57 file which can be visualized in various software tools.
Input: E57n file and association RDF
Output: Colorized E57 point cloud file
Example:

```
docker run --rm -v /home/user/work:/work
   paulhilbert/duraark_diffdetect --input /work/a.e57n --assoc
   /work/association.rdf --output /work/differences.e57n
```

2.4 Usage in DURAARK WorkbenchUI

During the ingest phase in the DURAARK Workbench UI, a difference analysis of two datasets (either BIM against point cloud, or point cloud against point cloud) can be performed as part of the "Geometric Tools" step.

The process starts with the selection of the first dataset on the left via the + sign. This opens up the available geometric enrichment tools for the selected file type on the right. From the list switch on *Difference Detection*. Next to the selected file a new tile appears, which allows you to select a second dataset to compare the first one with via a drop down box. All point cloud and BIM datasets which were uploaded to the building session are available here. After selecting a dataset from the box click the *Start* button which starts the registration, association, and difference detection components in the background. See Figure 2 for a screenshot of the described step.
When the processing is finished the tile's header is switching from *Processing ...* to *Show Result*. Click on *Show Result* to see the annotated result as depicted in Figure 3. The differences are color-coded. In this case the furniture was present in the point cloud scan, but not in the according BIM model.

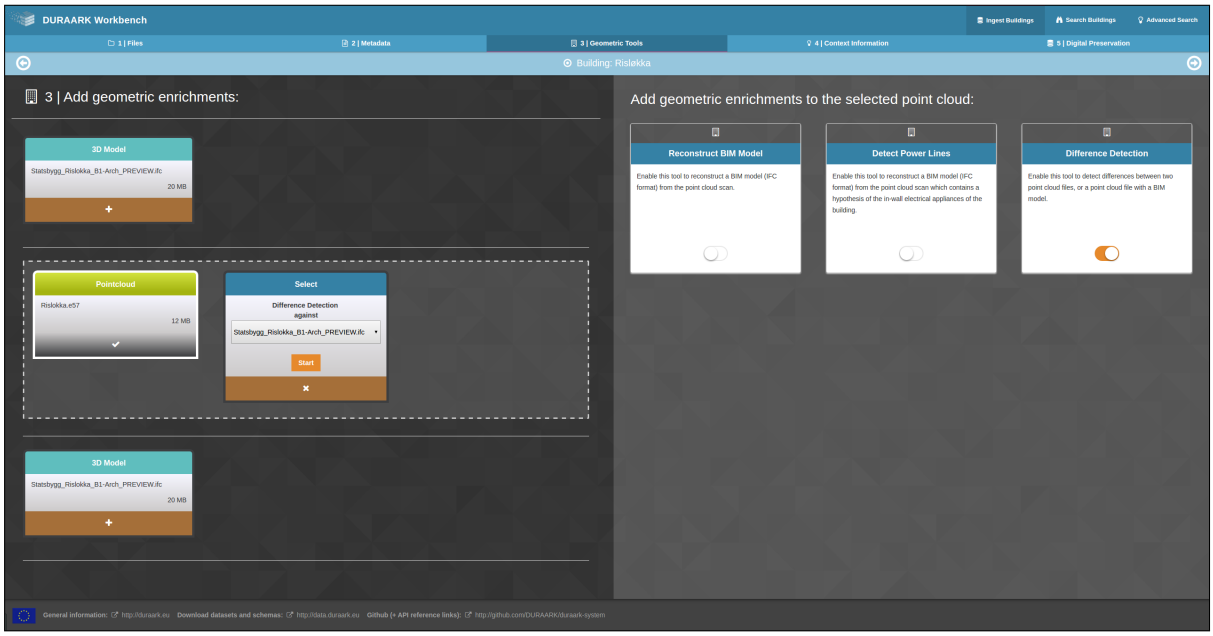

Figure 2: Difference detection is initiated by selecting a dataset on the left and then switching on the Difference Detection component on the right.

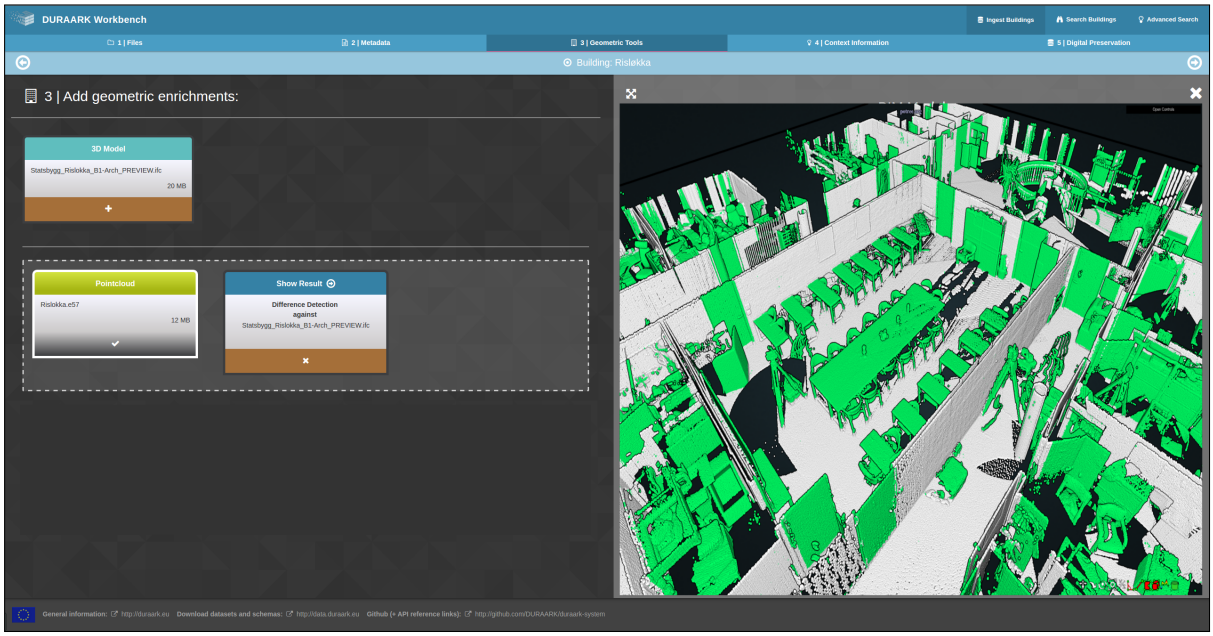

Figure 3: After processing is finished, the difference detection result can be inspected as a color-coded point cloud.

3 Registration

3.1 Recap of Developments in Year 1 and Year 2

The previous version of the software prototype for spatial alignment of different representations of the same building allowed the user to load e.g. a point cloud scan and a BIM model for registration. Subsequently, a coarse, manual pre-alignment had to be done before an automatic fine alignment based on an Iterative Closest Point (ICP) algorithm [1] could be performed (Figure 4). The resulting transformation between the datasets which realized the final alignment could then be exported to a RDF-based format for use in other components which require registered datasets.
Evaluation and feedback from stakeholders showed that a fully automated processing chain is highly desired. Despite the original planning, work on a fully automated registration component has therefore been continued in project year 3 as previously announced in deliverable D4.2. This allows to run subsequent steps (i.e. association, difference detection) in a fully automated processing chain without requiring manual user intervention to align the datasets.

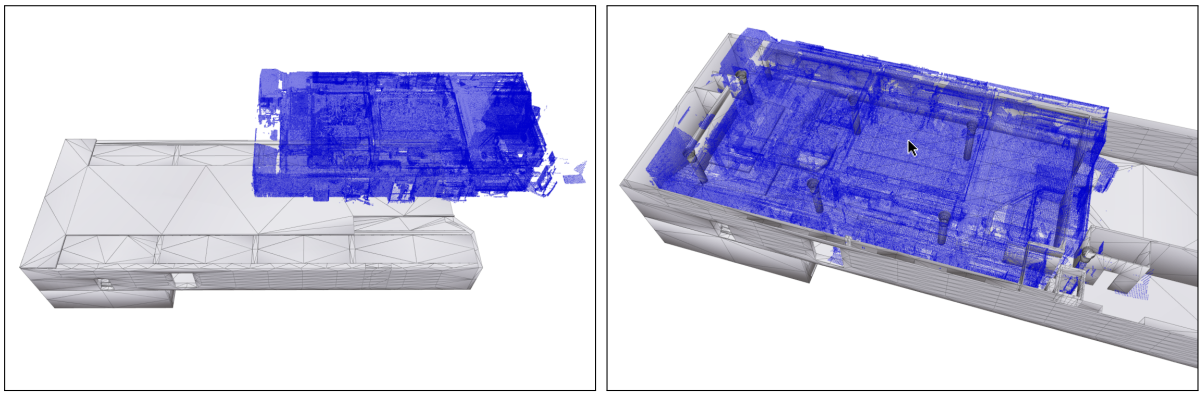

Figure 4: Manual registration process in the previous version of the software prototype. Left: BIM and point cloud datasets after loading. Right: A coarse pre-alignment had to be done manually.

3.2 Problem Description and Challenges

The goal of the third version of the prototype was to replace the aforementioned manual pre-alignment with a fully automatic method that does not require user intervention. As a consequence, other components which require registered datasets as their input are

then also able to run in an automated manner. In the context of the overall DURAARK system, this enables fully automatic processing of datasets by a chain of microservices without requiring additional user input.
A problem of the widely used ICP algorithm is that it requires at least a coarse pre-alignment of the datasets which is then iteratively refined. This pre-alignment is necessary since corresponding points between the datasets need to be determined in order to find a rigid transformation that minimizes the total distance between corresponding point pairs. Without a faithful pre-alignment, the determined minimum will generally be some undesired local minimum that does not result in a meaningful alignment of the datasets. The main challenge is thus to provide automated means to perform the pre-alignment which can then be further improved using e.g. ICP. While this problem is hard to solve in the general case (i.e. given arbitrary, possibly partially captured geometry), we can use heuristics that are specific to the architectural domain in order to make this problem feasible.

3.3 Implementation and Workflow

Our implementation of a fully automatic, coarse pre-alignment of different (possibly partial) representations of the same building is based on successive alignments of images in the two-dimensional domain (Figure 5). This idea is roughly based on work by Makadia et al. [2] who use correlations between Extended Gaussian Images in the Fourier domain to achieve a coarse alignment which is subsequently refined using ICP. While the approach by Makadia was tested (amongst other datasets) on point clouds of single rooms, we argue that performing an alignment of whole buildings as is the case in the scope of the DURAARK project is prohibitively expensive due to the used voxelization. Since we can make certain (weak) assumptions on our datasets, we restrict ourselves to a successive alignment in the two-dimensional domain which greatly reduces computational costs.
The only assumptions on the datasets are that the "up" direction is known beforehand and that *most* wall and floor structures are vertical or horizontal, respectively. In the following, the "up" direction is assumed to be the z axis, and the horizontal plane is thus parallel to the xy-plane. In a first step, the datasets are aligned in the xy-plane without considering possible deviations along the z axis. To this end, vertical planes in both representations are extracted and their dominant normal directions are determined by clustering the directions of projections of the extracted planes into the xy-plane. The

largest cluster of directions is then considered to be the most dominant direction, and the second dominant direction is constructed orthogonally to this direction. In case of BIM models, the planes are extracted directly from the surfaces of wall entities; in case of point clouds, a Random Sample Consensus (RANSAC) plane detection is performed.
The dominant directions in both datasets provide hints for possible rotational alignments of the datasets around the z axis, i.e. we assume that aligning the dominant directions to each other yields a valid rotational alignment of the datasets. In practice, we also consider additional rotation angle offsets of 90°, 180°, or 270° since the most dominant direction may deviate between the representations, especially in only partial scans. This still sufficiently restricts the search space in the rotational domain since only four different rotations need to be considered.
For each possible rotation around the z axis, the projections of vertical planes are rendered as lines in two grayscale images. The task is then to find an alignment of these two images which maximizes the overlap between the rendered lines (i.e. wall surfaces). The advantage of using images is that we can use a Fourier transform based approach (using the OpenCV image processing library) for efficiently solving this maximization problem.
For selecting the correct rotation angle, we consider the response (i.e. overlap) of the image alignment method; the alignment with the largest response is selected. The computed image alignment transformation is then transferred back into world coordinates and applied to the building representations. This results in an alignment of the datasets in the xy-plane.
For determining the translational alignment along the z axis, a similar approach is used: Horizontal planes that potentially correspond to floor surfaces (i.e. planes with upwards facing normal) and vertical planes are rendered into an image, viewing the building representations from the side. Aligning the images to each other then results in a translational offset which is applied to the building representations.
Figures 6 and 7 show example results of our automatic registration approach.

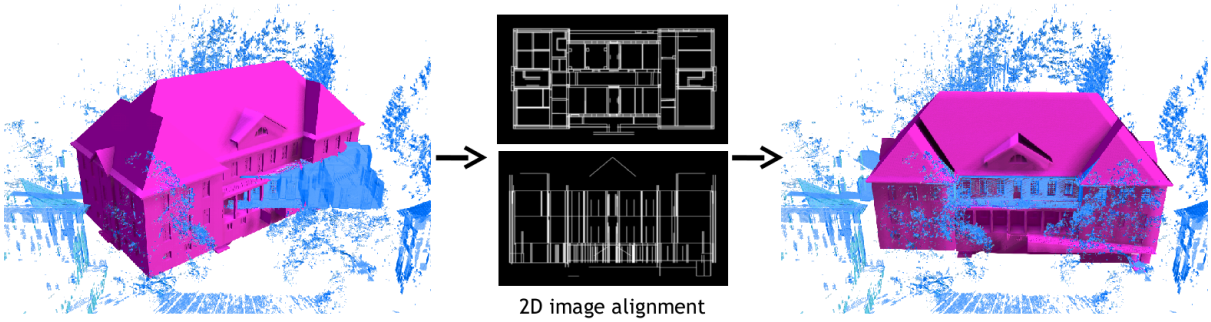

Figure 5: Schematic of the automatic registration process. The BIM model is shown in magenta, the point cloud of one story is shown in blue. Note that the point cloud contains many outlier points outside of the building. Left: Unaligned BIM and point cloud datasets. Middle: 2D images of projected planes are used for performing an alignment in the xy-plane and along the z-axis. Right: Datasets after alignment.

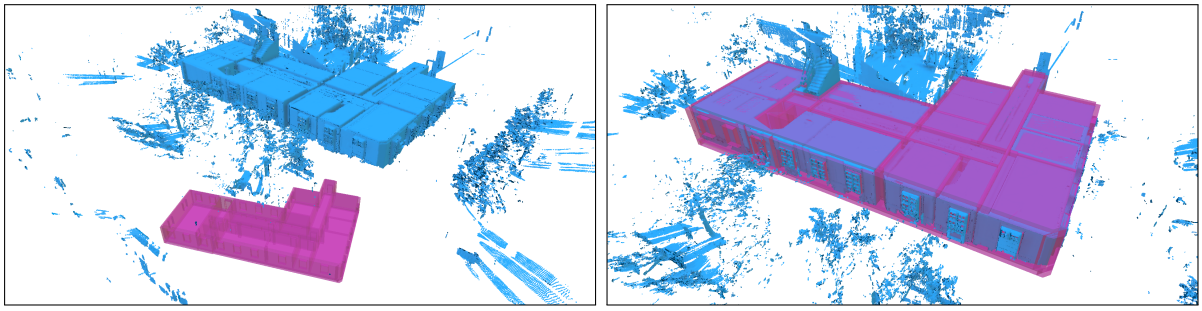

Figure 6: Auto-registration example on a smaller dataset (Byg72). The BIM model is shown in semi-transparent, magenta color. Left: Before alignment. Right: After alignment.

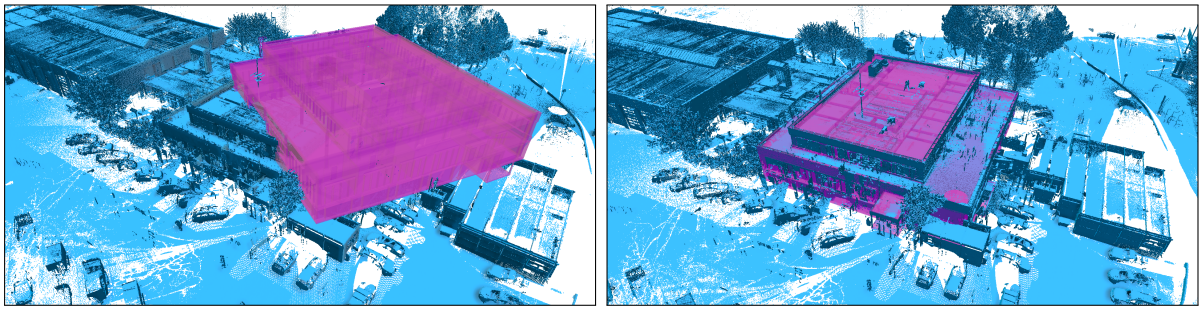

Figure 7: Auto-registration example on a large dataset (Risløkka Trafikkstasjon). Note that our registration finds a valid alignment although the BIM model only covers part of the building and the point cloud also contains large parts of the outside area.

4 Transfer of Structure and Semantics

4.1 Recap of Developments in Year 1 and Year 2

The second software prototype for the transfer of structure and semantics provided means to determine correspondences between different representations of the same building. Corresponding parts (i.e. BIM entities such as walls and subsets of a point cloud) are associated with each other such that a mapping between them is obtained (Figure 8). This information can then be used for e.g. finding parts that do not have correspondences in the respective other representation.

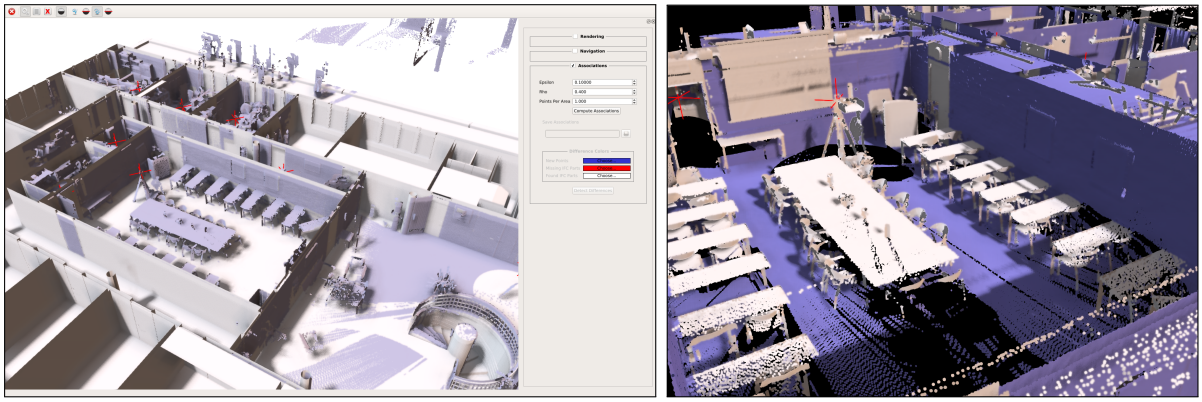

Figure 8: Example result of the previous version of the association component. Left: Aligned BIM model and point cloud. Right: Colorized point cloud; blue points have been associated with BIM entities, white points are unassociated.

4.2 Problem Description and Challenges

Structuring for visualization and navigation The visualization of large-scale point cloud datasets is a challenging problem for various reasons: First, the sheer amount of data usually forbids to load the whole dataset at once due to memory and/or bandwidth constraints. Second, the complexity of – particularly indoor – datasets often makes it difficult for a user to navigate point clouds in a simple manner. Third, effects inherent to the used scanning techniques such as scan shadows (and thus holes in the point cloud datasets) sometimes makes it hard for a user to comprehend the visualized data, e.g. to differentiate between real holes due to openings in walls or missing data due to occlusions. To mitigate the aforementioned problems, we demonstrate that a suitable transfer of semantics between a high-level (possibly automatically generated) BIM model and the

low-level point cloud data can help to structure the point cloud data in a way that improves visualization and navigation. This structuring is also used to perform a meaningful prior indexing for compression of point cloud datasets as described in D5.5. Figures 9 and 10 show examples for how an association between the low level point cloud data and a high level model (in this case automatically generated by our reconstruction approach, see D5.3 and D5.5) can help the user to work with the data by automatically hiding outliers and selectively hiding rooms.

Association of doors In the previous software prototype, doors in the BIM model have explicitly been excluded since the state of the corresponding physical door observed in the point cloud scan is generally unknown (i.e. whether it is closed or how far it is opened). Therefore correspondences between the observed geometry and the planned geometry cannot be determined in the same way as for static geometry.
To overcome this limitation, means to associate potentially dynamic (i.e. movable) objects need to be incorporated. In general, this is a challenging problem when considering objects such as furniture elements in a point cloud that can be moved almost arbitrarily. Determining true correspondences between different observed states of a scene may become a prohibitively hard computational problem.
In case of doors however, meaningful constraints can be incorporated into the detection process in a sense that most doors present in our datasets are hinged at a specific location and thus only have one degree of freedom, namely a rotation angle. By exploiting this additional, domain-specific constraint, the problem of determining correspondences between the state of a door in a BIM model and a point cloud becomes tractable since the search space is restricted to a small region with low dimensionality of possible movement. An example for associated door geometry is shown in Figure 11.

Point cloud to point cloud association The previous software prototype performed associations between BIM models and point clouds. While this provided an insight into correspondences and differences between “as-planned” models and “as-built” measurements, it did not directly provide information about changes between point cloud measurements taken at e.g. different points in time. Since the determination or monitoring of changes to buildings over time (e.g. during a renovation phase) was identified an important use case in the architecture as well as LDP domains (see D2.1), the third prototype has been extended to perform assocications between two different point cloud datasets in

addition to associations between BIM and point clouds.

4.3 Implementation and Workflow

Structuring for visualization and navigation We exemplify the applicability of transferred semantics from BIM models to point clouds for improved visualization and navigation using models generated automatically by our reconstruction prototype (D5.3, D5.5). It is important to note that the models generated by our reconstruction approach are not purely geometric mesh models, but they provide deeper insight into a building story's room layout: An intermediate data structure of the reconstruction process is the representation of a building story as a halfedge graph in which faces are interiors of different rooms and edges are walls between adjacent rooms. The neighborhood relation of rooms is also directly contained in this structure, i.e. two rooms are neighboring if and only if they share a common edge. This representation allows for fast queries to which high-level building element (room, wall, floor, ceiling) a given 3D point belongs. This allows us to associate all measured points of a point cloud to the aforementioned building elements. This association yields a segmentation of the point cloud on a high semantic level of building elements.
In a visualization setting, this allows a user to e.g. selectively highlight and inspect point subsets corresponding to a single room – or a room and its neighbors – while hiding parts of the data which is irrelevant. In addition to conserving memory and allowing for higher resolution visualization of relevant parts of the data, it also improves understandability of the rendered data. The underlying, light-weight graph representation used for segmenting the point cloud and/or the reconstructed BIM model can also be used to improve navigation of the point cloud data: For instance, the user can quickly and easily navigate the building using the more abstract model before loading the detailed point cloud data when desired. Finally, the obtained segmentation can also be used to index the point cloud data in a meaningful manner for compression and storage as described in D5.5.

Association of doors For determining correspondences (or missing correspondences) between doors in different building representations, means to detect the geometry (location, size) of doors are required. In case of BIM models, door entities are annotated with the required information such that it can directly be accessed after loading an IFC file. In case of point clouds however, this information is not immediately available but

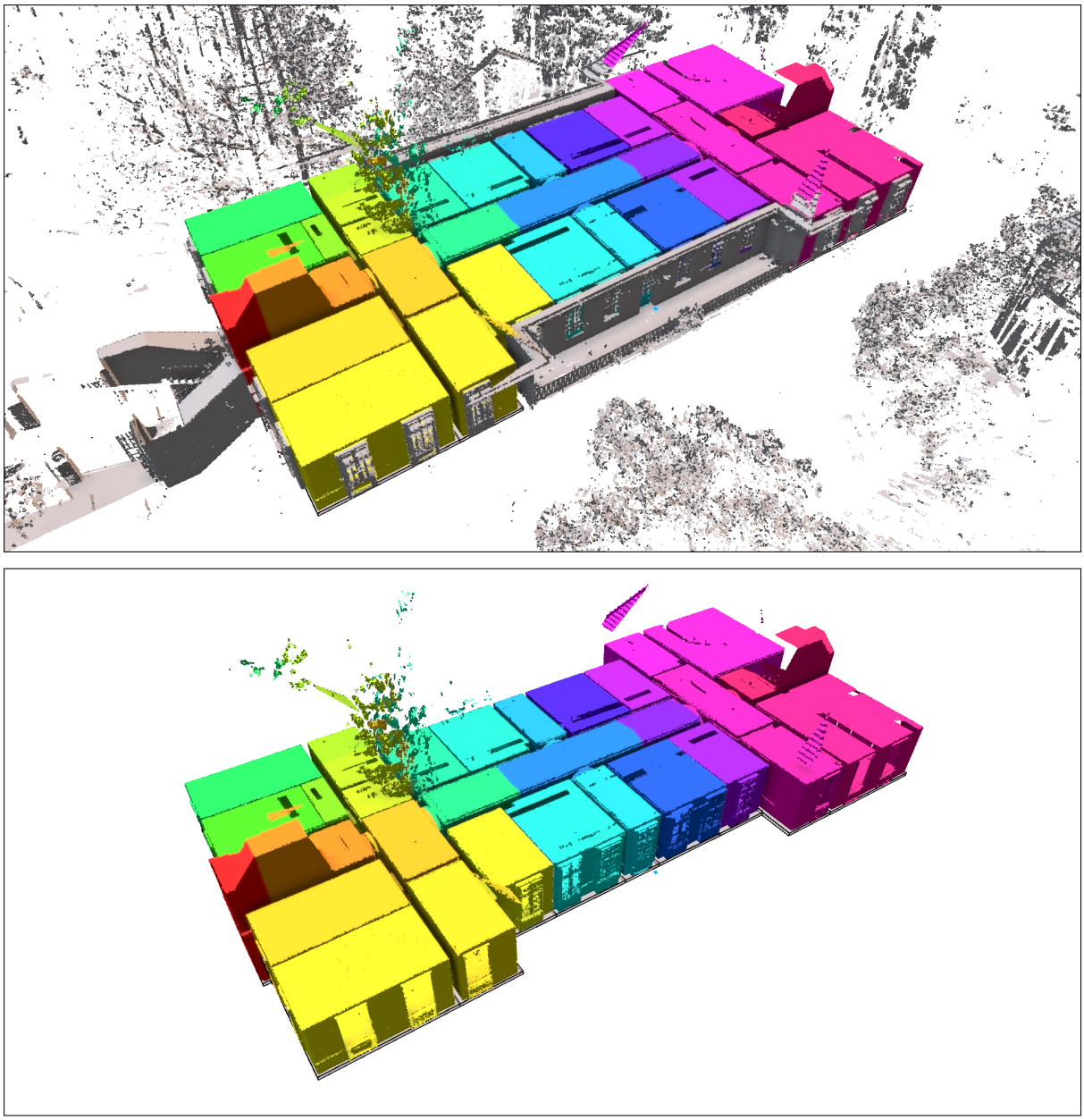

Figure 9: Transferring the room layout from a BIM model (in this case automatically generated by our reconstruction component, see D5.3 and D5.5) allows the user to work with the initially unstructured point cloud data on a high level. Top: The point cloud is colorized with respect to the separate detected rooms; grey points are outliers. Bottom: Hiding the automatically detected outliers already provides a much cleaner view on the point cloud data.

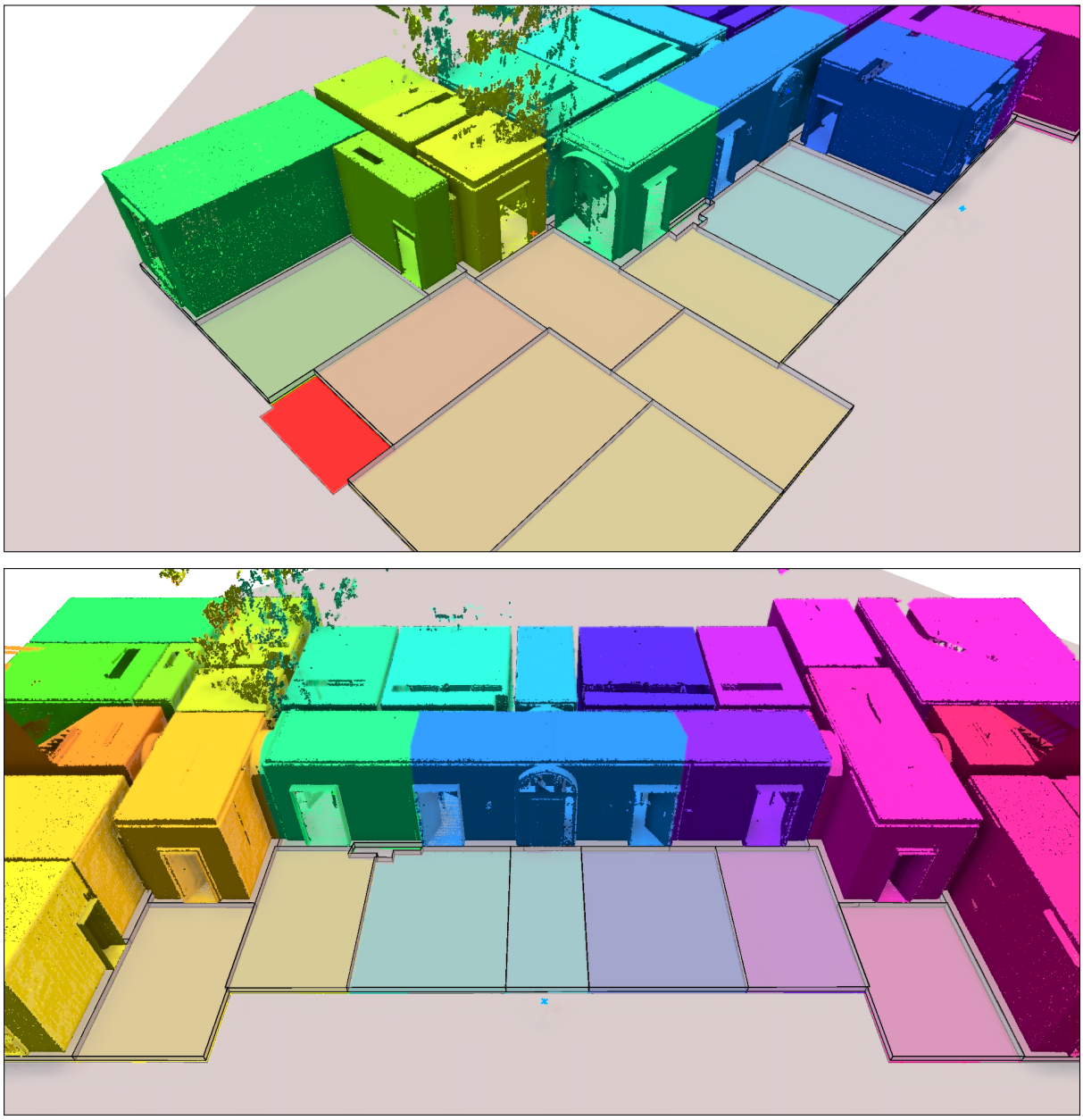

Figure 10: Combining high level semantics such as the room layout and measurements of the real building can help the user to better understand and navigate the data. These examples show a hybrid visualization of the automatically detected room layout and the corresponding point cloud. The user can selectively hide rooms (hidden rooms are indicated by colored slabs) to get a better view of relevant parts of the measured data.

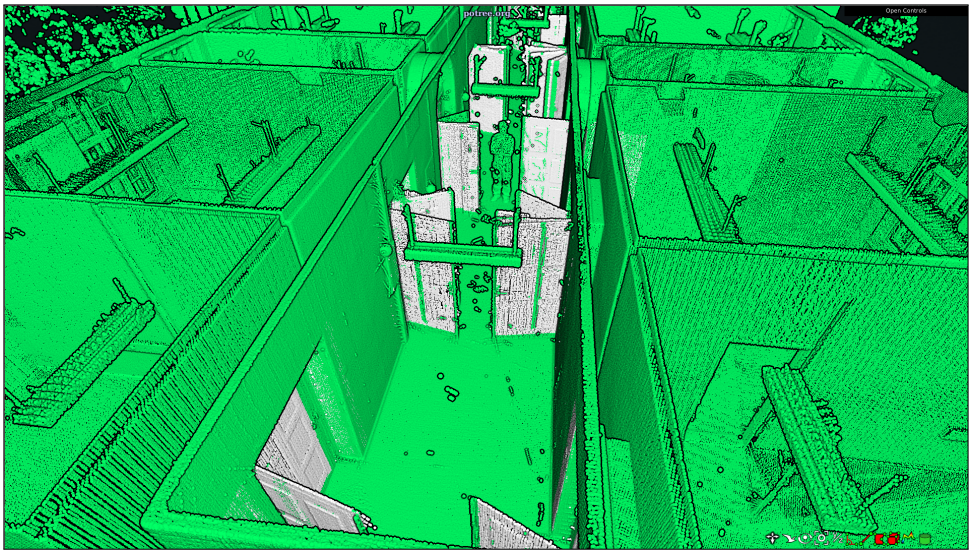

Figure 11: Door geometry measured in the point cloud scan can now be associated with door entities in an IFC model. White points highlight the point cloud subset that has been associated with door entities.

needs to be estimated from the measured building geometry. For this, we use a similar approach as for the newly implemented determination of opening directions in the BIM reconstruction prototype (see D5.5): Given a point cloud scan and an (aligned) BIM model of the same building, consider the locations of door entities in the BIM model. For each door location, a subset of the point cloud scan in the vicinity of the door is considered to search for corresponding measured geometry of the door. To this end, a plane detection is performed in the respecitve point cloud subset extracted around the door entity. Approximately vertical planes with sufficient support by measured points are considered as candidates for the corresponding door geometry. As an additional constraint, the detected plane needs to be close to the location where the door hinges are located (this information usually is part of the meta-information contained in the BIM model). If a suitable candidate for measured door geometry is found, the point cloud subset supporting the detected plane is associated with the corresponding door entity.

Point cloud to point cloud association For performing an association between two different point cloud datasets A and B of the same building, we use a simple yet effective geometric approach: For each measured point p in point cloud A, we test whether at least one corresponding measured point lies in the vicinity of the position of p in point cloud B. If there exists a point in vicinity, p is said to have a correspondence. To accelerate

this procedure, a search tree structure (e.g. octree) is built over the point cloud B. It should be noted that since the nearest neighbor point query operation is not symmetrical, the result of associating A against B and the result of associating B against A is also generally not symmetric.

5 Identification of Inconsistencies

5.1 Recap of Developments in Year 1 and Year 2

The previous software prototype for the identification of inconsistencies between the as-built and as-planned states of a building provided means to detect and highlight parts of the building which are either missing from one of the representations, or only present in one of them. Figure 12 shows an example result generated by the previous software prototype.

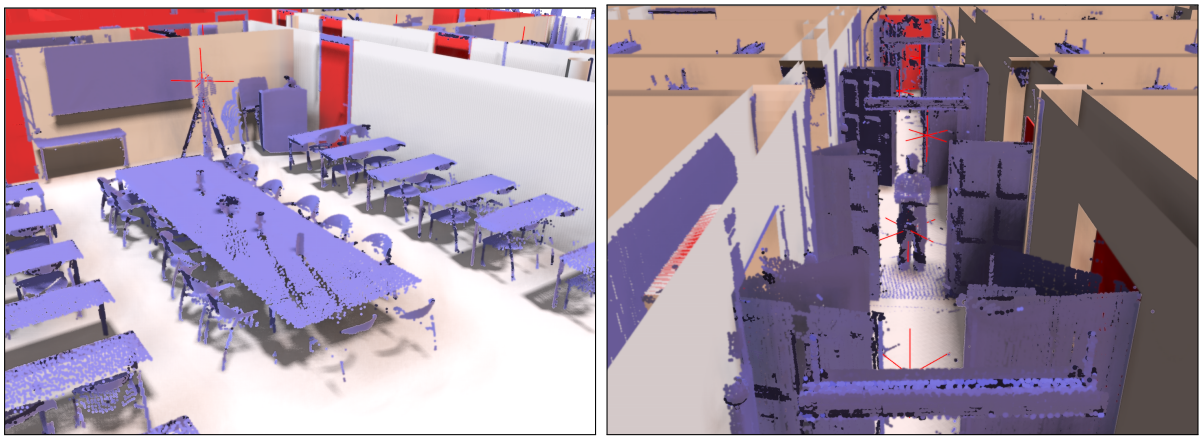

Figure 12: Example result of the previous version of the identification of inconsistencies. BIM entities with sufficient corresponding measured points are shown in white, entities without sufficient correspondence in red, and unassociated points are shown in blue.

5.2 Problem Description and Challenges

Annotation of inconsistencies in E57 files To provide an universal means to store detected inconsistencies in point cloud data, we provide a software tool which annotates point subsets in standard E57 files given an input point cloud and a corresponding RDF file computed by the association component. The output is a new, annotated E57 file which can be used in many standard tools for inspection of the difference detection results. Figures 13 and 14 demonstrate this by visualizing detected differences in the WebGL-based point cloud viewer "Potree"[3].

[3] `http://potree.org/`

Detection of inconsistent door openings As described in Section 4, door entities have been excluded from the difference detection in the previous software prototype since the moving geometry of doors was not not taken into account. The newly incorporated assocication of door geometries with the corresponding BIM entities allows the inconsistency identification component to test for missing correspondences of door elements.

5.3 Implementation and Workflow

Annotation of inconsistencies in E57 files Given an input E57 point cloud and a corresponding RDF file containing the indices of points which were detected to have correspondences by the transfer of semantics component, points which have or do not have correspondences are annotated in the E57 file. This annotation is performed by using the color attribute of points which is part of the E57 file format. The resulting, annotated point cloud is written as a new E57 file which can be used by many standard software tools which support the E57 format.

Detection of inconsistent door openings In order to implement the detection of inconsistent door openings, no particular change had to be made to the inconsistency detection component. This is due to the fact that associations between measured door geometry and door entities in the BIM model are encoded in the association RDF file in the same way as for associations with e.g. wall entities. Thus the added associations are directly taken into account by the inconsistency detection.

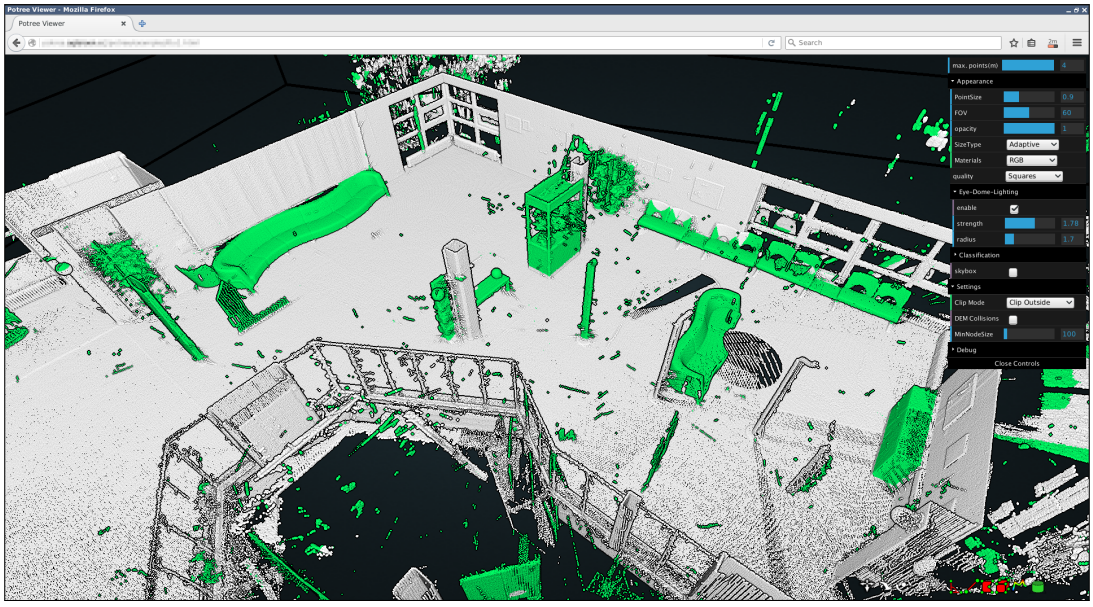

Figure 13: Exporting annotated (colorized) point clouds allows us to visualize differences in a variety of software, for example the WebGL-based "Potree" point cloud viewer. The dataset shown here is the comparison of the first and second scanning session during renovation of the LTU building. As is clearly visible, furniture and plants have been removed between the scans.

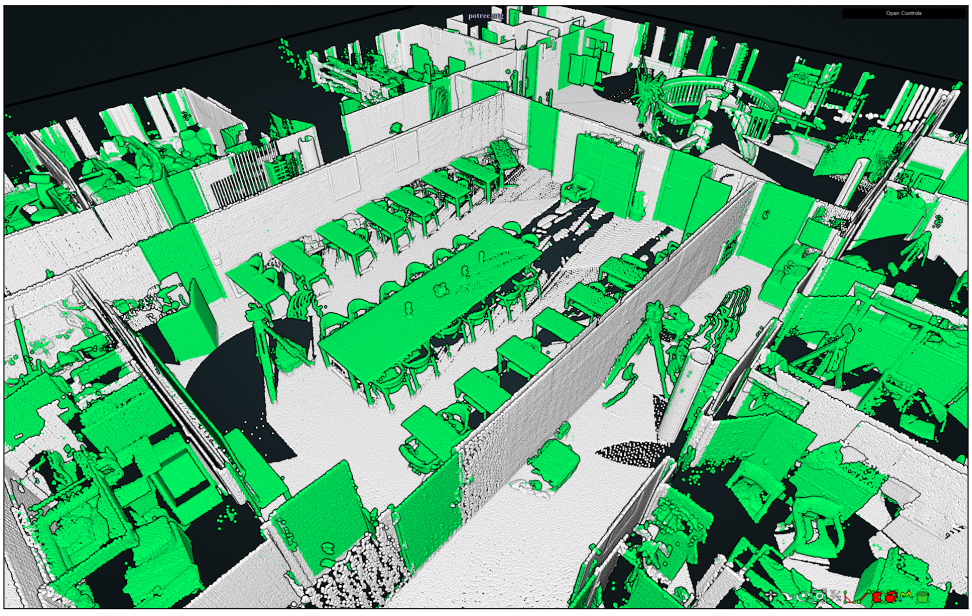

Figure 14: An example for a comparison between a point cloud of the Risløkka Trafikkstasjon dataset with a corresponding BIM model. Objects such as chairs, tables, cabinets have been identified as differences since they are not present in the model.

6 Example Scenario: LTU A-square renovation

In D4.2 the dataset from the redesign at Luleå University of was introduced, the final scan has now been performed and can now be used as a demonstrator. The complete data set consist of IFC files of the building before the renovation as well as several IFC files for the renovation. The IFC files for the current status (before redesign) is reconstructed and based on 2D drawings, measurements and old documents such that the quality and therefore the accuracy of them is not that high. The IFC files for the renovation is more detailed. Figure 15 below presents the IFC models before and after the redesign.

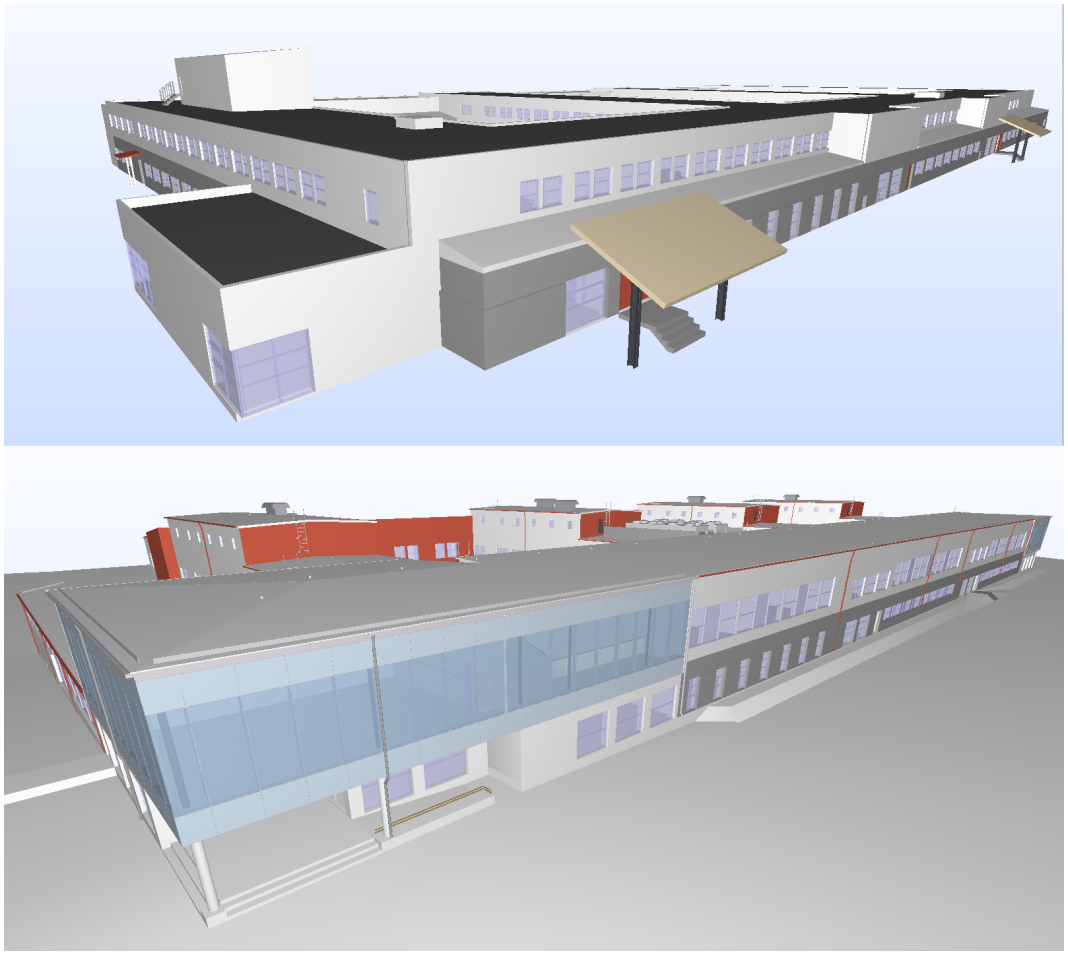

Figure 15: LTU A-house before (top) and after (bottom) the rebuilding.

6.1 Point Cloud Datasets

The A-square consists of several separate areas on two different floors. Scans have been performed on three different occasions, on June 2014 (with some furniture), August 2014 (without furniture) and a final scan with furniture when the redesign has been completed (October 2015). Each scan contains about 360M points. Scans performed before and after the renovation phase can bee seen in Figure 16.

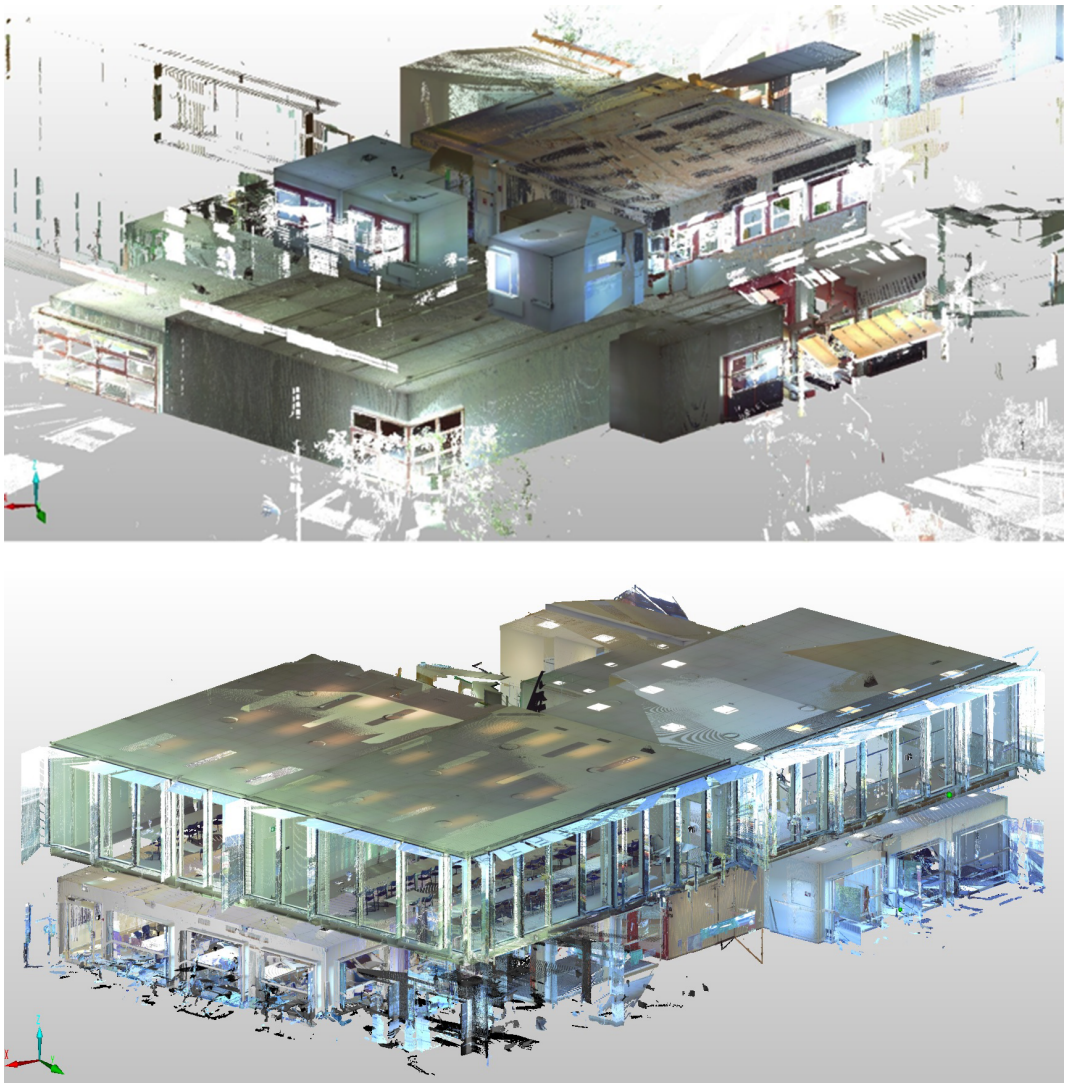

Figure 16: LTU A-house before (top) and after (bottom) the rebuilding.

6.2 Difference detection result

Figure 17 shows an example result of the difference detection performed between the point cloud datasets from the second and third scan sessions, i.e. before and after the renovation took place. The newly constructed parts of the building are clearly visible by the points highlighted in green color, but also details like newly placed lamps at the ceiling are detected.

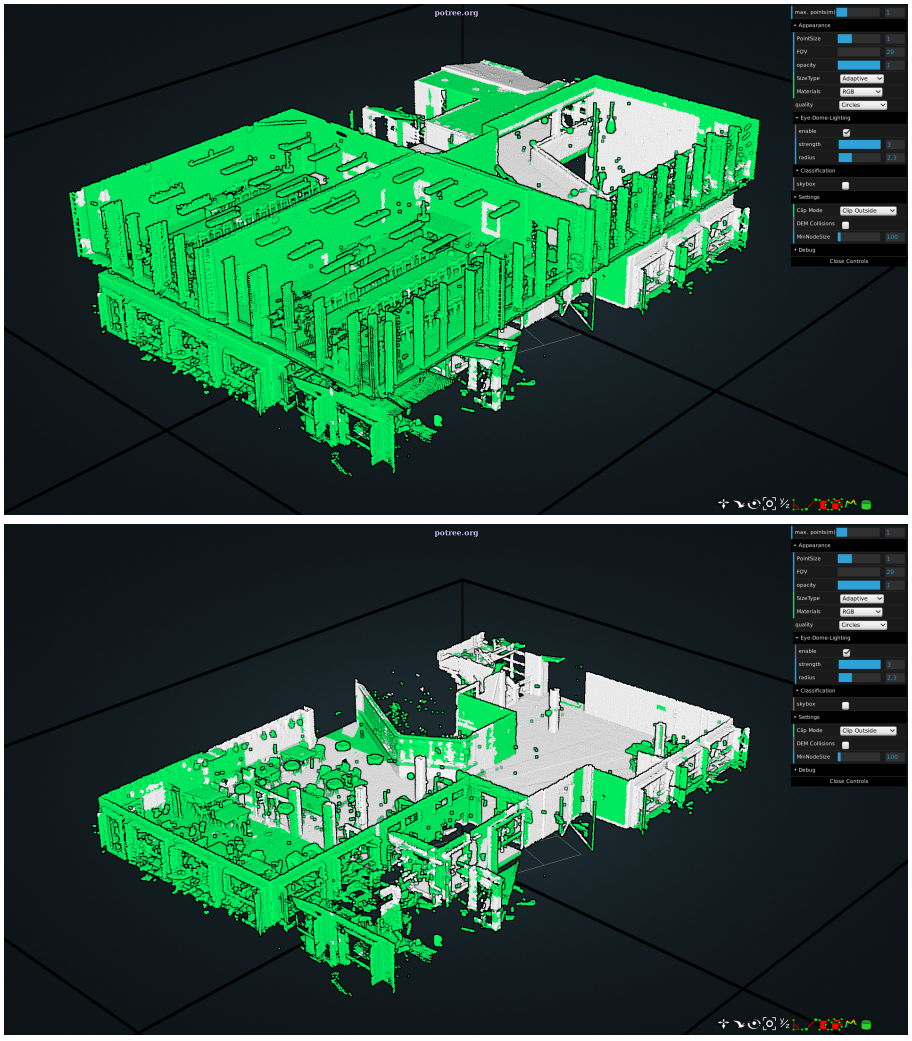

Figure 17: Difference visualization showing the point cloud of the third scan session (after renovation) with differences to the second scan session highlighted. Top: Upper floor, bottom: lower floor. White points show areas which have been detemined to be similar in both scan sessions, green points are detected as being different. Especially in the upper floor, the newly constructed parts of the building (see also Figure 16) are clearly visible.

7 Decisions & Risks

The technical decisions and risks discussed in this Section are an amendment to the risk assessment in the previous deliverable D4.2.

7.1 Technical Decisions

Component deployment as Docker images During development of the DURAARK Service Platform, it was decided to use the Docker containerization platform to allow for platform-independent deployment of the DURAARK microservices (see D2.5). Consequently, the software components in WP4 and WP5 are deployed as Docker images which are publicly available through Docker Hub.

Storage of Docker images in Docker Hub For easy installation of all software components, the Docker images are stored in the publicly available Docker Hub. In addition to storing the built images, Docker Hub also provides means to have images built automatically from e.g. GitHub repos containing the necessary Docker image description as a "Dockerfile".

7.2 Risk Assessment

Obsolescence of the Docker platform

- **Risk Description:** The Docker platform becomes obsolete and is superseded by alternative containerization solutions.
- **Risk Assessment:**
  - **Impact:** Medium
  - **Probability:** Low
  - **Description:** The Docker platform is relatively new and there exist competing containerization frameworks such as "rkt" or "LXD". It thus cannot be ruled out that Docker will be superseded by other containerization/virtualization solutions.

- **Contingency Solution:** Each of the Docker images fundamentally is a self-contained Linux-based operating system including the respective software component. As such, porting the components to similar frameworks should be relatively easy to do. Alternatively, the software could be run in full virtualization environments such as VirtualBox.

Abandonment or unavailability of Docker Hub

- **Risk Description:** The Docker Hub (but not Docker) is being abandoned or otherwise unavailable.
- **Risk Assessment:**
  - **Impact:** Low
  - **Probability:** Low
  - **Description:** Since the Docker Hub repositories are separate from the core development of the Docker framework, it is possible that the repository infrastructure is abandoned while the Docker platform remains available. Also, Docker Hub could be temporarily unavailable for various reasons.
- **Contingency Solution:** Usage of the Docker Hub for storing and building images is mostly a convenience. Docker images can also be built manually provided the respective "Dockerfile" which contains the information how a Docker image is built. In our case, all necessary Dockerfiles are publicly available on GitHub and thus the necessary images can be built with relatively little effort without using Docker Hub.

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	zlib	zlib/libpng License	http://opensource.org/licenses/zlib-license.php
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
used	OpenCV	3-clause BSD	http://opencv.org/license.html
used	Primitive Shapes	Custom, see below.	—
generated	E57 processor	2-clause BSD	http://opensource.org/licenses/BSD-2-Clause
generated	IFC mesh extract	2-clause BSD	http://opensource.org/licenses/BSD-2-Clause
generated	E57 metadata	2-clause BSD	http://opensource.org/licenses/BSD-2-Clause
generated	Registration shared library & command line tool	2-clause BSD	http://opensource.org/licenses/BSD-2-Clause
generated	Association shared library & command line tool	2-clause BSD	http://opensource.org/licenses/BSD-2-Clause
generated	Difference detection shared library & command line tool	2-clause BSD	http://opensource.org/licenses/BSD-2-Clause

Primitive Shapes license:

Copyright 2009 Ruwen Schnabel (schnabel@cs.uni-bonn.de), Roland Wahl (wahl@cs.uni-bonn.de).
This software may be used for research purposes only.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIB-

UTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License Implications

It shall be noted that some of the used software components restrict usage of the respective software to research purposes (e.g. Primitive Shapes), or usage in free applications unless a commercial license is obtained (e.g. NVIDIA OptiX). This implies that commercial usage of the software developed by us requires e.g. the re-implementation of some of the functionality provided by the respective libraries and/or the purchase of a commercial license where applicable.

9 Conclusion, Impact and Outlook

We have presented the third version of the software prototypes for the registration of datasets, association BIM and point cloud data, and detection of differences between different datasets of the same building. The registration component now works in a fully automated manner which enables components that are dependent on spatially aligned datasets to run as an automated processing chain. The association component has been extended to support more kinds of correspondences with respect to dynamic objects, and we have exemplified that the association of point cloud data and (possibly automatically generated) BIM data can help to visualize and navigate complex datasets in a more comprehensible manner.

UBO and CITA are currently in contact with multiple partners from industrial and scientific domains who show great interest in the software prototypes developed during the DURAARK project. Software prototypes and datasets are being shared between the partners to further evaluate the possibilities of the developed components and plan further steps beyond the end of the DURAARK project.

References

[1] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[2] A. Makadia, A. Patterson, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1297–1304. IEEE, 2006.

Glossary

BIM Building Information Modeling

IFC Industry Foundation Classes

LDP Long-term Digital Preservation

RDF Resource Description Format

ICP Iterative Closest Point

RANSAC Random Sample Consensus