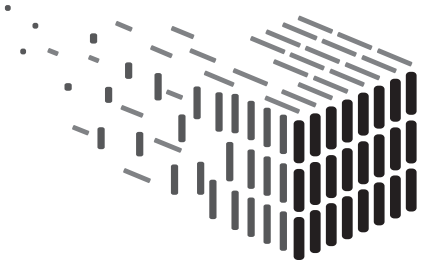

D5.3 Recognition of Architecturally Meaningful Structures and Shapes

Software prototype v2

DURAARK

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

Date: 2015-03-31
Version 1.0
Document id. : duraark/2015/D.5.3/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/2015/D.5.3/v1.0
Title of document	:	Recognition of Architecturally Meaningful Structures and Shapes – 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)
Responsible editor(s)	:	Sebastian Ochmann <ochmann@cs.uni-bonn.de> (UBO) Richard Vock <vock@cs.uni-bonn.de> (UBO)
Quality assessor(s)	:	Jakob Beetz <j.beetz@tue.nl> (TUE) Michelle Lindlar <michelle.lindlar@tib.uni-hannover.de> (LUH)
Approval of this deliverable	:	Stefan Dietze <dietze@L3S.de> (LUH) – Project Coordinator
Distribution	:	Public
Keywords list	:	Curation, Building Semantics, Access Copies, Structuring, Geometric Enrichment, BIM, Point Cloud, E57, IFC

Executive Summary

This report presents the second version of the software prototype for the recognition of architecturally meaningful structures and shapes, as well as an overview of its integration into the overall DURAARK system. Building on the first prototype, we present a highly automatic method for the reconstruction of high-level BIM models from raw indoor point cloud scans. This approach is especially desirable for the legacy building stock for which BIM models from the planning phase are not available. The generated models may be stored alongside the point cloud data in a long-term preservation system for e.g. facility management tasks throughout the life span of a building. Furthermore, we present a point cloud compression method which is suitable for the integration of point cloud parts directly into IFC files.

Table of Contents

1 Introduction

The geometric processing tools developed in WP5 deal with two kinds of three-dimensional building representations: High-level, semantically annotated models for usage in a Building Information Modeling (BIM) setting, and low-level, mostly unstructured point cloud scans. Both representations constitute important building blocks for the long-term preservation of architectural data since BIM models describe the "as-planned" state from an architect's point of view, and point clouds represent a snapshot of the "as-built" state of a building. The availability of both representations in a long-term preservation system enables different use cases, for instance analyzing and comparing representations of the same building using the toolsets developed in WP4 in order to detect any intentional or unintentional deviations, and updating BIM models from observations of the real object in form of point cloud datasets.

One particular challenge is that high-level BIM models are not available for most of the legacy building stock. Since generating such models from scratch is a very time-consuming and cumbersome task, methods for the automatic reconstruction and generation of BIM models from unstructured point cloud data are highly desirable.

Building on the first software prototype for the recognition of architecturally meaningful structures and shapes (D5.1), we have developed a highly automatic method for reconstructing BIM models from raw indoor point clouds. While the first prototype only provided a segmentation of the point cloud into separate rooms and a simple opening detection, our new method is able to reconstruct complete, high-level BIM models. The reconstructed models represent the building using native BIM entities (walls, floor slabs, doors, windows) whose extents are estimated directly from the data (e.g. wall thickness), as well as relations between building elements (e.g. how walls are connected). This recognition of a building's structure (Task 5.1) and shapes like doors and windows (Task 5.2) enables high-level editability of the model in industry-standard architectural software. We demonstrate the applicability of our approach in our second software prototype developed during the second project year. Figure 1 shows an example of a reconstruction produced by our method. For details how to obtain the software prototype for BIM reconstruction, please see Appendix A.

Furthermore, we present an extension to a point cloud compression method (Task 5.4) which increases compression speed and reduces memory requirements of the original approach. These extensions will be beneficial for the integration of point cloud parts directly

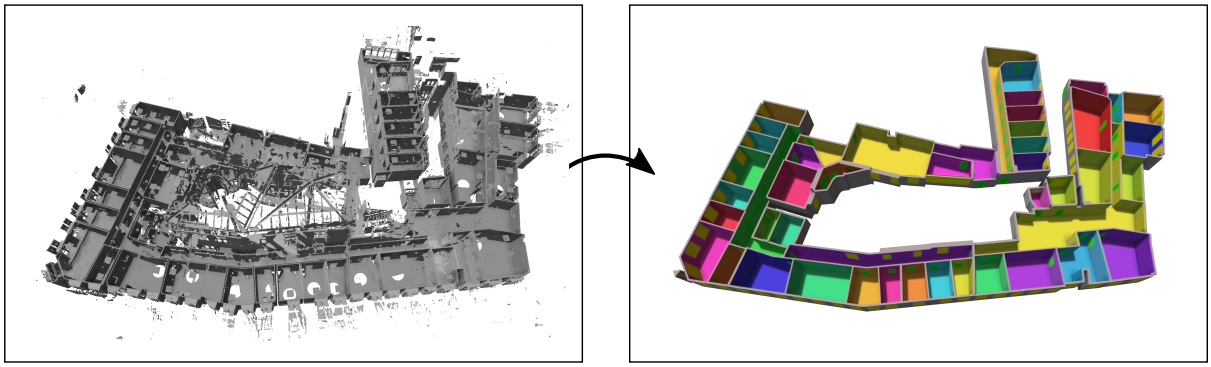

Figure 1: Example result of our method for the reconstruction of BIM models from point clouds. Left: An input point cloud consisting of 67 scans. Right: Our reconstruction of wall elements and openings. Clutter points outside of the building due to points measured through windows (here e.g. in the center of the point cloud) are effectively filtered out by our algorithm. Doors and windows detected in wall elements are shown in green and yellow color, respectively.

into IFC files which will be developed in the third project year. This novel approach for the combination of point cloud datasets and entities in high-level BIM models will lift point cloud data to the semantic level of Building Information Models and allow seamless, simultaneous usage of both kinds of representations. This method will be developed as a joint-effort between WP3 (TU/e), WP4, and WP5.

Figure 2 shows an overview of the overall DURAARK system and puts the components developed in WP5 in context of the workflow. Basically all Tasks and components of WP5 (generation of BIM models, detection of hidden structures, compression) work on point cloud data as their input (i.e. E57 files). IFC files generated by the reconstruction components may either be ingested into the archive, or forwarded to WP3 components for extraction of meta-information such as the number or area of rooms. Compressed versions of point clouds may e.g. be used as part of the "Search & Retrieve" phase for fast previewing. Linking to the use cases defined in D2.2.3, WP5 particularly tackles

- UC2: Search and retrieve archived objects (compression),
- UC7: *Plan*, *document* and verify retrofitting/energy renovations (generated BIM as basis for planning).

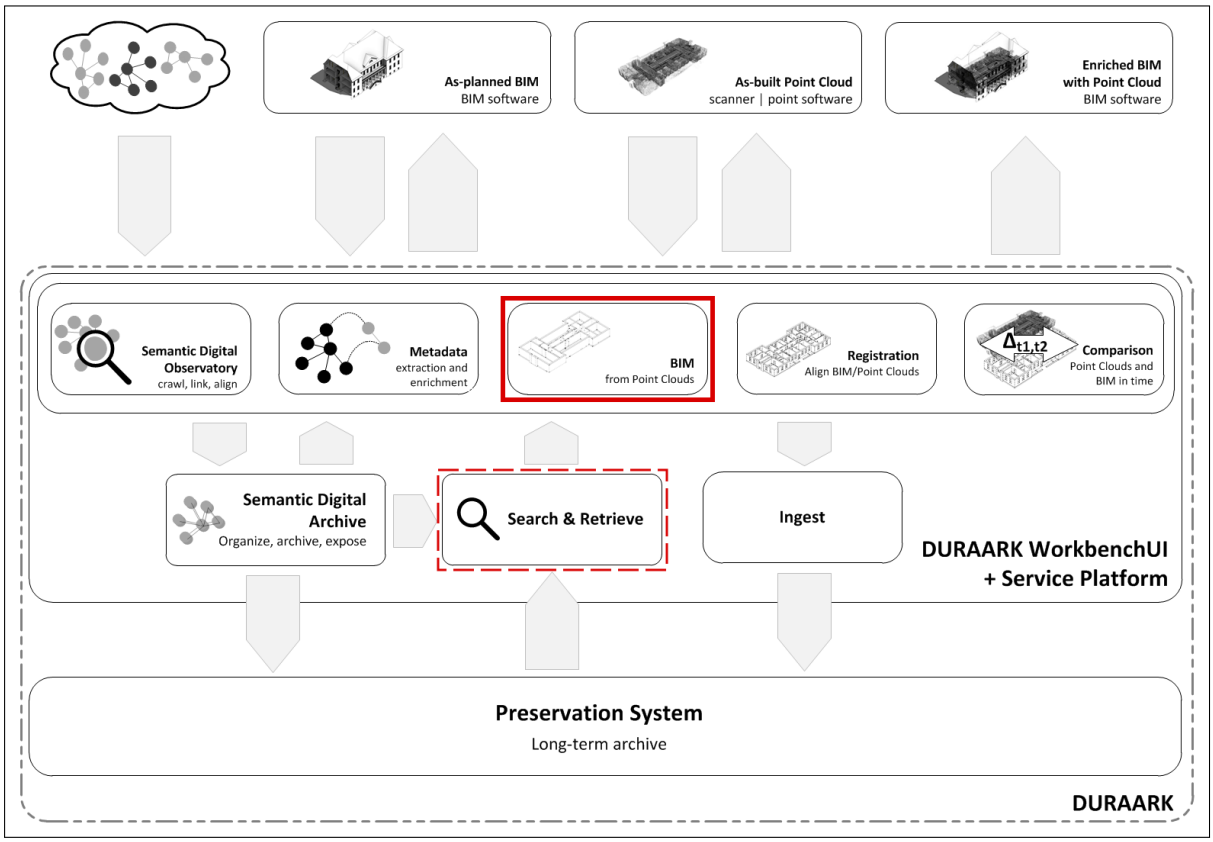

Figure 2: Overview of the DURAARK system. Components related to WP5 are highlighted. The components developed as part of Tasks 5.1-5.3 deal with the reconstruction of BIM models from indoor point clouds. The results of Task 5.4 may e.g. be used for generating access copies that can be used for fast browsing of datasets in the Search & Retrieve phase.

2 Reconstruction of BIM Models from Point Clouds

2.1 Problem Definition

The first software prototype (D5.1), which was developed during the first project year, provided a segmentation of a point cloud dataset into separate rooms, and a detection of openings between adjacent rooms. While the obtained representation has several applications as described in [7], it does not provide a geometric reconstruction of the building in the form of an editable model which can be used in a BIM setting for further processing or analysis in industry-standard BIM software.

The second prototype for the recognition of architecturally meaningful structures and shapes tackles the problem of generating high-level, editable BIM models from raw indoor point clouds. The main challenge of this task is to reconstruct a globally plausible BIM model from a mostly unstructured, point-wise sampling of observed surfaces obtained from (laser) scans in the real building. While several methods for reconstructing three-dimensional models from indoor point clouds exist, none of them represent buildings using native BIM entities and their relations (e.g. how walls are connected) which hinders the usage of the results for e.g. facility management in which a model usually needs to be editable in a meaningful way.

Building on our first prototype, we now developed a highly automatic approach for the reconstruction of 3D models from indoor point cloud scans which reconstructs buildings using native BIM entities (e.g. walls, doors, and relations between entities, e.g. where walls are connected). This allows us to export the resulting model directly in form of IFC (Industry Foundation Classes) files for further processing in architectural software, as well as storage in the long-term preservation system alongside the point cloud representations. In addition to reconstructing the building's floor, ceiling and wall structure (detection of structure, Task 5.1), we also detect openings in reconstructed wall elements (detection of shapes, Task 5.2). While our first prototype performed a simple opening detection, we are now also able to distinguish between door and window openings which are classified by means of a machine learning approach.

2.2 Related Work

In this Section, we give a brief overview of related reconstruction methods which have been published since the first version of this deliverable. Most of the related work described in D5.1 is also relevant for the reconstruction of 3D models from point clouds.
Mura et al. [2] reconstruct indoor scenes with arbitrary wall orientations by building a 3D Delaunay tetrahedralization of the input dataset and partitioning inside and outside using a diffusion process on a matrix encoding affinities of tetrahedron pairs. A binary space partitioning is also done by Oesau et al. [4] by first splitting the input dataset horizontally at height levels of high point densities and then constructing a 2D arrangement of projections of detected wall surfaces. The space partitioning into inside and outside is performed by means of Graph-Cut. Mura et al. [3] first extract candidate wall elements while taking into account possibly occluded parts of the surfaces to determine the real wall heights for filtering out invalid candidates. After constructing a 2D line arrangement, they use a diffusion embedding to establish a global affinity measure between faces of the arrangement, and determine clusters of faces constituting rooms. The result is a labeled boundary representation of the building's rooms. Turner et al. [8] use a volume carving approach to reconstruct simplified mesh models from point clouds captured by backpack-mounted mobile scanning devices. They first generate a 2D floor plan and perform a room labeling. This plan is subsequently extruded to constitute a 2.5D representation of the building. Finally, a texture mapping of images captured during scanning is performed. The resulting models can be used e.g. for navigation purposes.
While most previous approaches are able to faithfully represent the building's geometry and generate 3D models (e.g. meshes) which have several applications in e.g. energy monitoring or navigation, none of them represent buildings using parametric building elements which are suitable for generating a high-level, editable BIM model. In contrast, our proposed method regards the reconstruction of buildings using native BIM entities and their relations as a central component.

2.3 Workflow

This Section gives an overview of the workflow of our reconstruction software prototype from the user's perspective. Two versions of the software are available: A standalone, graphical desktop application version which guides the user visually and interactively through the processing of the datasets, and a command line version which allows processing of datasets e.g. on servers that do not have a graphical user interface. The internal reconstruction algorithm is the same for both versions.

2.3.1 Graphical Desktop Application Version

The user interface of the reconstruction prototype extends our previous point cloud segmentation prototype which is based on the same GUI (graphical user interface) framework. The workflow starts by selecting an input point cloud dataset of a building storey e.g. in E57 format which consists of multiple indoor scans that are registered (i.e. spatially aligned) in a common coordinate system. The unit of measurement and the "up" axis are assumed to be known; this information is usually contained in the E57 file format. Note that our reconstruction algorithm currently works on single building storeys[1]. The task of reconstructing multiple storeys simultaneously poses distinct challenges, rendering this a harder problem which is a topic for future work.

After the point cloud dataset is loaded, the user may inspect it interactively using a 3D visualization (Figure 3, top). Addressing first feedback from stakeholders, we improved the user experience by largely facilitating the overall workflow: In contrast to the first prototype, our reconstruction algorithm does no longer require scans taken within the same room to be merged manually; an automatic merging step is performed as part of the BIM model reconstruction. However, manual, interactive merging of scans is still possible to help the algorithm in difficult cases.

The automatic BIM model reconstruction can now be started by the click of a button. While some steps of our reconstruction algorithm may be tweaked using several parameters, we found that a fixed set of parameters worked well across a variety of real-world datasets that we used for our experiments. This makes the reconstruction method highly automatic and user interaction is usually not required.

[1]While a sorting of scans depending on the storey they belong to is a common practice among stakeholders when performing scanning sessions, not all of our datasets include such a separation into separate storeys. Thus a manual sorting of scans may currently be required if this information is not available.

Depending on the size of the dataset, our prototypal implementation takes from a few seconds to a few minutes for processing a dataset. After the reconstruction has been performed, wall elements, as well as detected doors and windows are visualized for inspection. The user also has the opportunity to visualize the resulting model and the point cloud simultaneously for comparison, as well as several intermediate steps of the algorithm. In addition, simple editing tasks may be performed directly in the software prototype: Wall elements and connection points between walls may be moved interactively (Figure 3, bottom), and the types of detected openings (doors, windows) may be changed. If desired, it is also possible to tweak advanced parameters and re-run the reconstruction algorithm.
Once the user is satisfied with the results, the reconstructed BIM model may be written out as an IFC file which can subsequently be imported in architectural software such as Autodesk Revit.

2.3.2 Command Line Version

In addition to the GUI version, we also provide a command line version of our reconstruction prototype. This version is tailored for batch processing of datasets on systems which may not have a graphical user interface, such as servers. The core reconstruction algorithm is identical to the GUI version.
Currently, the command line version does not allow tweaking of parameters of the reconstruction algorithm but uses the same default parameters as the GUI version. The program only requires two file paths as arguments, namely a file path of the input E57 file, and a file path where the resulting IFC file shall be written to.

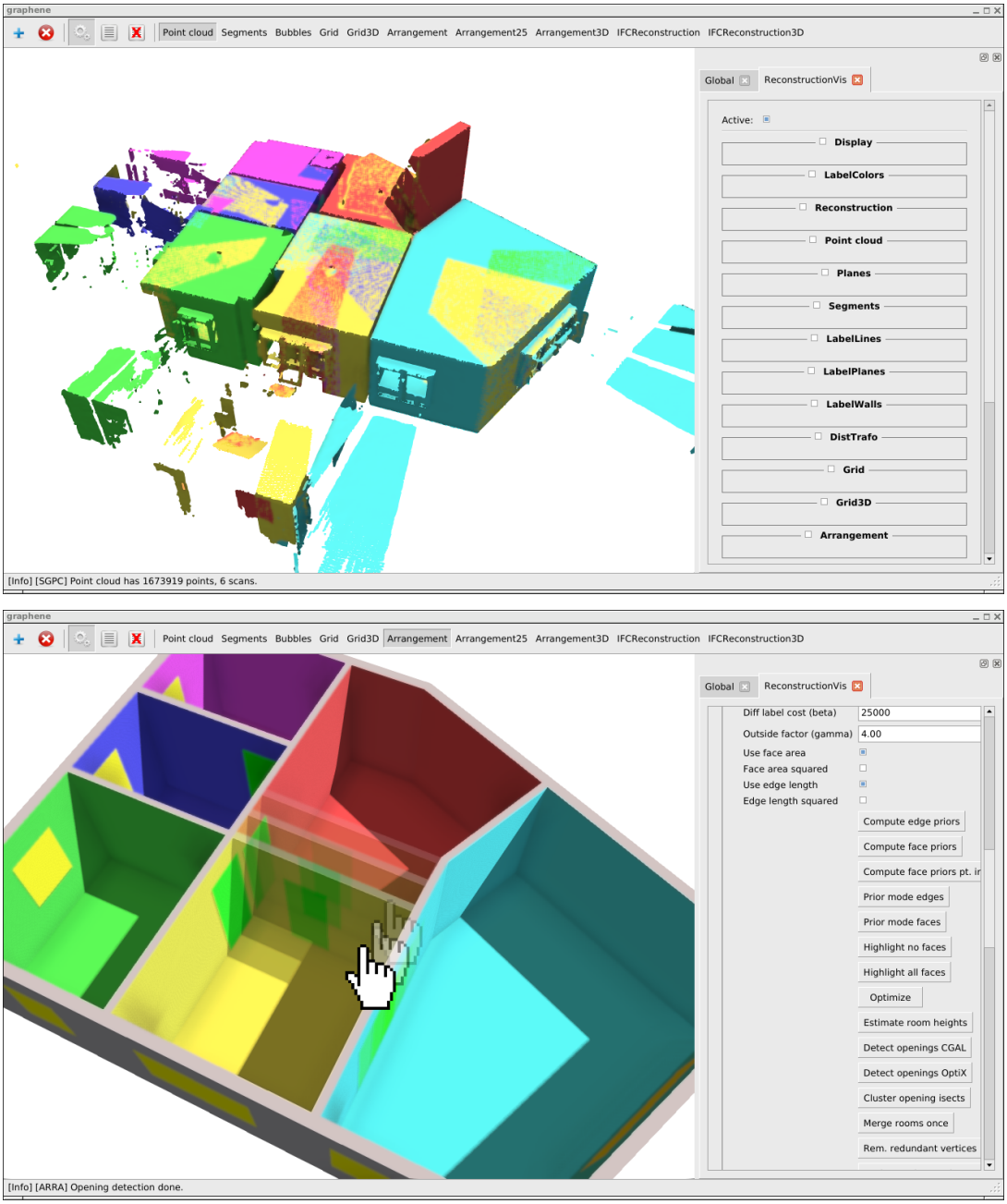

Figure 3: **Top:** A point cloud dataset loaded into our reconstruction prototype. Different scans are shown in different colors. The user may inspect the point cloud and (optionally) merge multiple scans within the same room to support the reconstruction algorithm. **Bottom:** After reconstruction, the user may inspect the resulting model and perform simple editing tasks within the prototype. The final model can now be exported as an IFC file for further processing in architectural software or storage in the archive.

2.4 Method Overview, Implementation Details

In this Section, we give a concise overview of our method for reconstructing BIM models from raw indoor point clouds. A detailed description of our approach will be submitted shortly as a full paper. For this reason, we cannot disclose all details at this point.

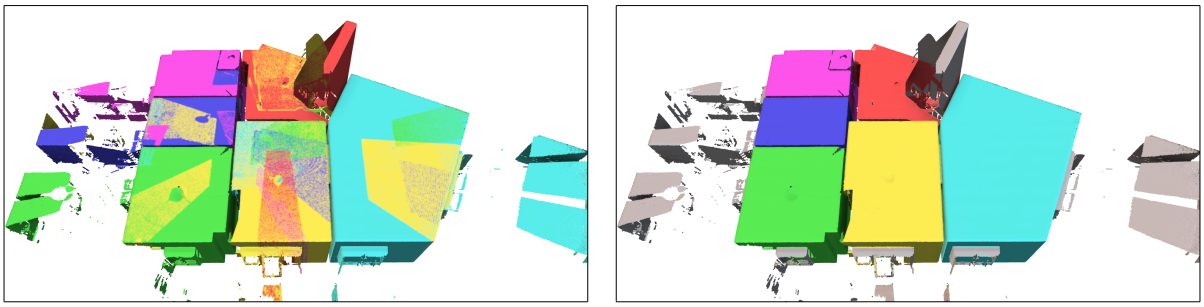

Figure 4: Automatic refinement of point associations to separate scans. Left: An initial assignment of points to different scans (visualized in different colors) yields a coarse partitioning of the building in different rooms. However, overlaps between scans lead to a mix of differently labeled points in large regions. Right: Using the method developed in the first project year, the assignment are automatically refined such that point labels are homogeneous within rooms. We extended our method to also filter out clutter outside of the building (light gray).

The starting point of our approach are indoor point clouds of a building storey consisting of multiple scans, including scanner positions. Figure 4 (left) shows an example of a point cloud in which the separate scans are highlighted in different colors. While this association of points to scans gives coarse hints where rooms are located, large overlaps between scans exist. In order to mitigate this problem, a coarse partitioning of the point cloud into rooms is performed using the method developed in the first project year. The result after this step is demonstrated in Figure 4 (right): Point associations are now homogeneous within each room of the building. We extended our previous method such that outliers outside of the building (e.g. measured points scanned through windows) are filtered out automatically. In the Figure, outlier points are shown in light gray color. These points are ignored in subsequent steps.

In the next step, candidates for *potential* locations of walls separating rooms from the outside area, as well as for walls separating adjacent rooms are generated. These wall candidates are generated from planar, vertical structures detected in the set of measured points. The previous partitioning of the point cloud allows us to associate these wall candidates with rooms of the building which yields important information for determining

which rooms are separated by a particular wall in later steps. Figure 5 (right) depicts the set of wall candidate centerlines which have been generated for the point cloud.

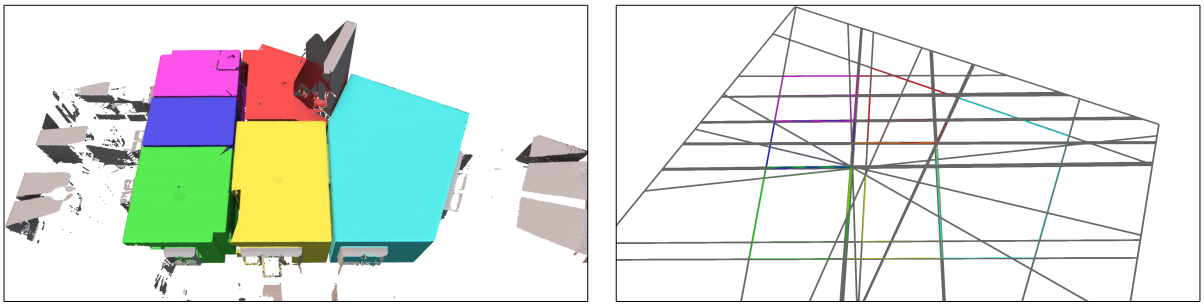

Figure 5: Potential locations for vertical walls are extracted from planar, vertical surfaces observed in the point cloud data (right). At this point, each wall candidate is (conceptually) infinitely long. Finite wall candidate segments, which constitute a globally plausible configuration of wall elements, are extracted in subsequent steps.

Having obtained candidates for possible wall entities, the task is to determine a suitable subset of wall candidate segments which faithfully represent the building's real walls. An energy minimization approach is used to find a globally plausible configuration of walls, including their connectivity and an estimation of wall thickness directly from the point cloud data. The prior point cloud partitioning, as well as the association of wall candidates to single rooms or pairs of rooms, constitute important priors for the definition of the used energy functional used for the optimization. The result at this point is a graph in which edges represent centerlines of extracted wall entities, and vertices represent positions at which wall entities are incident (Figure 6 (right)).

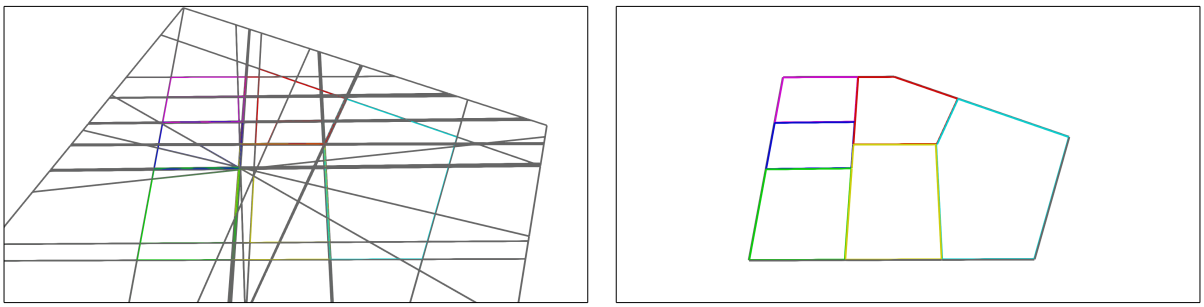

Figure 6: Using the generated wall candidates (left), an energy minimization approach is used to extract a subset of wall candidate segments which yields a globally plausible configuration of walls. The result is a connected graph in which edges represent wall elements, and vertices represent locations where wall elements are incident (right).

For extruding the wall entities vertically, the height of each room of the building is

estimated heuristically by determining how many measured points belonging to approximately horizontal, planar structures (i.e. floor and ceiling surfaces) are located within each face of the extracted wall graph. Each wall entity is extruded such that its height spans the floor and ceiling heights of the incident rooms. The thickness of reconstructed wall entities is estimated from the support points of the wall candidate from which the respective wall entity originates. Figure 7 shows an example of a wall graph and the extruded wall model.

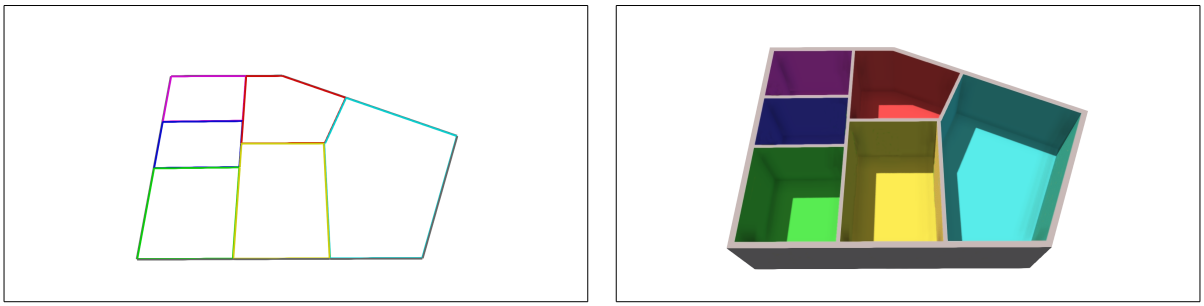

Figure 7: For extruding the wall elements vertically (right), we first estimate room heights for each face of the extracted wall graph, and extruding incident wall elements accordingly. Wall thickness is determined by measured points from which the respective wall candidates originate.

After walls have been reconstructed, openings (doors, windows) are detected by clustering intersection points of simulated laser rays with the reconstructed walls. In order to distinguish between different kinds of intersection point clusters, a machine learning approach is used. Figure 8 (left) shows an example for detected openings and their classification (green = door, yellow = window, red = invalid). The final reconstructed model (Figure 8 (right)) can now be exported to IFC format for further processing and analysis in BIM software, and for storage in the long-term preservation system alongside the point cloud.

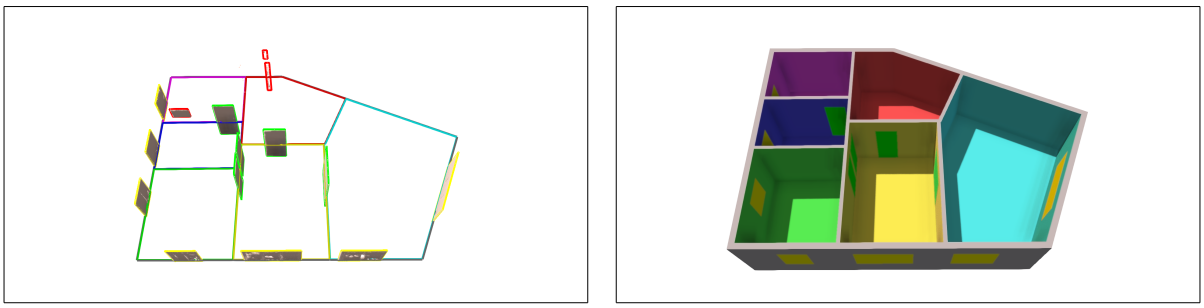

Figure 8: Openings in walls are determined by clustering intersection points of simulated laser rays from the scanner positions to the measured points with reconstructed wall elements (left). The determined clusters are classified as doors (green), windows (yellow), and invalid clusters (red) by means of supervised learning. The final model (right) can now be exported as an IFC file.

2.5 Evaluation

To evaluate our approach, we have tested our method on a variety of real-world datasets which have been collected and captured within the DURAARK project, in particular by CITA and LTU. In this Section, we present exemplary results on some of the datasets we used. A more in-depth evaluation is presented as part of D7.2 by CITA[2], and in our paper which will shortly be submitted to a journal.

Each of the following Figures shows the input point cloud on the left hand side (with most points belonging to ceiling surfaces removed for visualization) and the resulting reconstruction of our algorithm on the right hand side. In the reconstructed model, different rooms are shown in different colors; detected doors are shown in green color, detected windows are shown in yellow color. Note that the given number of points refer to the subsampled version of the point cloud that is actually used for processing and not the number of points of the original dataset.

[2]Please note that the evaluation by CITA is based on a slightly earlier version of the prototype.

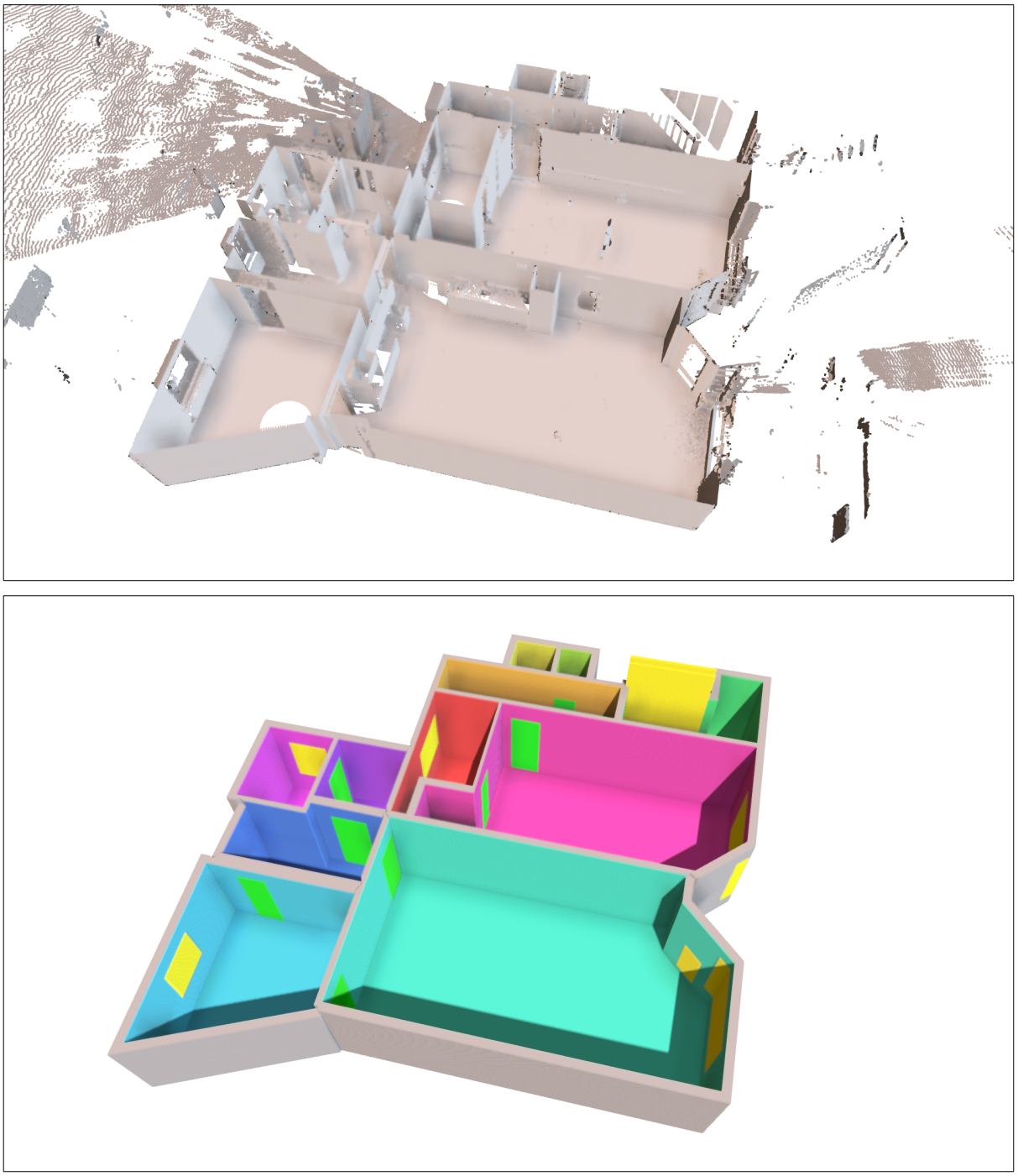

Figure 9: Subset of the *Gormsgade* dataset, 2470678 points in 11 scans.

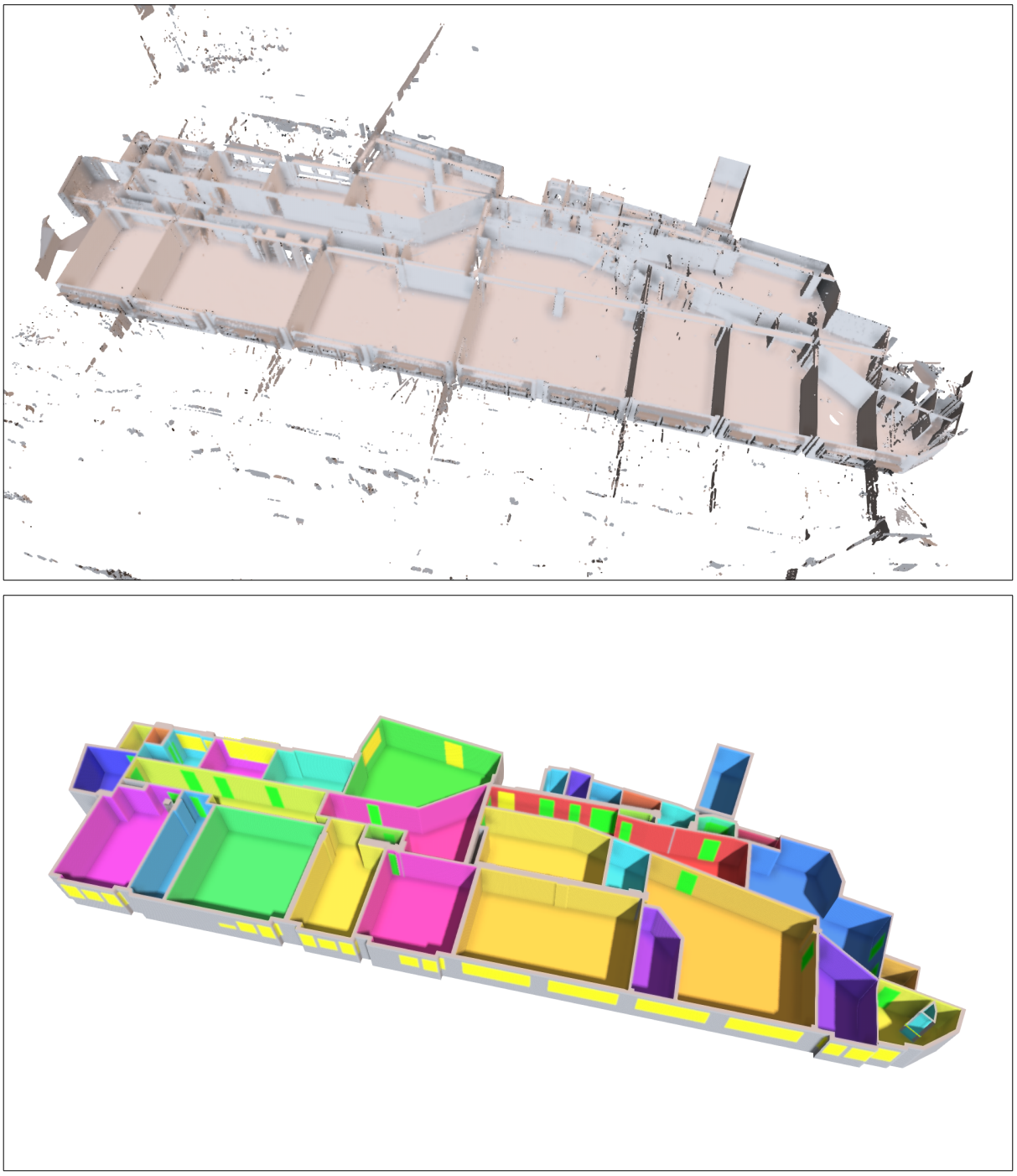

Figure 10: Subset of the *Vestergade* dataset, 19769647 points in 51 scans.

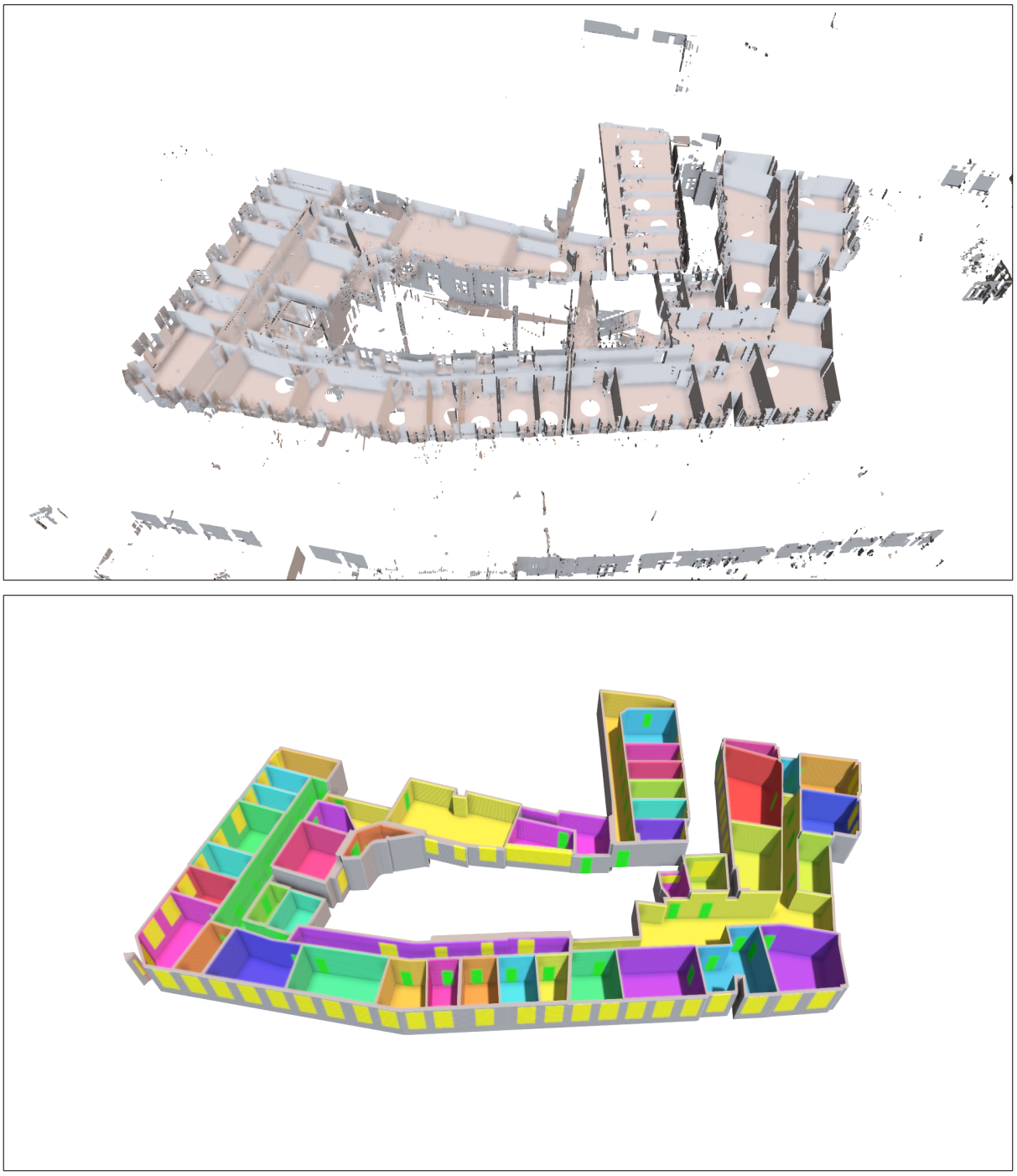

Figure 11: Subset of the *Hojbroplads* dataset, 22757718 points in 67 scans.

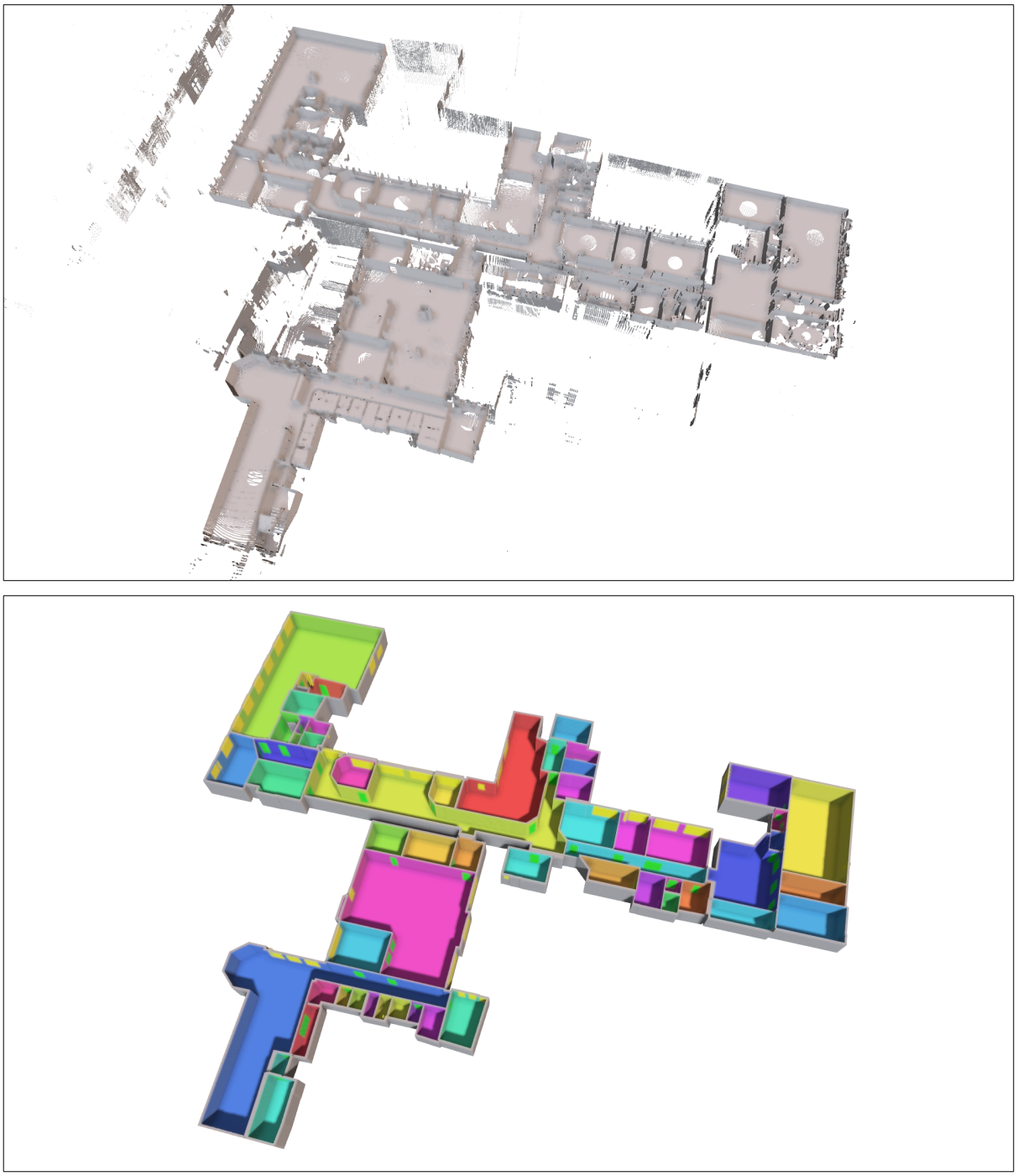

Figure 12: Subset of the *Nygade* dataset, 19001890 points in 88 scans.

2.6 Integration

Both the graphical desktop application version and the command line version of the reconstruction prototype have been provided for integration into the DURAARK system to WP2. The command line version is part of the Service Platform which allows to access the tool's functionality via a REST interface (Figure 13, bottom-left). As such it is part of the "server-side" toolset which usually does not require direct user interaction. It is provided as a Linux executable in a Docker[3] container which allows easy deployment of the software component in a self-contained image. The graphical version is deployed as a standalone Windows executable. The software component (Figure 13, right) is accessible via the WorkbenchUI and allows an interactive processing of locally available datasets as described in Section 2.3. For a description of the Service Platform and WorkbenchUI, please refer to D8.5, Section 2.2.

Figure 14 provides a workflow-centric localization of the developed components into the overall system.

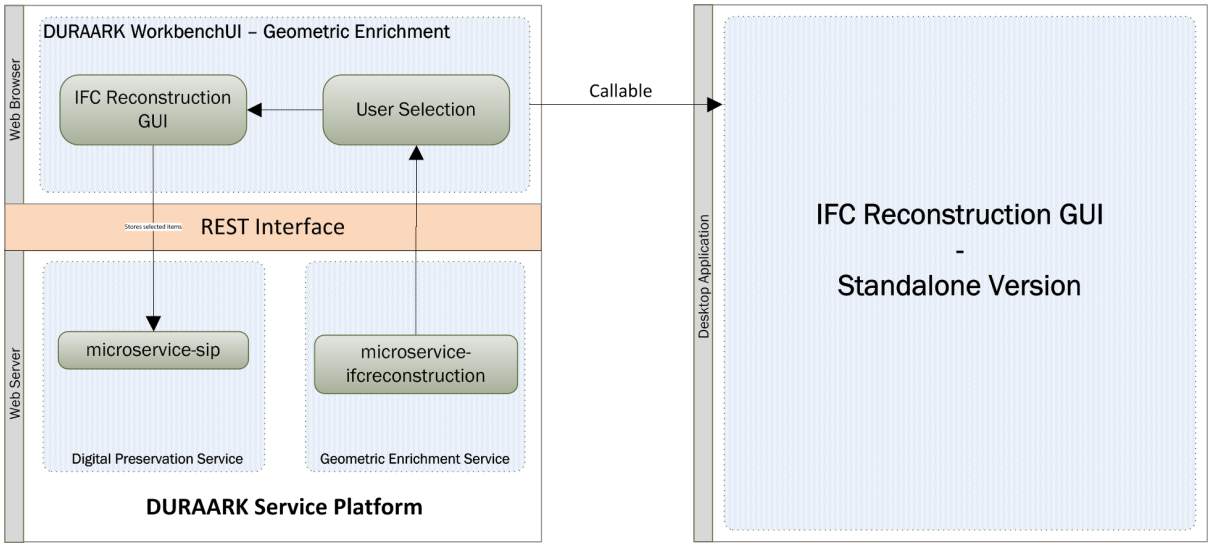

Figure 13: Integration of the reconstruction components into the DURAARK system. The reconstruction prototype is available as an interactive, graphical version (right) which will be callable from WorkbenchUI, and a "headless" command-line version which will be part of a microservice in the Service Platform (bottom-left).

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

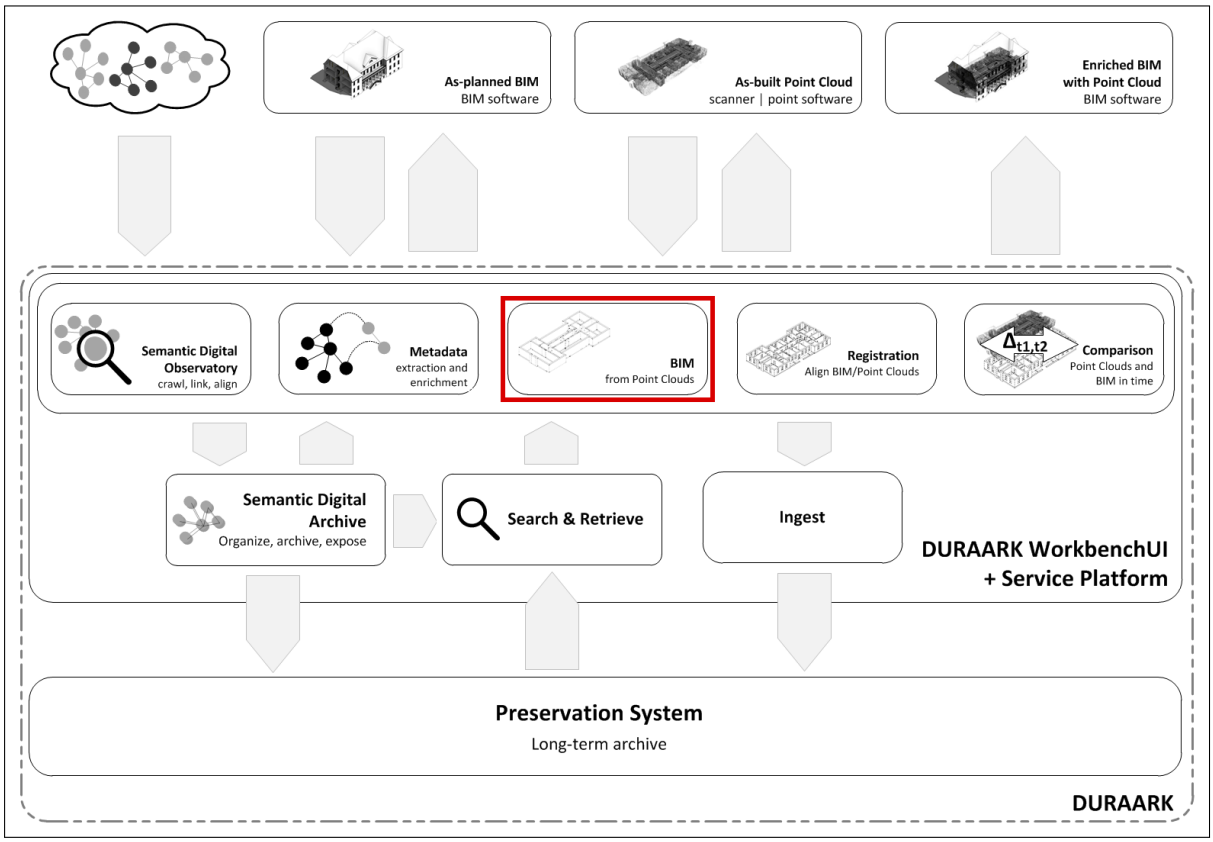

Figure 14: Location of the BIM reconstruction components in the overall DURAARK workflow. The input of the component are indoor point clouds; the resulting BIM models (i.e. IFC files) may be ingested into the preservation system or e.g. further processed by the semantic enrichment tools for extracting meta-information such as number of rooms, area, etc.

3 Point Cloud Compression

In this part, we present our current developments regarding point cloud compression techniques. In addition to the planned generation of access copies for fast previewing of stored datasets, we are also planning towards the integration of (subsets of) point cloud data into IFC files (which is also related to Task 4.3, transfer of structure and semantics). We argue that an association of point cloud parts with semantically high-level building elements of BIM models will be beneficial for closing the gap between as-planned and as-built geometry, and will allow to work with both representations simultaneously on a semantically high level.
In order to store point cloud data directly as part of an IFC file, a compression method which takes advantage of the available information and domain (e.g. man-made objects, large planar structures) is desirable. We thus extended a point cloud compression approach based on detected primitive shapes (e.g. planes). While the proposed method currently works by detecting primitives in the point cloud itself, we argue that geometric shapes of a corresponding BIM model could also be used for compressing parts of the point cloud after the association between BIM model and point cloud has been established.

3.1 Problem Definition

Modern point cloud capturing devices produce datasets which have very high precision and density. As a consequence, the resulting datasets are usually large and thus unhandy for certain tasks, e.g. fast previewing of stored datasets. In addition to usage of compressed point clouds as *access copies*, we are planning to integrate point cloud parts into IFC files, directly associating them with corresponding BIM entities. Depending on the requirements, compressed point cloud representations are desirable, and the used compression method shall take advantage of the available information, e.g. which BIM entity the point cloud is associated with.
We have pinpointed the following features which the compression method shall fulfill:

- Take advantage of the available information (association with BIM model) and domain (architecture, man-made objects).
- The desired level of detail shall be selectable, depending on the specific requirement (e.g. quick previewing vs. more detailed investigation).

- Compression and decompression shall be fast.
- The compression method shall yield sufficiently high compression ratio suitable for transfer over low-bandwidth networks.
- When multiple datasets of the same building are available, previously computed data from one dataset shall be re-used to decrease time and memory requirements when datasets are subsequently added.

While several methods for the compression of point cloud data have been proposed, none of them fulfills all of the aforementioned requirements. A promising candidate is the compression method by Schnabel et al. [6] which is based on primitive shapes (e.g. planes) detected in the point cloud. We extend this method to make it suitable for the envisioned use case.

3.2 Method Overview

In this Section, we give an overview of our proposed compression method. A detailed description is given in our paper [1] which was presented at the 19th International Workshop on Vision, Modeling, and Visualization (VMV 2014).

Our method is based on the compression technique by Schnabel et al. [6]. The original compression method performs a decomposition of the point cloud into clusters which are approximated by geometric primitives (e.g. planes). Each point is either associated to one of the detected geometric primitives, or discarded if it is not associated to any primitive shape. Fine details over the detected primitives (e.g. detail structures of surfaces) are represented by height fields constructed over each primitive. The height fields are stored in different resolutions in form of a Laplacian pyramid (i.e. differences between differently subsampled versions of the original height field) which allows for a level of detail reconstruction: Downsampled, lower-resolution representations are obtained by using only a subset of the stored data for reconstructing the height fields. For compressing the height fields, a vector quantization approach is used: The height-maps are split into tiles from which a codebook (or dictionary) of representative examples is built. The original height field tiles are then exchanged by links to example tiles in the codebook.

While the original algorithm fulfills three of our criteria mentioned in Section 3.1, namely exploitation of geometric shapes (e.g. large planes) common in man-made objects, selectable level of detail, and good compression ratios, the method is relatively slow since codebook generation is done anew from scratch on every dataset, and previously computed data is not re-used for compressing newly acquired datasets of a whole building, or part of a building.

In order to overcome these shortcomings, we propose the following extensions to the original algorithm. First, instead of performing the codebook generation anew for each dataset, we perform a prior offline "learning" phase in which a codebook is generated from a given set of training datasets. This precomputed codebook is then used for compressing newly acquired datasets which improves speed and decreases space requirements since a relatively small codebook may be re-used for a whole range of different datasets. Second, we replace the moving least squares approach used for interpolation in the height field generation step by a faster, GPU-based approach which further helps to speed-up our algorithm.

3.3 Evaluation

While an evaluation of using the proposed compression method for the integration of point clouds to IFC files is still pending, the approach has been evaluated on real-world datasets using primitives detected directly in the point clouds. The complete evaluation is given in our paper [1]; in this Section we summarize findings which are particularly important for our envisioned usage scenario.

The compression and error rates achieved with the modified algorithm are generally comparable to those of the original method while our modifications (e.g. usage of GPU-accelerated height field computation) yielded a significant speed-up by a factor of about 5. Both compression quality and time requirements depend on the selection and size of the codebook used for height field quantization, but the method showed to be especially robust on test point clouds from an urban setting even when using small codebook sizes. We postulate that the method will also work well on other architectural datasets.

Especially when dealing with indoor scans of a building in which large amounts of points represent relatively simple structures (e.g. planar wall surfaces), using only a small amount of representative codebook examples should yield good compression results while keeping memory and time requirements low.

4 Decisions & Risks

4.1 Technical Decisions

Choice of programming language The reconstruction 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 (using OpenMP), 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 output, 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, implemented and maintained by a consortium partner) is freely available.

4.2 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.
- **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 (Intel Embree, or our previous, custom OpenCL-based implementation) which can be used almost as drop-in replacements.

Obsolescence of data formats used

- **Risk Description:** The file formats used (E57, IFC) are abandoned/replaced by newer alternatives.
- **Risk Assessment:**
  - **Impact:** Medium
  - **Probability:** Medium
  - **Description:** The E57 and/or IFC file formats 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.

5 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	Primitive Shapes	Custom, see below.	—
used	NVIDIA OptiX	Inclusion in free applications allowed; commercial use requires Commercial License	https://developer.nvidia.com/optix
used	CGAL	LGPL/GPL, or commerical licensing	http://geometryfactory.com/products/licenses/
used	mpfr	LGPL v3	http://www.mpfr.org/
used	GCoptimization	Custom, see below.	—
generated	Reconstruction shared library	GPLv2	http://www.gnu.org/licenses/gpl-2.0.html

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 CONTRIBUTORS 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.

GCoptimization license:

Copyright 2007-2010 Olga Veksler (olga@csd.uwo.ca), Andrew Delong (andrew.delong@gmail.com)
This software and its modifications can be used and distributed for research purposes only. Publications resulting from use of this code must cite publications according to the rules given above. Only Olga

5.1 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, GCoptimization), 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.

6 Conclusion, Impact and Outlook

We presented the second software prototype for the recognition of architecturally meaningful structures and shapes in point cloud datasets. Building on the first version of our prototype, which provided a segmentation of point clouds into separate rooms and a detection of openings, we developed a highly automatic method for the reconstruction of high-level BIM models from raw indoor point clouds. In addition to detecting wall entities and their connectivity, it also detects and distinguishes door and window openings. By exporting the resulting model as an IFC file, the model can be directly used in industry-standard architectural software, and stored alongside the corresponding point cloud datasets in a long-term preservation system.

First evaluation of the developed software by CITA (see also D7.2) shows that the results of this early prototype are already beneficial to the stakeholder community. Further evaluation in Year 3 will provide a more in-depth analysis which will help to improve and stengthen our novel reconstruction approach.

Furthermore, we presented an extension to a point cloud compression technique based on primitive shapes (e.g. planes). Our extensions reduce time and memory requirements of the approach while maintaining compression quality comparable to the original method. In the next steps, the proposed method will be used for integrating parts of point cloud datasets into IFC files in collaboration with TU/e which will enable simultaneous and semantically high-level usage of both kinds of representations.

References

[1] T. Golla, C. Schwartz, and R. Klein. Towards Efficient Online Compression of Incrementally Acquired Point Clouds. In J. Bender, A. Kuijper, T. von Landesberger, H. Theisel, and P. Urban, editors, *Vision, Modeling & Visualization*. The Eurographics Association, 2014.

[2] C. Mura, A. Jaspe Villanueva, O. Mattausch, E. Gobbetti, and R. Pajarola. Reconstructing complex indoor environments with arbitrary wall orientations. *Proc. EG Posters.*, 2014.

[3] C. Mura, O. Mattausch, A. Jaspe Villanueva, E. Gobbetti, and R. Pajarola. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, 44:20–32, Nov. 2014.

[4] S. Oesau, F. Lafarge, and P. Alliez. Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90:68–82, Apr. 2014.

[5] R. Schnabel and R. Klein. Octree-based point-cloud compression. In *Proceedings of Symposium on Point-Based Graphics (SPBG 2006)*, pages 111–120, July 2006.

[6] R. Schnabel, S. Möser, and R. Klein. Fast vector quantization for efficient rendering of compressed point-clouds. *Computers & Graphics*, 32(2):246–259, 2008.

[7] M. Tamke, I. Blümel, S. Ochmann, R. Vock, and R. Wessel. From point clouds to definitions of architectural space. In *Proceedings of eCAADe 2014*, pages 557–566.

[8] E. Turner, P. Cheng, and A. Zakhor. Fast, Automated, Scalable Generation of Textured 3d Models of Indoor Environments.

A Software Manual

This Section describes basic usage of the graphical desktop application version of the reconstruction prototype. Note that the software uses the same GUI framework as the first version of the prototype. Basic controls (e.g. camera controls) are described in D5.1. The software prototype is available at `ftp://ftp.cg.cs.uni-bonn.de/pub/outgoing/duraark/d5_3_prototype.zip` (anonymous login).

After starting our visualization tool `graphene.exe`, the user is greeted with a dialog for choosing an E57 point cloud file for reconstructing a model from (Figure 15). Alternatively, the user may leave the file selection empty at this point and load a point cloud file later. The "Start" button will bring the user to the main interface.

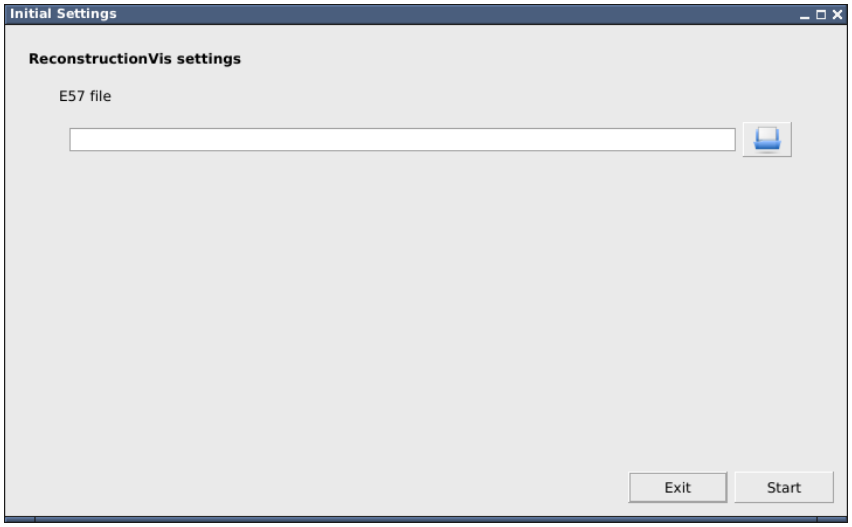

Figure 15: Initial dialog. The user may select an E57 file to load here.

If an E57 file was selected in the start dialog, it will be loaded and visualized in the main interface (Figure 16). If no file is loaded yet (or another file shall be loaded), the "Load E57 file" widget on the right-hand side of the interface may be used. The user may inspect the loaded dataset interactively, similar to the first version of the software prototype. The reconstruction algorithm may then be started using the "Perform reconstruction" button. Status messages will be written out to the status bar at the bottom of the main window, and to the console.

When the reconstruction is done, the resulting model is shown and may be inspected by the user (Figure 17). The point cloud can also be shown by enabling the "Point cloud" checkbox within the "Display" section on the right-hand side. Note that detected doors

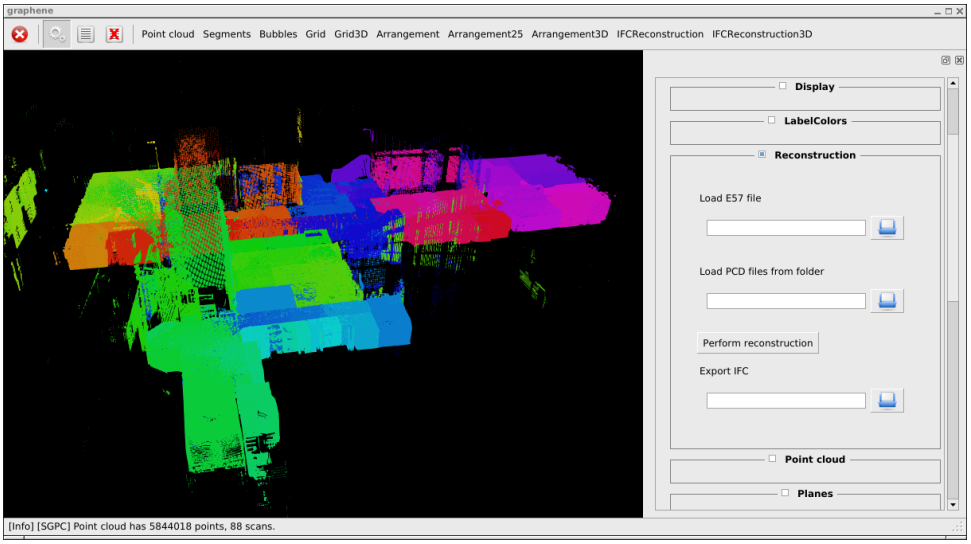

Figure 16: A point cloud loaded into the reconstruction tool.

and windows are shown as green and yellow boxes on the walls, respectively.

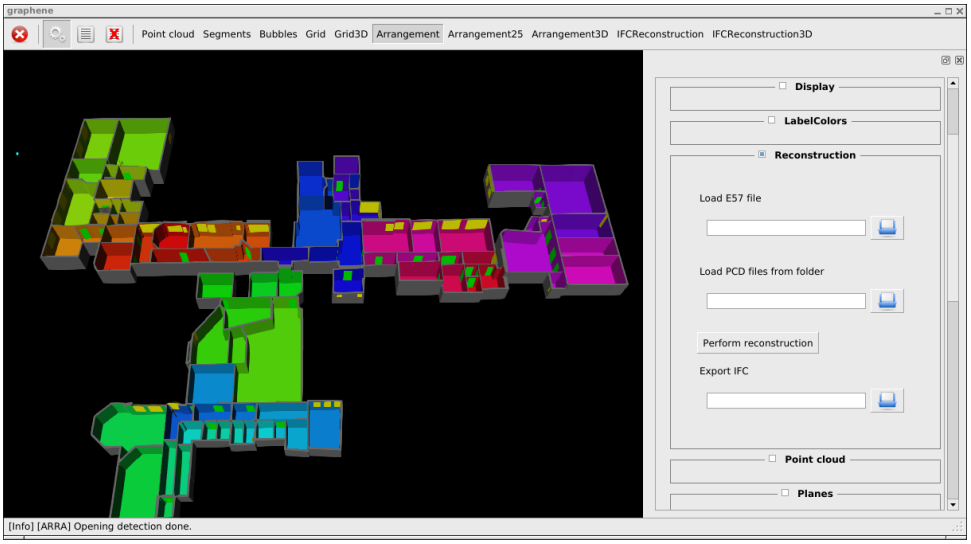

Figure 17: The resulting model after reconstruction has been performed.

Simple editing tasks may be performed within the prototype, e.g. moving walls and endpoint of walls (Figure 18). For this, the "Arrangement" button at the top of the main window needs to be enabled. Afterwards, wall elements or endpoints of walls may be dragged while holding down the left mouse button.
Finally, the resulting model may be exported as an IFC file using the "Export IFC" file selector on the right-hand side of the main window.

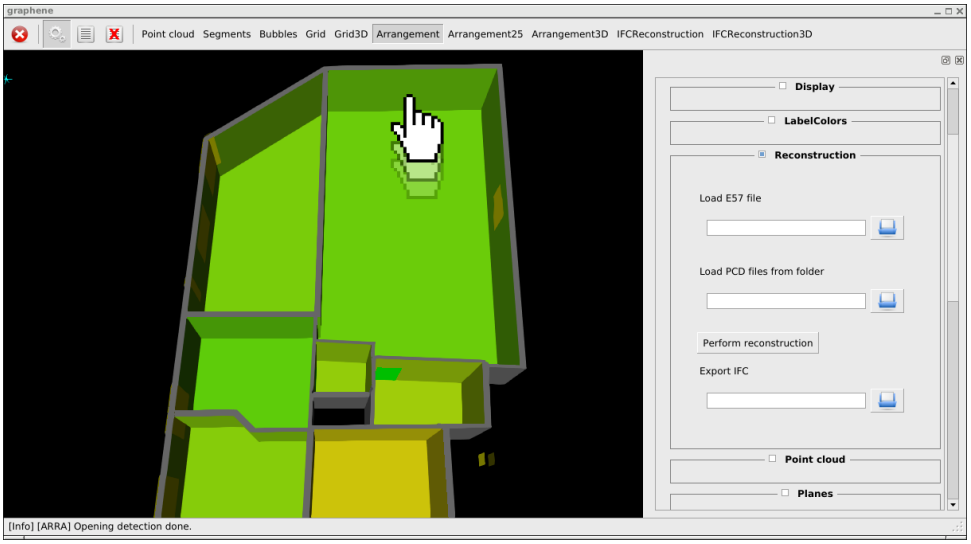

Figure 18: Simple editing tasks may be performed within the tool.

B Addendum to D5.1

In our previous deliverable D5.1, we argued that the evaluated octree-based point cloud compression method by Schnabel et al. [5] may be used to perform (pseudo-)lossless compression. As a follow-up to the reviewer's recommendations from Year 1, we present exemplary results of the compression of a subset of the Risløkka dataset including a comparison of compression rates and precision of the resulting compressed point cloud representation.

The notion of losslessness fundamentally depends on the envisaged usage scenario of the compressed point cloud datasets. While scanning devices often are able to capture point clouds at very high spatial precision (depending of the type of capturing device and used settings), such high precision may not be necessary or even desired in some scenarios. As an example, the IFC reconstruction prototype presented in this deliverable uses subsampled versions of the input point clouds in order to keep computational costs manageable. Concretely, for processing point clouds using our method we subsample the datasets such that within a cubic volume with a side-length of 2-4 mm, at most one point of the input data is retained. We will use this exemplary use case to put the achieved precision after compression at different octree levels into perspective.

For our experiment, we used a non-subsampled subset of the Risløkka dataset consisting of 1.635.960 points. Table 1 shows an overview of the total data size ("total bits") at different octree levels, the estimated number of bits per point, and the worst-case displacement[4] (in mm) of a measured point given the respective octree level ("Max. spatial error"). Figures 19 and 20 show charts of the max. spatial error and bits per points, respectively; we only show values from octree level 8, since lower levels are usually not meaningful due to very coarse precision.

In the exemplary use case of our IFC reconstruction method, a maximum displacement error of about 2-4 mm may be regarded as lossless in a sense that the measured points of the compressed representation are equivalent to the original point cloud for our purpose. As can be seen in Table 1, an octree level of 12-13 yields the desired precision while yielding a data size of about 4.5-7.6 bits per point. Note that the desired precision may be increased almost arbitrarily by adjusting the maximum octree depth such that "lossless" results may be achieved for any demanded worst-case error margin.

[4]Since the compression method replaces measured points by the center of octree cells, we estimate the maximum possible displacement as half the length of the diagonal of an octree cell at the respective subdivition level.

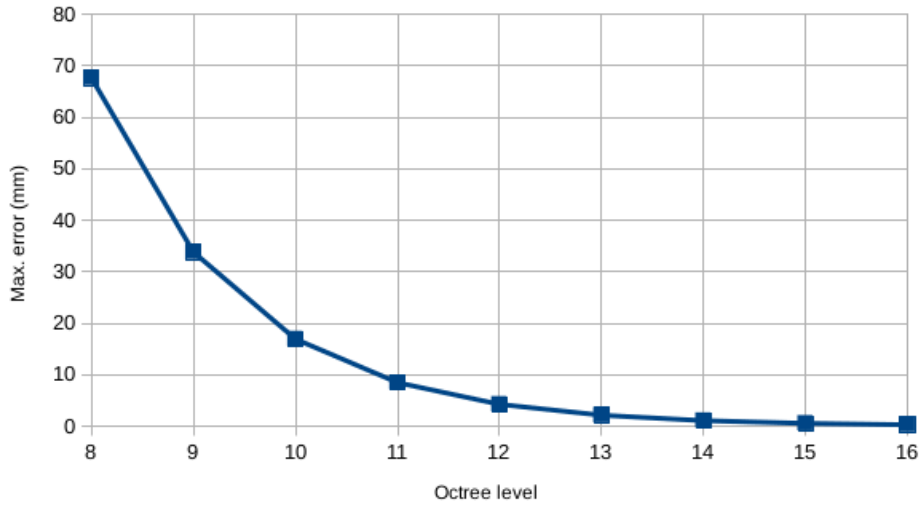

Figure 19: Max. spatial error at different octree levels.

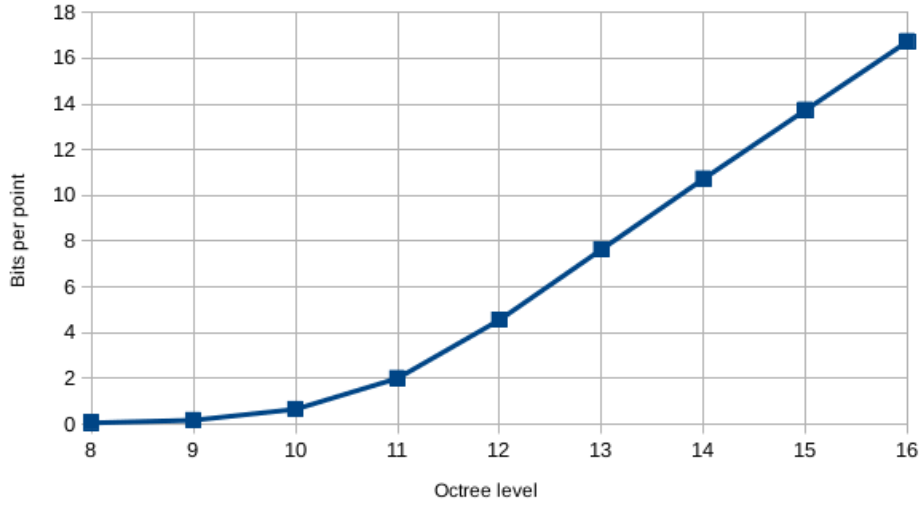

Figure 20: Bits per point at different octree levels.

Octree level	Total bits	Bits per point	Max. spatial error (mm)
1	7	2.87E-005	8660.254
2	32	4.40E-005	4330.127
3	150	0.00011614	2165.063
4	678	0.000438886	1082.531
5	2461	0.00152877	541.265
6	7961	0.00489071	270.632
7	24976	0.0152913	135.316
8	77884	0.047632	67.658
9	272457	0.166567	33.829
10	1055293	0.645085	16.914
11	3273142	2.00077	8.4572
12	7446706	4.55191	4.2286
13	12496061	7.63839	2.1143
14	17549047	10.7271	1.0571
15	22472791	13.7368	0.5285
16	27380578	16.7367	0.2642

Table 1: Compression results