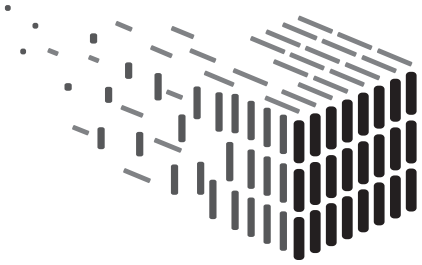


D5.4 Shape grammars for almost invisible structures

Software prototype v2

DURAARK

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

Date: 2015-07-31
Version 1.0
Document id. : duraark/2015/D5.4/v1.1

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/2015/D5.4/v1.1
Title of document	:	Shape grammars for almost invisible structures – Software prototype v2
Deliverable type	:	Software prototype
Contractual date of delivery	:	2015-07-31
Actual date of delivery	:	2015-07-31
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)	:	Jakob Beetz <J.Beetz@tue.nl> (TUE) Richard Vock <vock@cs.uni-bonn.de> (UBO)
Approval of this deliverable	:	Stefan Dietze <dietze@L3S.de> (LUH) – Project Coordinator Marco Fisichella <fisichella@L3S.de> (LUH) – Project Manager
Distribution	:	Public
Keywords list	:	object detection, shape grammar, computer vision, point clouds

Executive Summary

This report presents the new version of the tool **RISE** (**R**eveal almost **I**nvisible **S**tructur**E**s), the second software prototype for the detection of almost invisible structures from point cloud data and images. This deliverable D5.4 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 into a formal grammar, which is used together with the detections to generate a hypothesis for the invisible in-wall elements of the electrical appliances using a discrete optimization approach.

Table of Contents

1 DURAARK RISE

The RISE component (Reveal Invisible StructurEs) is composed of a set of related modules for the detection of almost invisible structures from point cloud data and images.

1.1 Motivation

A study conducted on the productivity of the construction industry in North America 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. However, 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 [6].

Studies confirm however a great interest for BIM in the electrical construction field. A survey conducted in the USA in 2009 [2] gives a ranking of the afterthought features of BIM for this field: clash detections, visualization of electrical design, space utilization, partial trade coordination, shop drawings review, virtual mock-ups, shop fabrication process, walk-throughs, design validation and energy analysis (See also Figure 1).

One can conclude, that there is an interest for working with BIM in the electrical construction field, but, that this interest is currently not fullfilled in the area of as-built documentation and renovation projects - which makes for 75% of the EU- building market share. [1]. Furthermore, the Deliverable 7.2 illustrates that existing workflows and tools around legacy BIM data and PointClouds are not akin to BIM information.

The DURAARK RISE component addresses this issue by giving the stakeholder the ability to automatically extract information about position of electrical sockets and hidden

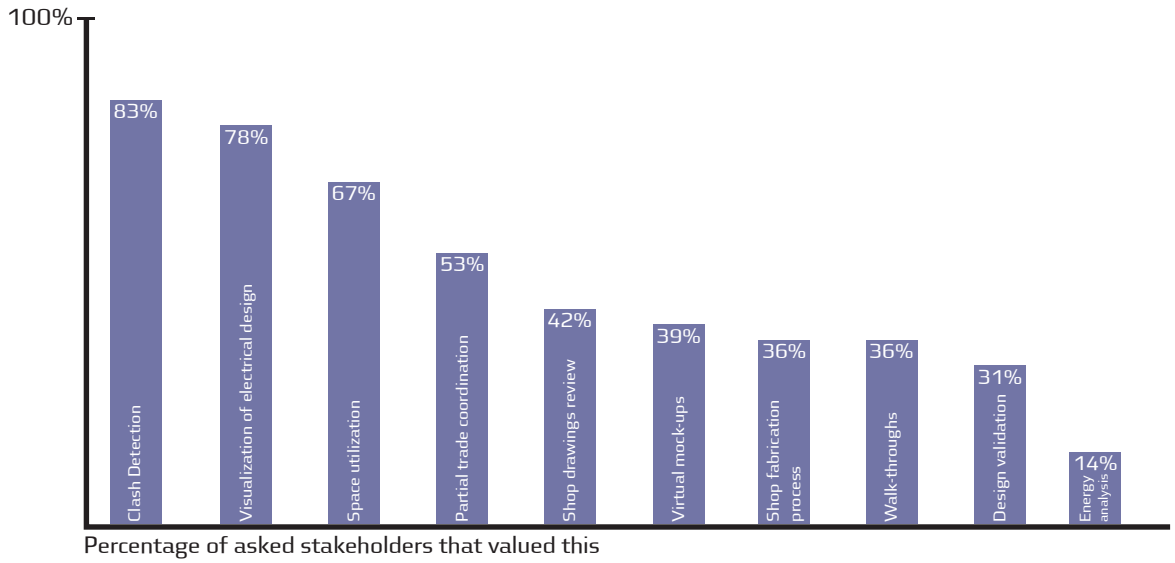


Figure 1: Percentage of stakeholders who valued these BIM functionalities in a survey conducted in USA in 2009. Graph based on [2]

cable paths.

This information can then be used by the stakeholder, for example to visualize the current electrical design (interest of 78%), conduct the planning of new installations utilizing a clash detection algorithm (interest of 83%) and coordinate with partners (interest of 53%) (see also Figure 1).

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
Source code:	`http://github.com/DURAARK/wiregen`

Geometric Enrichment Service

Type:	Web service
License:	MIT
Description:	D2.5, Section
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 Addressed Use Cases

For the DURAARK Rise component we see three stakeholders with particular interest herein. These stakeholders are:

- Electrical engineers and electrical construction companies
- Building owners and Facility managers
- Architects

The use cases for this component are shortly described here as sub-use cases to

- UC7: Plan, document and verify retrofitting/energy renovation of buildings (D2.1)

Use case 1: A renovation project. Whenever new electrical appliances/installations have to be added, the system can be used to create the installation zones and aid the planner. The stakeholder, a constructing architect or electrical engineer, retrieves an up-to-date point cloud and an IFC model with legacy information from the DURAARK long term archive for the planning of a renovation project. The stakeholder then automatically detects sockets and the probable cable paths and outputs this as IFC objects. After this the stakeholder imports them into a desired BIM software to visualize and e.g. conduct clash detections with the

planned electrical system. This data is then ingested into the DURAARK long term archive and retrieved by other partners of the renovation project. In this way the DURAARK long term archive and the components become an integrated part of the daily workflow of the construction industry.

Use case 2: Maintenance of a building. Whenever a re-organisation of e.g. an office building is needed, the system can create a visualisation of the current electrical sockets and probable cable path based on existing point cloud scans. The created ifc-model can be used for a quantity take-off, where planners can automatically generate a count of sockets and switches per room, level etc. A task, which is currently done manually and error prone.

Use case 3: Re-organisation of a building. The stakeholder, a building owner or facility manager, retrieves an up-to-date point cloud and IFC model with required information from the DURAARK long term archive for the planning of the re-organisation of a building. The stakeholder uses the DURAARK Rise to detect electrical sockets and probable cable paths. This information can then be used to aid the re-organisation of e.g. tables in office spaces. Or the information can provide cues to decide which walls to remove.

1.4 User Manual

This software prototype is directly integrated into the DURAARK WorkbenchUI which is available at `http://workbench.duraark.eu`. The first step is to select an E57 point cloud file from the provided list after starting a new pre-ingest session[1] (see D2.5, Section 2.1 for a detailed description of the pre-ingest workflow). Figure 2 shows the file selection

[1]Note, that D2.5 contains a very similar description of the steps for using RISE. For consistency the description is repeated here to keep the document self-contained.

in the WorkbenchUI.

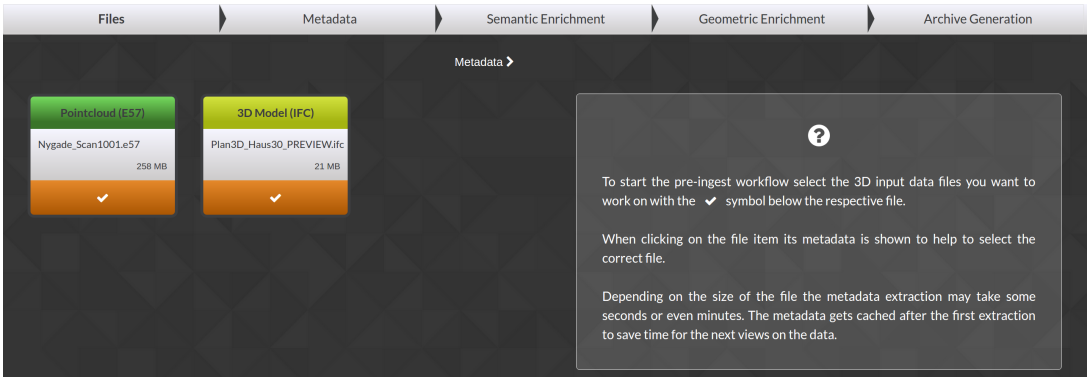


Figure 2: Selection of an E57 point cloud file from within the WorkbenchUI.

After the point cloud selection the stakeholder moves on to the "Geometric Enrichment" part of the pre-ingest workflow. Here he gets a selection of available enrichment components when clicking the "plus" sign below the previously selected point cloud file. The WorkbenchUI currently provides two components, the *IFC Reconstruction* and the *Electrical Appliance Detection* component. The Electrical Appliance Detection component is the one described within this deliverable under the name "RISE". The name in the user interface was picked to make its purpose obvious to the stakeholder. Figure 3 shows the geometric enrichment component selection.

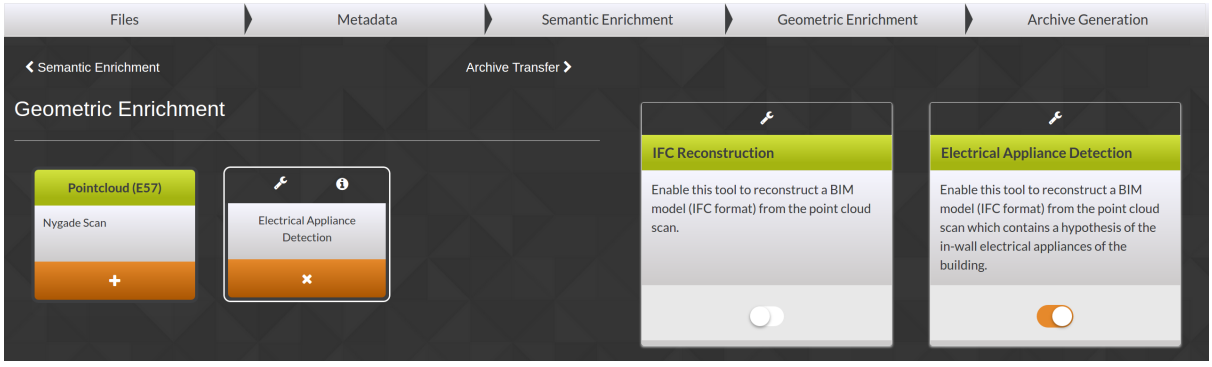


Figure 3: **Selection of the RISE component.**

As a stakeholder you have to enable the tool in clicking on the respective item as shown in

Figure 3 first. An additional item is appearing next to the E57 point cloud file, indicating that the detection of the in-wall appliance started in the background. A loading spinner appears during processing. In the background the module *OrthoGen* is used to create orthographic images of walls from the image information of the point cloud scan. The module *ElecDetect* detects the observable electrical appliances, and *WireGen* generates a hypothesis of in-wall wirings. An information icon in the header shows that the process has successfully finished.

Clicking on the icon presents the stakeholder with an overview of the results (see Figure 4), where the extracted hypothesis, the installation zones according to the used rule set, and the detection results of the observable sockets is shown. In M30, the rule set is fixed to the one described in Section 2.6.3, in M36 the stakeholder will be able to choose from a set of different rule descriptions.

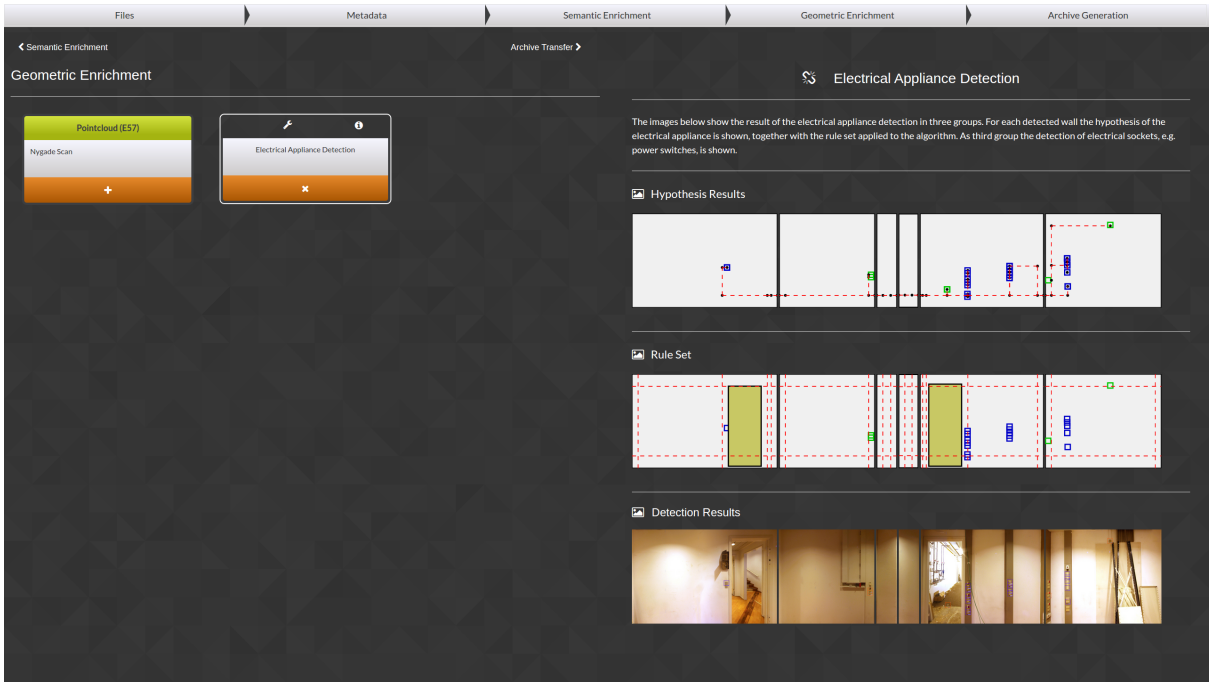


Figure 4: **Electrical applicance detection (RISE) results**

2 Components

2.1 Integration into the WorkbenchUI

The RISE component is directly integrated into the Service Platform's Geometric Enrichment Service. The component provides an API which is directly exposed by the Geometric Enrichment Service. The workbench is consuming the functionality of RISE via the Service Platform. The documentation of the Geometric Enrichment Service can be found at `here`.

2.2 Overall Architecture

The overall architecture of the RISE component is shown in Figure 5. The application consists of the modules *Geometry Extraction*, which extracts proxy geometry and adjacency information from the input point clouds, *OrthoGen*, which creates an orthographic image per wall segment, given proxy geometry and panoramic images taken from the scanning positions. *ElecDetect* performs detection of visible parts of electrical appliances (power sockets, light switches), and *WireGen* synthesizes a hypothesis for a highly probable wire routing inside the walls. Lastly, the results are injected back into the IFC for output.

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 adja-

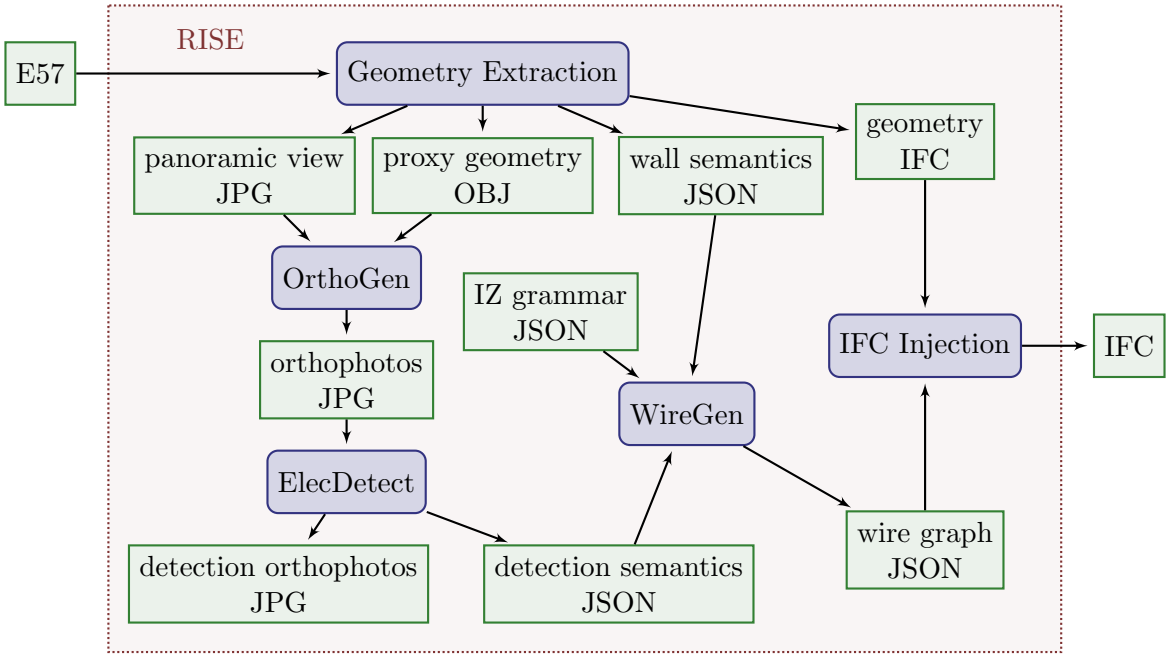


Figure 5: Overall Architecture of the RISE component. Modules (processes) correspond to blue nodes with rounded corners, and data (e.g. intermediate results) corresponds to green nodes.

cency relations are extracted and encoded in a *wall semantics* JSON file. (At the moment this is semi-automatic, but in the final release it is planned to extract this information automatically from the IFC).

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 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 6. For an extensive description of the method see the publication *Automatic*

Texture and Orthophoto Generation from Registered Panoramic Views [2] [10].

2.4.1 Commandline Interface

The commandline interface to the application is shown in Table 1.

`--help`	show help message
`--im arg`	(required) input panoramic image [.jpg]
`--ig arg`	(required) input geometry [.OBJ]
`--res arg`	resolution [mm/pixel], default 1mm/pixel
`--trans arg`	transformation [x,y,z]
`--rot arg`	rotation quaternion [w,x,y,z]
`--elevation arg`	elevation angle bounds [min..max], default [-PI/2..PI/2]
`--azimuth arg`	azimuth angle bounds [min..max], default [0..2*PI]
`--exgeom arg`	export (textured) geometry [OBJ] 0/1, default 0 (false)
`--exquad arg`	export extracted quads as (textured) geometry [OBJ] 0/1, default 0 (false)
`--exsphere arg`	export panoramic sphere [OBJ] 0/1, default 0 (false)
`--scale arg`	scale of input coordinates (mm/cm/m), default m
`--ncluster arg`	geometry element normal direction clustering window size, default 0.3
`--dcluster arg`	geometry element planar distance clustering window size in m, default 0.1

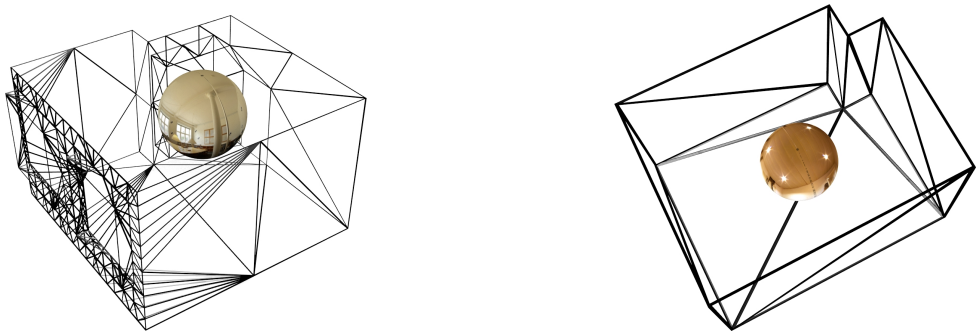
Table 1: All *OrthoGen* commandline options

2.5 ElecDetect

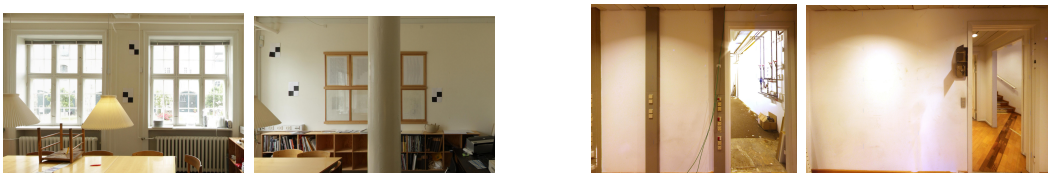
The *ElecDetect* module performs detection of relevant observable electrical appliances: instances of sockets and switches. In a preprocessing step, the module is trained with the visual appearance of the desired object classes. When performing the geometric enrichment with RISE, the module is used to detect instances of these classes on the orthophotos generated with *OrthoGen*. A detailed description of the contents of this

[2] `http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-5-W4/131/2015/isprsarchives-XL-5-W4-131-2015.html`

(a) Panoramic Views



(b) Proxy Geometry



(c) Exemplaric extracted Ortho Views

(d) Textured 3D model

Figure 6: 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).

Figure 7: The *ElecDetect* module performs the detection of sockets and switches (right) on orthophotos generated by the *OrthoGen* module (left).

module, as well as its related work were given in deliverable D5.5.2 - Shape grammars for almost invisible objects, software prototype v1 (see also the deliverable report).

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 four *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 this 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.

2.6.1 Related Work

Installation Zones. In many countries, standardization procedures for the installation of electrical appliances exist. The "Deutsche Industrienorm" (DIN), the german industry standard specification, specifies so-called *installation zones* as the preferred method to decide the actual position of cable routings in walls in residential areas, as described in DIN 18015 [4]. Several additions to this standard exist, for example for rooms with bathtubs or showers as described in DIN VDE 0100-701 [5].

An example of these installation zones for residential and office buildings described in the DIN can be seen in Figure 8. It can be observed that the creation of these zones follows strict rules, as they are constrained by minimum and maximum distances from neighboring walls and "restricted" areas, e.g. doors and windows.

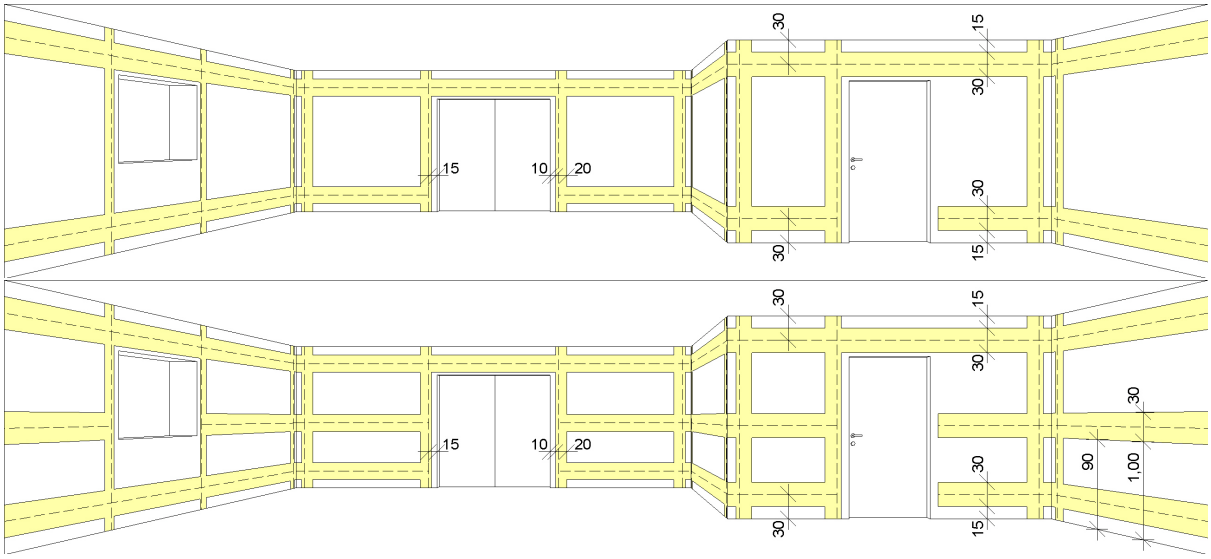


Figure 8: installation zones for residential buildings (top) and office rooms (bottom).
source: wikimedia commons

Formal languages and Shape Grammars. Formal language theory [3] is used in compiler construction to formally define the syntax of a programming language and verify syntactical correctness of given source code. A formal grammar G is a tuple $G = (N, T, P, S)$, where N is the set of nonterminal symbols, T is the set of terminal symbols, P is the set of production rules and S is the starting symbol. A context-free grammar allows only one nonterminal on the left side of the production rule. An attribute grammar defines a method to associate attributes to symbols (both nonterminal and terminal), evaluation is carried out when the production rules are processed.

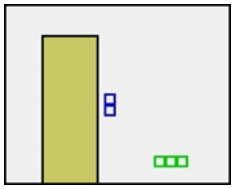
Shape grammars have been developed by Stiny et al. [12], originally to capture a formal specification of the design of paintings. The concept is based on methods from formal languages; a grammar consists of set of rules, a word in this grammar is a specific instance of an object and all possible words of the grammar correspond to the whole design space. Shape grammars have recently gained popularity in the computer graphics community as a method for the automatic generation of variations of an object class (e.g. buildings in a city) [11].

Graph theory and discrete optimization. A graph consists of a set of elements (nodes) together with a binary relation that is defined on the set [13]. Graphs can be visually represented by diagrams in which the elements are shown as points and binary relations as lines (edges) joining pairs of points.

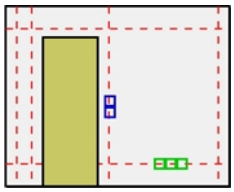
Discrete or combinatorial optimization [8] concerns itself with finding optimal solutions in discrete problem domains. Such problems often exhibit a seemingly simple structure, but finding the optimal solution boils down to an exhaustive search of the solution space, which is often not feasible (e.g. the famous traveling salesman problem). Therefore, approximative algorithms are used in practice that may not find the global optimum but terminate in an acceptable amount of time.

2.6.2 Overview

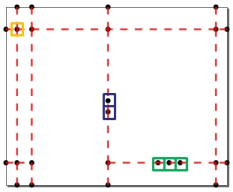
In principle, the module performs the following steps to create the result:



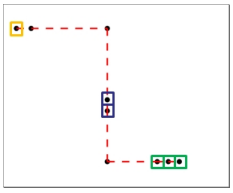
As input serves a JSON file that contains wall semantics, i.e. wall openings (e.g. doors and windows) and adjacency relations to neighboring walls. Furthermore, the file contains the detections from the *ElecDetect* module.



An installation zone grammar JSON file is supplied to the module that contains the rules which specify the prior knowledge of the applying standards for the creation of installation zones. The system evaluates the grammar, which leads to the creation of horizontal and vertical installation zone elements per wall (dashed lines).



After grammar evaluation, a graph is built that encodes all possible routings: crossing installation zones yield vertices, and detections (blue and green rectangles) are either placed as vertex on an existing edge, or connected by the shortest path to an adjacent edge. Adjacent zones from adjacent walls are connected together. Furthermore, at least one power root (orange rectangle) has to be specified.



Finally, a subgraph (specifically a forest) is extracted from this graph that connects all marked endpoints (detections) to a power root, with the assumption that the overall length of power lines, which equals to material costs, is minimized.

2.6.3 Installation Zone Grammar

As has been observed from the from standard specifications of installation zones, installation zones are placed with respect to wall boundaries or forbidden zones, e.g. windows or doors. We define the installation zone grammar as an context-free attribute grammar $G_{IZ} = (N, T, P, S)$. Nonterminals are written in uppercase letters (e.g. $WINDOW$) and terminals are written in lowercase (e.g. *hzone*). A production rule is written

$$\langle nonterminal \rangle \rightarrow \langle terminal | nonterminal \rangle \{\text{attribute definitions}\} \ldots \quad (1)$$

The left side of a production rule contains exactly one nonterminal, the right side can consist of any number of nonterminal and terminal symbols, together with attribute definitions in curly brackets. All symbols on the right hand side automatically inherit all attributes from the left hand side, additional attribute definitions are carried out afterwards.

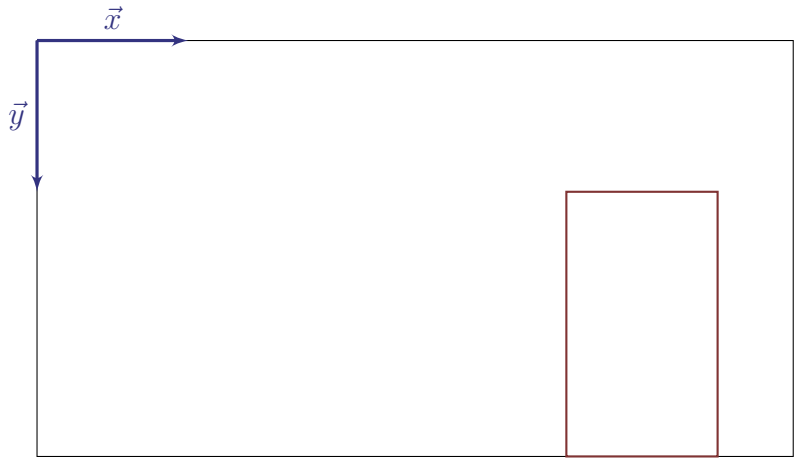


Figure 9: All positional information in the grammar is encoded by attributes relating to a local 2D coordinate frame per wall segment (*wall coordinates*), with origin in top left. Positional information of an symbol related to a wall (e.g. a door) is encoded by an axis aligned bounding box with attribute names `left`, `right`, `width`, `height`.

The starting production rule produces nonterminal symbols according to all elements that influence installation zone placement (walls, doors, windows, detections). All positional information is stored in attributes, with respect to *wall coordinates* (see also Figure 9) that are defined for each wall segment. The terminal symbols of this grammar correspond to horizontal and vertical installation zones, as well as forbidden zones for the placement of wirings.

The nonterminals produced by the starting rule are expected to contain the following default attributes: a bounding box specified by the attributes `left`, `top`, `width`, `height`. Furthermore each wall has an unique identifier, called `id`. Each terminal is expected to contain an attribute `wallid` that references the wall this symbol belongs to. Given the starting rule that contains the wall detection semantics from a dataset, the production rules are evaluated until the list of nonterminal symbols is consumed and only terminal symbols are left. The terminals with a special meaning for the optimization system are `hzone, vzone, root`, the others are treated as an endpoint if it contains the attribute `endpoint=true`.

As an example we provide a simple residential installation zone grammar, (DIN 18015):

$$
\begin{aligned}
WALL \rightarrow & vzone\{pos = left + 150; wallid = id\} \\
& vzone\{pos = left + width - 150; wallid = id\} \\
& hzone\{pos = top + 300; wallid = id\} \\
& hzone\{pos = top + height - 300; wallid = id\} \\
& wall\{zone\_width = 200\}
\end{aligned}
$$

$$
\begin{aligned}
DOOR \rightarrow & vzone\{pos = left + 150\} \\
& vzone\{pos = left + width - 150\} \\
& door\{forbidden = true\}
\end{aligned}
$$

$$
\begin{aligned}
WINDOW \rightarrow & vzone\{pos = left + 150\} \\
& vzone\{pos = left + width - 150\} \\
& window\{forbidden = true\}
\end{aligned}
$$

$$SOCKET \rightarrow socket\{endpoint = true\}$$

$$SWITCH \rightarrow switch\{endpoint = true\}$$

$$ROOT \rightarrow root\{root = true\}$$

2.6.4 Wire Routing Hypothesis

A wire routing is represented by a graph $G_W = (V, E)$ that contains a number of vertices $v \in V$ and edges $e \in E$, where a vertex is associated to a position inside a wall segment, and an edge connects two vertices. Vertices can also be associated to either an endpoint (sockets, switches) or a power root.

The graph of all possible routings is created from the list of terminal symbols as follows: For each wall, the graph is built from the arrangement of lines formed by the horizontal and vertical installation zones. Then, any edges that intersect terminal symbols that are marked as *forbidden* are removed. After this step, graphs from adjacent walls are connected together at corresponding installation zone endpoints.

Under the assumption that the planning was carried out minimizing material costs, creating a hypothesis for actual routing is extracted from this graph by the following optimization: We want to extract the hypothesis subgraph $G_H \subseteq G_W$, where G_H is a forest that connects all vertices that are marked as endpoints to a power root. This graph is minimal with respect to an edge cost function defined on the edges of G_W, where the cost function simply corresponds to the euclidean length of the edge. This problem is also called the minimum steiner tree problem in graphs, and is known to be NP-complete [7]. There are however algorithms that run in polynomial time, although they might not deliver the global optimal solution, the structure of our problem suggests that the solutions will yield a good solution. For this software deliverable we implemented an algorithm similar to [9], which runs in approximately $O(n^2 * m)$ where n is the number of edges and m is the number of endpoints.

2.6.5 Commandline Interface

The application has three commandline arguments

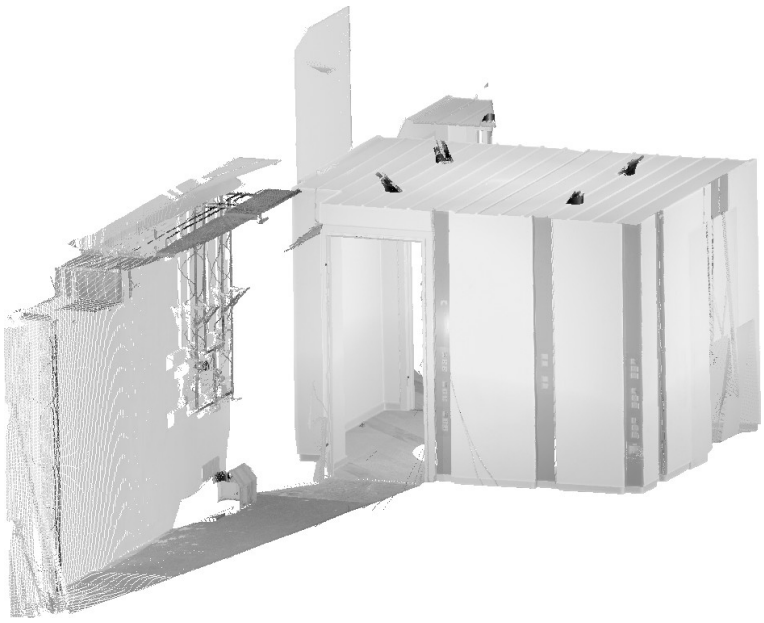
`-g, --grammar`	(required): Set Installation Zone Grammar (JSON)
`-i, --input`	(required): Set room layout input symbols created by the first rule (JSON)
`-o, --output`	(optional): Set output directory.

Table 3: The three *WireGen* commandline options.

For an example of the input JSON see the technical appendix in Section 6.

2.7 Results

We demonstrate the application of each pipeline step using the Nygade dataset (Figure 10). The orthophoto extraction using *OrthoGen* yields 6 wall elements (Figure 11a), the electrical appliance detection using *Elecdetect* finds 23 elements (sockets and switches as seen in Figure 11b. Using the installation zone grammar, *WireGen* produces installation zones (Figure 11c) and extracts the final wiring hypothesis (Figure 11d).
The same dataset is a showcase in the WorkbenchUI and can be inspected by starting the session *Nygade*.



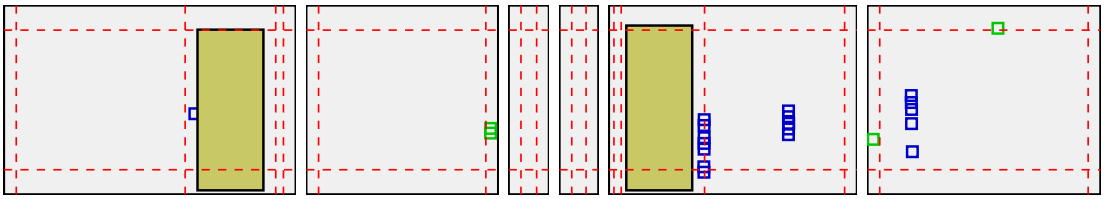
(a) Input E57 point cloud

(b) Panoramic View

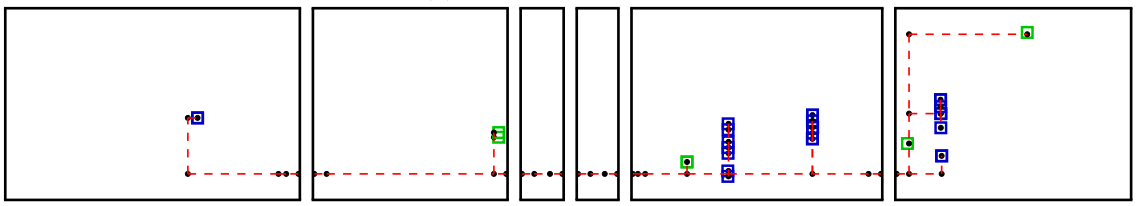
Figure 10: The input pointcloud and panoramic image for the test case.

(a) Orthographic Views

(b) Electrical appliance detection



(c) Installation Zones



(d) Electrical wiring hypothesis

Figure 11: Intermediate results and final hypothesis from the RISE component.

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.

---

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

5.1 ElecDetect

Other than some bugfixing, the detection component has been developed further, optimizing training and classification parameters for the specified objective.

5.2 OrthoGen

The OrthoGen component produces orthographic views given a scanner position and panoramic views that have been acquired at its position (hence registered panoramic views). At the moment the module can only handle one scanner position/panorama, which results in quantization artifacts (blurry orthophotos) for walls that are far away from the scanning position, or worse, wrong projections if the projected wall cannot be seen from the scanners' position. In the M36 prototype we will add functionality that OrthoGen will be able to cope with multiple scans (e.g. by projecting the color information from the closest scanner position).

5.3 WireGen

WireGen will create a wire routing hypothesis given room layout and the detections of the OrthoGen component. The optimization procedure has been tested on smaller datasets, but still has to be evaluated on a larger dataset. These will include a hierarchy of rooms, as for instance corridors and connected rooms, and can here make use of the room-graph, which is created in the DURAARK IFC-reconstruction tool from WP5.

5.4 Impact

By using the RISE component, a stakeholder will be able to create a hypothesis for wire routings in wall from the measurements (point clouds, panoramic photos). The system can also be used to just 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. 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 possible 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 involving the use of shape grammars as prior knowledge representation.

5.6 Future Work

The next steps are the evaluation of the pipeline using more complex datasets, and the extension of the *OrthoGen* component to combine the information from several scanning positions (panoramas). and the creation of more rule sets that guide the installation zone generation. Furthermore, as the detection of observable electrical appliances will not be able to detect all instances in some cases, it might be necessary to manually modify or add detections or the position of doors and windows before applying the installation zone grammar. Selection of rulesets and modification of detections will be integrated in the WorkbenchUI. The RISE component will be evaluated within the overall evaluation of the WorkbenchUI in WP7 (upcoming D7.4).

6 Technical Appendix

In this section we give exemplaric input for the JSON format developed in the WireGen component.

6.1 Installation Zone Grammar

This JSON file describes the installation zone grammar as described in Section 2.6.3.

```
{
  "WALL" : [
    { "label":"vzone", "attributes":{ "pos":"att.left+150","wallid":"att.id" } },
    { "label":"vzone", "attributes":{ "pos":"att.left+att.width-150","wallid":"att.id" }
        },
    { "label":"hzone", "attributes":{ "pos":"att.top+300","wallid":"att.id" } },
    { "label":"hzone", "attributes":{ "pos":"att.top+att.height-300","wallid":"att.id" }
        },
    { "label":"wall",  "attributes":{ "zone_width":"200" } }
  ],

  "DOOR" : [
    { "label" : "vzone", "attributes" : { "pos": "att.left-150" } },
    { "label" : "vzone", "attributes" : { "pos": "att.left+att.width+150" } },
    { "label" : "door" }
  ],

  "WINDOW" : [
    { "label" : "vzone", "attributes" : { "pos":"att.left-150" } },
    { "label" : "vzone", "attributes" : { "pos":"att.left+att.width+150" } },
    { "label" : "window" }
  ],

  "SOCKET" : [
    { "label" : "socket" , "attributes" : { "endpoint":"true" } }
  ],

  "SWITCH" : [
    { "label" : "switch" , "attributes" : { "endpoint":"true" } }
```

```
28    ],
29
30    "ROOT" : [
31      { "label" : "root" , "attributes" : { "root":"true" } }
32    ]
33  }
```

6.2 Example Input Symbols

This section shows the JSON file that corresponds to the example shown in the *WireGen* overview, see Section 2.6.2.

```
1
2   [
3     {
4       "label": "WALL",
5       "attributes": {
6         "id" : "wall01_1_01",
7         "left" : 0,
8         "top" : 0,
9         "width" : 3000,
10        "height" : 2400,
11        "origin" : [ 0, 0, 2400 ],
12        "x" : [ 1, 0, 0],
13        "y" : [ 0, 0, -1]
14      }
15    },
16    {
17      "label": "DOOR",
18      "attributes": {
19        "left" : 500,
20        "top" : 415,
21        "width" : 735,
22        "height" : 1985,
23        "wallid" : "wall01_1_01"
24      }
25    },
```

```
26  {
27    "label": "SWITCH",
28    "attributes": {
29      "left" : 1335,
30      "top" : 1350,
31      "width" : 128,
32      "height" : 128,
33      "weight:" : 0.8,
34      "wallid" : "wall01_1_01"
35    }
36  },
37  {
38    "label": "SWITCH",
39    "attributes": {
40      "left" : 1335,
41      "top" : 1200,
42      "width" : 128,
43      "height" : 128,
44      "weight:" : 0.8,
45      "wallid" : "wall01_1_01"
46    }
47  },
48  {
49    "label": "SOCKET",
50    "attributes": {
51      "left" : 2000,
52      "top" : 2036,
53      "width" : 128,
54      "height" : 128,
55      "weight:" : 0.8,
56      "wallid" : "wall01_1_01"
57    }
58  },
59  {
60    "label": "SOCKET",
61    "attributes": {
62      "left" : 2150,
63      "top" : 2036,
64      "width" : 128,
```

```
65        "height" : 128,
66        "weight:" : 0.8,
67        "wallid" : "wall01_1_01"
68      }
69    },
70    {
71      "label": "SOCKET",
72      "attributes": {
73        "left" : 2300,
74        "top" : 2036,
75        "width" : 128,
76        "height" : 128,
77        "weight:" : 0.8,
78        "wallid" : "wall01_1_01"
79      }
80    },
81    {
82      "label": "ROOT",
83      "attributes": {
84        "left" : 150,
85        "top" : 300,
86        "width" : 0,
87        "height" : 0,
88        "wallid" : "wall01_1_01"
89      }
90    }
91  ]
```

References

[1] 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.

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

[3] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956.

[4] Electrical installations in residential buildings, sep 2013.

[5] Low-voltage electrical installations - part 7-701: Requirements for special installations or locations - locations containing a bath or shower (iec 60364-7-701:2006, modified); german implementation hd 60364-7-701:2007, oct 2008.

[6] 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.

[7] F. K. Hwang and D. S. Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992.

[8] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 4th edition, 2007.

[9] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15(2):141–145, 1981.

[10] U. Krispel, H. L. Evers, M. Tamke, R. Viehauser, and D. W. Fellner. Automatic texture and orthophoto generation from registered panoramic views. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W4:131–137, 2015.

[11] P. Müller, P. Wonka, S. Haegler, U. Andreas, and L. Van Gool. Procedural Modeling of Buildings. *Proceedings of 2006 ACM Siggraph*, 25(3):614–623, 2006.

[12] G. Stiny and J. Gips. Shape Grammars and the Generative Specification of Painting and Sculpture. *Best computer papers of 1971*, 1:125–135, 1971.

[13] R. J. Wilson. *Introduction to Graph Theory*. John Wiley & Sons, Inc., New York, NY, USA, 1986.