A Service-Oriented Approach for Classifying 3D Points Clouds by Example of Office Furniture Classification

Vladeta Stojanovic
Hasso Plattner Institute
University of Potsdam, Germany
vladeta.stojanovic@hpi.de

Matthias Trapp
Hasso Plattner Institute
University of Potsdam, Germany
matthias.trapp@hpi.de

Rico Richter
Hasso Plattner Institute
University of Potsdam, Germany
rico.richter@hpi.de

Jürgen Döllner
Hasso Plattner Institute
University of Potsdam, Germany
juergen.doellner@hpi.de

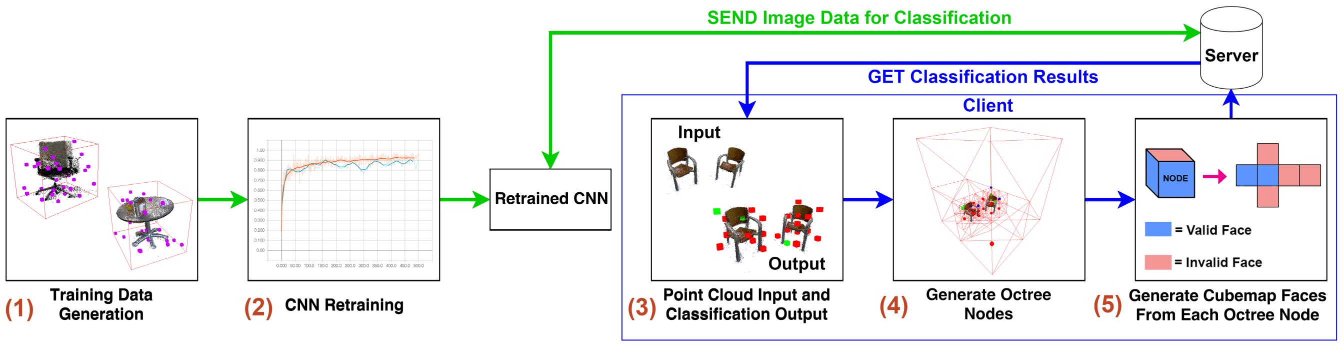

SEND Image Data for Classification
Server
GET Classification Results
Client
Input
Output
NODE
= Valid Face
= Invalid Face
Retrained CNN
(1) Training Data Generation
(2) CNN Retraining
(3) Point Cloud Input and Classification Output
(4) Generate Octree Nodes
(5) Generate Cubemap Faces From Each Octree Node

Figure 1: High-level overview of the presented point cloud classification approach using image-based classification. The process shows: (1) generation of the training data, (2) retraining of the CNN, (3) user input of point cloud for classification and the visualized output, (4) resulting generation of octree nodes, (5) generation of cubemaps for the image-based classification.

ABSTRACT

The rapid digitalization of the Facility Management (FM) sector has increased the demand for mobile, interactive analytics approaches concerning the operational state of a building. These approaches provide the key to increasing stakeholder engagement associated with Operation and Maintenance (O&M) procedures of living and working areas, buildings, and other built environment spaces. We present a generic and fast approach to process and analyze given 3D point clouds of typical indoor office spaces to create corresponding up-to-date approximations of classified segments and object-based 3D models that can be used to analyze, record and highlight changes of spatial configurations. The approach is based on machine-learning methods used to classify the scanned 3D point cloud data using 2D images. This approach can be used to primarily track changes of objects over time for comparison, allowing for routine classification, and presentation of results used for decision making. We specifically focus on classification, segmentation, and reconstruction of multiple different object types in a 3D point-cloud scene. We present our current research and describe the implementation of these technologies as a web-based application using a services-oriented methodology.

CCS CONCEPTS

• **Computing methodologies** ⟶ *Computer graphics*; *Neural networks*; • **Human-centered computing** ⟶ *Visualization systems and tools*;

KEYWORDS

Indoor Models, 3D Point Clouds, Machine-Learning, BIM, Service-Oriented

ACM Reference Format:
Vladeta Stojanovic, Matthias Trapp, Rico Richter, and Jürgen Döllner. 2018. A Service-Oriented Approach for Classifying 3D Points Clouds by Example of Office Furniture Classification. In *Web3D '18: Web3D '18: The 23rd International Conference on Web3D Technology, June 20–22, 2018, Poznan, Poland.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3208806.3208810

1 INTRODUCTION

Establishment of Building Information Modelling (BIM) within the building lifecycle has created a need to bridge digital assets and traditional documentation approaches [Eastman et al. 2011]. This also requires the Facility Management (FM) sector to make use of digital documentation for building Operation and Maintenance (O&M). According to Roper and Payant [2014], the use of building automation within an existing IT infrastructure is the main cornerstone of an Integrated Workplace Management System (IWMS). These IWMSs must be able to communicate and provide an informative analytical output of a given FM operation status to stakeholders. One requirement for such a system is frequently updated physical representations of the built environment - in our case this would include frequently updated *snapshots* of physical interiors of office spaces. These snapshots would be used to capture the 3D physical representation of the environment to a reasonable degree of accuracy and allow for further inspection, update, and analysis. An approach to generate these snapshots is with the use of point clouds.

Using laser scanning and photometry-based methods, point clouds can provide 3D and textured up-to-date physical representations of the built environment up to high resolutions (millions of points per square meter), and featuring intricate details [Discher et al. 2018]. The ability for FM stakeholders to be able to capture, classify and collaboratively annotate point-cloud representations of indoor environments can increase engagement, encourage collaboration, and enhance decision making concerning generation of digital FM documentation.

1.1 Problem Statement and Contribution

Recurring generation of up-to-date representations and documentation of building interiors creates problems when capturing the current state of the built environment. Apart from historical 2D and 3D data of buildings (e.g, 2D floor plans and 3D CAD models), which may not have been updated since the design stage of the building, manual generation and updating of BIM data is an expensive and time consuming process. The use of 3D point clouds provides an efficient, affordable, and manageable alternative to capturing the physical state of the built environment. Less detailed scans can also be made routinely and can be used to assess physical changes over time. A 3D point cloud consists of what can be defined as *non-interpreted data*, meaning data that is open to visual interpretation but does not have any semantics associated with it.

For example, a point-cloud representation of an office will place all the captured chair, desks, and computers into the same dataset category (Figure 2 (a) provides an example of an office point-cloud scan). The use of mobile devices for capturing and generation of 3D point-cloud representations of interior environments has also become a viable option (e.g., Google Tango specification compatible consumer mobile devices). The main advantage of using commodity consumer mobile devices with depth-sensing cameras is the ability to provide an affordable, flexible, and simple approach for capturing 3D point clouds of interiors, in comparison to more expensive and professional laser scanning devices [Froehlich et al. 2017]. To generate up-to-date digital documentation, point cloud data needs to be segmented and classified and often reconstructed as 3D geometry: segmentation is required to distinguish different geometric features in the data and classification is used to add semantics to the segmented data. Subsequently it can be used as basis for primary *Level of Detail* (LOD) representation or as source data for *as-is* BIM. Figure 1 illustrates a high-level overview of our approach to this problem.

We want to enable FM users to capture, classify, annotate, and discuss routinely captured physical representations of interior spaces from an online portal. Figure 3 illustrates the proposed process using the methodology described in this paper and the stages of our approach. With respect to this, the main challenges are:

(1) Accurate and efficient documentation generation without having to deal with large volumes of point-cloud data and time-consuming processing.
(2) Users should be able to capture point-cloud data using both high-end and commodity point-cloud acquisition hardware.
(3) Allow for frequent generation of digital documentation from point-cloud scans.
(4) Documentation must be centrally accessible to involved stakeholders via a service-oriented platform.

We present our approach for automatic classification of indoor point clouds containing multiple different office furniture objects. We present this using a prototypical web-based application that enables classification, segmentation, and reconstruction of point-cloud scenes of office interiors. Finally, we discuss the implementation details of our method using a service oriented approach and provide preliminary test results.

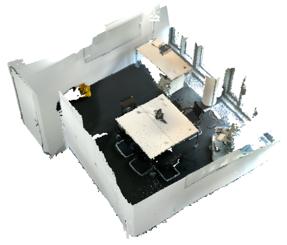
2 RELATED WORK

2.1 BIM In the Context of FM

The two key features of using BIM in FM are information sharing between stakeholders and clash detection, which potentially can help reduce operational costs [Teicholz et al. 2013]. According to Kincaid [1994], two important factors of integrated FM are (1) management of the organization, (2) management of the support and critical services of the organization. *As-is* BIM data is used to capture the current state of the building, in contrast to *as-designed* BIM data created during the design and construction stages [Anil et al. 2013]. The visualization-based analytical output of combined *as-designed* and *as-is* BIM, point cloud and sensor data can be combined to create a *Digital Twin* [Grieves 2014], that allows for historical and current real-time representations of building lifecycle-management topics within the emerging Industry 4.0 realm [Lasi et al. 2014].

2.2 As-is BIM Data from 3D Point Clouds

Current methods for automated classification and reconstruction of *as-is* BIMs and 3D geometry from 3D point clouds still require a degree of manual user input as well as pre- and post-processing, in order to generate a usable dataset required for digital documentation [Ochmann et al. 2016; Xiong et al. 2013]. Previous research by Tang et al. [2010] has provided reviews and discussion of current methods used for automated generation of usable *as-is* BIMs from point clouds. According to Qu and Sun [2015], the required point-cloud processing can be too complex for routine FM use. As


(a)

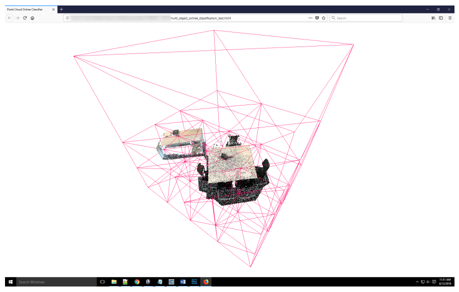

(b)

Figure 2: (a) Example of a 3D point cloud of an office environment (222 084 points for approximately 20 square meters), captured using a Google Tango compatible mobile phone, and (b) the same point cloud, running on a modern WebGL compatible web-browser.

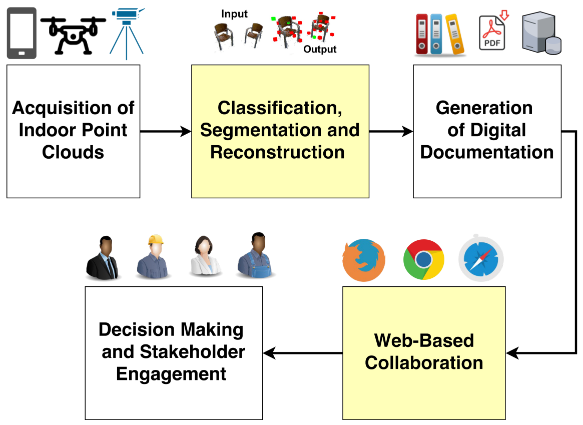

Input
Output
PDF
Acquisition of Indoor Point Clouds
Classification, Segmentation and Reconstruction
Generation of Digital Documentation
Decision Making and Stakeholder Engagement
Web-Based Collaboration

Figure 3: The proposed process of deriving of office furniture objects from point clouds and generating digital documentation for collaborative stakeholder engagement. The yellow highlighted stages are addressed by our approach.

noted by Dimitrov and Golparvar-Fard [2015], raw point clouds can contain a number of possibly undesired artefacts, including: (1) noise introduced due to surface roughness and surface material reflections, (2) visual clutter (e.g. small objects), (3) occluding elements, (4) partially and/or incorrectly captured data and (5) increased point-cloud density due to overlapping scans. The majority of the previous research on reconstruction has been focused on reconstructing empty interior spaces at a given BIM LOD, which does not include things such as office furniture; rather the reconstruction often focuses on the structural, non-furnished interiors, registered equipment (in case of manufacturing or maintenance) and Mechanical, Electrical and Plumbing (MEP) components of the building [Volk et al. 2014].

2.3 Service-Oriented 3D Visualization

Applications of 3D point clouds, BIM and 3D visualization for FM are currently at an early stage of adaptation in comparison to other stages in the building lifecycle, though successful case studies have been described [Teicholz et al. 2013]. This is mainly due to the demanding requirements of dealing with large volumes of generated digital data (including storage and retrieval) and complexities of integration with existing IT infrastructure and CAFM (Computer-Aided Facility Management) approaches [Pärn et al. 2017]. There are advantages of using 3D visualization in comparison to using static 2D images (such as floorplans), in terms of enhancing communication between different stakeholders [Fischer et al. 2003]. Majority of current approaches for 3D visualization in FM rely on desktop-based applications that do not offer the flexibility of service-oriented applications, such as streaming to mobile thin clients [Hagedorn and Döllner 2007]. Apart from processing visualization outputs, research by Zhao et al. [2015] describes the use of a service-oriented architecture to update BIM and energy models during the design phase of a building construction project. Further, Scully et al. [2015] describes how web-based 3D visualization can be used for tracking of related 3D assets through a service-oriented and BIM-enabled online repository. Lee et al. [2016], describe an online FM portal for collaborative visualization and decision making using web-based 3D visualization technologies. There is a clear paucity for the use of service-oriented approaches for enabling FM stakeholders to generate, document, track and discuss digital representations of interior building environments centrally.

2.4 Multiview-Based Classification Methods

With the current advancements in machine learning driven by computer vision and robotics applications, there is an ongoing research in the classification and generation of semantics for and from 3D point-cloud data. Most notable approaches rely on machine-learning methods for training and classifying 3D Convolutional Neural Networks (CNNs) as well as Deep Learning methods [Ioannidou et al. 2017]. The development and release of Google's Inception V3 CNN model and TensorFlow API allows for more practical implementation and application of machine-learning-based methods for classification of 2D and 3D data [Abadi et al. 2016; Szegedy et al. 2016]. The two mainstream methods for classification of 3D data include 3D voxelization and 2D image-based classification. The former uses a 3D CNN trained on the voxelized representation of a given 3D object in order to distinguish between geometric features (examples include Qi et al. [2017] and Wang et al. [2017a]), while the later uses a 2D CNN image classifier and uses 2D images of 3D objects for classification. However, in terms of both training and classification, using a 2D image-based approach is still considered to be faster and more optimal [Wang et al. 2017b].

There is a recent trend towards using *multiview-based* methods for classification on 3D data, with notable approaches described by Ma et al. [2017], Boulch et al. [2017], and Kanezaki et al. [2016]. Multiview approaches generate 2D training data for classifying 3D objects by taking consecutive screen captures around a 3D object. The way the images are generated are largely left to the programmer to decide (e.g., using spherical projection images of 3D objects described by Cao et al. [2017]). This makes multiview-based classification methods suitable for web-based 3D visualization using a service-oriented approach, where 2D image data is smaller and faster to classify than 3D point clouds. This is especially important for 3D point clouds since their data can be very large in size, e.g., a few terabytes for one building. Our research expands on the approach explored by Dietze et al. [2017], for classification of single 3D geometry objects based on 2D images and 3D models using a service-oriented architecture.

3 APPROACH AND IMPLEMENTATION

We have implemented a prototypical 3D web application based on the service-oriented architecture paradigm. It enables users to load and classify point-cloud data via an user interface running in any WebGL compatible browser (Figure 2 (b)). We can currently load and classify scenes with up to 1.5 million points for approximately twenty square meters. We focus on a simpler 2D image-based classification approach, in contrast to dealing with larger and more complex 3D point-cloud data types and structures, to accommodate our methodology for service-oriented use. The primary technologies used for our approach are HTML5, WebGL (via Three.js), Node.js, Websockets, Python and the Google Tensorflow machine-learning framework.

The client-server model allows for communication between the back-end processing (implemented as a Node.js server interfacing Python image classification scripts), and the front-end for displaying the classification results (Section 3.1.3). The output of our web-based application is a numerical and visual classification result that is presented to the user in the form of an interactive 3D scene for inspection. In addition, our application can approximate spatial positions where placeholder 3D models of classified 3D models can be inserted. We can also extract point clusters from classified octree nodes, thus implementing basic segmentation (Sections 4.2.1).

3.1 Service-Oriented Architecture Overview

The client-side is responsible for the following operations: (1) loading and displaying of point clouds, (2) generation and displaying of an octree for loaded point clouds for visual debugging, (3) generation and validation of cubemap faces, (4) notifying the server that the generated data is ready for classification, (5) receiving and displaying the classification results from the server, (6) segmentation of point-cloud data, and (7) k-means clustering and 3D object placement (reconstruction).

Figure 4 illustrates this processing pipeline of the client-server model. We currently perform the majority of data generation on the client-side. While this approach limits data throughput to and from the server, it allows us keep all the generated data locally on the client machine and facilitates better digital rights management. Using this approach, sensitive data (except the cubemap images) does not need to be sent to a remote server for classification. The cubemap images that are sent could possibly be encrypted by the client and decrypted on the server prior to classification.

Using the client-side interface, a user is able to load in a raw point cloud for classification. We use RGB point-cloud data in the PLY file format. The web application then automatically partitions the point cloud using an octree of adjustable complexity. The octree complexity is determined by the sampling size of points set by the user. The more points that are sampled, the more sparse the generated octree becomes. At each octree node that contains a point cluster, a virtual camera position is computed. Generation of cubemap images is performed on the client-side, while the classification is computed on the server-side using Python (with TensorFlow being installed as a separate add-on for Python). The generation of the cubemaps is handled via a fragment shader and saved as a JPEG image using the HTML5 Canvas 2D image processing components. Communication from the server JavaScript code and the Python code is enabled through Node.js using an extension (called *PythonShell*). Each of the generated cubemap images is then classified using a retrained version of the Inception v3 CNN model via TensorFlow. The classification average value of a node is obtained by summing up each object type classification percentage for valid cubemap faces, and dividing the total number of its valid cubemap faces. The average classification results are sent back to the client application to be presented as a percentage-based value for each object type at each octree node that contains a given point cluster. Nodes that have completely invalid faces are not used for classification.

Once the 3D point cloud has been classified, we use k-means clustering to generate placeholder nodes for 3D objects from the classified octree nodes. The number of placeholder nodes is based on the number of objects that are set prior to classification. This allows us to place 3D models in approximated positions in 3D space to enhance visualization and enable comparison between the original point cloud scan. The extraction of point clusters from classified nodes is also possible and allows us to perform basic segmentation if required (Section 4.2.1).

3.1.1 Spatial Partitioning Using Octrees. One of the main requirements for this project was to be able to classify multiple furniture types in a captured 3D point cloud. A typical scene may contain more than one object type, such as multiple chairs, a table, and a sofa in the background. While 3D and 2D CNNs can be used successfully to classify single objects, including multiple objects for classification requires partitioning of the 3D scene in order to treat each partition as a cluster belonging to a given object type. The use of an octree structure for partitioning and discretization has previously been used by Wang et al. [2017a] for training a 3D CNN using voxelized 3D objects, and by Qi et al. [2017] for multiview-based classification. Octrees have also been researched by Bassier et al. [2017] for segmentation of unstructured point cloud scans of building interiors using region growing-methods (particularly random fields for sampling octree clusters for segmentation). We decided to use an octree for spatial partitioning mainly because it provides a reasonable performance in terms of node traversal in comparison to other solutions such as kd-trees and BSPs, and scales sufficiently to the spatial distribution of point clouds. We also needed to use an octree instead of a quadtree due to dealing with data with variable point distribution in 3D space. Figure 5 (a) shows

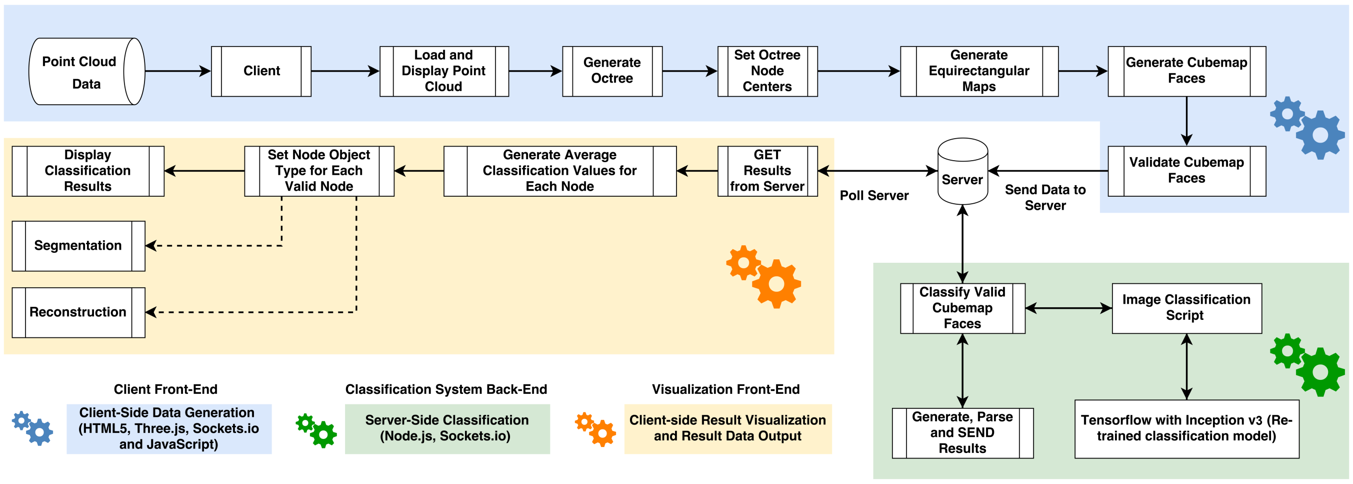

Point Cloud Data
Client
Load and Display Point Cloud
Generate Octree
Set Octree Node Centers
Generate Equirectangular Maps
Generate Cubemap Faces
Validate Cubemap Faces
Send Data to Server
Server
Poll Server
GET Results from Server
Generate Average Classification Values for Each Node
Set Node Object Type for Each Valid Node
Display Classification Results
Segmentation
Reconstruction
Classify Valid Cubemap Faces
Image Classification Script
Generate, Parse and SEND Results
Tensorflow with Inception v3 (Re-trained classification model)
Client Front-End
Client-Side Data Generation (HTML5, Three.js, Sockets.io and JavaScript)
Classification System Back-End
Server-Side Classification (Node.js, Sockets.io)
Visualization Front-End
Client-side Result Visualization and Result Data Output

Figure 4: The presented client-server model showing the data flow, processes, and systems used for our approach.

the results of an octree for an input point cloud and the generation of the classified octree nodes for the classification result.

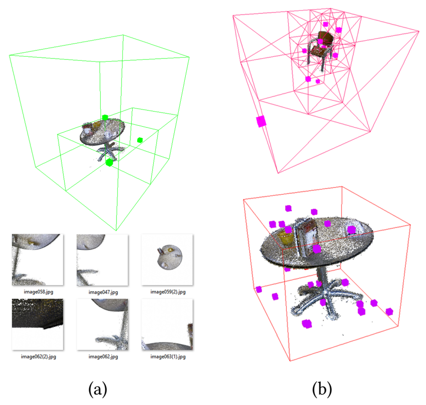


Figure 5: (a) Example octree used for image-based classification with generated valid cubemap faces, and (b) example of method used to generate 2D image training data using octree node positions (top) and random positions (bottom).

3.1.2 Generation of Cubemap Images. At each octree node that contains a point cluster, a virtual camera position is computed. This virtual camera generates a cubemap of the location, while the visibility of other octree nodes is disabled (this prevents occlusion and points from other nodes intruding the current node whose cubemap is being generated). This approach allows us to capture the complete environment around each node center as a single image. We also check if the generated cubemap faces have an average color value of less than 250 for each of the RGB color channels. These faces are marked as being valid, otherwise they are marked as invalid and we do not classify whitespace (blank) faces with none or too few points. One current limitation of this this approach is that it works only on RGB color values that are not bright. A remedy to this would be to use a binary coverage map along with RGB color values (Section 5).

This image is then converted to cubemap faces using the HTML5 Canvas API and sent to the server for image classification. The generated cubemap faces are resized to 300 × 300 pixels size and saved as JPEG images (recommended size and file type for image classification using the Inception CNN). File names of the generated cubemap faces that are marked as valid are given a numerical ID that corresponds to a specific octree node in the scene. This allows us to map calculated classification result averages to each corresponding octree node. Due to security reasons, the generated cubemap face images and any segmented point-cloud clusters must be explicitly downloaded by the user via the client webpage. We remedy this by using a default assets directory with the web browser to where we download all the generated data.

3.1.3 Client-Server Model Implementation. The server is implemented using Node.js and communication with the server is established via the Sockets.io library, using an echo client/server architecture. The server listens to any communication by the client from a given port, such as incoming SEND responses for receiving data and GET responses for sending classification results. The server calls the image classifier script implemented in Python 3.5 using Tensorflow. Once the classification results have been generated by calling the image classifier Python script, the server loads in the classification results (stored as JSON data and saved as text files on the server), parses them and sends the results back to the client. The client then averages the result for each corresponding node that the valid cubemap faces were generated from. The server then removes the generated cubemap faces and results once the classification has been completed.

3.2 CNN Retraining Methods and Testing Criteria

The training point-cloud data was obtained from the RGB-D Object Dataset from University of Washington [Lai et al. 2011] and the IRML point-cloud dataset [Gómez 2018]. The image data sets used for retraining the CNN model were generated using two approaches. With the first approach, the set of training data was generated by randomly sampling 32 different locations around each point-cloud object and then generating the corresponding 2D images of cubemap projections at those points. For the first approach the axis-aligned bounding box for a given point-cloud object is calculated, then the random points used as camera positions are generated within the bounding box. This approach was selected as it closely resembled how the point clusters may be partitioned within an octree structure, but is faster. The second approach is based on the actual node locations using the generated octree partition of the point-cloud cluster. This approach is identical to the main classification approach, except only the first two stages of the classification pipeline are used. For the retraining of the CNN model, we used between 300-700 images per object category. For each of the datasets, two different testing categories were established. First, there is a simple testing category that features no more than two types of objects in the same 3D space. The second, more complex category, features multiple instances of any of the three object categories arranged in various positions in the same 3D space (this complexity factor is based on the number of different object types in the scene and not on the complexity of the point cloud itself). The out-of-core point-cloud data was considerably lower resolution than the dataset constructed using the in-core point-cloud test data (on average 100 times more sparse). For each of the datasets and their complexity categories, three different octree complexity presets were chosen (low, medium, and high). Overall we had 72 different test cases (36 for each of the two test datasets). Figure 5 shows examples from both training data generation methods.

3.2.1 CNN Model Retraining Configuration. The Inception V3 CNN model was retrained via TensorFlow (Inception V3 was originally trained using ImageNet dataset [Russakovsky et al. 2015]). The retrained CNN configuration used was based on Google [2018]. For our classification tasks we only had to retrain the last bottleneck layer of the CNN with the new image categories. The training data input vector size is $300 \times 300 \times 3$ elements. Random distortion of training data (brightness, scale, and cropping) was not utilized, as the generated point cloud 2D image data is fairly uniform in terms of visual elements (point size, color, and opacity). The predicted classification accuracy with this approach is 81.3%, using 4000 training steps with a learning rate of 0.01. We use a linear softmax function for generating the classification probability scores for the input image data. The specific hyper-parameters used for the retraining of the last bottleneck layer of the Inception V3 CNN are included in Table 1, along with the recommended default hyper-parameters for complete retraining. For practical fine-tuning of the hyper-parameters, other possible configurations are described by Bengio [2012].

3.3 Visualization Features

We use Three.js for the client-side rendering system. It is based on WebGL and allows for the use of the OpenGL ES 2.0 and 3.0 API specifications within a compatible web-browser [Cabello et al. 2010]. The use of high-level JavaScript classes hides underlying WebGL code and allows for advanced real-time 3D features such as model loading (including PLY file format support for 3D point clouds), scene navigation, 3D data structures (octrees), and GPU-based rendering. Further, it features a flexible scene-management system where each component of the scene is added to a scene graph and accessed using a hierarchical function call system. The rendering of the point clouds is enabled through the use of a built-in point material shader. While the shader code for default materials is not exposed, input parameters such as point size and opacity can be adjusted. Three.js also supports loading the COLLADA 3D file format for visualizing the 3D placeholder models. We also implemented debug rendering of the octree and the associated valid nodes for easier verification output. One limit of Three.js for visualizing point clouds is the lack of support for out-of-core rendering of massive amounts of point-cloud data, thus it can only be used to visualize point-cloud scenes in real-time with approximately 1 million points, without resorting to the use of more sophisticated scene and memory management methods [Richter et al. 2015; Schütz 2016]. While we do use an octree for classification purposes, we have not focused on using it as an acceleration structure for rendering large point-cloud datasets.

Table 1: Default model retraining hyper-parameters and the specific hyper-parameters used for retraining the last bottleneck layer of the Inception V3 CNN.

Default/Retrained	Hyper-Parameter	Value
Default	RMSPROP_DECAY	0.9
Default	MOMENTUM	0.9
Default	RMSPROP_EPSILON	1.0
Default	INITIAL_LEARNING_RATE	0.1
Default	NUM_EPOCHS_PER_DECAY	30.0
Default	LEARNING_RATE_DECAY_FACTOR	0.16
Retrained	RETRAINING_LEARNING_RATE	0.01
Retrained	BOTTLENECK_TENSOR_SIZE	2048
Retrained	MODEL_INPUT_WIDTH	300
Retrained	MODEL_INPUT_HEIGHT	300
Retrained	MODEL_INPUT_DEPTH	3
Retrained	TRAINING_STEPS	4000
Retrained	TESTING %	10%
Retrained	VALIDATION %	10%
Retrained	EVAL_STEP_INTERVAL	10
Retrained	TRAIN_BATCH_SIZE	100
Retrained	TEST_BATCH_SIZE	1
Retrained	VALIDATION_BATCH_SIZE	100

4 RESULTS AND DISCUSSION

This section presents the preliminary results that we have obtained by testing our approach. We tested for the general accuracy of the classification model, the visualization accuracy (how well the distinctions between different object types are made visually), segmentation and reconstruction results, and finally the basic performance. Our approach addresses the key issues stated in Section 1.1 by:

1. Allowing users to classify scans of indoor offices using a retrained model able to classify specific object types.
2. Our implementation supports point-cloud resolutions that can typically be acquired using commodity hardware (e.g., mobile phones with a depth-sensor camera).
3. Users can load and classify different point clouds multiple times without having to reset the system or adjust parameters unless needed to do so.
4. Service-oriented architecture can be extended to include collaborative and centralized documentation access.

Table 2: True positive classifications for each of the 6 test scenes for each octree complexity category in the two in and out-of-core test datasets. The average number of generated nodes for each of the six scenes for the given (low, medium, or high) octree complexity is included.

Datset	Average Nodes	Test Scenes	True Positive
In-core_Set_1	42	6	5
In-core_Set_2	49	6	5
In-core_Set_3	55	6	5
In-core_Set_4	87	6	6
In-core_Set_5	100	6	5
In-core_Set_6	111	6	6
Out-of-core_Set_1	35	6	4
Out-of-core_Set_2	49	6	3
Out-of-core_Set_3	56	6	3
Out-of-core_Set_4	70	6	2
Out-of-core_Set_5	97	6	4
Out-of-core_Set_6	121	6	1

Table 3: Set 1 correspondence between octree complexity and the classification percentage for each number of object types in the scene (N = Number of valid generated nodes, S = Sofa, T = Table and C = chair).

Model	N	%S	%T	%C	Points
Set_1_Scene_1	23/27	4.34	17.39	78.26	203894
Set_1_Scene_1	30/34	6.7	26.7	67.7	203894
Set_1_Scene_1	31/40	3.22	12.9	83.87	203894
Set_1_Scene_2	80/100	16.25	23.75	60	640662
Set_1_Scene_2	91/115	24.17	24.17	51.64	640662
Set_1_Scene_2	101/130	31.68	17.82	50.5	640662
Set_1_Scene_3	3/14	0	33.3	66.6	4244
Set_1_Scene_3	5/22	0	20	80	4244
Set_1_Scene_3	6/28	0	33.3	66.6	4244
Set_1_Scene_4	6/37	16.6	50	33.3	12524
Set_1_Scene_4	14/49	0	21.4	78.6	12524
Set_1_Scene_4	14/53	0	28.57	71.43	12524

Table 4: Set 2 correspondence between octree complexity and the classification percentage for each number of object types in the scene (N = Number of valid generated nodes, S = Sofa, T = Table and C = chair).

Model	N	%S	%T	%C	Points
Set_2_Scene_1	28/34	3.6	64.28	32.14	589638
Set_2_Scene_1	13/22	7.7	61.53	30.77	589638
Set_2_Scene_1	9/14	0	55.55	44.44	589638
Set_2_Scene_2	12/20	58.33	33.33	8.33	212947
Set_2_Scene_2	7/9	85.71	0	14.3	212947
Set_2_Scene_2	4/6	75	0	25	212947
Set_2_Scene_3	13/21	46.15	38.46	15.38	32893
Set_2_Scene_3	10/14	40	40	20	32893
Set_2_Scene_3	8/11	25	50	25	32893
Set_2_Scene_4	16/23	12.5	25	62.5	10974
Set_2_Scene_4	7/10	0	14.28	85.71	10974
Set_2_Scene_4	6/6	0	10	90	10974

4.1 Classification Model Accuracy

We have aggregated the data from the 72 different test cases, divided into 36 test cases using the in-core data and 36 using the out-of-core data for testing (Table 2). We used 6 different test scenes for each dataset at three different octree complexities (low, medium, and high complexity). This was tested with the initial version of retrained CNN model (using randomly generated camera node positions for generating the training data). We used two different point-cloud datasets for testing the accuracy of the classification method. In this paper we presented the true-positive classification accuracy results only for the initial version of the classification model. The first dataset was composed of the original in-core training data, but edited so that multiple objects were merged in the same file, thus essentially creating a new dataset. Whether or not the set is classified correctly, is determined by comparing the classification percentage of a specific object classification based on the actual number of such objects in the scene. The test scenes created using the in-core data have an 88.8% classification success rate, while the test scenes created using out-of-core data have a significantly lower classification success rate of 42.2%. For the worse classification rate for the out-of-core data, it shows that our system currently works best for classification of objects containing similar visual features as the training data. Therefore, our approach is not suitable for general classification. Since most office buildings have specific furniture styles, in reality for each use case our system would need to be retrained for specific office building furniture.

4.2 Empirical Results

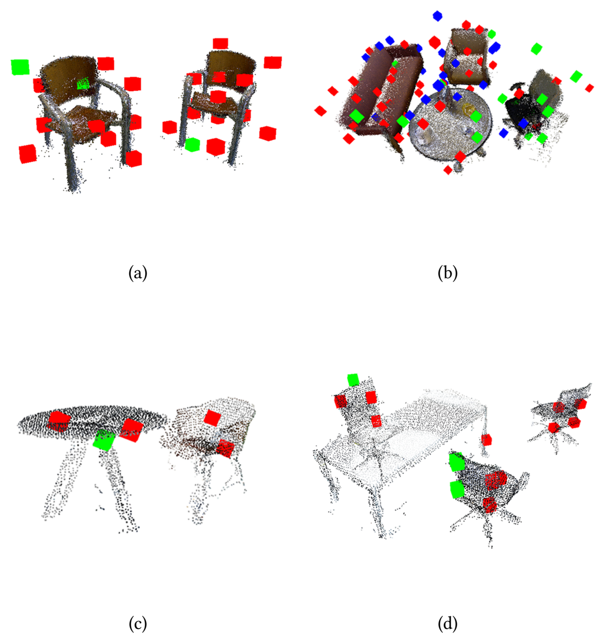


Figure 6: Set 1 empirical results (red nodes = chair, green nodes = table and blue nodes = sofa): (a) Set_1_Scene_1 (n = 31/40 valid nodes), (b) Set_1_Scene_2 (n = 91/115 valid nodes), (c) Set_1_ Scene_3 (n = 5/22 valid nodes) and (d) Set_1_Scene_4 (n = 14/49 valid nodes).

The presented empirical results show that the image-based classification system is able to provide a sufficient description of the composition of objects in the scene (Figure 6 and Figure 7, Table 3 and Table 4). The empirical results were obtained using two versions of the retrained CNN model. The first version is the same CNN model used to obtain the classification model accuracy results in Section 4.1 (training data created using only randomly generated camera node positions). The second version uses both the

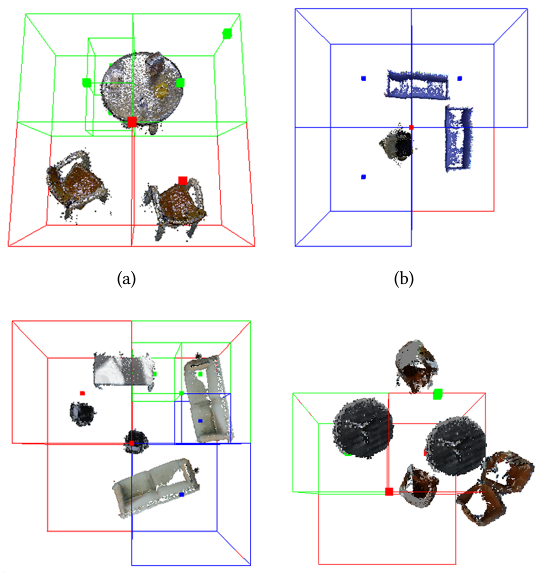

(a) (b) (c) (d)

Figure 7: Set 2 empirical results (red nodes = chair, green nodes = table, and blue nodes = sofa). The bounding boxes of valid nodes of the generated octree are rendered, in order to show the lower octree complexity used for classification. Portions of outer octree bounding boxes have been removed to show better focus on the visual classification result in the center. (a) Set_2_ Scene_1 (n = 9/14 valid nodes), (b) Set_2_ Scene_2 (n = 7/9 valid nodes), (c) Set_2_Scene_3 (n = 8/11 valid nodes) and (d) Set_2_ Scene_4 (n = 6/6 valid nodes).

randomly generated camera-node positions and those captured from the octree object representation. The two sets of data used for the empirical results have the following properties and test configurations:

1. *Set_1_Scene_1* and *Set_1_Scene_2* were created using in-core data, while *Set_1_Scene_3* and *Set_1_Scene_4* were created using out-of-core data, and using the initial version of the retrained CNN model.
2. *Set 1* results were acquired using a more complex octree (more nodes for classification).
3. *Set 2* is composed also of four test scenes, but all four test scenes were created using in-core data.
4. *Set 2* results were acquired using a less complex octree, and using the second version of the retrained CNN model.

The test data we used for the empirical results consists of variations of the three different furniture types, with different spatial arrangements and rotations of multiple objects in a given scene. The results from *Set 1* show that increasing the density of the octree and the number of generated nodes does not guarantee increase in the accuracy of the classification method. Tables 2 and 3 provide a comparison of the obtained results. We can conclude from the comparison of the two set results that a sparser octree is better suited for classification. The points contained in these sparser and larger node volumes are easier to distinguish based on their color and spatial distribution, rather than a sparser point sample that would be captured for classification if using a higher resolution octree with higher segmentation. All of the test data was edited prior to classification to remove walls, ceilings, floors, and other non-classifiable objects. For sparse point clouds we found that increasing the point size during rasterization provides a better classification result. This approach works fairly well if the overall shape of the object can be preserved. We also observed empirically that there was not a noticeable difference in the classification results due to using two different versions of the retrained CNN models. This is possibly due to using a smaller sampler size of training images per category (less than 1000).

4.2.1 Segmentation, Reconstruction and Performance Results. Our segmentation approach allows for point clusters from classified nodes to be extracted and exported as 3D point coordinates in the XYZ file format. The reconstruction of classified point clouds is possible by placement of 3D furniture models. The placement of these 3D models is based on the calculated *k*-means centers, and allows us to provide a better visual approximation of the spatial arrangement of furniture objects. Figure 8 illustrates examples of the reconstruction and segmentation methods. In terms of performance, we have measured preliminary computation performance. We can classify a scene of less than 50 nodes and 30 000 points within two minutes average. The development desktop computer used is a commodity PC with an Intel Core i5-6500 CPU at 3.2GHz, 8GB RAM and an NVIDIA GeForce GT 630 graphics card with 2GB of dedicated video memory.

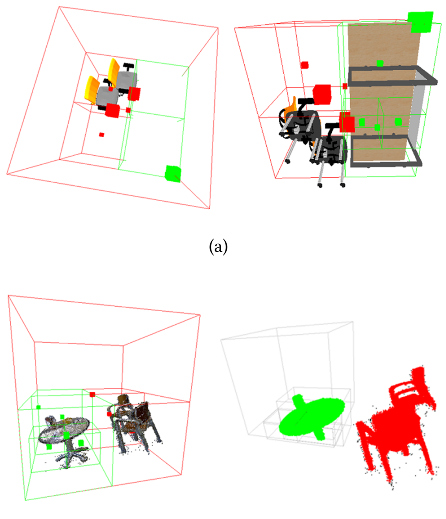

(a) (b)

Figure 8: (a) Reconstruction by placement of 3D models at *k*-means centers (larger cubes, models obtained from SketchUp 3D Warehouse), and (b) example segmentation of a desk object (green point cluster).

5 CONCLUSIONS AND FUTURE RESEARCH

In this paper we have demonstrated the feasibility of our idea for automatic deriving of indoor office spaces from point clouds for potential FM and BIM use. Our image-based classification approach can be used for fast labeling and extraction approximations of indoor spaces. Our approach also allows for the classification of image data, streaming, and viewing of the results using a service-oriented approach. Potential for parallel computation of convolutions and max pooling, as well as using multiple layers utilizing GPU and cloud computing, will be investigated for CNN model retraining (along with feature map merging and using larger sample sizes for object category training data). We also want to include the use of a binary coverage map in addition to checking the RGB values to detect faces for brightly colored surfaces and sparse point clusters that may otherwise be marked as invalid. Finally, we want to expand our client web application to allow multiple users to upload, classify, inspect and annotate point cloud scans of office interiors.

ACKNOWLEDGMENTS

This work was funded by the Research School on *Service-Oriented Systems Engineering* of the Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

Engin Burak Anil, Pingbo Tang, Burcu Akinci, and Daniel Huber. 2013. Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data. *Automation in Construction* 35 (2013), 507–516.

M Bassier, M Bonduel, B Van Genechten, and M Vergauwen. 2017. Segmentation of Large Unstructured Point Clouds Using Octree-Based Region Growing and Conditional Random Fields. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2017), 25–30.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*. Springer, 437–478.

Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. 2017. Unstructured point cloud semantic labeling using deep segmentation networks. In *Eurographics Workshop on 3D Object Retrieval*, Vol. 2. 8.

Ricardo Cabello et al. 2010. Three. js. *URL: https://github. com/mrdoob/three. js* (2010).

Zhangjie Cao, Qixing Huang, and Karthik Ramani. 2017. 3D Object Classification via Spherical Projections. *arXiv preprint arXiv:1712.04426* (2017).

A Dietze, Marcel Klomann, Y Jung, Michael Englert, S Rieger, A Rehberger, S Hau, and Paul Grimm. 2017. SMULGRAS: a platform for smart multicodal graphics search. In *Proceedings of the 22nd International Conference on 3D Web Technology*. ACM, 17.

Andrey Dimitrov and Mani Golparvar-Fard. 2015. Segmentation of building point cloud models including detailed architectural/structural features and MEP systems. *Automation in Construction* 51 (2015), 32–45.

Sören Discher, Rico Richter, Matthias Trapp, and Jürgen Döllner. 2018. Service-Oriented Processing and Analysis of Massive Point Clouds in Geoinformation Management. In *Service Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management*, Jürgen Döllner, Markus Jobst, and Peter Schmitz (Eds.). Springer.

Charles M Eastman, Chuck Eastman, Paul Teicholz, and Rafael Sacks. 2011. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons.

Martin Fischer, John Haymaker, and Kathleen Liston. 2003. Benefits of 3D and 4D models for facility managers and AEC service providers. *4D CAD and visualization in construction-developments and applications* (2003), 1–32.

Mark Froehlich, Salman Azhar, and Matthew Vanture. 2017. An Investigation of Google Tango® Tablet for Low Cost 3D Scanning. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, Vol. 34. Vilnius Gediminas Technical University, Department of Construction Economics & Property, 1–8.

Google. 2018. TensorFlow for Poets. (2018). https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/

Michael Grieves. 2014. Digital twin: Manufacturing excellence through virtual factory replication. *White paper* (2014).

Benjamin Hagedorn and Jürgen Döllner. 2007. High-level web service for 3D building information visualization and analysis. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*. ACM, Article 8, 8:1–8:8 pages.

Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. 2017. Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 20.

Gómez J. 2018. IRML Source Code and Database repository. (2018). https://github.com/jvgomez/irml

Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. 2016. RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints. *arXiv preprint arXiv:1603.06208* (2016).

David Kincaid. 1994. Integrated facility management. *Facilities* 12, 8 (1994), 20–23.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. 2011. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 1817–1824.

Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. 2014. Industry 4.0. *Business & Information Systems Engineering* 6, 4 (2014), 239–242.

Wan-Li Lee, Meng-Han Tsai, Cheng-Hsuan Yang, Jhih-Ren Juang, and Jui-Yu Su. 2016. V3DM+: BIM interactive collaboration system for facility management. *Visualization in Engineering* 4 (2016), 5.

Yanxin Ma, Bin Zheng, Yulan Guo, Yinjie Lei, and Jun Zhang. 2017. Boosting Multi-view Convolutional Neural Networks for 3D Object Recognition via View Saliency. In *Chinese Conference on Image and Graphics Technologies*. Springer, 199–209.

Sebastian Ochmann, Richard Vock, Raoul Wessel, and Reinhard Klein. 2016. Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics* 54 (2016), 94–103.

EA Pärn, DJ Edwards, and MCP Sing. 2017. The building information modelling trajectory in facilities management: A review. *Automation in Construction* 75 (2017), 45–55.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* 1, 2 (2017), 77–85.

Tan Qu and Wei Sun. 2015. Usage of 3D Point Cloud Data in BIM (Building Information Modelling): Current Applications and Challenges. *Journal of Civil Engineering and Architecture* 9, 11 (2015), 1269–1278.

Rico Richter, Sören Discher, and Jürgen Döllner. 2015. Out-of-core visualization of classified 3d point clouds. In *3D Geoinformation Science*. Springer, 227–242.

Kathy O Roper and Richard P Payant. 2014. *The facility management handbook*. AMACOM Div American Mgmt Assn.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

Markus Schütz. 2016. Potree: Rendering large point clouds in web browsers. *Technische Universität Wien, Wieden̈* (2016).

Timothy Scully, Jozef Doboš, Timo Sturm, and Yvonne Jung. 2015. 3drepo. io: building the next generation Web3D repository with AngularJS and X3DOM. In *Proceedings of the 20th international conference on 3D web technology*. ACM, 235–243.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.

Pingbo Tang, Daniel Huber, Burcu Akinci, Robert Lipman, and Alan Lytle. 2010. Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in construction* 19, 7 (2010), 829–843.

Paul Teicholz et al. 2013. *BIM for facility managers*. John Wiley & Sons.

Rebekka Volk, Julian Stengel, and Frank Schultmann. 2014. Building Information Modeling (BIM) for existing buildings - Literature review and future needs. *Automation in construction* 38 (2014), 109–127.

Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. 2017b. Dominant Set Clustering and Pooling for Multi-View 3D Object Recognition. In *Proceedings of British Machine Vision Conference (BMVC)*. 12.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017a. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 72:1–72:11.

Xuehan Xiong, Antonio Adan, Burcu Akinci, and Daniel Huber. 2013. Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction* 31 (2013), 325–337.

Hu Zhao, Zoltán Nagy, Daren Thomas, and Arno Schlueter. 2015. Service-oriented architecture for data exchange between a building information model and a building energy model. In *Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale*. LESO-PB, EPFL, 761–766.