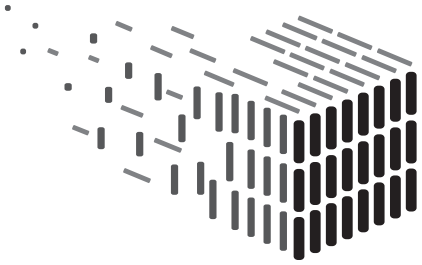


D5.6 Shape grammars for almost invisible structures

Software prototype v3

DURAARK

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

Date: 2016-01-31
Version 1.4
Document id. : duraark/2016/D5.6/v1.4

Grant agreement number	:	600908
Project acronym	:	DURAARK
Project full title	:	Durable Architectural Knowledge
Project's website	:	www.duraark.eu
Partners	:	FhA – Fraunhofer Austria [AT] CITA – KUNSTAKADEMIETS ARKITEKTSKOLE [DK]
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/D5.6/v1.4
Title of document	:	Shape grammars for almost invisible structures – Software prototype v3
Deliverable type	:	Software prototype
Contractual date of delivery	:	2016-01-31
Actual date of delivery	:	2016-01-31
Lead beneficiary	:	FhA
Author(s)	:	Ulrich Krispel <ulrich.krispel@vc.fraunhofer.at> (FhA) Martin Hecher <martin.hecher@vc.fraunhofer.at> (FhA) Martin Tamke <martin.tamke@kadk.dk> (CITA)
Responsible editor(s)	:	Ulrich Krispel <ulrich.krispel@vc.fraunhofer.at> (FhA)
Quality assessor(s)	:	Sebastian Ochmann <ochmann@cs.uni-bonn.de> (UBO) Mateusz Zwierzycki <mzwi@kadk.dk> (CITA)
Approval of this deliverable	:	Stefan Dietze <dietze@L3S.de> (LUH) – Project Coordinator
Distribution	:	Public
Keywords list	:	object detection, shape grammar, computer vision, point clouds

Executive Summary

This report presents the third software prototype of the tool RISE (**R**eveal almost **I**nvisible **S**tructur**E**s). This final deliverable D5.6 is part of WP5, "Recognition of Architecturally Meaningful Structures and Shapes". The software prototype is a semantic enrichment tool for detecting hidden structures like in-wall electrical appliances. The information of the found structures is provided in a form so that it can be stored in a long-term archival system. The technical approach is to use point cloud and image data from a laser scan to detect the visible parts of an electrical appliance (e.g. power sockets, light switches, etc.) via computer vision algorithms. The prior knowledge (i.e. constraints) of the placement of power lines is encoded as a formal grammar, which is used together with the detection results to generate a hypothesis for the invisible in-wall elements of the electrical appliances using a discrete optimization approach. We report the final improvements to the pipeline and the documentation of all the components.

Table of Contents

1 DURAARK RISE

The RISE component (**R**eveal **I**nvisible **S**tructur**E**s) is composed of a set of related modules for the detection of almost invisible structures; i.e. not immediately observable or deducible from point cloud data and images.

1.1 Motivation

A study conducted on the productivity of the construction industry in North America [4] shows that the industry is struggling with a lack of coordination, in particular the electrical construction companies. The study points at the implementation of Building Information Modeling (BIM) in these fields as a solution for this problem. Its application in this field would reduce conflicts and improve coordination. The study simultaneously points at little actual implementation of BIM in the electrical construction field. This is supported by the finding that 59% of the companies, who actually use BIM, have only three or less years of experience with this technique. Another survey conducted in the USA [3] gives a ranking of BIM features for this field, e.g. clash detections, visualization of electrical design or space utilization. However, 79% of the participating companies responded that they are not using BIM, with the main reasons being not knowing about BIM, lack of technological experience, software incompatibility, and implementation costs. The 21% that use BIM reported positive savings in time and cost. While these studies show that there is an interest in the utilization of BIM for electrical construction, this interest is currently not fulfilled in the area of as-built documentation and renovation projects - which makes for 75% of the EU building market share [2].

Past and current work practices of companies installing the electrical wiring of buildings use 2D drawings as a base for the work on site. These drawings contain the necessary information for workers to execute the work in the building, such as the position of

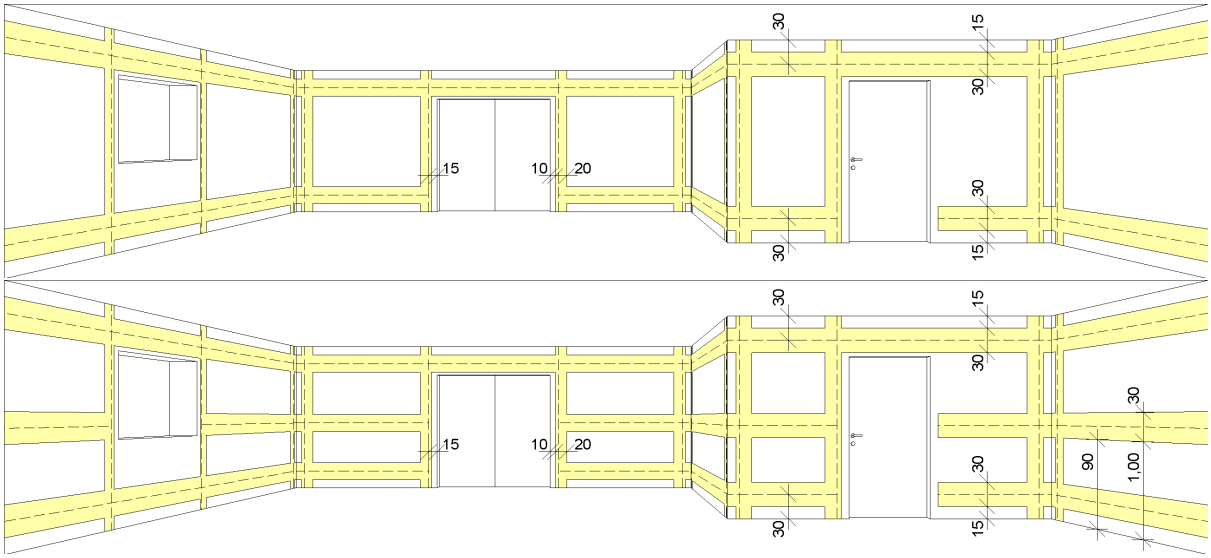


Figure 1: installation zones for residential buildings (top) and office rooms (bottom).
source: wikimedia commons

switches, fuse boxes, as well as the principal connection between these. The workers themselves know about the height, at which switches are to be mounted in the walls and the way that cables shall be laid. These rules are setup by national authorities and today harmonized on the European level. Technical standards, as the DIN 18015 in Germany (see also Figure 1), or the "Stærkstrømsbekendtgørelsen" in Denmark, specify so called installation zones, which should be preferred for routing of cables. Workers might adapt the routing to special features on local level, but will generally follow the concept of the norms.

This situation provides the base scenario of this work and other recent research, where the location of sockets in 2D floorplans of buildings were used to estimate the amount of copper wire in the walls – a valuable resource and a real asset in the evaluation of a buildings value before demolition [8]. This previous research utilizes standards to create a wiring hypothesis for estimating the copper amount. The prior knowledge is "hardcoded" in the algorithm, however, the norms may differ between countries, or may be revised. Therefore, the method presented in the *WireGen* component proposes to encode this

knowledge in a set of rules using a formal grammar system. For an extensive explanation of the grammar formalism used in the component we refer to the document D5.4. The grammar ruleset can be adapted to reflect changes in norms, or encode different norms, typically by a domain expert. Within RISE, the grammar is used to create installation zones for a given scan, and extract a wiring hypothesis from the graph of all possible connections within the generated zones.

1.2 Accessing the Software Prototypes

The RISE component consists of the subcomponents OrthoGen, ElecDetect and WireGen, which are accessible on the DURAARK Github repository. The integration into the Service Platform is done in the Geometric Enrichment Service.

OrthoGen

Type:	C++ Component
License:	BSD
Description:	D5.4, Section 2.5
Source code:	http://github.com/DURAARK/orthogen

ElecDetect

Type:	C++ Component
License:	BSD
Description:	D5.2
Source code:	http://github.com/DURAARK/elecdetect

WireGen

Type:	NodeJS Component
License:	BSD
Description:	D5.4, Section 2.6 & D5.6
Source code:	http://github.com/DURAARK/wiregen

Geometric Enrichment Service

Type:	Web service
License:	MIT
Description:	D2.5, Section 4.4
API documentation:	http://data.duraark.eu/services/api/geometricenrichment
API endpoint:	http://data.duraark.eu/services/api/geometricenrichment
Source code:	http://github.com/DURAARK/microservice-geometric-enrichment

1.3 User Manual

A user interface for the RISE component is available within the WorkbenchUI, which uses the underlying Geometric Enrichment service to access RISE's functionality. This section provides a user manual for the RISE component.

To access the WorkbenchUI open the webpage http://workbench.duraark.eu. From there select *Open Building* to open a list of available demo buildings, as depicted in Figure 2. From the list select *Bygade 72 - 2nd scan*.

The selected building is opened in the WorkbenchUI, initially showing the *Files* section which lists the 3D files currently stored for this building (see Figure 3). From here go to the *Geometric Tools* which is located in the header of the page.

After selecting the *Geometric Tools* section the screen shows a *Pointcloud (E57)* file. Click on the + sign below the file to see the geometric tools available for the *Pointcloud (E57)* file. From the list appearing on the right switch on *Detect Power Lines*, which is the human friendly name for the RISE component. Next to the *Pointcloud (E57)* file a new tile appears, which means that the detection of the in-wall electrical appliances is started in the background. When the processing is finished the item's header is switching from *Processing ...* to *Show Result*, as depicted in Figure 4.

Click on the *Show Result* header to inspect the result on the appearing result widget on the right side of the screen. For a better user experience we recommend to click the fullscreen button on the result widget before continuing. After that the widget shows a floor plan of the building (which was also extracted from the pointcloud file) on the left. Double clicking on any of the rooms opens the room as a 3D preview with the detected in-wall cables in red. Multiple rooms can be shown at the same time when double clicking on additional rooms on the floor plan. The result should look similar to Figure 5.

Scrolling down the result widget also shows each wall of the currently selected room with the result of the in-wall cables, the installation zones defined by the shape grammar and the original images taken from the point cloud scanner as depicted in Figure 6.

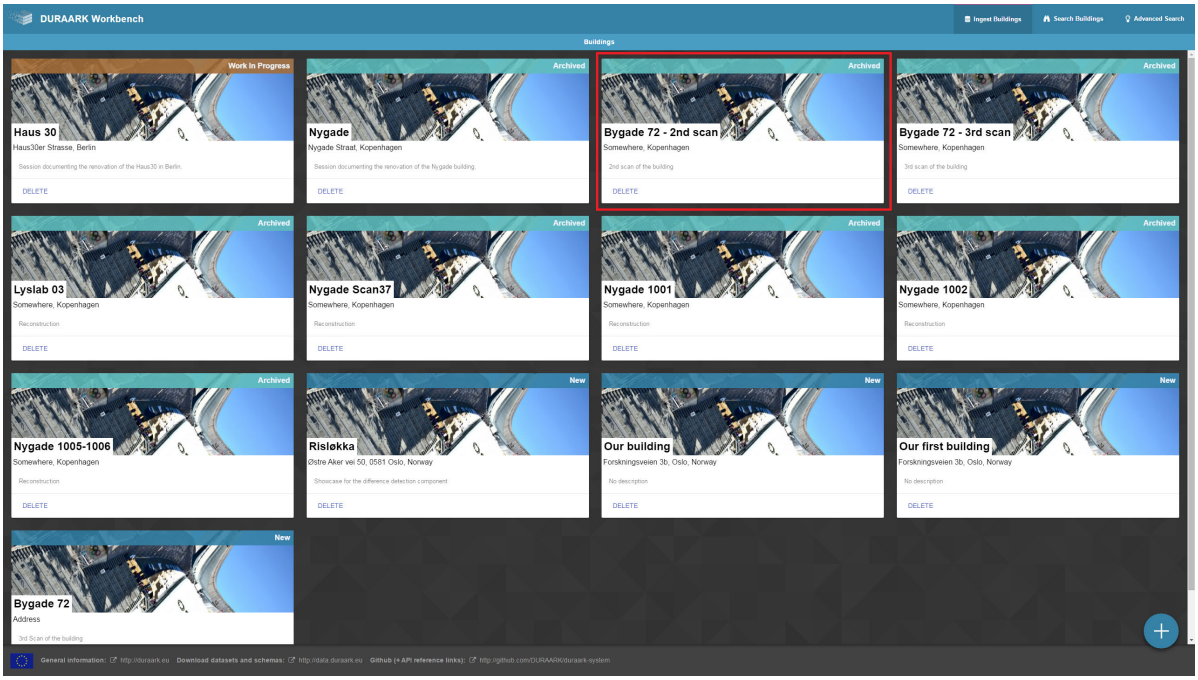


Figure 2: Dataset selection in the WorkbenchUI. For this manual please select *Bygade 72 - 2nd scan*.

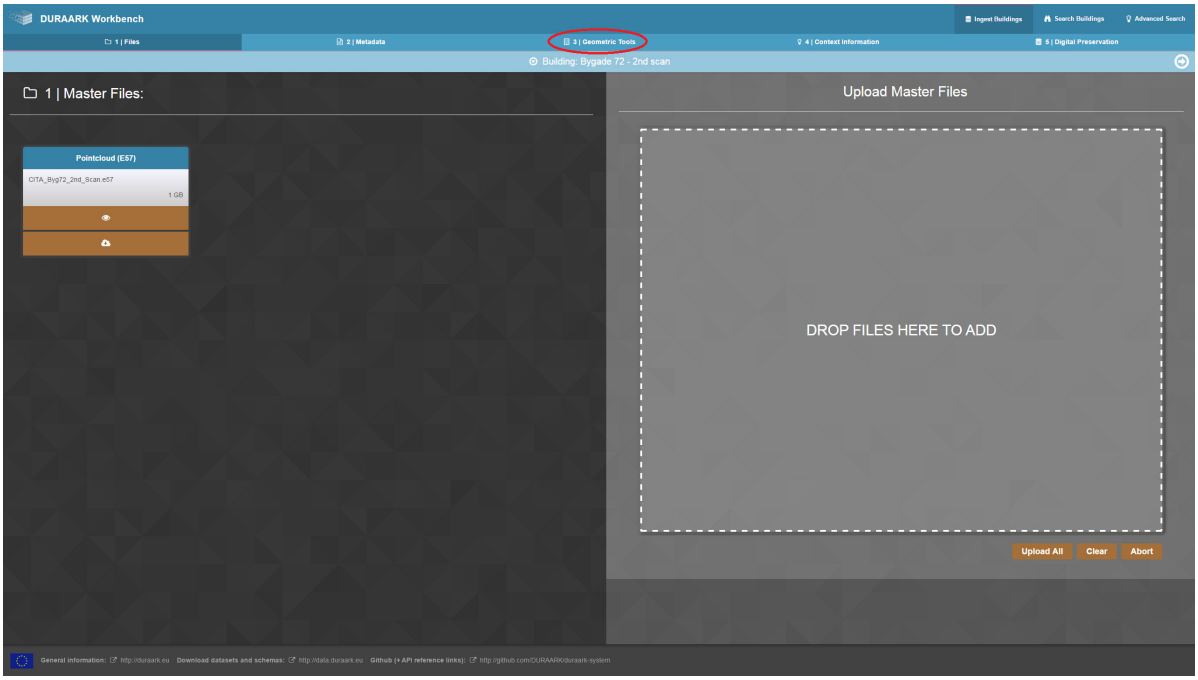


Figure 3: *Files* section showing currently stored 3D files. Proceed to the *Geometric Tools* section from here.

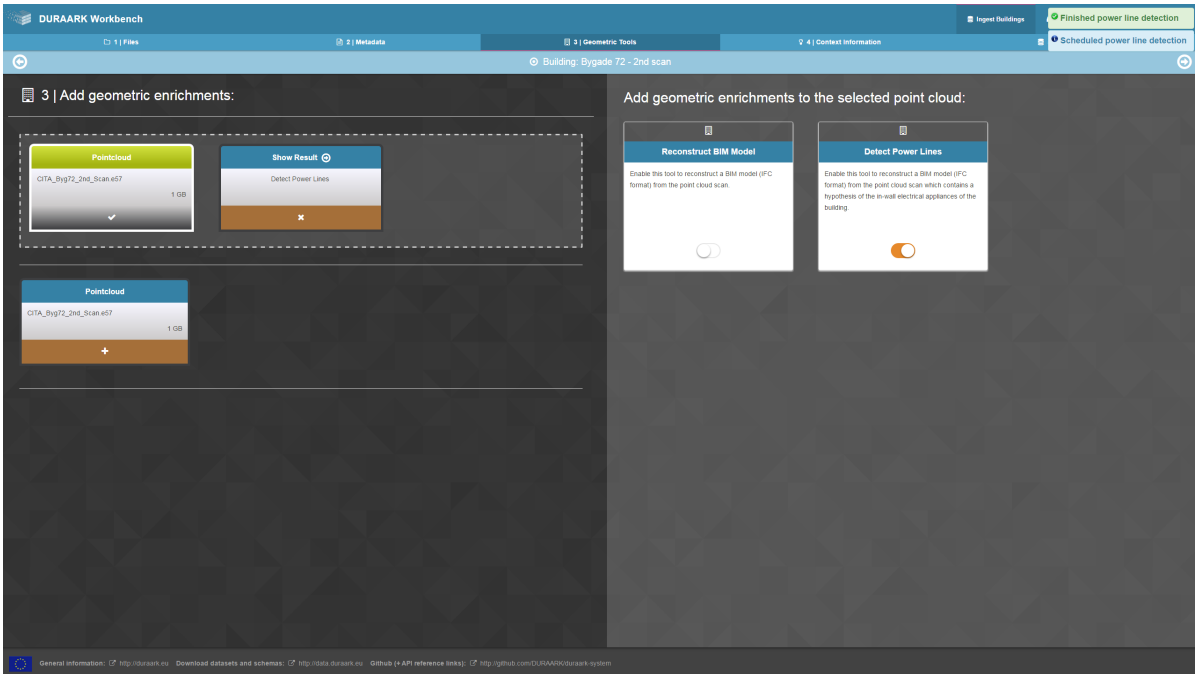


Figure 4: Screenshot after a successful detection of the in-wall electrical appliances.

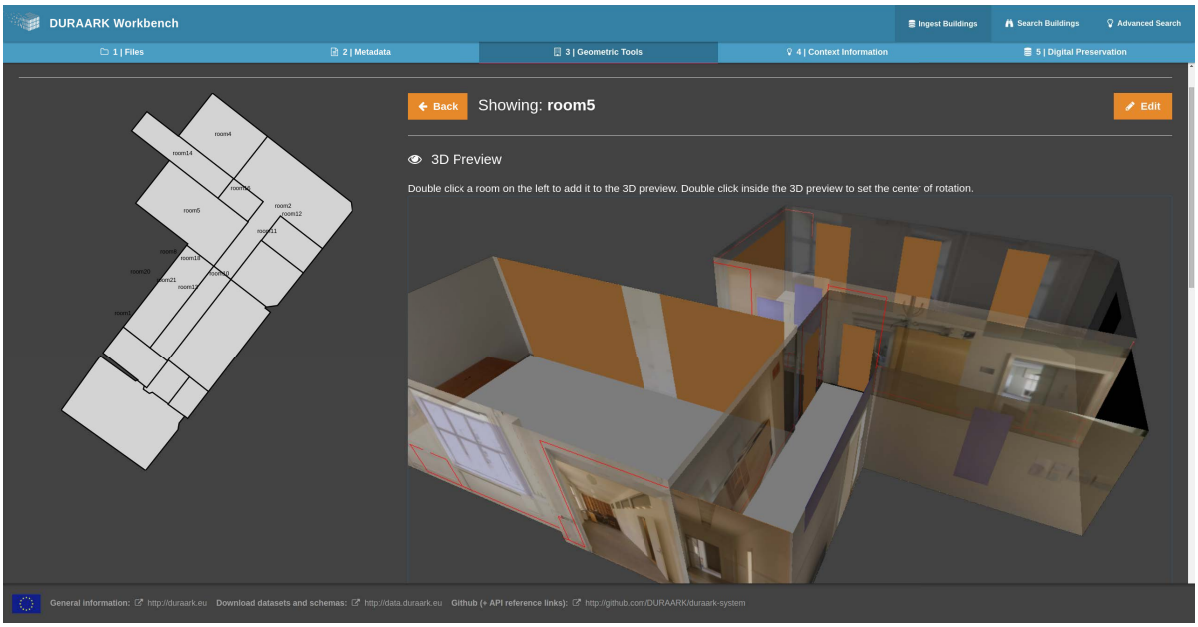


Figure 5: Screenshot showing the 3D preview of multiple rooms with the in-wall cable detections in red.

Figure 6: Screenshot of the 2D wall previews for the currently selected room.

2 Components

2.1 Integration into the WorkbenchUI

The functionality of the RISE component is directly integrated into the DURAARK Service Platform via its geometric enrichment service API[1]. The documentation of the API can therefore be found as part of the geometric enrichment service API documentation. The WorkbenchUI includes a graphical user interface to RISE, as was described in the previous section.

2.2 Overall Architecture

The overall architecture of the RISE component is shown in Figure 7. The application consists of the following modules:

- *Geometry Extraction*, which extracts coarse geometry and adjacency information from the input point clouds,
- *OrthoGen*, which creates an orthographic image per wall segment, given wall information and panoramic images taken from the scanning positions,
- *ElecDetect* performs detection of visible parts of electrical appliances (power sockets, light switches),
- *WireGen* synthesizes a hypothesis for a highly probable wire routing inside the walls,
- *Rise2X3D* injects the results are back into the IFC for output, or converts it to X3D for visualization in the geometric enrichment service

[1] https://github.com/DURAARK/duraark-geometricenrichment

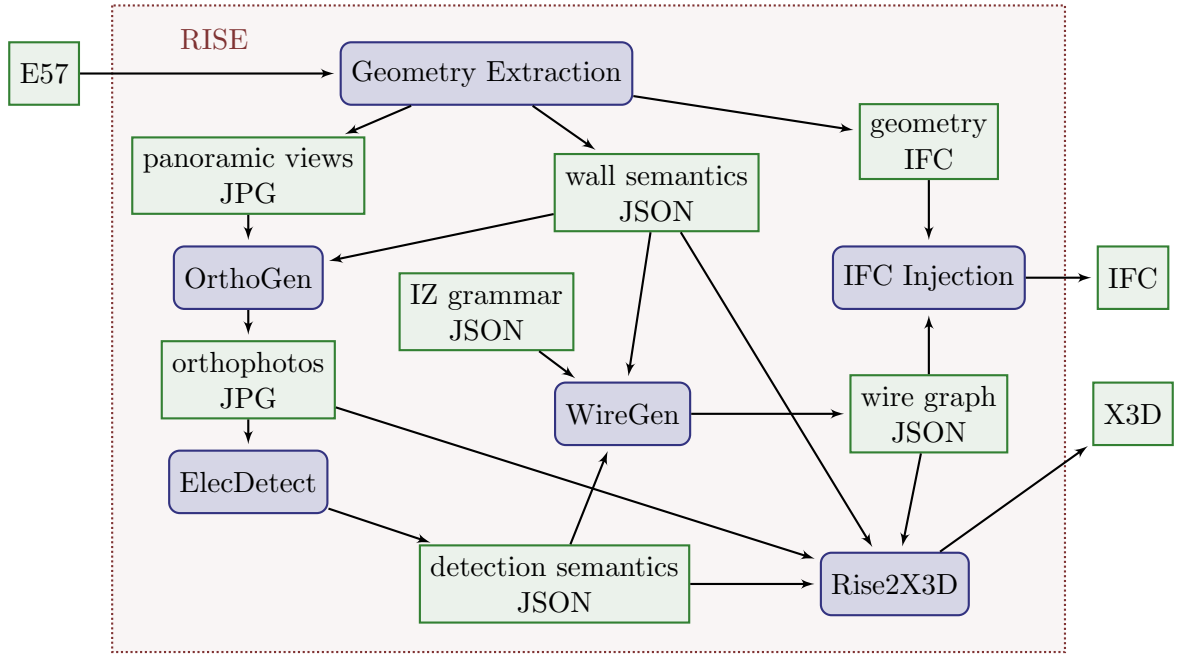


Figure 7: Overall Architecture of the RISE component. Modules (processes) correspond to blue nodes with rounded corners, and data (e.g. intermediate results) corresponds to green nodes.

2.3 Geometry Extraction

The Input E57 File contains registered scanned data, plus any panoramic images that have been acquired while scanning, either directly from the scanner or manually by stitching a panoramic sphere from images taken at the scanner position.

In the geometry extraction stage, proxy geometry that corresponds to a coarse room layout is extracted from the point clouds. Furthermore, position of walls and their adjacency relations are extracted and encoded in a *wall semantics* JSON file, which is automatically created by the IFC extraction component (D5.5).

Wall Semantics JSON Format The data directly serves as input symbols for the shape grammar evaluation in Wiregen. It consists of an object that contains an array for a specific class of geometric instance, namely *Walls* and *Openings* for positions of windows and doors.

Positional information is encoded in the wall coordinate system (origin top left) described in D5.4, Section 2.6.3. An exemplaric wall entry would be

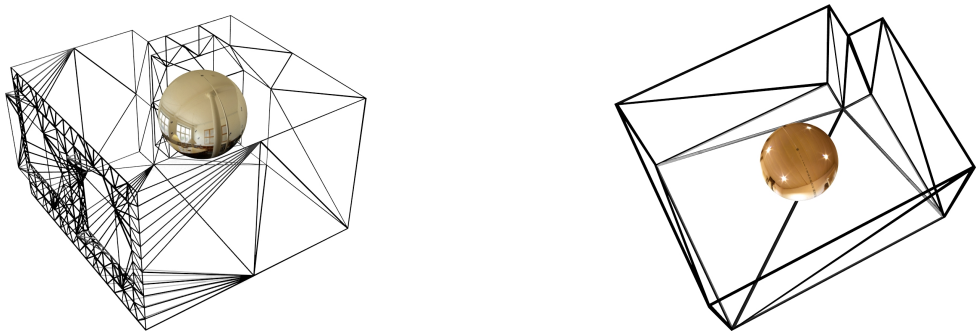
```
{
            "label": "WALL",
            "attributes": {
                "id": "wall0",
                "width": 2330.40234375,
                "height": 2294.59521484375,
                "thickness": 200,
                "origin": [ -179.1, -2182.8, 142638.6 ],
                "x": [ 0.8, 0.6, 0 ],
                "y": [ 0, 0, -1 ],
                "roomid": "room0"
            },
            "left": 0,
            "right": 5,
            "crosslink": []
        }
}
```

Id specifies the unique wall identifier, *width*, *height* specify the wall size in wall coordinates, *origin* specifies the top left point (as seen from inside the room) of the wall in world coordinates, and x and y specify the wall coordinate system directional vectors as described in D5.4, Section 2.6.3. Furthermore, walls are associated to a room via *roomid*. Adjacency information is encoded with *left* and *right*, which specify a connection point to other walls in this room (e.g. wall0 is connected left to point 0, and wall4 is connected right to point 0). Any walls that are connected to this walls by other means (e.g. an orthogonal wall in another room on the backside of the wall) are specified via the *crosslink* list.

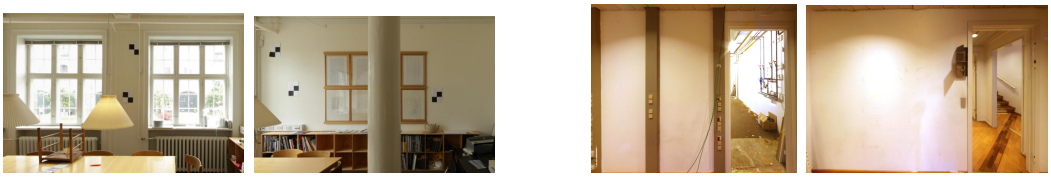
2.4 OrthoGen

The *OrthoGen* module is responsible to extract orthographic views that correspond to the wall segments in the input data. The basic method is that the module is supplied with panoramic spheres, acquired at the scanning positions, and proxy geometry that

(a) Panoramic Views



(b) Proxy Geometry



(c) Exemplary extracted Ortho Views

(d) Textured 3D model

Figure 8: This Figure shows the input and output of the *OrthoGen* module on two datasets (left and right column). *OrthoGen* extracts orthographics views (c) from panoramic images (a) and proxy geometry (b). These orthophotos can also be used to texture the proxy geometry (d).

corresponds to wall segments, expressed as rectangular patches in 3D. For each patch, the corresponding color information is projected from the panoramic sphere. See also Figure 8. The method was published in *Automatic Texture and Orthophoto Generation from Registered Panoramic Views* [7].

For the updated D5.6 version, *OrthoGen* now is able to integrate the information from multiple scans. In contrast to the D5.4 pipeline, *OrthoGen* now directly reads the E57 metadata file to extract all scanning positions, and the wall JSON file. As the wall JSON file contains the information which walls correspond to which rooms, *OrthoGen* first associates each panoramic image to a room, and considers all panoramic images of a room for each wall of that room. For every pixel on a generated orthophoto, the *nearest* scan, i.e. the scan with shortest distance from the scanning position to that pixel, is used to project to this pixel. Therefore, the orthophotos contain the information with highest resolution wherever possible from the obtained data.

2.4.1 Commandline Interface

The commandline interface of the application is shown in Table 1. It requires as mandatory input

- a JSON file that contains the e57 metadata information from pointcloud scans
- a JSON file that contains the wall and room adjacency information from the IFC reconstruction
- a path to the panoramic images

Orthogen assumes that at least one panoramic image called

`<scan_name>_faro.jpg`

`--help`	show help message
`--im arg`	(required) input panoramic image [.jpg]
`--help`	show this help message
`--align arg`	align executable
`--e57metadata arg`	e57 metadata json [.json]
`--walljson arg`	input wall json [.json]
`--exgeom arg`	export textured geometry as .obj [0]/1
`--exroombb arg`	export room bounding boxes as .obj [0]/1
`--output arg`	output filename [.jpg] will be appended
`--panopath arg`	path to pano images
`--resolution arg`	resolution [1mm/pixel]
`--scanoffset arg`	translation offset
`--usefaroimage arg`	use pano from faro scanner
`--verbose arg`	print out verbose messages
`--scan arg`	if specified, only this scan will be considered
`--wall arg`	if specified, only this wall will be considered
`--bbfitnormalprecision arg`	normal encoding precision for oriented bounding box fit for rooms [8]

Table 1: All *OrthoGen* commandline options

is available per scan at the specified path. Optionally, if a high resolution panoramic image called

`<scan_name>_aligned.jpg`

is available, it will be used for orthophoto projection. If there exists an image called

`<scan_name>_Manual.jpg`

in the pano directory, *OrthoGen* will try to execute an automatic panorama alignment using the executable *panoalign* that is also part of the *OrthoGen* project.

2.5 ElecDetect

The *ElecDetect* module performs detection of relevant observable electrical appliances: instances of sockets and switches. In a preprocessing step, a classifier is trained with

the visual appearance of the desired object classes. When performing the geometric enrichment with RISE, the module is used to detect instances of these classes on the orthophotos generated with *OrthoGen*. A detailed description of the contents of this module, as well as its related work were given in deliverable D5.2 - Shape grammars for almost invisible objects, software prototype v1. This component has not been changed for M36.

2.5.1 Commandline Interface

The commandline interface to the application is shown in Table 2.

`-m, --mode`	(required): execution mode, which can be either "train" for training mode, or "detect" for detection mode
`-c, --config`	(required): relative path of the XML configuration file that is created in training mode or read on detection mode
`-d, --dir`	(required): the image directory containing the training patches at training or the query images, depending on the mode
`-i, --ini`	(optional): relative path to the ini file that specifies application dependent settings. Per default, the program uses the config.ini file.

Table 2: The *ElecDetect* commandline options.

2.6 WireGen

The *WireGen* module performs a wire routing hypothesis, given the observable endpoints of electrical appliances. To be able to generate a reasonable hypothesis, meaningful assumptions about the structure have to be made. The assumptions made in this module are twofold: first, we assume that planning and construction of the building has been made with respect to a standardization that applies to building construction, and second, that the planning was carried out with minimizing material cost, i.e. the length of wirings in walls.

The prior knowledge that is defined by the standardization is represented by a set of rules, which facilitates the usage of different rule sets for different types of standardization or prior knowledge, e.g. standards in different countries, or standards that applied at different times. The methodology used for this component was submitted for publication and is currently under review [6].

The grammar presented in D5.4 (see also the appendix section of the D5.4 document) was context free, as the rules stated by the standards are also context free. However, after applying the method to some real word data it was observed that installation zones can also be placed at positions where the measurements suggest the existence of a zone, i.e. at horizontal or vertical arrangements of endpoints.

Therefore, the grammar was extended to context sensitive rules that allows pairs of non-terminal symbols on the left side of a production rule (written "NT1:NT2", as extension to the labels described in D5.4), with an optional attribute condition expression. Pair rules are always prioritized until no pair of symbols matches, context free rules are matched and executed afterwards. This allows us to formulate rules that group horizontal and vertical arrangements together; groups that exceed a specific size will generate an additional installation zone, as can be seen in Figure 9.

2.6.1 Commandline Interface

The application has the commandline arguments

`-i, --input`	(required): Set room layout input symbols created by the first rule (JSON)
`-g, --grammar`	(required): Set Installation Zone Grammar (JSON)
`-o, --output`	(optional): Set output directory.
`-p, --prefix`	(optional): Orthophoto prefix string.

Table 3: The *WireGen* commandline options.

2.7 Rise2X3D

For visualization purposes, an X3D export module was implemented. X3D is an ISO standard for declarative representation of 3D computer graphics [5] based on XML, i.e. a file format. Furthermore, X3D can be embedded directly into webpages using x3dom [1].

The exporter is integrated into the geometric enrichment service, and creates a textured 3D model containing floor and wall geometry, combining the following information:

- wall and floor geometry from a JSON file
- low resolution orthophotos as textures
- *ElecDetec* detection results as colored rectangles
- final wiring hypothesis as colored line segments

An example for the Nygade1001 dataset can be seen in Figure 11.

2.8 Results

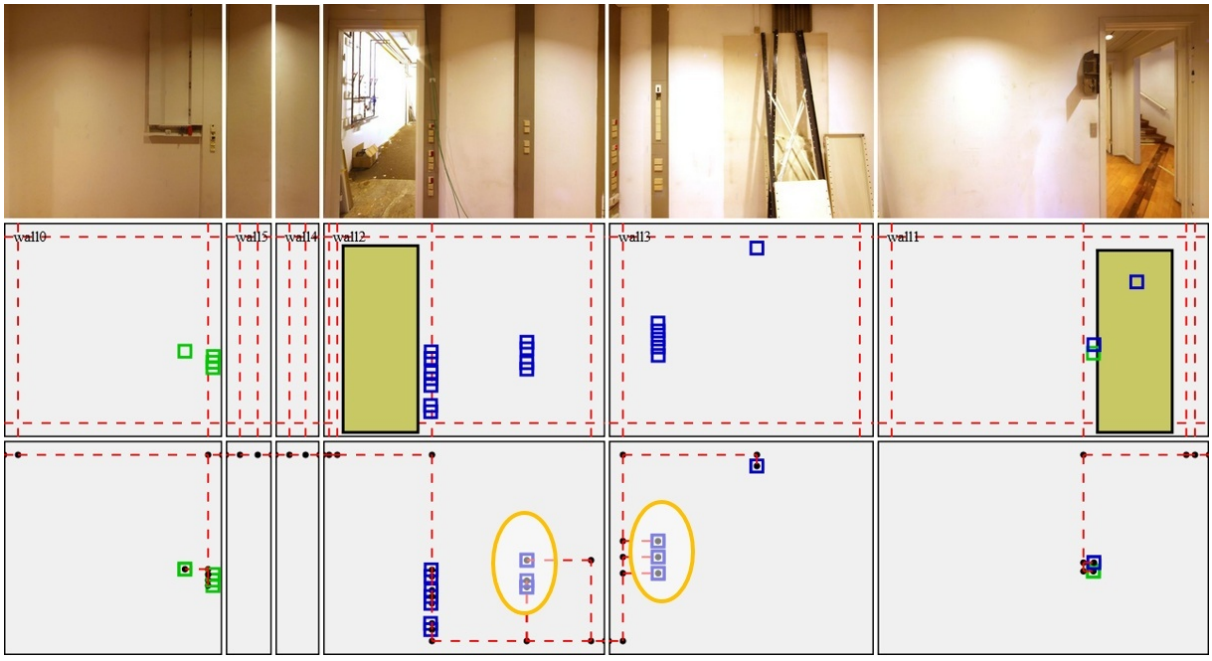
We demonstrate the application of each pipeline step using the Nygade dataset (top row in Figure 13). The orthophoto extraction using *OrthoGen* yields 6 wall elements (Figure 12a), the electrical appliance detection using *Elecdetect* finds 23 elements (sockets and switches). Using the installation zone grammar, *WireGen* produces installation zones (Figure 12d) and extracts the final wiring hypothesis (Figure 12e).

The dataset is also a showcase in the WorkbenchUI and can be inspected by starting the session *Nygade*.

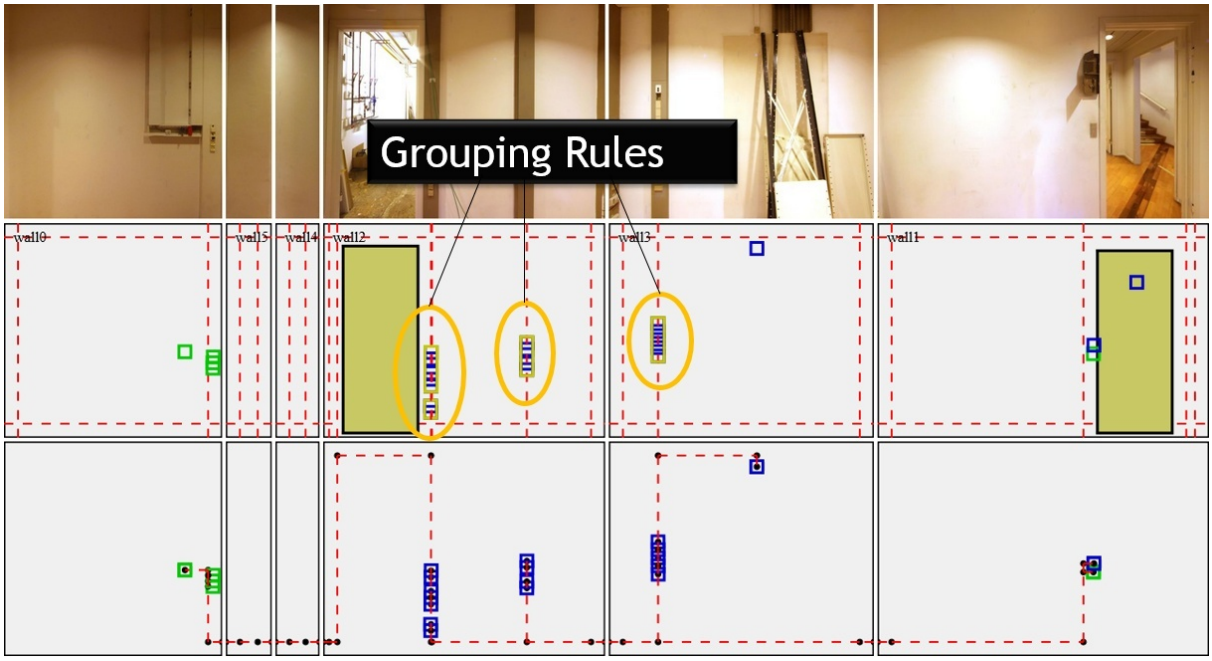
The results on this dataset are satisfactory and demonstrate the overall functionality of the pipeline. It should be noted that, as is expected for any pipeline structure, that errors

made in early stages of the pipeline propagate through the chain, as it was observed that problems at the measurement stage (e.g. out of focus / blurred photographs) will lead to wrong detections, and therefore a wrong input state for wiregen. This also happens with false positives, where the detection finds objects looking similar to sockets or switches, as was the case with the lyslab03 dataset.ffl Furthermore, while the openings detection works quite well, in some cases wrongly detected openings would lead to unroutable configurations for some detections. The optimization tries to connect such detections as isolated groups in this case.

Therefore, we do see applicability of the approach not in a complete automatic pipeline, but an approach that allows manual intervention at different stages of the pipeline. For example, correcting the results of the electrical endpoint detection, or inserting a window or door that was closed while scanning, and therefore not detected by the IFC extraction component. Furthermore, it seems that the installation zone grammar is a good possibility to specify building specific data, therefore it would also make sense to allow building-specific editing of the grammar rules up to the user.



(a) context free grammar



(b) context sensitive grammar

Figure 9: Using a context free grammar allows to create installation zones only at positions as suggested by the standards, i.e. minimum distances from windows and doors, which might lead to incorrect connections (a). Using a context sensitive grammar with grouping rules that identifies horizontal and vertical arrangements of endpoints allows to suggest additional installation zones at these positions (b).

Figure 10: The *ElecDetect* module performs the detection of sockets and switches (right) on orthophotos generated by the *OrthoGen* module (left).

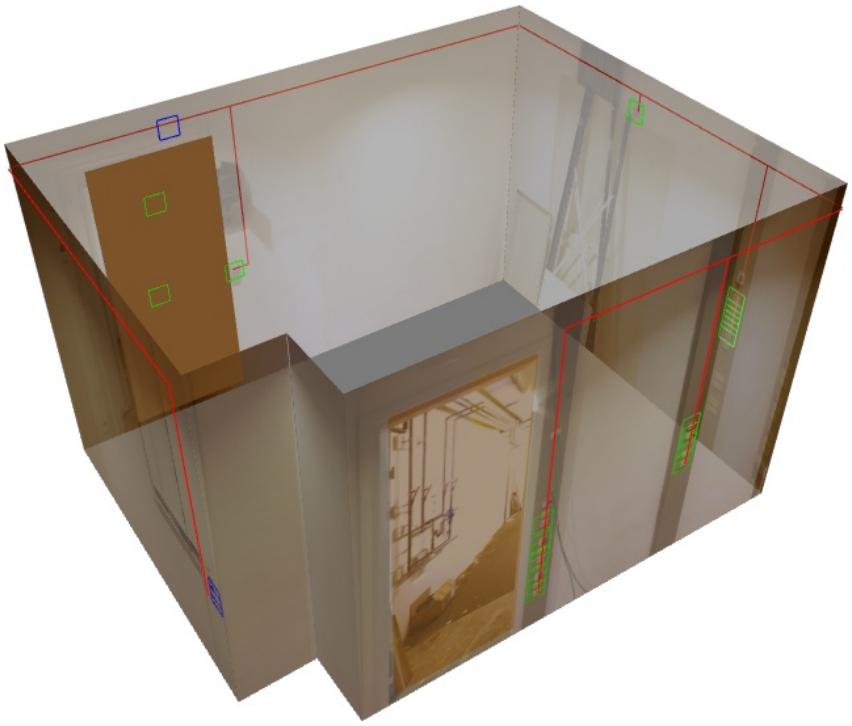
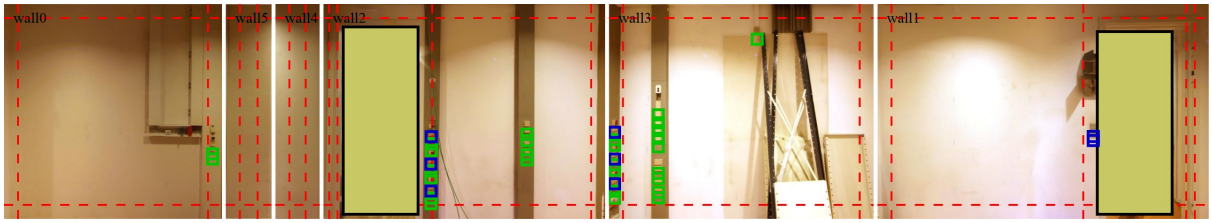
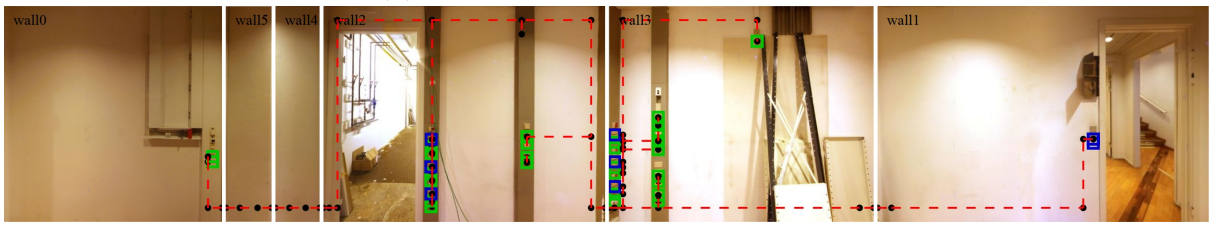


Figure 11: The result of the X3D export can be displayed in a WebGL capable web browser. Detections are shown as blue and green rectangles, and the wiring hypothesis as red lines.

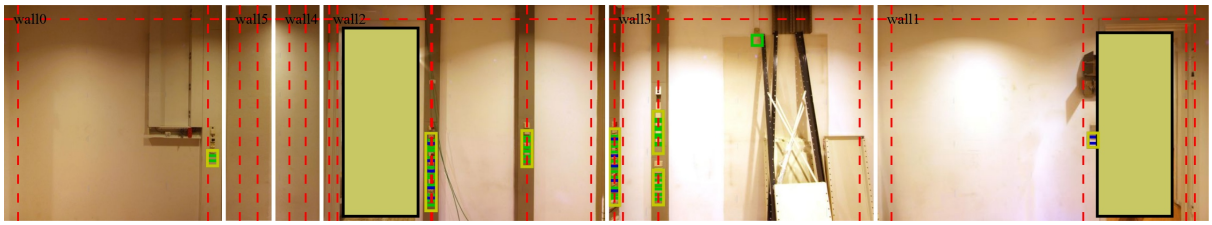
(a) Groundtruth



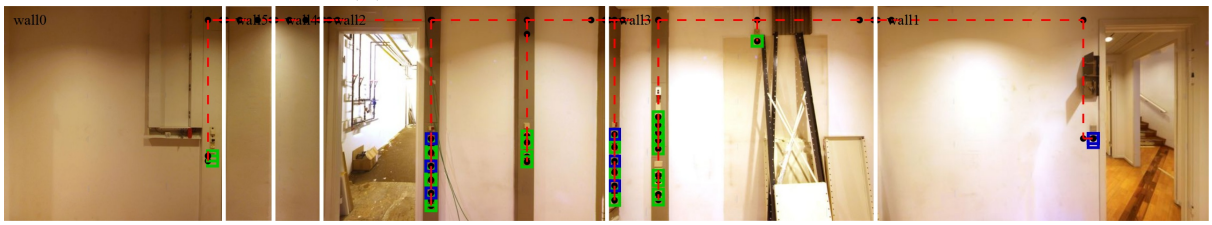
(b) Context free installation zones



(c) Context free hypothesis



(d) Context sensitive installation zones



(e) Context sensitive hypothesis

Figure 12: A groundtruth for nygade1001 has been manually acquired (a). A context free grammar (b) yields wrong results (c) in regions where groups of sockets and switches suggest an additional zone. A context sensitive grammar (d) that creates additional zones where groups are found (yellow rectangles) yields a correct result (e).

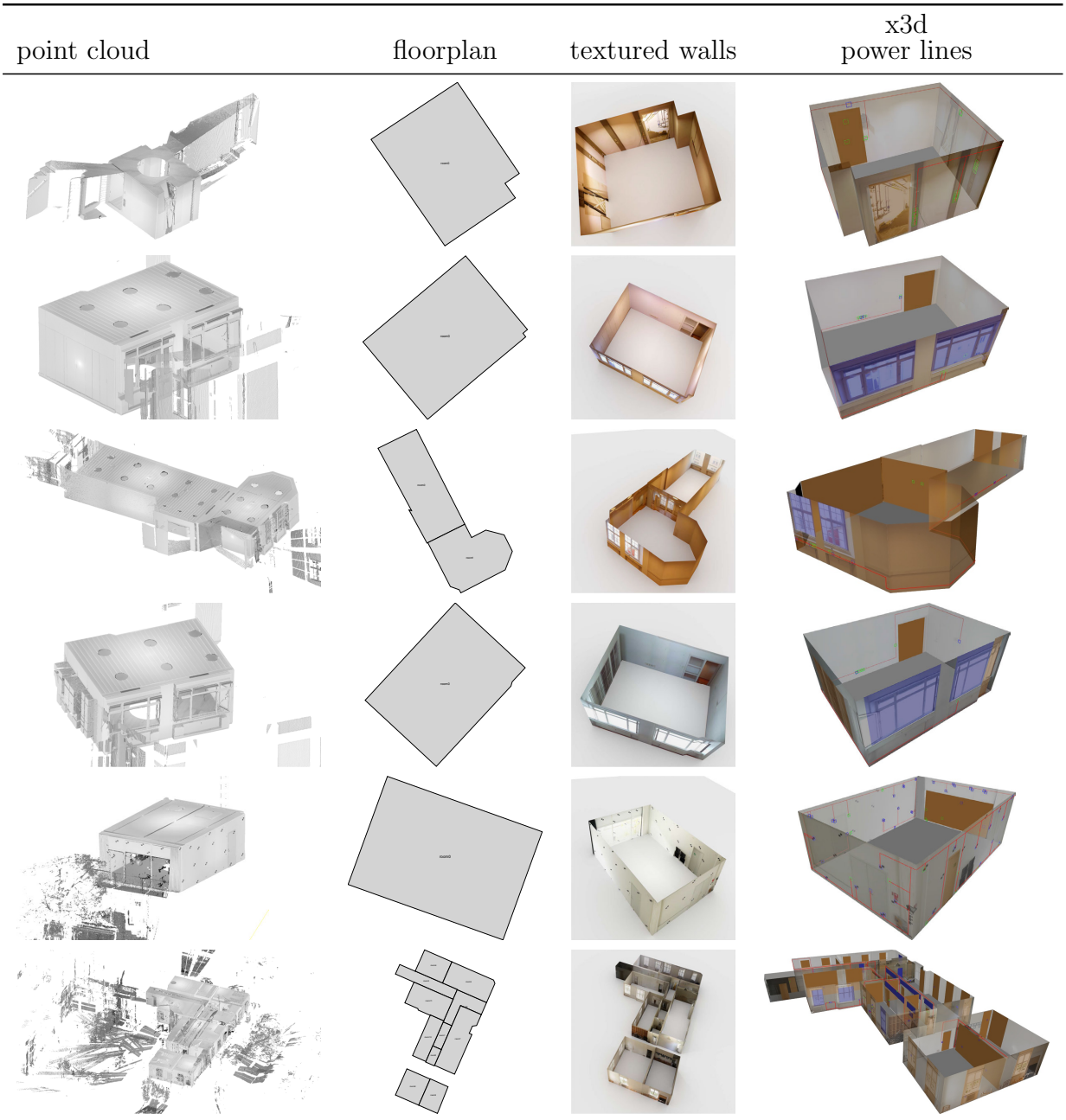
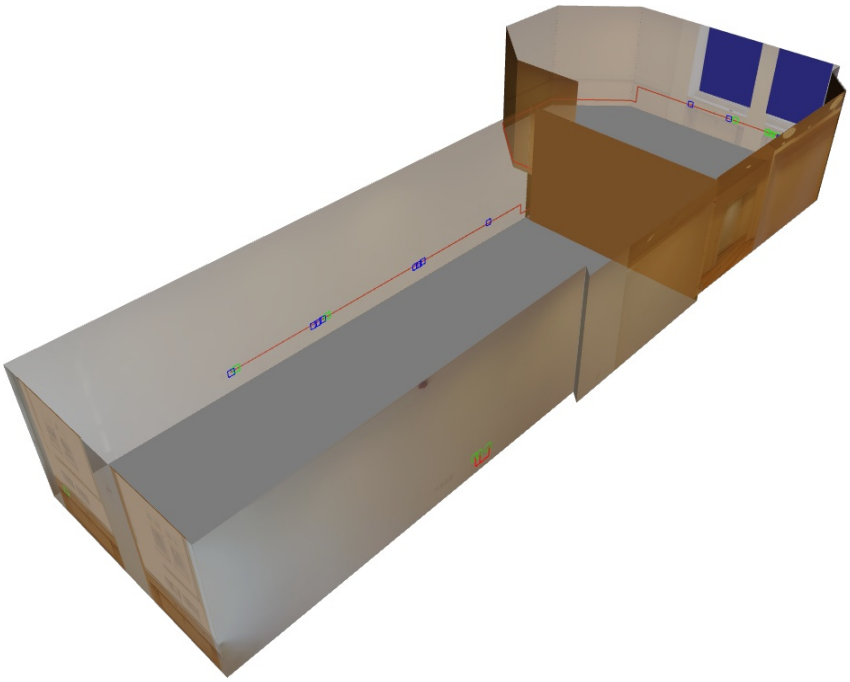
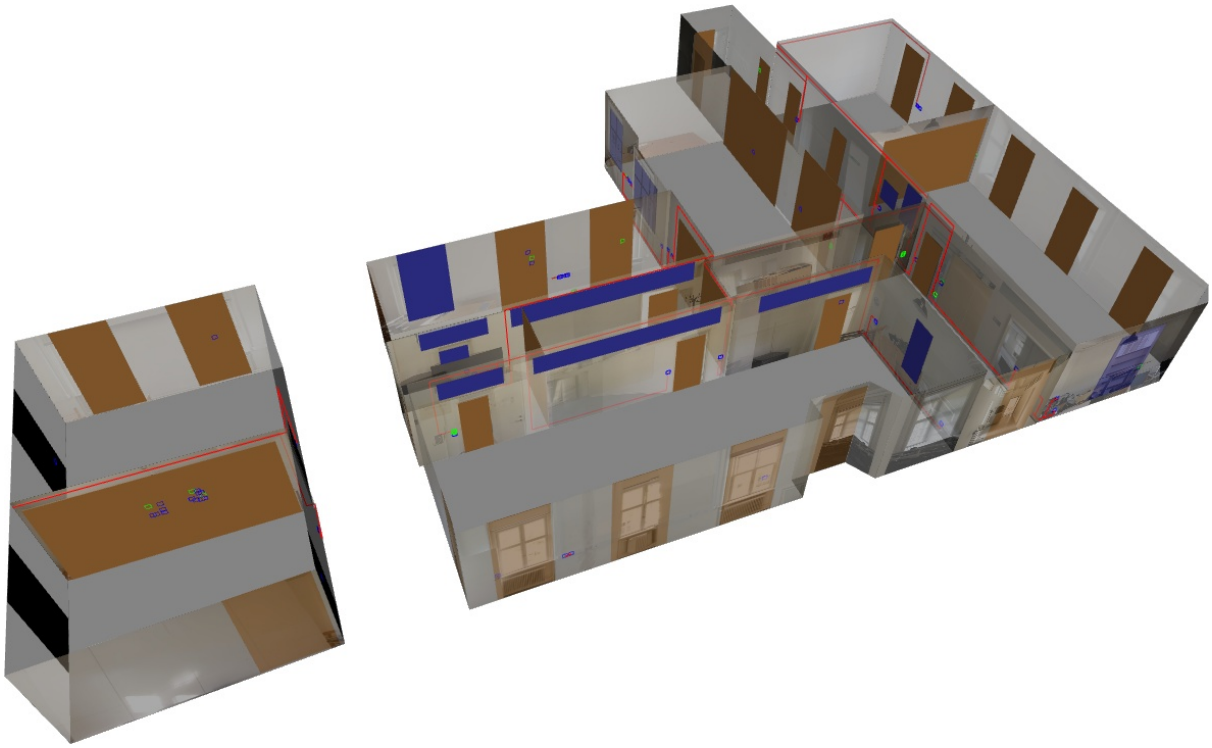


Figure 13: All 6 datasets that have been processed. From top to bottom: nygade1001, nygade1002, nygade1005-1006, nygade037, lyslab03 and bygade72-2ndscan.



(a) Nygade1005-1006



(b) Bygade72

Figure 14: Resulting X3D visualizations for datasets Nygade1005-1006, which consists of 2 scans, and Bygade72, which consists of 14 scans.

3 Decisions & Risks

3.1 Technical decisions and impacts

Pipeline-based Architecture

The pipeline based approach chosen and motivated in D5.2 proved to be suitable for the chosen approach. Furthermore, the well-defined interface between the modules in the pipeline allows to reuse intermediate results in other parts of the pipeline (e.g. visualizing the electrical appliance detections in orthophotos).

Grammar-based approach

Using a formal language to encode the prior knowledge has the advantage that the system is adaptive and should be ready for incoming future changes when more types of prior knowledge should be incorporated. Future additions should be possible without changing the system or the optimization method, but rather by providing a new installation zone grammar.

WireGen Implementation

WireGen is implemented using the server framework "NodeJS". The problem representation and optimization method was fully implemented without the use of any third party library, which eliminates the need for special licenses. Furthermore, as the Geometric Enrichment Service is also implemented in NodeJS, the component integrates seamlessly into the service platform.

3.2 Risk assessment

This section lists the discussed technical risks, consequence and treatment action:

Risk Description The prior knowledge encoded in the shape grammar is not sufficient for the stakeholders needs.

Risk Assessment

Impact Low

Probability Medium

Description The method used to synthesize the structure of almost invisible objects (power lines) naturally requires some assumptions with respect to the hidden structures. These assumptions are encoded into the grammar, and reflect a method, or standards/specifications that guide the hypothesis. The DURAARK workbench provides a few predefined rule sets that reflect some standards, (e.g. DIN), but a stakeholder might want to create a hypothesis using a different standard.

Contingency Solution As the WireGen component allows to encode the prior knowledge in grammar rule sets, the stakeholder just has to create the necessary rules that create installation zones using the desired method. While the rule editing consists of editing textual descriptions in this prototype, it should be possible to incorporate the rule editing in a graphical editor in an accessible manner for the user, as the rules have strong two-dimensional relations.

---

4 Software Licenses

The following table gives an updated overview of the software licences generated and used for the RISE components. The services are written in C++ or Javascript with the server framework "NodeJS", which is a very common combination in the web development world. The majority of open source projects which are using the same technology stack as DURAARK is preferring the MIT license, as it is a very permissive and community friendly license. The components of RISE use MIT whenever possible, and BSD when necessary, which is also a permissive license.

IPR Type	**IP used or generated**	**Software name**	**License**	**Information**
software	generated	ElecDetect	BSD	D5.2
software	generated	OrthoGen	BSD	D5.4
software	generated	WireGen	BSD	D5.4
software	used (ElecDetect)	OpenCV	BSD	http://opencv.org
software	used (ElecDetect)	tclap	MIT	http://tclap.sourceforge.net
software	used (ElecDetect)	tinyxml	zlib	https://github.com/leethomason/tinyxml2
software	used (OrthoGen)	Boost	Boost Software License	http://www.boost.org/
software	used (OrthoGen)	Eigen	MPL2	http://eigen.tuxfamily.org/

5 Conclusions & Impact

This report presents the third version of the software prototype D5.6 in form of the RISE component and its modules *ElecDetect* (D5.2) that is used to detect endpoints of electrical appliances, *OrthoGen* (D5.4), which is used to create orthographic views, given panoramic images and proxy geometry, as well as the improved *WireGen* (D5.4 and D5.6), which, given a room layout, detections of sockets and switches and an installation zone grammar, synthesizes a most probable routing of the electrical wiring inside walls.

5.1 ElecDetect

This component has not been changed.

5.2 OrthoGen

OrthoGen improvements addressed stability and the ability to combine information from several scans to achieve better orthophoto projection if multiple scans are available per room.

5.3 WireGen

This component has been improved by an context sensitive grammar that creates installation zones at places where the measuerements suggest them, i.e. horizontal or vertical arrangements of sockets/switches.

5.4 Impact

By using the RISE component, a stakeholder will be able to create a hypothesis for wire routings from the measurements (point clouds, panoramic photos). The system can

also be used to generate the position of installation zones, by making the intermediate results of the installation zone grammar evaluation available. This may be helpful in planning scenarios, as a planner could be presented with a 3D view of possible installation zones, given a scanning of a building. The grammar approach seems to be a viable approach to represent prior knowledge, however, there is no single grammar that captures all information. Rather, we believe that a grammar that captures regional norms and regulations for a building serves as an starting point. Based on that, a domain expert gradually refines the grammar to match the actually applied installation zones for during the construction of the building. Experiences with the tested datasets and with talks to stakeholders showed, that norms and regulation are often adapted for specific buildings and therefore deviating from the official specs. By adding additional interfaces in the future to CAD standards, or planning systems,the installation zone hypothesis could also be created for planned-only models, which opens a greater market exploitability.

5.5 Sustainability

The software is fully integrated in WP2 and will be publicly available at `http://workbench.duraark.eu` beyond the project period. The source code of the components *OrthoGen*, *ElecDetect*, and *WireGen* is publicly available on GitHub (see Section 1.2). FhA has a strategic interest in continuing the development of the WireGen component, due to the thematic proximity to similar projects [9] involving the use of shape grammars as prior knowledge representation.

5.6 Future Work

The pipeline has been completed and works with the presented precision for the acquired sample datasets. Naturally, there are technical improvements possible in any parts of the

pipeline.

As there have been vast improvement in the machine learning sector in the recent years, the *ElecDetect* pipeline could be improved by using a more recent algorithm, as this component was developed in the first year. For *OrthoGen*, the depth information that is available from the single scans could be used to create a depth image for each high resolution panoramic image, and perform true ortho projection using this information. The *WireGen* component could be enhanced by a visual editor of the grammar rules, to make it more accessible to end users. Another interesting point of future research is the incorporation probability values for installation zones over the whole wall.

References

[1] x3dom http://www.x3dom.org/.

[2] B. Atanasiu, C. Despret, M. Economidou, J. Maio, I. Nolte, and O. Rapf. Europe's buildings under the microscope - a country-by-country review of the energy performance of buildings. *Buildings Performance Institute Europe (BPIE)*, 2011.

[3] S. Azhar. Bim for electrical construction: Benefits and current trends. *JBIM*, Fall:28–29, 2009.

[4] A. S. Hanna, M. Yeutter, and D. G. Aoun. State of practice of building information modeling in the electrical construction industry. *Journal of Construction Engineering and Management*, 140(12):05014011, 2014.

[5] X3d v3.3 abstract specification, iso/iec is 19775-1:2013, http://www.web3d.org/standards, November 2013.

[6] U. Krispel, H. L. Evers, M. Tamke, and T. Ullrich. An automatic hypothesis of electrical lines from range scans and photographs (under review). In *16th International Conference on Computing in Civil and Building Engineering*, 2016.

[7] U. Krispel, H. L. Evers, M. Tamke, R. Viehauser, and D. W. Fellner. Automatic texture and orthophoto generation from registered panoramic views. *ISPRS - Inter-*

national Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-5/W4:131–137, 2015.

[8] P. J. Petkova and U. Rüppel. A graph-based prediction method for electrical wiring in old residential buildings as a part of bim for urban mining purposes. *eWork and eBusiness in Architecture, Engineering and Construction*, pages 109–113, 2014.

[9] H. Riemenschneider, U. Krispel, W. Thaller, M. Donoser, S. Havemann, D. W. Fellner, and H. Bischof. Irregular lattices for complex shape grammar facade parsing. In *CVPR*, pages 1640–1647, 2012.