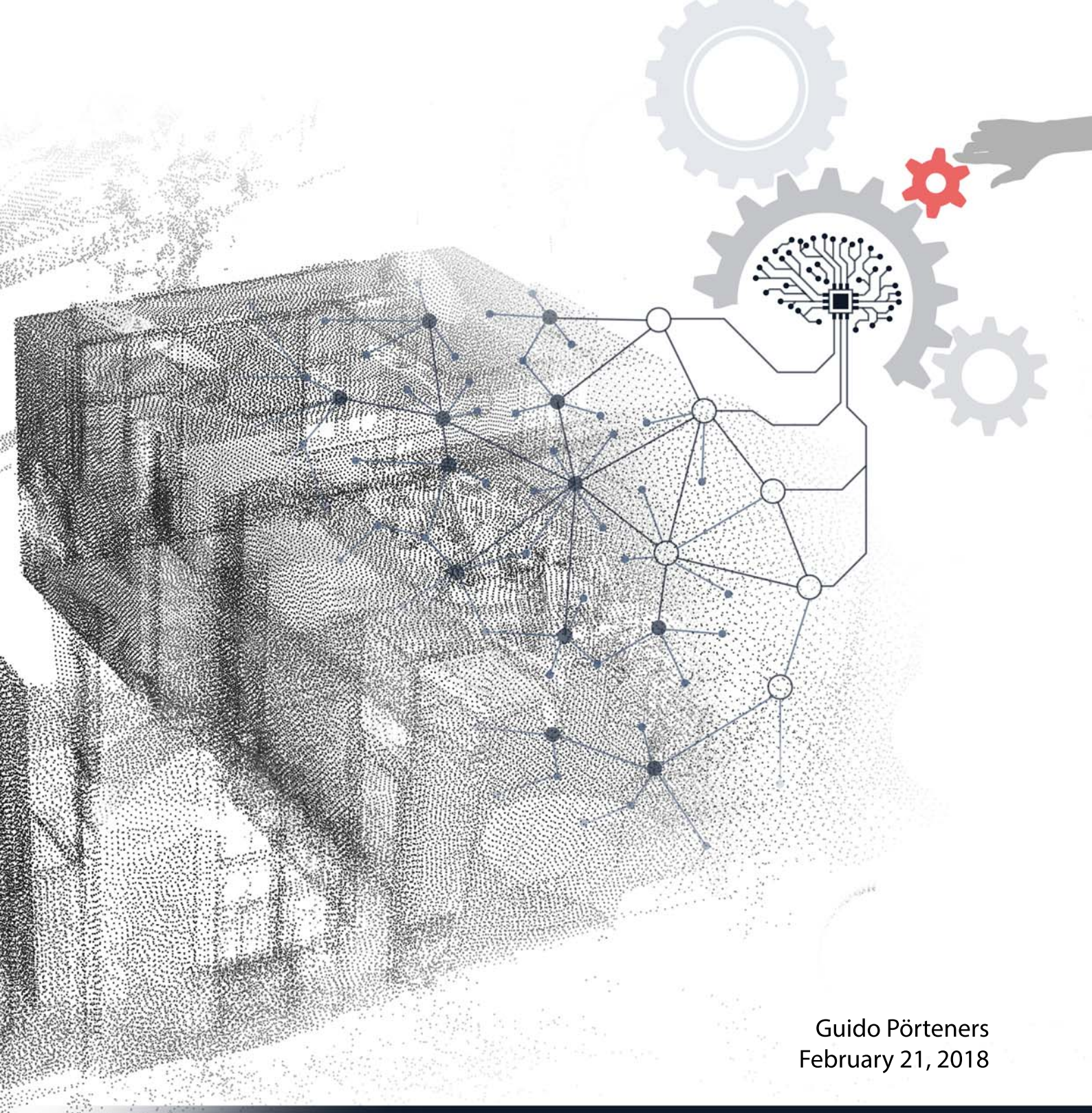


Guido Pörteners
February 21, 2018

Towards the automation of semantically enriching point cloud data sets

An interactive method incorporating machine learning for fast semantic classification of building indoor scenes

MSc Construction Management & Engineering
Eindhoven University of Technology

Towards the automation of semantically enriching point cloud data sets

An interactive method incorporating machine learning for fast semantic classification of building indoor scenes

Author

Name	Guido Pörteners
Student number	0909713
Email	guidoport@hotmail.com
University	Eindhoven University of Technology
Master program	Construction Management & Engineering

Graduation Committee

Chairman	Prof. Dr. ir. Bauke de Vries
1st supervisor	Prof. Dr. Jakob Beetz
2nd supervisor	Thomas Krijnen

Thesis defense date	21-02-2018

Technische Universiteit Eindhoven University of Technology

Preface

I am very happy to present my thesis that has evolved from my graduation project for the Master program Construction Management & Engineering at the Eindhoven University of Technology. The past months have been full of experience and gaining knowledge from out of my comfort zone including the associated mental challenges that came with it, but I am happy to say it was quite the journey and I am pleased with the result!

The topic I selected as a graduation project evolved from my interest in the automation of processes in the AEC industry, and especially because the world around us is evolving very quickly where automation is implemented in almost any form imaginable. However, when looking at construction management practices, the amount of opportunities for automation is almost endless. Although, when you look a little closer it becomes clear why, as the beauty of construction projects is in their complexity, but is in this regard also their curse. The idea in this thesis originated from fully-automatic progress monitoring on construction sites, but it rapidly became clear that the underlying processes for doing so, are complex puzzles on their own, to most of which belong to individual research department. The knowledge and experience I have gained from these other departments, ready to be applied for construction management are invaluable to me, and have kindled my enthusiasm towards automation even more.

Of course I would like to thank my supervisors. Especially Thomas Krijnen for sticking with me till the end as my (nearly) solo supervisor, I would like to thank him for the steady coaching and feedback during meetings throughout my graduation project, and the interesting, sometimes even psychological discussions about machine learning and point cloud interpretation we have had. Also I want to thank Jakob Beetz for the good discussions we have had during the first few months, as those have been very valuable for the formation of my idea and for refining my graduation topic.

Last but not least, I want to thank my family for the supporting base they always have been, and I want to thank my friends for the endless fun times, which created many awesome memories during my graduation and life as a student!

Table of Contents

I. Summary

Current practical implementations of point cloud data within the AEC industry and construction management sector are very limited, partially attributed to point cloud data being a primitive data representation and the modest amount of research for point cloud data enrichment. The uprising of BIM in the last decade stimulated an impulse to many new ideas, research and developments to aid construction management practices, but even though there are many use cases for point cloud data – with Scan-to-BIM, comparing the as-built and as-designed situation for progress monitoring, analysis of structural deviations and quality control being the most notable – manual processes and workflows require significant experience and time. Point cloud datasets need semantic information to be useful, but the current workflows for doing so, where segmentation and classification being the crucial ones, seem to be intensive, experience reliant and open for interpretation.

There is a limited amount of literature that proposes workflows for automatically segmenting and classifying point cloud datasets, which use hierarchical schemes and contextual reasoning. However, these methods are inflexible and error prone, and break down fast as buildings are not standard (e.g. walls are not always straight), building elements are often ambiguous and can converge into one another (e.g. conjunctions between beams and columns) and interpretation of the user and the consideration of their goals is not taken into account (e.g. different level of details or specific classification categories).

Fortunately, machine learning and its' recent re-introduction and fast adaptation within the fields of computer vision and object recognition has introduced a powerful way of automation. More recently, deep learning and neural networks are most notable for their performance, both in the scientific community (such as Genomics, Oncology, Biology, Finance and Marketing) and practical implications (such as image searching engines, spam filters, data mining or autonomous vehicles). However, research for the automatic segmentation and classification of point cloud datasets is quite limited. Current State-of-the-Art work, such as PointNet++ by Qi et al. (2017) and PointCNN by Li et al. (2018) appear to be very promising, but are exceptionally complex for adaptation to produce useful results regarding the use cases within the AEC industry. Additionally, these methods are static and do not involve any user involvement during or after the process, and their incorporation of using contextual and semantic relationships for classification is poorly understood.

The objective of this thesis project is to develop a hybrid method for segmenting and classifying point cloud datasets for the AEC industry, which combines both machine learning and user interaction to exploit their advantages and to balance out their disadvantages.

As machine learning method, a deep neural network is used and acts as a base model for succeeding user involvement. It is expected that deep learning picks up on contextual and semantic relationships between the different classes. Furthermore, the dataset used for learning and testing is the Byg72 dataset from the DURAARK online data repository, which included 22 indoor scans of raw point cloud data. To be able to train and test, the raw data is manually classified to distinguish 10 different class categories: beams, ceilings, chairs,

columns, doors, floors, occlusion, tables, walls and windows. Point cloud datasets rely heavily on xyz-values in their representation, however when used to train a machine learning model, the slightest change in rotation or scale would result in erroneous segmentation and classification results. Therefore, the dataset is enriched with features that are pose invariant and provide additional information for proper learning of class distinguishability. The respective features are point normal vectors, curvature and point feature histograms (FPFH), which enrich data points with vital information of their surrounding point neighborhoods.

The deep learning model is trained and tested and evaluated by two test cases. Unfortunately, the result of the trained model on the test cases are rather lower than expected, as predicted accuracies are 33% and 15% over all class predictions. Through thorough analysis, it seemed that the model struggles to distinguish floors, tables and ceilings in particular, which is sensible as their respective features share significant similarities. Furthermore, the neural network is quite sensitive to noise, which is coherent as normal vector orientation, curvature and FPFH are highly dependent of their point neighborhood. Also, the deep neural network has not been able to incorporate context as initially assumed, as it does not learn contextual or geometrical relationships between different classes and their respective features.

The deep learning model outputs a categorical distribution for predictions, where the class with the highest probability is assigned to that class. However, the underlying probabilities of the remaining classes are still valuable information. The second part of the method introduces a region-growing algorithm that is able to produce region-based selection and segmentation, and introduces the user interaction into the hybrid method as a complementation to machine learning.

While the initial predictions of the deep neural network are of poor quality, analysis shows that the underlying soft probabilities of the other classes are often not that far off, and therefore provides a valuable new set of parameters to quickly select specific construction elements. However, the practical results are mixed as there are also significant limitations. The region-growing works well on classes that are easily distinguishable, such as walls, floors, tables, ceilings and windows to a certain extent, and it provides a manual but quick solution to the issue of ceiling, table and floor distinguishability. Additionally, regions can sometimes contain patches of other object or element classes, but are conceptually easy to solve by introducing region-de-growing and adjusting the parameters accordingly. Another limitation regarding practicality, is that initial seed selection is vital to the quality of region-growing output. Additionally, it gives varied results, as parameters for region-growing are different from class to class, and currently a matter of trial-and-error for proper region selection.

All in all, while the research and tool development in this thesis give mixed results, the idea is proven to be feasible and therefore showing promise that it can serve as an initiator and cornerstone for further research and developments regarding hybrid systems for the semantic enrichment of point cloud datasets in the AEC industry. To expand on a broader notion, the application of deep learning for object recognition and appropriate classification is still an on-going subject in research. Furthermore, hybrid methods are still very under-represented, while they show significant potential and benefit as an extension to machine learning for practicality.

II. Samenvatting

De huidige praktische implementaties van pointcloud datasets ter assistentie van de bouwmanagement sector is behoorlijk beperkt. Dit komt doordat pointclouds een erg primitieve vorm van data zijn die in beginsel geen intelligente informatie bevatten, en er weinig onderzoek is gedaan om deze data te prepareren om de bruikbaarheid te verbeteren. Sinds de opkomst van BIM en de groeiende vraag naar digitalisering en automatisering in de bouwsector, is er echter wel vraag naar een efficiënte manier om de huidige situatie van gebouwen in kaart te brengen. Binnen de bouwmanagement sector hebben pointclouds grote potentie, zoals het transformeren van een pointcloud naar een volwaardig BIM model, en het vergelijken van de geplande situatie en gerealiseerde situatie van een gebouw waarmee bouwprocesvoortgang bewaakt kan worden, kritieke structurele afwijkingen ontdekt kunnen worden en kwaliteitscontroles kunnen worden uitgevoerd. De processen die hiervoor ontwikkeld zijn nemen echter veel tijd in beslag en zijn sterk afhankelijk van de ervaring van de modeleur. Voor het bewerken van pointclouds en met name het inbrengen van semantiek om de praktijkvoorbeelden die hierboven beschreven zijn te verwezenlijken, zijn segmentatie en classificatie cruciale taken.

In de huidige literatuur is er onderzoek naar gedaan om deze taken te automatiseren, en zijn er enkele methodes en processen ontwikkeld om pointcloud data automatisch te segmenteren en classificeren. Deze methodes blijken echter niet flexibel en zelfs ongeschikt, aangezien gebouwen sterk t.o.v. elkaar kunnen verschillen, bouwelementen vaak slecht te onderscheiden zijn, en zelfs in elkaar kunnen overgaan (zoals bij kolommen en balken). Daarnaast hebben modelleurs baat bij flexibelere methodes om segmentatie en classificatie naar wens uit te voeren, maar worden bij huidige methodes niet centraal gesteld.

Sinds de herintroductie van machine learning in het afgelopen decennium, zijn de ontwikkelingen in de automatisering vanuit de computer wetenschappen en object herkenning in een stroomversnelling geraakt. Met name deep learning en de daarbij horende familie van neurale netwerken zijn het meest opvallend in hun prestaties, zowel binnen de wetenschap (zoals in de Genomica, Biologie, Oncologie, Marketing en Econometrie) als in de praktijk (zoals zoekmachines voor afbeeldingen, spam filters, data mining en autonome voortuigen). Onderzoek voor het automatisch segmenteren en classificeren van point cloud datasets is hierin echter gelimiteerd. Huidig grensverleggend onderzoek zoals PointNet++ door Qi et al. (2017) en PointCNN door Li et al. (2018) zien er veelbelovend uit, maar zijn te complex om deze methodes aan te passen om goede resultaten te boeken voor potentiele use cases binnen de bouwmanagementsector. Daarnaast bieden deze methodes geen gebruikersinteracties aan om gemakkelijk aanpassingen te maken, er het is niet duidelijk hoe deze methodes nou daadwerkelijk semantisch classificeren kunnen leren.

Het doel van dit onderzoek is om een hybride methode te ontwikkelen voor het segmenteren en classificeren van point cloud datasets specifiek voor de bouwmanagementsector, waarbij machine learning en gebruikersinteractie worden gecombineerd om te profiteren van hun voordelen en hun nadelen tegen elkaar uit te spelen.

Een deep neural network wordt gebruikt als machine learning methode die als basis fungeert voor gebruikersinteractie. Van het neural network wordt verwacht dat deze tijdens het trainen, contextuele en semantische relaties tussen verschillende element categorieën zal oppikken. Om het neural network te trainen en tests uit te voeren, wordt de Byg72 point cloud dataset gebruikt van de DURAARK online data archief welke bestaat uit 22 scans van binnenruimtes van een gebouw. Daarnaast is deze dataset volledig handmatig geclassificeerd, waarbij onderscheid is gemaakt tussen 10 categorieën: balk, plafond, stoel, kolom, deur, vloer, occlusie, tafel, muur en raam. Point clouds worden in hun 3D representatie gevisualiseerd op basis van hun xyz-coordinaten, maar zijn onbruikbaar om direct te gebruiken voor machine learning, aangezien rotaties en verschalingen in onjuiste classificaties leidt. Daarom worden er features toegevoegd aan de point cloud dataset die punten lokaal karakteriseren en robuust zijn tegen transformaties, die vervolgens ook het neural network ondersteunen om categorieën te onderscheiden. De desbetreffende features zijn normaal vectoren, curvature en point feature histograms (FPFH).

Het deep neural network is getraind door het overgrote deel van de dataset, waarbij er twee scènes worden overgelaten ten gebruik voor de test cases. De resultaten van het getrainde neural network op de automatische classificatie van de twee test cases is lager uitgevallen dan verwacht, met een nauwkeurigheid van 33% en 15% over alle categorieën gemiddeld. Uit analyse blijkt dat het model moeite heeft met het onderscheiden van vloeren, plafonds en tafels, veroorzaakt doordat deze categorieën features bevatten die sterk op elkaar lijken. Daarnaast blijkt het model erg gevoelig voor ruis, die oneffenheden veroorzaakt in het berekenen van de normaal vectoren, curvature en FPFH. Daarnaast is gebleken dat tegen verwachtingen in, het neural network geen contextuele of semantische relaties tussen elementcategorieën heeft geleerd.

Het neural network geeft voor elk datapunt een categorische distributie als output, waarbij aan elke categorie een voorspellingspercentage wordt toegekend aan elk afzonderlijke categorie. Daarbij geldt de categorie met de hoogste voorspelling als desbetreffende categorie van dat punt. De resterende voorspellingspercentages zijn echter nuttig voor de gebruikersinteractie. Het tweede gedeelte van de methode gebruikt een region-growing algoritme, dat snel slimme selecties kan maken door middel van een voorselectie en een uitbreidingsmethode met gebruiker gedefinieerde drempelwaardes. Dit onderdeel introduceert de gebruikersinteractie aan de hybride methode die als toevoeging fungeert aan machine learning.

Alhoewel de classificaties die door het neural network gedaan zijn van mindere kwaliteit zijn, blijkt uit analyse dat vaak de onderliggende voorspellingspercentages van de werkelijke categorieën niet ver van de voorspelde categorieën liggen. Dat resulteert in een nieuwe waardevolle set aan attributen die gebruikt kunnen worden voor het alsnog slim selecteren van specifieke bouwelementen. Uit praktische resultaten van de test cases blijkt dat de methode goed werkt bij elementen die goed te onderscheiden zijn, zoals wanden, vloeren en tafels. Het geeft ook een simpele oplossing voor het onderscheidingsprobleem van de tafel, vloer en plafond categorie. Daarnaast kunnen selecties nauwkeuriger worden gemaakt door het conceptuele region-de-growing, waarbij oneffenheden makkelijk gedeselecteerd kunnen

worden. Het is echter wel van belang om een goede voorselectie te maken, aangezien de kwaliteit van het neural network invloed heeft op de kwaliteit van region-growing. Daarnaast zijn de parameters voor selecteren verschillend per categorie, en zijn de waarden een kwestie van trial-and-error.

Het doel van het onderzoek was om een methode te ontwikkelen voor het segmenteren en classificeren voor point cloud datasets, waarbij machine learning en gebruikersinteractie zijn gecombineerd, en geconcludeerd kan worden dat het doel behaald is. Alhoewel de tests gemixte resultaten opleveren, is het oorspronkelijke idee wel degelijk uitvoerbaar en toont daarbij potentie. Daarnaast dient het onderzoek als fundament voor vervolgonderzoek naar het semantisch verrijken van point cloud datasets voor de bouwsector, en laat als relevante toevoeging zien dat hybride methodes die machine learning en gebruikersinteractie combineren, wel degelijk praktische potentie hebben.

III. Abstract (max. 350 words)

Point clouds are a valuable datatype for the AEC industry and construction management sector, as they can be used for accurate Scan-to-BIM processes, and allow construction managers to compare the as-built and as-designed situation of construction projects. This provides practical implications such as progress monitoring, analysis of structural deviations and quality control. However before being pragmatic, raw point cloud data needs to be enriched with semantic information, for which segmentation and classification are key activities. Manual processes require a significant experience and time investment, and automation is therefore desired. This is a contemporary studied problem in the field of construction management and several methods are proposed that incorporate contextual reasoning and a hierarchical workflow. However, these methods break down as buildings are vastly different and construction elements are often ambiguous. Therefore, researchers turned to methods incorporating machine learning for object recognition, specifically deep learning and neural networks, as they appear to closely resemble human reasoning and learning contextual and semantic relationships. Unfortunately, current state-of-the-art methods incorporating deep learning are not tailored to the AEC industry, and because of their complexity, are poorly understood and impractical for construction management practitioners to work with.

This research proposes a hybrid method for segmenting and classifying point cloud datasets for the AEC industry, which combines both machine learning and user interaction to exploit their advantages and to balance out their disadvantages. This is achieved by training a deep neural network as a base model, which outputs a valuable set of parameters which can be used for a region-growing algorithm to operate as a smart and quick selection tool for selecting specific element class categories.

The developed method gives mixed results, as the performance of the base model is lower than initially expected, which directly affects the quality of selections produced by region-growing. Also, the deep neural network has been unable to lean contextual or semantic relationships between different element classes, contrasting initial expectations. However, the hybrid smart selection tool does perform adequate on certain elements, and with further refinement, could show significant potential. Furthermore, the research has shown that the combination of user interaction and machine learning can be definitely powerful.

IV. List of abbreviations

2D	2-dimensional
2.5D	2.5-dimensional, common in range images consisting of a 2D matrix of points, although depth information is included
3D	3-dimensional
AEC	Architecture, Engineering & Construction
AI	Artificial intelligence
ANN	Artificial neural network
API	Application programming interface
BIM	Building information modeling
CNN	Convolutional neural network
CRF	Conditional random field
FPFH	Fast point-feature histograms
IFC	Industry foundation class
k-NN	K nearest neighbor
LIDAR	Light detection and ranging
MLP	Multi-layer perceptron
MRF	Markov random field
n-D	n-dimensional
PCA	Principal component analysis
PCL	Point cloud library
RANSAC	Random sample consensus
RGB-D sensor	Sensor that can capture color and depth information
RNN	Recurrent neural network
SHOT	Signatures of Histograms of Orientations
SVM	Support vector machine

V. List of figures

VI. List of tables

1. Introduction

With the introduction of BIM in the AEC sector, many new developments and applications in this digital environment have emerged. One of those developments is laser scanning. Laser scanning has been developed in the 1960s for the purpose of accurately recreating the surface of objects, but it has only been adopted by the engineering industry during the late 90's, where Cyra Technologies (today Leica Geosystems) produced one of the first scanners used by engineers. However, storage space and bandwidth has always been a limiting factor in the continued development of laser scanning, since the data produced by laser scanners, depending of the accuracy, is often huge (Ebrahim, 2014). Additionally, laser scanning has limited factors for accurately digitalizing objects, such as objects that have mirroring, shiny or transparent properties and occlusions that are unwanted in the final output of a laser scan: point clouds.

Point clouds are geometric data structures where every point in the Euclidian space $\mathbb{R}^3$ is represented by their corresponding XYZ coordinates, usually on a Cartesian coordinate system, but not limited to other coordinate systems. Although point clouds give a very accurate geometric representation of objects, such as buildings, construction sites or the geographical environment around it, they are not widely adopted yet by the AEC industry. The main advantage of using point clouds (and also laser scanning) in the AEC sector is to have a digital model of the actual and precise representation of the interior and exterior of a building, often including its environment. This digital documentation can be utilized through several use cases, defined by Beetz et al. (2015): (1) Deposit 3D architectural objects, (2) search and retrieve archived objects, (3) maintain a semantic digital archive, (4) detect differences between the planning state and as-built state, (5) monitor the evolution of a structure over time and monitoring the construction process, (6) identify similar objects within a point-cloud scan, (7) plan, document and verify retrofitting/energy renovations, (8) exploit contextual information for urban planning, and (9) enrich BIM/IFC models with metadata from a repository.

The utilization of point clouds in some of these use cases can have a direct positive impact on the AEC industry. Monitoring the construction progress is a critical factor in the success for projects to be delivered on time, within budget and within all requirements and finishing (pursued quality). Unfortunately, this seems to be also the most difficult task, because the construction process is a very complex process, in which all activities are interdependent of each other (Zhang et al., 2009). Current approaches for quality control on construction sites are time-consuming and rely heavily on repeated data input which includes onsite inspection and data collection, data analysis and potential defects detection, communication with relevant project participant for resolution, and execution (Wang et al., 2015). This is very labor intensive, and due to human error (whether it being data gathering, reporting or communicating), delays and cost overruns are not managed correctly. The time it takes to identify the differences between the as-planned and as-built model is proportional to the cost and the difficulty to implement measures to deal with these differences (Navon & Shpatnitsky, 2005). Regardless of the introduction of BIM to the AEC industry, costly mistakes are still

made, and there are many opportunities to assist managers with a tool to ease and smooth the case of process monitoring. There is always room for improvement.

Besides the cases of using point clouds in comparing and integrating the as-built and as-planned models of construction projects, they can also be used for other use cases such as historical buildings. These buildings often have ornaments, statues and decorations that have parametric surfaces beyond the possibilities that IFC offers. Remodeling these ornaments is usually a painstaking process for which conventional BIM tools are unsuitable, since the documentation of these elements besides dimensions (in the form of meshes or polygons) do not include any other information. Another practical use case for using point clouds is the monitoring of structural health, where point clouds are used to analyze deviations in bearing structures and to check for deformations in structures that impact the structural robustness of a building, as pointed out by Krijnen & Beetz (2017). In a broader sense, the aforementioned use cases are associated with Scan-to-BIM, transforming laser scans to BIM models, a process that has gained traction on the practical level only recently.

There is a wide variety of software available, both commercial and open-source, for the manipulation (as in: meshing, vectoring, classifying and segmentation) of point clouds, such as: LIS, VisionLidar, RealityCapture, CloudCompare, Point Cloud Library and VRMesh, although none of these are specific for the segmentation and annotation of construction elements, or usable for the variety and intensity of clutter that is apparent in buildings currently under construction. Buildings in general contain a large amount of clutter and occlusion, and can be considered on several levels of detail. Furthermore, the Scan-to-BIM process expects a finalized BIM model that is empty, i.e. without furniture, people, supporting elements during construction etc. The issue with current software is that they do not contain tools to deal with these problems quickly and accurately. Additionally considering construction elements, they are not ambiguous, and sometimes it is not clear where an element stops, and another element continues, further impeding the scan-to-BIM process.

Considering the aforementioned limitation in the use cases and software, there is a need for a tool that can automatically segment point clouds, tailored to point cloud data taken from building scenes.

Automation can be done with the use of algorithms to segment and classify point clouds. There are several techniques available for segmentation and classification. The most easy to understand and widely used segmentation techniques are: (1) edge-based, by detecting edges in a point cloud by analyzing rapid changes in local surface properties such as point normal vectors and curvature, and cluster points within these edges; (2) region growing, by assigning seed points within the point cloud and allowing them to grow by assigning neighboring points with similar properties into the same group; (3) segmentation by model fitting, such as Hough Transform (HT) (Ballard, 1981) and Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981), which are algorithms that fit primitive shapes into point cloud data and segment the points which conform to that specific shape. Additionally, (4) machine learning such as Support Vector Machines (SVM) or Artificial Neural Networks (ANN), which learn to segment point clouds based on training data.

Once the segmentation part has been completed, these segmentations can be classified and be given a semantic meaning to a cluster of previously meaningless xyz-points. Classification is usually done through two different approaches: supervised and unsupervised. The difference between supervised and unsupervised learning is that supervised learning (classification) uses input from a training set of data, which is used to predict the classification of separate similar data, while cluster analysis groups data with (near-)similar attributes or parameters.

Another approach for classification is the interactive approach where the user can interactively have input in the classification process with feedback signals. User interaction is already apparent in the medical field for discovering potential cancer cells in MRI scans or in the field of geology for the classification and segmentation on geology maps. Additionally, as mentioned by Weinmann et al. (2015), most approaches suggested in research of segmentation and classification processes of point clouds consider both separately. However, features and spatial context are interrelated and it is interesting to explore how to connect both processes.

Furthermore, machine learning is part of Artificial Intelligence (AI) which is currently a hot topic applied in a lot of ongoing development, but also stimulating completely new developments. Although it has been around since the early 1960s, the introduction of the internet, the exponential growth in data storage and processing, and the ever growing set of open source tools and research papers, has made it easier than ever to use machine learning to automate classification processes.

1.1. Problem definition and research gap

Point clouds are a very primitive data type which need to be enriched with information to be truly useful for practical implementations in the construction management industry. To add semantic information to point cloud datasets, segmentation and classification are key processes. The automation of these processes is highly desired, but are mostly done manually as humans have a relative significant advantage in accuracy. This is caused by the fact that the human eye in combination with previous experience and knowledge about the geometries of (construction) elements, including the matter of deciding which level of detail is appropriate for segmentation and classification and cognitive imagination, is very accurate in deciding where an object begins and ends, and what an object is or could be, based on the relation to its' surroundings and position in the 3D space. Especially in the building industry, where there is only a limited standardization in the geometries of construction elements and everything on all levels of detail are usually custom made, makes it seemingly impossible to segment and classify point clouds by hand.

The question "what makes a [...insert construction element...], a [...insert construction element...]?" is an interesting one for asking oneself to decide which cluster of points belong to a certain element. For example, a column has a cylindrical or horizontal cuboid-like shape, with a floor attached at the bottom, and a ceiling or beam attached at the top. Simultaneously, a wall has the floor attached to its bottom, and the ceiling to its top, and has the property of

vertically segmenting empty spaces within the point clouds also known as rooms. This seems to be a trivial way of thinking for humans, but for computers, this is unfortunately not the case. This kind of reasoning is a part of a human's cognitive ability including imagination that computers inherently lack.

The fact that it is impossible to automate the classification process of point cloud data to be 100% accurate, or to classify the point cloud datasets rather quickly manually, impacts the accuracy and includes the need for simplicity that is required for several of the aforementioned use cases. This leaves the matter to be a two-tailed problem: on one hand there is a need for automation for the segmentation and classification of point clouds, as the blunt computational speed by computers is essential for dealing with a large amount of data formats that are in addition large in storage size. On the other hand, there is a need for human input in semantic classification, for which accurate segmentation seems to be the breaking factor that is needed for effectively applying point clouds in the AEC industry.

Although there is plenty of research that has been conducted for creating several processes and algorithms to segment point clouds, classification in this regard is a topic that is still in its infant state. Research in classification can be distinguished in two parts: one part aims for developing a (machine learning) algorithm to classify point clouds based on their own use case. The other part of researchers optimize existent classification algorithms to gain the highest classification accuracy on the same dataset, the most prominent and widely known ModelNet40 and ModelNet10 dataset of Princeton University. A partial overview of the research conducted on these datasets is shown below in Table 1. This dataset contains objects from a list of the most common objects in daily environments. However, research conducted on these datasets are optimized on this dataset and none of the algorithms are specifically designed for the AEC industry.

Table 1: Princeton ModelNet40 leaderboard

ALGORITHM	MODELNET40 CLASSIFICATION (ACCURACY)	MODELNET40 RETRIEVAL (MAP)	MODELNET10 CLASSIFICATION (ACCURACY)	MODELNET10 RETRIEVAL (MAP)
ECC	83.2%		90.0%	
PANORAMA-NN	90.7%	83.5%	91.1%	87.4%
MVCNN-MULTIRES	91.4%			
FPNN	88.4%			
POINTNET	89.2%		77.6%	
MVCNN	90.1%			
WANG ET AL.	93.8%			
VOXNET	83%		92%	

To conclude, point cloud datasets seem to have promising applications, but their usefulness is limited because there is not much research done for automatically segmenting and classifying these point cloud datasets. There is a need for an interactive segmentation and classification method specifically designed for the AEC industry, which overcomes the

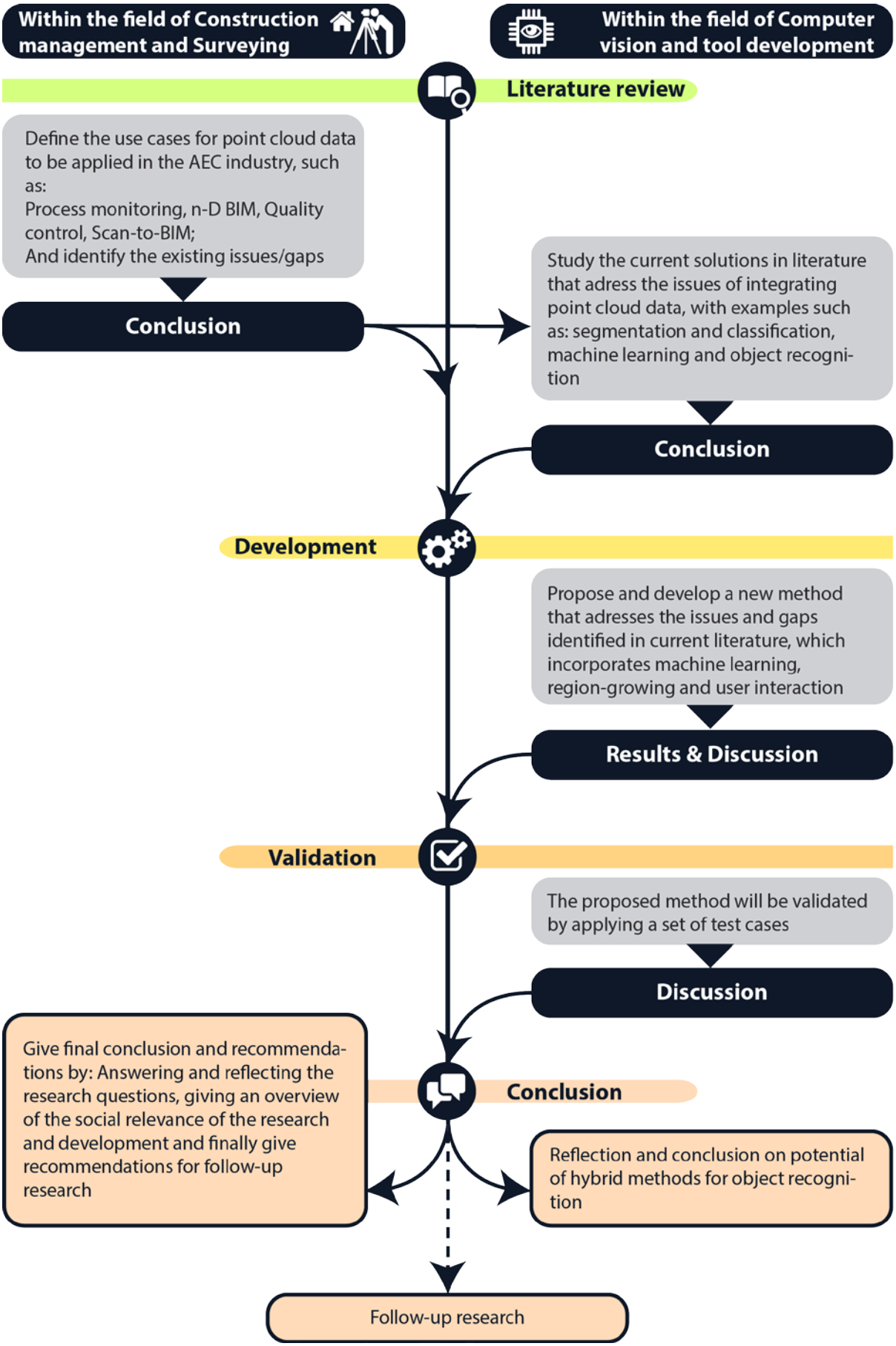
problems of inaccuracy in machine learning applications but takes advantage of the computational speed of these algorithms, and on the other hand involves the user interactively to increase both accuracy and keeping the freedom for users with their own goals and interpretations in mind. Moreover, current tools are designed for specific purposes and do not incorporate a broad functionality to classify point clouds automatically. A method designed for executing this process should be simple to use and open-source, for which it can be used by other research as a building block for further development purposes, and open to input, improvements and further development by research departments and industries.

1.2. Research scope

The research is divided into four stages: the literature review, the development of a new method, method validation and finally the conclusion. The literature review consists of two parts: the scope of the first part will lie in the field of construction management and surveying where the applicability and use cases for point clouds in the AEC industry are defined, and where the current issues for the incorporation of point clouds within construction management processes will be elaborated. The scope of the second part of the literature review lies within the field of computer vision and tool development, as the current methods in literature which incorporate machine learning, and classification and scene understanding, propose solutions for the issues determined from the first part of the literature review.

As a succession to literature review, which will elaborate throgoughly on the current research gaps and issues, a new method will be proposed that is tailored to the AEC industry and will introduce a way of incorporating user interaction for more flexible point cloud classification. The proposed method will be thoroughly explained and validated in the methodology chapter within the scope of computer vision and tool development.

As the fourth and last stage, the conclusion will give an answer to the research questions and will reflect on the new proposed method from a construction management point-of-view. Furthermore, it will include an explanation for the societal and practical relevance of the research of this thesis, and will further elaborate on the implications and possibilities of further research.


Within the field of Construction management and Surveying
Within the field of Computer vision and tool development
Literature review
Define the use cases for point cloud data to be applied in the AEC industry, such as:
Process monitoring, n-D BIM, Quality control, Scan-to-BIM;
And identify the existing issues/gaps
Conclusion
Study the current solutions in literature that adress the issues of integrating point cloud data, with examples such as: segmentation and classification, machine learning and object recognition
Conclusion
Development
Propose and develop a new method that adresses the issues and gaps identified in current literature, which incorporates machine learning, region-growing and user interaction
Results & Discussion
Validation
The proposed method will be validated by applying a set of test cases
Discussion
Give final conclusion and recommendations by: Answering and reflecting the research questions, giving an overview of the social relevance of the research and development and finally give recommendations for follow-up research
Conclusion
Reflection and conclusion on potential of hybrid methods for object recognition
Follow-up research

1.3. Research questions

The introduction to the topic of this research and the problem definition defined in section 1.1 motivates the following main research question:

1. **How can a point cloud scan be segmented and classified semi-automatically through machine learning, within the context of the AEC industry, while retaining user interaction?**

To answer this main research question, the following sub-questions are appropriate for a well-founded answer:

1. **What is the importance of using point cloud data regarding construction management?**
2. **What are the current techniques for automatic segmentation and classification of point cloud data?**
3. **To extend on sub-question 2, what role does machine learning have in this matter?**
4. **What is the current state-of-the-art regarding machine learning and deep learning for automatic classification of point cloud data in current literature?**
5. **What are the limitations of current methods for automatic point cloud segmentation and classification?**
6. **What is the added value of user interaction for point cloud segmentation and classification?**
7. **How can machine learning and user interaction be combined with the aim of increasing practicality?**

The following three paragraphs will further elaborate on answering these research questions, by using the "why, how, and what?" to define the research more clearly in its relevance, design and goal.

1.4. Relevance of research

The foremost relevance of this research and development project is bridging the gap between theory and practice, and more specifically the practice towards the possibility of incorporating the API in future developments or even software applications. The development part will tackle a multitude of problems that are relevant in the bigger picture. This bigger picture can be seen as a world quickly moving towards automation, but where the construction industry is still conventional and lacking behind. Digitally documenting a construction project including its environment can be seen as a complementation of BIM and be used to gain advantage in building processes, where time, money and pursued quality are the limiting factors (Golparvar-Fard, Peña-Mora, & Savarese, 2015). Many processes within a construction process can actually be automated or refined by using automation, taking away complexities and resulting in smoother processes and workflows.

Scientifically, this research and development project closes the gap that is currently existing in literature of the AEC industry, taking knowledge from existent research in computer science

and computer vision to apply this knowledge directly in the AEC industry and AEC industry related applications. It also aims to create a core foundation for follow-up research, as that is certainly needed, which will be elaborated later in the conclusions.

1.5. Research design

The research is split into four parts, with the conclusion and recommendations as the fourth part. The first part is the literature review, which situates the research focus, and holds a critical view against the already available and related research, and further identifies and explains the research gap in current literature. Through the literature review, it was concluded that the original research proposal that has been created, seemed to scratch only the surface of the underlying problems for answering the research questions, and identified even wider variety of surfaces that could benefit from answering the research questions in a narrower scope. Thus, changes have been made to the original research questions. The answers to these questions, including the fulfillment of the "What" of this research, creates a core foundation for follow-up research and further development in the automation of the AEC industry.

The second part of this research aims to develop a classification and segmentation tool specifically for the AEC industry, where the user can interact with the tool to gain a higher accuracy for successfully preparing point clouds for use cases such as comparing the as-built and as-planned situation, construction process monitoring, creating accurate point cloud construction element repositories or checking buildings for structural deviations or deficiencies. With a wide variety of new machine learning methods and an increase in computational power, new methods can be developed that are faster, smarter or more accurate.

The developed tool from the second part will be analyzed and validated through two test cases, where the results will be used for the discussion of these test cases. The last part will be the conclusion, based on the results of the developed tool, and will reflect on a broader view within the topic of semantic point cloud processing and reflect on hybrid methods that combine user interaction and machine learning within the field of object recognition.

1.6. Research goal and expected results

The goal of this research is to create an interactive segmentation and classification method for point cloud datasets, specifically designed for the AEC industry, which overcomes the problems of inaccuracy and relative inflexibility of machine learning applications but takes advantage of the computational speed of these algorithms. On the other hand, it involves the user interactively by producing results which give support in the freedom of interpretation and goals within the point cloud classification process. Meanwhile, the tool should be open-source and easy to use, closing the research gap defined in the introduction and elaborated in the literature review, and acts as an initiator and building block for further research and development purposes in practice. Additionally, the research provides the sufficient

instructions to use the tool for users that are familiar with the key concepts within computer vision and machine learning.

Due to the complexity and scope of this research problem, the goal of this research is NOT to explain the broad topic of automation in the AEC industry or give introductions to several important key concepts, such as algorithms, BIM or software applications and programming languages. There are many good tutorials, books and online sources available that can explain these concepts more clearly than this thesis.

1.7. Reading guide

This report is structured into four parts, with every part being a separate chapter. The first chapter gives the introduction to the topic, problem definition and research design. The second chapter is the literature review, where relevant literature and theoretical concepts will be explained and elaborated, and will end with a conclusion. The third chapter will demonstrate the methodology, which describes the proposed method, test cases, analyses and discussion. The fourth and last chapter of this thesis is the conclusion, which will include the scientific and social relevance of the research and will end with the recommendations for further research.

2. Literature review

The following chapter reviews the state-of-the-art literature and provides the necessary background that is related to the research problem. Additionally, it gives an overview of the methods that are already proven and proposed by other research. Finally, the last section concludes the limitations and further elaborates on the research gap that is existent in current literature.

2.1. Point clouds

A point cloud is a geometric data structure where data points in the Euclidian space $\mathbb{R}3$ are represented by their corresponding X, Y and Z-coordinates, most often representing the surface of objects. In most software, the Cartesian coordinate system is used for visualization, although more coordinate systems are in place for different purposes, such as the spherical and polar coordinate system for point cloud acquisition. Additionally, point clouds can contain additional data such as color information, normal vectors and curvature, making the point cloud an n-dimensional data structure.

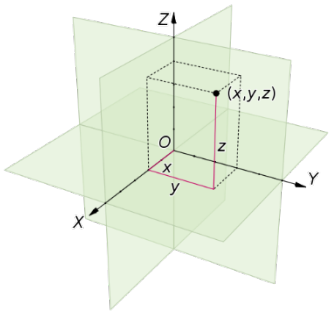


Figure 1: Illustration of the Cartesian coordinate system for 3D by Jorge Stolfi (2009)

There are several techniques for the acquisition of point clouds, such as laser scans (LIDAR), laser triangulation, stereo cameras, structured light cameras, time of flight cameras (ToF) and RGB-D cameras (such as the Microsoft Kinect). While LIDAR acquires data in true 3D information through laser pulsing, RGB-D cameras acquire point cloud data in a flat image format, with an added depth value for every pixel, hence the name RGB (color) and D (depth) image.

Additionally, point clouds (and depth information to be more precise) can be generated from images containing the same object or environment but with different viewpoints, or by using drone images or videos to capture larger objects. This technique is called structure from motion (SfM) or multi-view stereo method (MVS). As such, a depth image can be reconstructed, allowing for a 2.5D representation of the object, similar to that of RGB-D data. From these acquisition techniques, LIDAR is widely used in the AEC industry since it has significantly reduced in cost and allows for very accurate data acquisition. However, most LIDAR does not capture transparent or reflective objects, and therefore it is not possible to generalize which acquisition technique is better, as that depends on the goal of where the data is being used for.

An important distinction to make is the notion of an organized point cloud versus an unorganized point cloud. An organized point cloud is a set of arrays where a data point has a fixed position in that set and array, similar to pixels having a fixed position in a flat image.

Organized point clouds are mostly range images created by an RGB-D camera or other techniques that output an image with added depth information. An example is given in Figure 2, where on the left a segment of a panoramic RGB image is shown, and on the right side the same image with depth (or range) values is represented in greyscale, where black pixels are located close to the sensor and white pixels far from the sensor. Once more, a range image gives a 2.5D representation of the scene in this case, and to be shown in full 3D, every pixel is reconstructed in a 3D coordinate system based on the XY location in the image and the Z location (i.e. the depth/range value) with respect towards the sensor location. Thereafter, the resulting point cloud has to be scaled with a certain factor (usually included in the metadata of the image) to transform it to its true XYZ representation in meters.

Figure 2: RGB image with depth information from the Stanford 2D-3D Semantic Dataset by Armeni et al. (2017)

In contrast with organized point clouds, unorganized point clouds contain a set of data points that do not hold a fixed ordering in relation to each other, i.e. points do have a fixed position in the Euclidian space, but the arrangement of the data points in the set has no effect on the point cloud representation. To exemplify, every point p_i in a given unstructured point cloud dataset P I described by an array with the point coordinates $\langle x, y, z \rangle$, possible color information $\langle R, G, B \rangle$, point normal vectors $\langle u, v, w \rangle$, curvature $\langle \sigma \rangle$ and extendable to other features such as intensity and category. It is irrelevant if point p_i would be positioned before or after p_j in dataset P, since the position is given by the point coordinates $\langle x, y, z \rangle$.

The advantages of using organized point clouds is the efficient calculation of point normal vectors, as shown by Holz et al. (2011), faster computation times in nearest neighborhood searches (PCL, 2012), and their functionality within the machine learning domain for object recognition techniques, similar to that of regular images (Qi et al., 2016). However, combining multiple organized point cloud datasets is not possible without transforming the range image to full 3D, as every range image is normalized towards its respective sensor location, and it has limitations in incorporating contextual information regarding spatial relationships between objects and elements. This topic will be further elaborated on in section 2.3.

2.1.1. Point cloud processing

Since point clouds are mainly very large data structures that require large computational power, they can be efficiently visualized and modeled through the use of spatial index structures, such as R-trees (Guttman, 1984), k-d trees (Bentley, 1975) or octrees (Meagher, 1982). These structures divide the point cloud into hierarchical cells which contain subsets of the total point cloud. These cells are only visible when the visualization requires it, such that irrelevant points are not rendered (Krijnen & Beetz, 2017). Furthermore, Krijnen & Beetz (2017) remark that these acceleration trees are not tailored semantically to what the points represent, such that the cells defined by the spatial index structures cut through structures like walls, floors and other architectural spaces and elements. They note that for example in the IFC standard, a subdivision structure is used which divides the building in individual building storeys or spaces. This would be an interesting concept to translate to point clouds, as it would allow the possibility to quickly and efficiently show one segment of a point cloud that represents a building storey.

Another way of processing point cloud data is by using ‘voxels’. A voxel is a portmanteau of the word 'volume' and 'pixel', and allows for fast 3D visualization or rendering in games, since they can be efficiently stored in memory as they can have very little in-memory data. The same concept can be translated to point cloud data, where clusters of points can be grouped together in a voxel. A way of storing this ‘voxelized’ point cloud data is by using octrees, which is used in the work done by Vo et al. (2015) where they use voxelization for speeding up the point cloud segmentation process. Octrees recursively subdivide 3D space into eight octants, known as 'nodes'. For point clouds, these nodes contain a specified number of data points, and as such, dense point cloud chunks contain many small nodes. These nodes can be visualized in 3D by giving them volumetric properties, where each single node (the 'pixel') is given a volume dependent on the amount of parent nodes it contains.

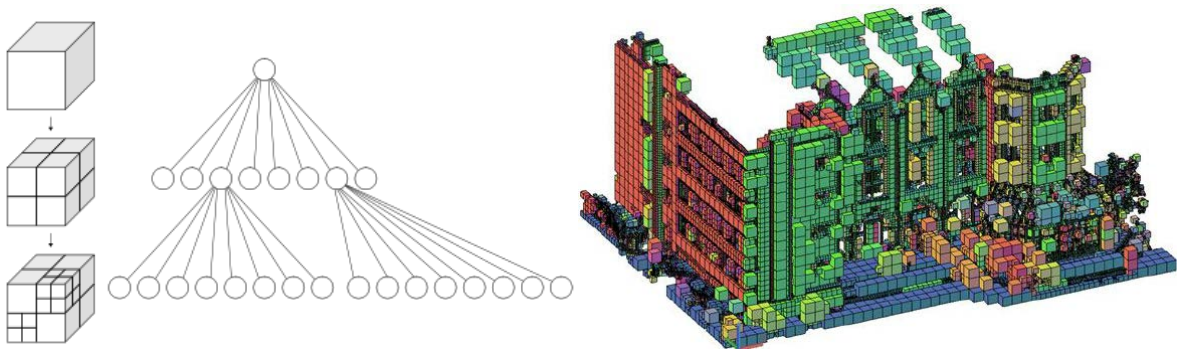


Figure 3: Example of octree decomposition and point cloud octree representation by Vo et al. (2015)

Segmenting a voxelized point cloud is a very fast process as shown in the work of Vo et al. (2015), since the parent voxel of a certain group of points entails similar properties of these points and the fact that neighboring voxels can be accessed quickly within the same parent node. However, the trade-off is that accuracy is exchanged for speed, as subtleties in point cloud data that characterize objects or elements are lost when voxel sizes are too large. Voxels are also used in classification algorithms for analyzing and visualizing medical scans such as in

MRIs or CTs, but also have gained popularity for training artificial neural networks (ANNs) due to the irregular data structure of point clouds, and the speed that voxels can generate. Examples for training machine learning algorithms with voxelized data, such as in Qi et al. (2016) and Li et al. (2016) will be discussed in section 2.4.

2.1.2. Applicability in the AEC industry

Point clouds give an accurate geometric representation of the environment, and with regard to buildings, give precise dimensions of construction elements and other objects, where in contrast, 3D BIM models provide the digital representation of how a building is planned to be built.

It is possible to compare both of these situations, to detect differences between the as-built (point cloud) and as-planned situation (BIM model) (Beetz, Krijnen, & Wessel, 2015). Beetz et al. (2015) define several use cases for the documentation of the as-built situation: (1) Deposit 3D architectural objects, (2) search and retrieve archived objects, (3) maintain semantic digital archive, (4) detect differences between planning state and as-built state, (5) monitor the evolution of a structure over time and monitoring the construction process, (6) identify similar objects within a point-cloud scan, (7) plan, document and verify retrofitting/energy renovations, (8) exploit contextual information for urban planning, and (9) enrich BIM/IFC models with metadata from a repository.

2.1.3. Connection to BIM and construction management

Now that as-planned 3D models are widely adopted in the AEC sector due to market demand and compliance towards national regulations, the demand for as-built models are increasing. As-planned models are used for a wide variety of applications such as project planning, analysis of energy consumption, structural analyses and facility management. Bassier et al. (2016), Wang et al. (2015), Tang et al. (2014), show several methods of transforming point clouds into BIM models for different purposes.

Simultaneously, Volk et al. (2014) acknowledge the fact that using BIM offers potential benefits, such as as-built heritage documentation, maintenance of warranty and service information, quality control, assessment and monitoring, energy and space management, emergency management, retrofit planning and potentially deconstruction processes. In Europe however, more than 80% of residential buildings have been built before 1990, and do not contain any documentation in BIM format. Consequently, Del Giudice and Osello (2013) point out that the need for renovation and refurbishment of cultural heritage and historical buildings is becoming more important than constructing new buildings. Therefore, the process of scan-to-BIM can help to capture the required information.

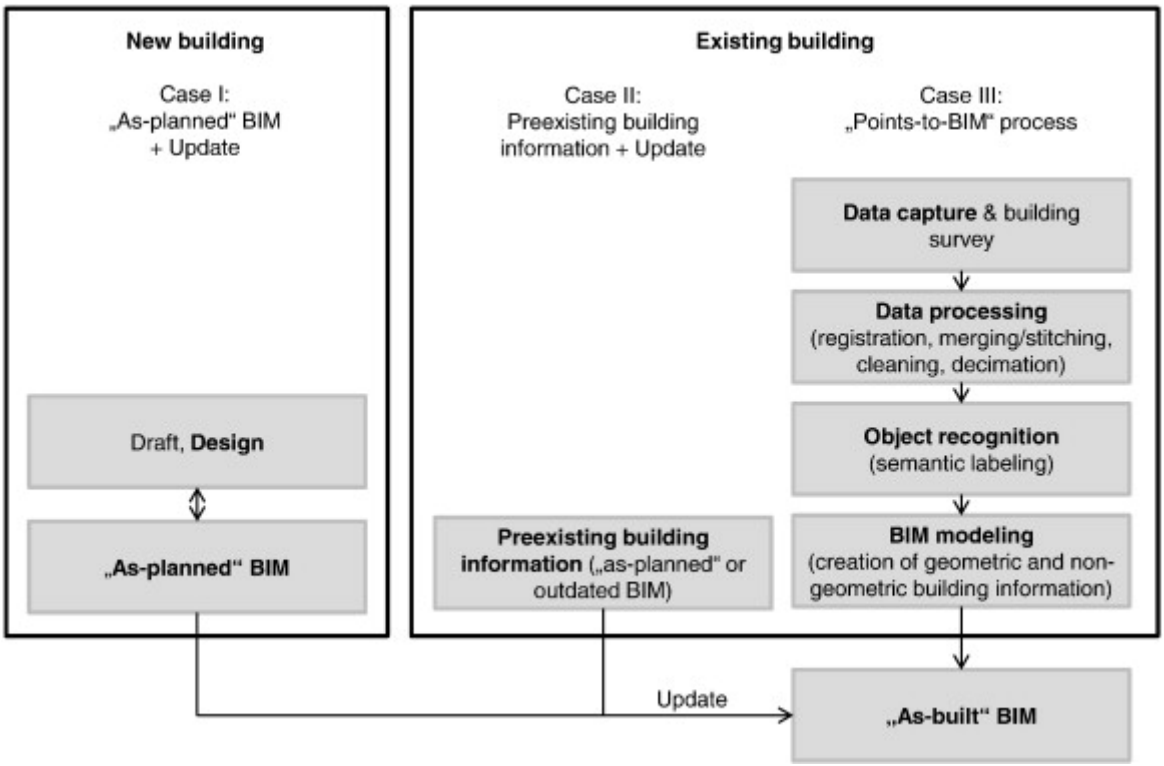


Figure 4: BIM creation processes for new and existing buildings by Volk et al. (2014)

Figure 4 represents the conversion workflow from building to as-built BIM. Data capture, data processing, object recognition and BIM modelling form the core of a complete as-built BIM documentation. There are several data capturing techniques available, with laser scanning, photogrammetry, RFID and barcode tagging being the most frequent ones. However, RFID and barcode tagging lack the interoperability with BIM and the main disadvantage of using laser scanning and photogrammetry are high operation costs (Volk et al., 2014).

Simultaneously, as pointed out by Tang et al. (2010), Zeibak-Shini et al. (2016) and Macher et al. (2017), transforming point clouds into 3D volumes is a long process to execute manually, and an inaccurate process to perform automatically, when point clouds contain noise and outliers. Even when surfaces are accurately extracted for the scan-to-BIM process, converting these surface models to volumetric models is not well explored for the problem of as-built modeling (Xiong, Adan, Akinci, & Huber, 2013). As-built modelling has a significant overlap with the field of geometry processing that incorporates algorithms for the volumetric analysis and reconstruction of (complex) 3D models. The issue is that generic processing tools designed for the Geometry Processing field cannot be applied immediately due to the high number of different elements and the immense amount of data points in building scenes.

Additionally, there is a large body of research within object recognition and related segmentation and classification processes for the field of Geomatics and urban scenes. However, as-built modeling requires a finer level of detail. Especially when complex building elements need to be distinguished, small details such as door handles for recognizing doors or window frames for detecting windows, are valuable information to improve the scan-to-BIM process. Therefore, conventional methods that are used for urban scenes cannot be applied directly onto the scan-to-BIM process (Patraucean et al., 2015). Even though object

recognition for as-built BIM is supposed to be a simpler problem than conventional object recognition in the Computer Vision field due to prior information available on the shape and distribution of the scene elements, as stated by Patraucean et al. (2015), the ambiguous interpretation of construction elements, and a lack of distinctive element boundaries result in generic object recognition tools from the Computer Vision field to fail such as seen in the work of Xiao & Furukawa (2012).

Other than the scan-to-BIM process, for most applications in the use cases of point cloud datasets mentioned by Beetz et al. (2015) in section 2.1.2, the conversion from acquisition to as-built 3D BIM model could be omitted. Several algorithms such as the Iterative Closest Point (ICP) algorithm inhabit sufficient practicality for point cloud and BIM model comparison, as shown in the work of Krijnen & Beetz (2017) where floor deviations can be detected. Additionally, the incorporation of point clouds into the IFC data format could possibly enrich the already existent BIM documentation. Both Belsky et al. (2016) and Krijnen & Beetz (2017) have proposed such an IFC schema extension and recognize the possibilities for enriching IFC with (semantic) point cloud data structures. Such an example includes the mapping of historical buildings which often contain ornaments, statues and other intricate decorations and details. Transforming these aspects is a painstaking process, for which documentation in a point cloud data format including semantic information, could prove to be a better substitute.

In other work, Zeibak-Shini et al. (2016) and Dong & Guo (2012) examine post-earthquake damage based on laser scanned data. They state that in the aftermath of an earthquake, structural engineers need to inspect the building, and have to classify it as safe, unsafe or dangerous, which is a tedious process. They state that laser scanning in combination with autonomous ground vehicles (AGV) proves to be a promising tool to inspect post-earthquake areas which are otherwise dangerous for human inspection. The major issue in this process is that point cloud data cannot be interpreted directly without information about the pre-existing building components of a building. While Dong & Guo (2012) only use aerial data to inspect roofs, Zeibak-Shini et al. (2016) further develop this idea on ground level where they compare the BIM model before the earthquake, with the post-earthquake model, created by transforming the point cloud data into a BIM model based on the semantic information given by the pre-earthquake BIM model.

2.1.4. Applicability in progress monitoring

To elaborate further on the applicability of integrating point clouds into construction management practices, progress monitoring can also benefit by detecting potential delays automatically. As stated by Golparvar-Fard et al. (2009), early detection of potential delay in a construction project is vital to its project management, which entails that project managers have to design, implement and maintain a systematic approach for monitoring such construction projects. This includes the case to identify, process and communicate these errors in an early stage between the as-planned state and the as-built state of a construction project. Wang et al. (2015), Kim et al. (2013), Turkan et al. (2012) and Bosché et al. (2015) further confirm this notion that these progress monitoring activities can be further refined by

introducing (semi-)automated processes for the comparison of the as-built and as-planned state. However, this process differs slightly from that of the scan-to-BIM process for the field of facility management, as point clouds are not required to be transformed into full 3D models for comparison.

Furthermore, it is possible to reconstruct sparse point clouds from pictures taken on construction sites without much effort, exemplified in the work of Golparvar-Fard et al. (2009). These pictures are converted to a point cloud through a Structure from Motion (SfM) method, and superimposed on the 4D BIM model. Consequently, the proposed method shows the completion rate in percentage, and uses augmented reality for real time process and quality control. However, the method is quite limited in the sense that it does not notify any deviations from the 4D BIM model, and where occlusion on construction sites such as cranes, construction workers, etc., give less accurate point clouds and consequently, less accurate process monitoring results. On a last note, the accuracy of the output of the proposed method is in direct relation with the amount of information acquired for input, i.e. sufficient pictures need to be collected to omit occlusion and to capture the entire construction site.

In related research, Turkan et al. (2012) fuses 3D object recognition algorithms with 4D schedule BIM data, and creates a feedback loop as shown in Figure 5. The 3D laser scan data provides the necessary information about the current as-built situation, whereas on the other hand, the 3D BIM model with integrated schedule information provides information for the as-planned situation. These situations are both compared and the progress is calculated from the recognition results. However, the feedback loop is limited as it only considers on-going work, thus object recognition and consequently progress calculation is only done for objects that are supposed to be completed.

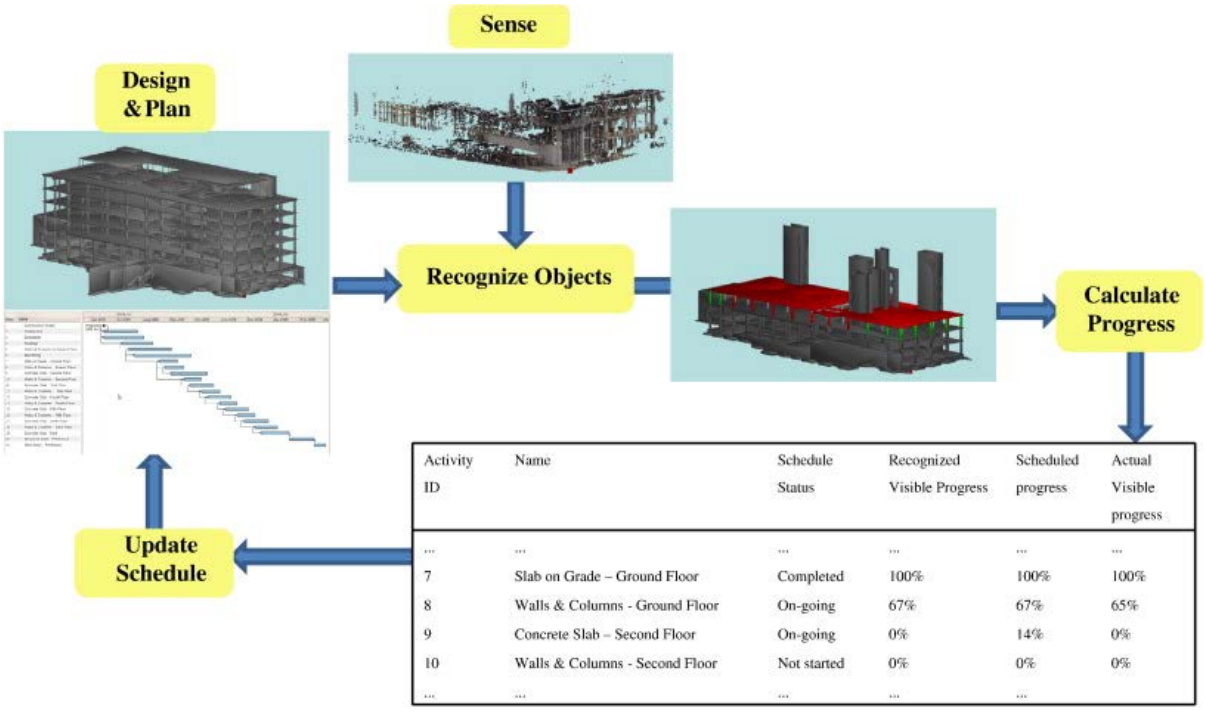


Figure 5: Procedure for automated progress calculation and schedule update by Turkan et al. (2012)

2.1.5. Conclusions regarding point cloud data in AEC industry

To conclude, there is a substantial amount of opportunities for incorporating point cloud data in construction management and the AEC industry in general. Point cloud data encompasses the ability to accurately describe the geometry of the as-is or as-built situation of construction work, existing buildings and/or their respective environment. The as-built situation can be used in a wide variety of use cases, with the most substantial ones being construction progress monitoring as studied in the works of Zhang et al. (2009), Golparvar-Fard et al. (2009), Turkan et al. (2012), Kim et al. (2013), Bosché et al. (2015), Han et al. (2015) and Gao et al. (2015), and Scan-to-BIM for mapping existing buildings as full 3D BIM models, such as studied in the work of Murphy et al. (2012), Xiong et al. (2013), Bassier et al. (2016) and Macher et al. (2017). According to Del Giudice and Osello (2013), the use of point cloud data for the scan-to-BIM process will become more prevalent as 80% of the buildings in Europe are built before 1990, and most probably do not contain any form of BIM of their current state, which would otherwise be useful for renovation, energy consumption analysis or creating a simple repository for facility management. Furthermore, point cloud data can be used for the management of structural damages as in the research of Dong & Guo (2012) and Zeibak-Shini et al. (2016), and be used to detect structural deficiencies as seen in the research of Krijnen & Beetz (2017).

However, transforming point cloud data to a BIM is a long process that is also heavily dependent on the interpretation and experience of the user, as stated by Tang et al. (2010), Zeibak-Shini et al. (2016) and Macher et al. (2017). Even when surfaces are accurately extracted for the scan-to-BIM process, the conversion to BIM is not well explored for the problem of as-built modeling (Xiong et al., 2013), and while there are several methods for urban scenes as in the work of Patraucean et al. (2015), they cannot be directly used for building indoor scenes. Additionally, while the conversion from point clouds to volumetric models has significant overlap with the field of geometric processing (or mesh processing), the processing tools designed for the Geometry Processing field cannot be applied directly due to the high number of different elements and the immense amount of data points in building scenes.

Also, it seems that in most work, such as in Krijnen & Beetz (2017), Golparvar-Fard et al. (2009) and Kim et al. (2013), the transformation of point cloud data into volumetric models can be omitted for the use cases such as progress monitoring, detecting structural deficiencies and enriching IFC/BIM data, but then the point cloud data needs to be enriched with additional meaningful information. There is a substantial amount of research done on point cloud data processing, such that it can carry sufficient information that makes the utility of point cloud data in the aforementioned use cases substantially easier. Most of this research comes from the field of Computer Science and Computer Vision, and is clarified in section 2.2 and section 2.3.

2.2. Introduction to relevant theory regarding point cloud data processing

The following section introduces several relevant topics that are used for the processing of point cloud data, in such a way that is becomes a usable concept for application in the AEC industry. Most relevant to this topic is introducing semantics to exploit the full capabilities of raw point cloud data, i.e. adding *meaning* to data or incorporating *relationships* between data that can be otherwise interpreted different ways. The following sections will further elaborate on what semantics entail, as it is a broad topic that can be incorporated in several was into point cloud data.

2.2.1. Machine learning

A method that is increasingly used for automating processes is machine learning. Originated in the field of Computer Science, it has widely spread to other fields of research as well, such as the fields of Medicine, Genomics, Biology, Finance, Robotics, Marketing, and has found its practicality in the automation of a wide range of tasks, with examples such as text, speech and object recognition, data mining, spam filtering, and advertising. Through the years, it has grown very close to the field of statistics, and is often termed as such. The following section will give a little background information about machine learning and the ways it is implemented for point cloud data.

The term machine learning was first coined in 1959, when it was a subfield of Artificial Intelligence (AI) and the study to pattern recognition to train machines to learn about their environment. Although since 1959, there have been two periods where the interest and funding in AI have hit rock bottom, also known as the "AI winters", the exponential growth in computing technology from the last two decennia allowed machine learning to develop further and faster than before. Machine learning is closely related to mathematical statistics, but machine learning deals with large complex datasets, where statistics would be otherwise impractical. Essentially, a supervised machine learning model requires three things: input data, examples of expected output and a way to check if the algorithm is doing a proper job. Therefore, a machine learning algorithm finds rules for meaningfully transforming data (whether it be segmenting, clustering, classifying etc.), and using a feedback system to adapt these rules accordingly (Chollet, 2017).

Machine learning methods can be divided into three categories, (1) supervised learning, (2) unsupervised leaning and (3) reinforcement learning. Choosing between either of these categories depends on the problem that one wants to solve (Bishop, 2006).

In supervised learning, the inner relations of the data that are being processed are unknown, but the output that is expected from the model is available. Regression and classification are both widely known supervised methods. As for the inputs, features are used that are believed to have properties that allow the model to correctly predict new data, thus selecting the right features is vital for a good prediction.

With unsupervised learning, it is not yet known what the outcome of the data is, but it is suspected that there are some kind of correlations or relationships within the dataset. Clustering is a form of unsupervised learning, where data with similar properties or near-similar properties are clustered within a group.

Reinforcement learning is a newer form of machine learning, where the machine learning model receives no correct input-output datasets, but learns to perform a certain way by reward signals. This resembles how humans and animals learn, although in a limited form, where the machine learning program gets rewarded when it gives a correct output.

A newer type of machine learning is deep learning which is gaining increased interest in research on multiple fields, which transcends over all three machine learning categories. As has been mentioned before, machine learning algorithms find rules for meaningfully transforming data. Deep learning puts an emphasis on learning multiple layers of rules for meaningfully transforming data, where the 'deep' in deep learning refers to these multiple layers. These layered representations are learned via "neural networks", and will be explained in section 2.4.1 and section 3.2.1. Machine learning and, more specifically deep learning, is considered to be a hands-on discipline as there is no “one-solution-fits-all” (Chollet, 2017). This is confirmed by the wide variety of models created and used for solving problems requiring deep learning, including the amount and variety of parameters that can be incorporated, which will be discussed in Chapter 3.

The aspect of machine learning presented in this thesis is mostly limited to the automation of object recognition (i.e. segmentation through bounding boxes) and object classification. Both object recognition and object classification, while in essence being independent tasks, share large overlap and are highly related. While object detection is responsible for the *identification* of objects or elements, object classification is responsible for the *categorization* of such identified object or elements. To give an example concerning point cloud data sets, Figure 6 gives an example of a chair, where the yellow points are identified as being a *separate entity* distinguished from the wall (blue points) and floor (green points), the segmented cluster of points is classified to a certain *category* of elements or objects.

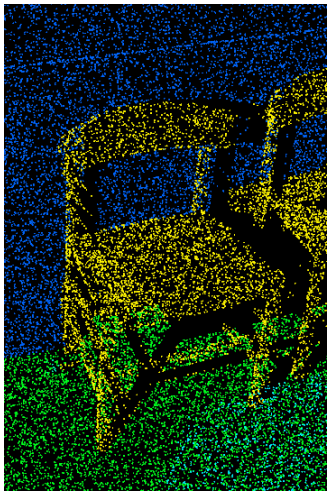


Figure 6: Example of point cloud object recognition and classification

For this example, the boundary between detection and classification is not very clear, as detection searches for distinctive features (e.g. the chair has four legs) that characterize a certain category, while classification tries to discover *objects instances* through the same distinctive features that are used for segmentation (e.g. the four legs perpendicular on the green floor points), and defines the (geometric) boundaries accordingly. This confirms the notion that both tasks are highly related, although keep in mind that both tasks can be considered independent entirely, e.g. segmentation can be done based on local properties such as edges, without the notion of being ‘semantic’. This will be further elaborated on in section 2.3 and section 2.4, as there are

many techniques for segmentation and classification, and methods in literature sometimes consider both tasks separately, while other methods do not.

To return on the topic of machine learning, the tasks of object recognition and object classification can be automated to a certain extent, with notable results in images, whether it be an online image search engine or self-driving cars. However, most segmentation techniques and classification techniques do not originate from machine learning, but from statistics for the segmentation and classification of datasets. For this reason, there are several algorithms proposed for these tasks, which will be elaborated on in section 2.3.

2.2.2. Segmentation and classification of point cloud datasets

The following section will give a small overview of segmentation and classification techniques, where they have originated and what their place is in current literature.

Segmentation techniques have been around since the early 1980s, with the Random Sample Consensus (RANSAC) algorithm originally designed by Fischler & Bolles (1981), and the generalized Hough Transform (HT) algorithm by Ballas (1981), as a basis for many point cloud segmentation methods in literature. To give an overview of segmentation methods, Grilli et al. (2017) created a diagram shown in Figure 7.

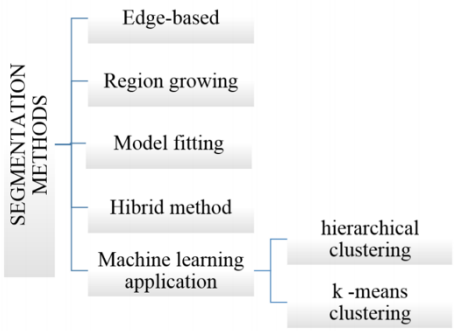


Figure 7: Overview of segmentation methods by Grilli et al. (2017)

The RANSAC algorithm has initially been introduced for the estimation of parameters in datasets for statistics, which contains outliers in such a way that the ordinary least squares method is unsuitable for line fitting for this given problem. Simultaneously, this algorithm can be used for outlier detection in datasets. Where the previous example demonstrates the use of RANSAC in 2D data, the same concept can be translated to 3D, where the same algorithm can be used for fitting planes and other geometric primitives. To recur on point clouds, the same plane fitting method can be used to iteratively detect planes in point cloud datasets, and subsequently cluster points within this plane and segment this cluster from the remainder of the point cloud data. The RANSAC algorithm proves to be robust in the presence of outliers, which is a beneficial attribute regarding the case that point cloud datasets can contain a significant amount of outliers.

However, as stated by Raguram et al. (2008), the RANSAC algorithm operates as a hypothesize-and-verify framework (i.e. a minimal subset of input points are randomly selected, for which a plane is fitted, and evaluated on the entire dataset), results in issues when it is applied in complex indoor or outdoor scenes for extracting planes. When a hypothesis is formed for a plane, the randomly selected points for hypothesis generation might not be part of the same plane, resulting in spurious planes. This is especially evident when the residue of data that is not yet parsed, gradually becomes smaller throughout iterations (Xu et al. 2015) (Li et al., 2017) (Awwad et al., 2010).

Generalized Hough Transform (HT) is an algorithm used mostly in 2D computer vision for shape and edge detection. Without going into too much detail, the initial state detects potential edges by drawing imaginary lines through every pair of potential edge points, which are then plotted in a parameter space (hence the name transform). Based on the parameter space plots, potential edge points are grouped that belong to the similar edge or shape by performing a voting procedure, where the edges are drawn as lines or polygons. The benefit of applying HT is that edges that contain missing points or noisy points, can still be found through this voting procedure. Points falling within these edges are considered to be part of the same cluster, thus segmented. While the same procedure can be applied in a 3D space, the extra dimension causes a computational complexity that is often infeasible, and therefore the technique is not often applied in 3D point cloud datasets.

Affiliated methods such as region-growing algorithms as originally proposed by Besl et al. (1988) and clustering methods based on k-means (not to be confused with k nearest neighbor method) presented by MacQueen (1967) are similar techniques for segmentation purposes. Region-growing is a simple technique that involves an initial selection of seed points based on a user specified criteria, for which a threshold is defined and for which neighbors are found with properties within this threshold. The neighboring points within the threshold are grouped with the initial selection of seed points and the neighboring search is repeated for, until no new points are added to the region.

To label patches segmented from techniques described above, one can use classification techniques that originated from both statistics as well as machine learning. Statistical learning and machine learning regarding this matter overlap each other considerably, although there is a subtle difference. Statistics focusses more on the hypothesis formulation and the assumptions that are formulated with it, while machine learning emphasizes more on the flexibility of a model, and its characteristic of letting algorithms discover relations between data. Although both statistics and machine learning come from different departments, both have become very close in its functionality through the years. To demonstrate, an overview of terminologies representing the same meaning between the two fields has been made by Wasserman (2004), visualized in Table 2.

Table 2: Overview of the same concepts in statistics and machine learning regarding the same concept

Statistics	**Computer Science**
Estimation	Learning
Classification	Supervised learning
Regression	Supervised learning
Clustering	Unsupervised learning
Covariates	Features
Data	Training sample

This thesis focusses on classification through machine learning, and therefore corresponding terminology is also used. In either way, classification is the identification of the category that new data (or observations) belongs to. The identification is done by a model that is either programmed be the user (e.g. an algorithm) or trained (e.g. by using training data that is already classified) to categorize new observations. It is an example of object or pattern recognition, and regarding machine learning, it falls under the concept of supervised learning. Classification can be either binary, where only two classes are distinguished, or multi-class where there are more than two classes are distinguished.

To be able to classify, data is required to contain adequate explanatory variables which are otherwise known as *features*. The features need to be sufficient in both quantity and quality, depending on the complexity of difference between classes and the amount of classes that one would like to distinguish for data categorization. Features can be any relevant characteristic of a data point, e.g. color, distance, position, shape, which can be efficiently used to distinguish and categorize. These can be the same features used for segmentation techniques as mentioned previously.

2.2.3. Features and feature extraction regarding point clouds

To give a better understanding of features, the following section will elaborate on what features can entail, specifically regarding the topic of point cloud data, and how new features can be discovered or extracted from data that is available at hand. As mentioned in section 2.2.2, features play an important role for segmentation and classification techniques. The features allow segmentation techniques to set thresholds and boundaries so that data can be distinguished and divided into clusters. Simultaneously, features are able to characterize certain classes, and are able to characterize and classify new data within these respective classes. These features are what draws the segmentation methods and the classification methods very close, as data that is directly classified by a model, can be seen as segmented (by category) and data that is directly segmented, could be seen as classified (although without any specific category).

The simplest features for point clouds are the coordinates $\langle x, y, z \rangle$, representing the location of data points in the Euclidian space. While at first glance they seem very good features for both segmentation and classification as location does characterize points very well, they are interpreted as ambiguous as their true values are not invariant. This would result that even the slightest rotation or scaling of a point cloud would give different segmentation or classification results. Therefore, it is important to look for features that are robust to transformations, in other words, features that are *pose invariant*.

2.2.3.1. Normal vectors

The most basic features for point cloud datasets are the normal vectors and curvature. Normal vectors indicate the direction of a point based on their hypothetical underlying surface, and thus based on the location of the neighboring points. This is done through Principal Component Analysis (PCA) and plane fitting, which essentially are least squares plane fitting problems. PCA is a statistical method for finding the underlying structure (the principal components), such as direction, variance and covariance in a dataset. The same can be applied to points within a point cloud data set.

To carry out PCA, a covariance matrix for every point within the defined point cloud neighborhood is assembled as:

$$C = \frac{1}{k}\sum_{i=1}^{k} \cdot (p_i - \bar{p}) \cdot (p_i - \bar{p})^T \; and \; \bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i$$

Where k is the number of neighbors, p_i is the point in question and $\bar{p}$ is the centroid of the point neighborhood, as depicted in Figure 8(a). Now, these matrices show the variance of a points local neighborhood, where the eigenvectors with smallest corresponding eigenvalues of these matrices indicate the normal vectors of planes best fitting to these neighborhoods. As depicted in Figure 8(b), all thee eigenvectors (v_0, v_1 and v_2, although v_2 is not visible) are perpendicular to each other, and the smallest eigenvector v_0 is perpendicular to the directions of the variation from the point neighborhood. This eigenvector is in fact the point normal vector.

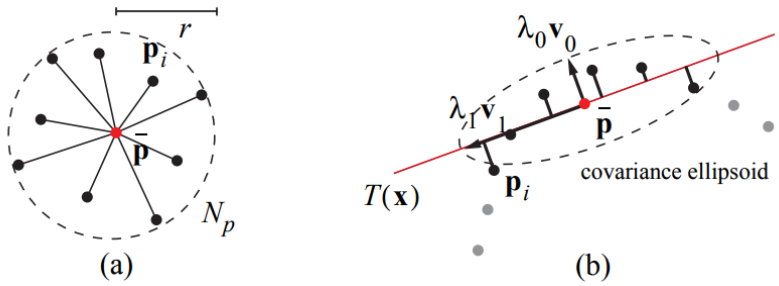


Figure 8: Schematic representation of (a) the centroid of a point neighborhood, and (b) the covariance ellipsoid by Pauly et al. (2002)

2.2.3.2. Curvature

The second basic feature is curvature, which is defined as the amount by which the underlying surface of a point with its neighboring points deviates from being a flat plane. In other words, it can similarly be defined as the variation of the normal vectors of neighboring points around a point in the point cloud data. Using the same covariance matrices from the normal estimation problem, three eigenvectors are extracted. The eigenvalues of these eigenvectors $(\lambda_0, \lambda_1, \lambda_2)$, indicate the variation of the normal vector v_0 of point p_i, from which the curvature can be calculated as:

$$\sigma(p_i) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$$

2.2.3.3. Histograms

While point normal vectors and curvature provide a solid base for segmentation or clustering techniques, classification methods need more powerful information to be able to distinguish data. The issue with normal vectors and curvature is that they are equivocal, i.e. orientation of normal vectors and values for curvature are prevalent on a wide range of classes, and therefore not explicitly tied to a certain class. They do not inhabit the discriminative power to identify characteristic points that are prevalent in indoor building scenes, e.g. edge points and corner points, which would otherwise be beneficial to distinguish construction elements, e.g. the distinction between walls and columns.

Rusu et al. (2009) developed a method to increase the information that point cloud data can contain by including a set of histograms to every point cloud data point, which store angular variations of neighboring points from a user specified radius. Rusu et al. (2009) developed Fast Point Feature Histograms (FPFH) initially for the 3D registration of overlapping point cloud views, with the purpose of using these views for object recognition for indoor robots. But they also show promising results to characterize points by creating histograms with information of their respective neighborhood. This way, points can be enriched with more informative characteristics, and it becomes clear that higher level classification characteristics can be discerned such as corners, edges, planes and spheres by detecting variations in their histogram distributions, such as exemplified in Figure 9.

Take note however that the features for the histogram are calculated based on the normal vectors and curvature. Therefore, the quality of the point feature histogram is heavily dependent on the parameters chosen for computing the normal vectors.

The method for calculating FPFH is as follows:

For every point p_i in the point cloud data set, a neighborhood is defined by a specified radius. All the points including their surface normal within this radius are paired, where p_i is the source point and p_j is the target point, and n_i is the normal of p_i and n_j is normal of p_j. Then for every point pair $p_i \; and \; p_j, where \; i \neq j$, a Darboux frame is constructed which calculates a set of vectors of the imaginary curve or surface between points $\langle p_i, p_j \rangle$ and their respective normal vectors $\langle n_i, n_j \rangle$, defined as:

$$u = n_i \; as \; the \; \boldsymbol{unit \; normal \; vector}$$

$$v = (p_j - p_i) \times u \; as \; the \; \boldsymbol{unit \; tangent \; vector}$$

$$w = u \times v \; as \; the \; \boldsymbol{tangent \; normal \; vector}$$

From the Darboux frame, three different angular variations are calculated as features as:

$$f_1 = v \cdot n_j$$

$$f_2 = \frac{u \cdot (p_j - p_i)}{\|p_j - p_i\|}$$

$$f_3 = \arctan(w \cdot n_j, u \cdot n_j)$$

The three features are binned into their respective histogram, resulting in three separate histograms, as exemplified in Figure 9. Figure 9 shows the difference in histogram distributions of synthetic geometric primitive surfaces, such as a plane, sphere, cylinder, edge and corner. The distinguishability between each primitive surface and their corresponding histograms are shown in an intersection kernel, ranging from white (100% similar) to black (0% similar).

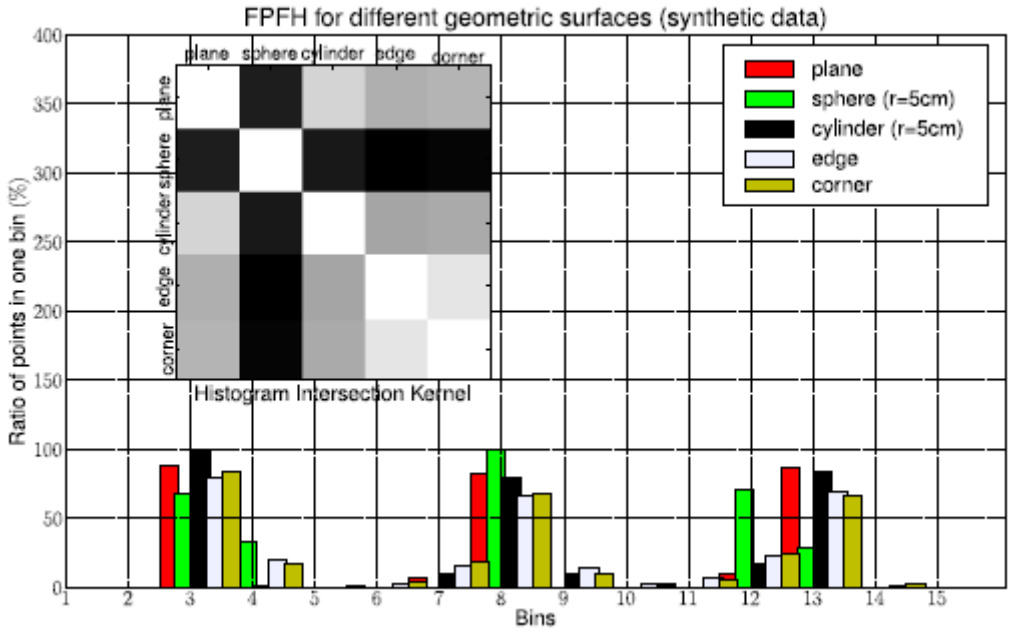


Figure 9: FPFH for different (synthetic) geometric surfaces by Rusu et al. (2009)

2.2.3.4. Other features

Color proves to be a valuable feature within certain classification methods, such as in the work of Qi et al. (2016), but is not always captured when acquisition techniques are used other than RGB-D sensors. Additionally, due to lighting issues and shadows, colors are not always accurately captured, as pointed out by Cho et al. (2014). Similar issues are found within other fields of object recognition, with a good example of sensors used in self-driving cars as in the work of Rasshofer et al. (2011), where snow, rain and fog can cause problems for object detection as colors can be distorted in depth sensors (and even can distort the geometric accuracy of LIDAR sensors). Moreover with respect to the AEC industry, the interpretation of color is ambiguous, as construction elements could contain any color when painted.

Another feature that is often captured by LIDAR sensors is intensity, and is often used for point cloud processing of urban scenes, as becomes evident from the work of Kashani et al. (2015) and Tatoglu & Pochiraju (2012). Intensity is acquired for every point, and indicates the strength of the laser pulse that generated the data point, partially dependent of the reflectivity of the material from which the point is acquired. However, due to intensity being relative to the position of the sensor, different values will be acquired from different scan positions, diminishing its descriptive capability.

A very powerful feature for object detection and object classification could be material, however, the research for this field seems to be a difficult classification problem on its own, as seen in the work of Kirchner et al. (2008).

2.3. Incorporating semantics for segmentation and classification of point cloud data

The following section will describe more extensively the current state of the art in literature that incorporates semantics into point cloud datasets for segmentation and classification purposes.

2.3.1. State-of-the-art techniques for point cloud data

Current segmentation and classification methods in literature offer methods that often separate the segmentation and classification process, where point cloud data is first segmented, and classified afterwards. One of such recent methods is proposed by Macher et al. (2017) which starts the segmentation process by analyzing the presence of vertical structural elements to divide the point cloud into several subspaces in an early stage. The corresponding floor map is superimposed over the point cloud, allowing for region growing techniques to divide rooms which as a result, creating a single point cloud per room. Subsequently, planes are extracted from these subspaces, which are then classified into three separate categories: walls, ceilings and floors. Since there are only three categories considered, the residual is considered as occlusion, which might contain information about other important elements, such as windows, doors, columns and beams. Additionally, there is still a certain necessary amount of manual work in the processing chain, such as superimposing the floorplan correctly, wall reconstruction when there is occlusion blocking wall parts, and correct plane extraction by using Maximum Likelihood Estimation Sample Consensus (MLESAC), an adapted variant of RANSAC.

Similar research for segmenting and classifying indoor spaces is done by Armeni et al. (2016). They introduce a hierarchical parsing method, as shown in Figure 10, that is able to have large point cloud scenes as input (i.e. multiple spaces within one building floor), where the rooms are segmented by detecting void space between walls. These void spaces are apparent in combined point cloud data, as scanners will only detect the surface of a space in which the scanner is positioned in. These void spaces are detected by iteratively creating density histograms, from which peak-gap-peak signatures are detected and extracted. The point cloud is then divided into spaces that are semantically meaningful. The method is very inventive and the comparison within the study shows improved results over RANSAC or HT approaches.

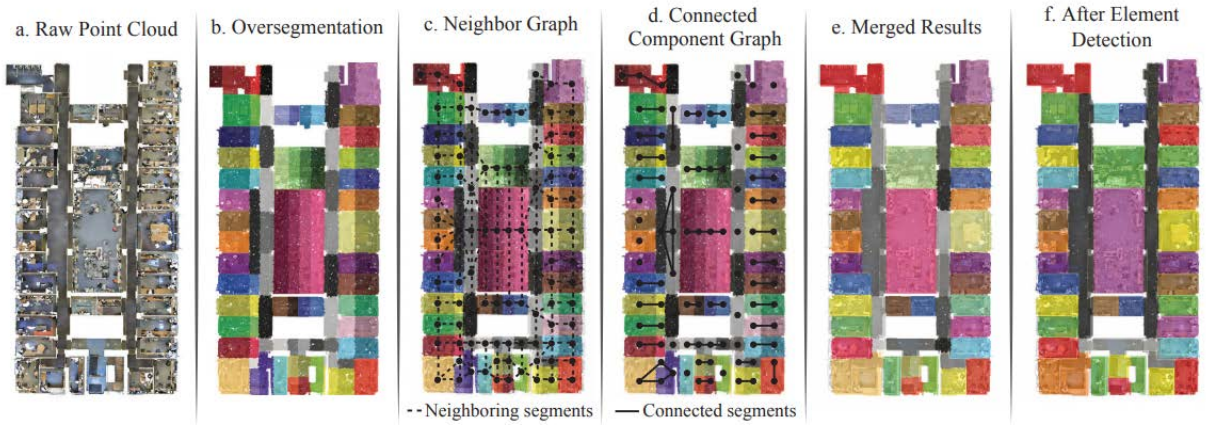


Figure 10: Workflow of semantic parsing of large scale indoor point cloud representations by Armeni et al. (2016)

Another method proposed and tested by Hackel et al. (2016) discerns that the large number of 3D nearest-neighbor queries for feature extraction seems to be the main bottleneck in processing speed within segmentation and classification techniques. They use an algorithm that incorporates a voxel grid to accelerate the nearest neighbor search by down sampling the point cloud by decreasing point density and generating a pyramid-like search structure, that can scale by level and computing separate search structures within these levels. This approximation method seems to be suitable enough for voxel based feature extraction.

The feature extraction is performed by PCA and constructing the covariance matrix as shown in section 2.2.3, however additional features are extracted, such as sphericity and planarity as seen in Figure 11 which gives an overview of the types of features Hackel et al. (2016) use for classification.

covariance	Sum	$\lambda_1 + \lambda_2 + \lambda_3$
	Omnivariance	$(\lambda_1 \cdot \lambda_2 \cdot \lambda_3)^{\frac{1}{3}}$
	Eigenentropy	$-\sum_{i=1}^{3} \lambda_i \cdot \ln(\lambda_i)$
	Anisotropy	$(\lambda_1 - \lambda_3)/\lambda_1$
	Planarity	$(\lambda_2 - \lambda_3)/\lambda_1$
	Linearity	$(\lambda_1 - \lambda_2)/\lambda_1$
	Surface Variation	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$
	Sphericity	λ_3/λ_1
	Verticality	$1 - \lvert\langle[0\,0\,1], \mathbf{e}_3\rangle\rvert$
moment	1st order, 1st axis	$\sum_{i\in\mathcal{P}}\langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_1\rangle$
	1st order, 2nd axis	$\sum_{i\in\mathcal{P}}\langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_2\rangle$
	2nd order, 1st axis	$\sum_{i\in\mathcal{P}}\langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_1\rangle^2$
	2nd order, 2nd axis	$\sum_{i\in\mathcal{P}}\langle \mathbf{p}_i - \mathbf{p}, \mathbf{e}_2\rangle^2$
height	Vertical range	$z_{\max} - z_{\min}$
	Height below	$z - z_{\min}$
	Height above	$z_{\max} - z$

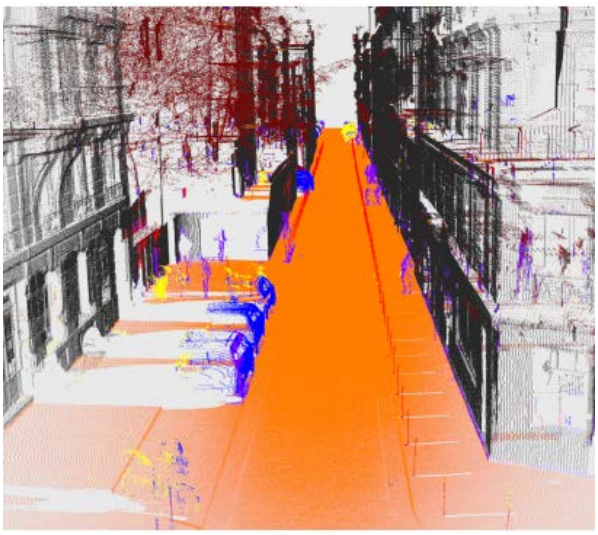


Figure 11: (a) Table of features used in urban point cloud classification in the research by Hackel et al. (2016) and (b) results of the classified urban point cloud scene by Hackel et al. (2016)

Their experiments classify point clouds of urban street environments with an overall classification accuracy of >90%. However, using normalized height is a very powerful feature for distinguishing point cloud data, and while mostly applicable in urban environments (e.g. buildings, are always above street level, cars and humans have a fixed range of height, etc.) cannot be used directly in indoor environments (e.g. columns, walls and floors do not have fixed heights). Moreover, their method is unsuitable for indoor environments as indoor environments require a different level of detail.

Although with better results, the research conducted by Hackel et al. (2016) is largely based on the work of Weinmann et al. (2015) and Weinmann et al. (2013), and therefore very similar and sharing the same issues for when the same method would be applied for indoor scenes.

There are also several workflows proposed in research that follow a strict manual encoded ruleset for the segmentation and classification process. Examples of such rulesets can be: floors are on certain heights, doors and windows have certain dimension constraints and are always an element of a wall, walls are parallel or orthogonal when referenced to each other, etc. Most notable is the research from Nüchter & Hertzberg (2008) which incorporate semantic maps for indoor scene recognition and point cloud classification, as seen in Figure 12 and the research from Pu & Vosselman (2012) which segment and classify building façades according to pre-defined rulesets (e.g. ground is on lowest position, windows are in same plane as walls, roofs are above walls). While these methods work well for the purpose they were created for, their usability in limited for point clouds that deviate from these rule sets, and as such, these methods are not generalizable for classification through scene understanding.

Similar but more recent research proposes a strict workflow of parsing point cloud data consisting multiple floors, by following strict workflows based on a set of filters, such as in the work of Macher et al. (2017) and Bassier et al. (2016). Both methods leave out non-structural elements such as furniture, as they are specifically designed for the scan-to-BIM process. Due to the incorporation of this semantic hierarchy proposed in the workflow (top-down), object recognition is not done based on contextual features that point cloud data can inhabit. This becomes more prevalent in section 2.4, as several researchers have challenged the use of rulesets, and have turned away from them by using approaches that incorporate machine learning for independent, context-based object recognition. Additionally, there is a wider variety of features that have been adopted in research to circumvent the use of strict rulesets, and increase generalizability of classification models. These will be discussed next in section 2.3.2.

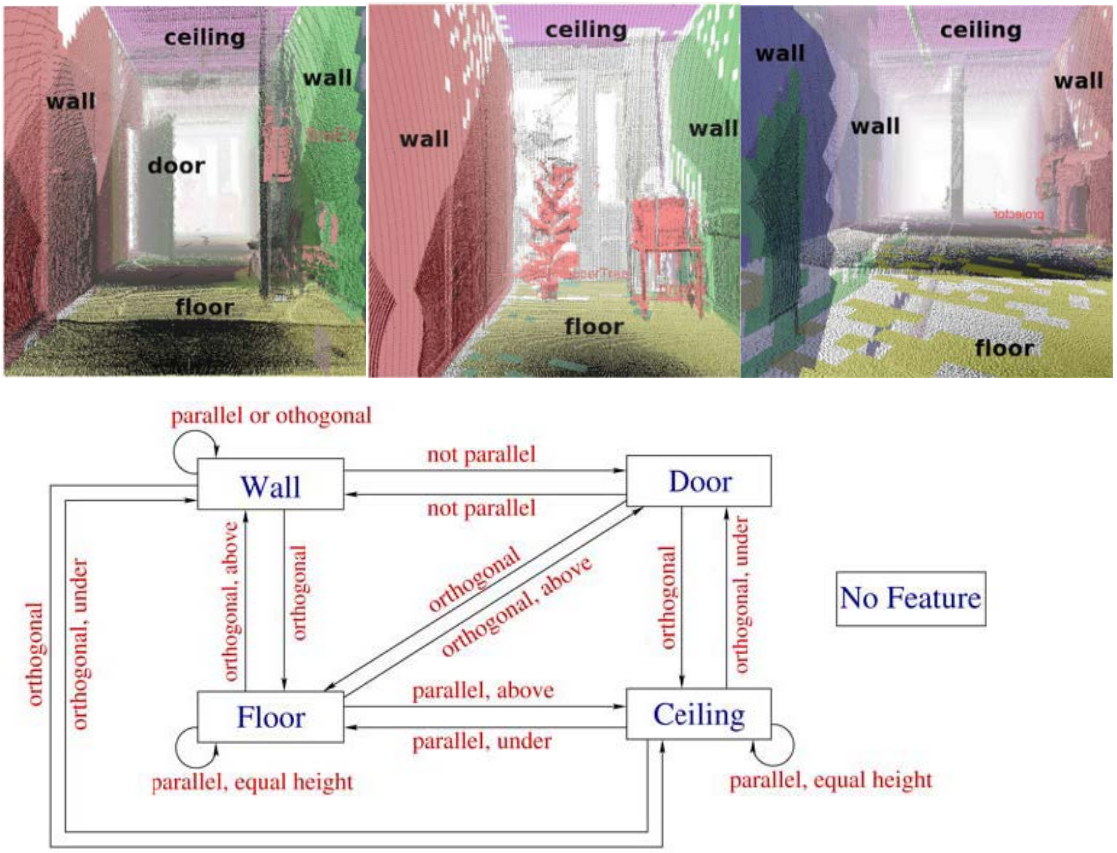


Figure 12: Example of a semantic map with point cloud scene interpretation by Nüchter & Hertzberg (2008)

2.3.2. Features for semantics – utility in object recognition and computer vision

As mentioned in section 2.2.1, point cloud classification is closely related to object recognition of 2D images, for which descriptive features (often dubbed as descriptors) are an important concept. Originally used to detect human faces, stitching panorama images and used in image search algorithms incorporated by internet search engines, it has been adopted for real-time applications such as autonomous vehicles and for robotics as well. The method of mapping images requires key points, and as such, most descriptors originated as algorithms for detecting key points for these fields.

As for point clouds, descriptors are able to describe the surrounding environment by using the relational properties of their surrounding neighbors as an indication of the position and structure of a certain point, i.e. they provide semantic information. Besides, it allows algorithms to map point clouds from different viewpoints to be mapped onto each other, which is the initial reason for the introduction of FPFHs, as mentioned in section 2.2.3, that are introduced by Rusu et al. (2009). Their research accentuates that from point cloud data, the underlying surface geometry of points and their respective point neighborhood can be captured in the form of histograms, and that point cloud scenes from different viewpoints can be mapped by tracing key points with identical histograms. This is in line with the … Which …

that point cloud data from different viewpoints could be seen as transformations of the same point cloud data set, and that therefore the descriptive features for matching are required to be pose invariant, which is highly valued in the field of object recognition.

Furthermore, they allow for extensions such as semantic mapping of 2D images onto 3D point clouds (Nüchter & Hertzberg, 2008) or mapping 3D point clouds onto 2D images as in the research of Golparvar-Fard et al. (2009) and Han et al. (2015) for progress monitoring in the AEC industry. In recent research however, such as Hackel et al. (2016), Wolf et al. (2015and , they are applied to every point in a point cloud data set to add semantics to the basic XYZ and RGB values, so that point cloud datasets can be classified more accurately.

Two groups of descriptors can be distinguished namely local and global descriptors. While local descriptors describe the local geometry around a certain point based on its neighbors, and do not have a direct notion of object definition, global descriptors encode object geometry and are used for classification and geometric analysis. Global descriptors are explicit representations of the shape of a mesh or cluster of points, and are not computed for every point within the point cloud data. The global descriptors are mostly used for 3D modelling or reconstruction, while local descriptors are used for object recognition and classification (Tang et al., 2010).

Table 3 gives an overview of the local descriptors that are used in relevant literature, Give overview of local descriptors used in literature.

Table 3: Overview of local descriptors used in relevant literature for point cloud segmentation and classification

Name	Type	Description
FPFH (Fast Point Feature Histograms)	Local	See section 2.2.3.3.
3DSC (3D Shape Context)	Local	3D spherical histogram of the topology/geometry of neighboring points within a defined radius, with the normal vector as base orientation, rotated *N* times because it is designed for point cloud mapping, and therefore the same point from 2 viewpoints could contain different normal vector orientations. (Frome et al., 2004)
SHOT (Signatures of Histograms of Orientations)	Local	Mostly used on volumetric 3D data for surface matching. Very similar to 3DSC, but contains improvements of 3DSC to increase uniqueness of the feature (Salti et al., 2010)
3D SIFT (3D Scale-Invariant Feature Transform)	Local	Originated for computer vision for 2D images by Lowe (2004). Used for 2.5D range images for key point detection and extraction, such as edges or border of objects, as seen in Figure 13
NARF (Normal Aligned Radial Feature)	Local	Used for 2.5D range images and underlying idea based on SIFT. Detects more key points as seen in Figure 13 (Steder et al. (2011)

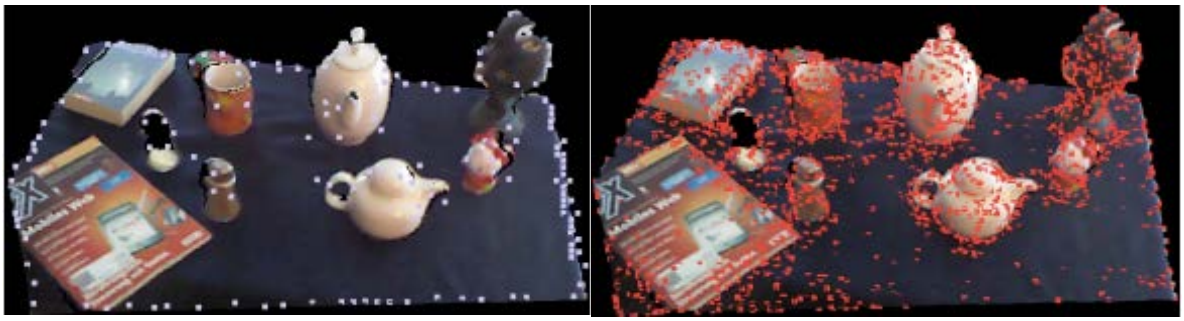


Figure 13: Example of 3D-SIFT feature key point detection (left) and NARF feature key point detection (right) in 2.5D point cloud data

The 3D-SIFT and NARF algorithms are specifically tailored for 2.5D range images and therefore cannot be directly applied to full 3D point clouds. Furthermore, even when sufficient changes would be made, only a limited set of points would be detected as key points, while for point cloud classification and the use of context, every point within the point cloud dataset should contain sufficient information about their local geometry.

This mapping of the local geometry from point neighborhood is done by both the 3DSC and SHOT descriptor. However, these descriptors are initially designed for the use of mapping and object detection of 3D volumetric data. Hackel et al. (2016) have been able to use them in their work but it becomes evident that on large point cloud datasets, the computation times for both of these descriptors becomes exceptionally large, and that their added value to classification accuracy is limited.

Global features are generally not used as they detect complete object instances (i.e. clusters of points), and only Armeni et al. (2016) and Soares et al. (2015) seem to be able to use global features for object detection and recognition. However, their proposed methods are too complex to implement directly for purposes within the AEC industry.

To sum up, FPFH is the most used in research, such as in Pang & Neumann (2016), Wang et al. (2014), Najafi et al. (2014) and Rusu et al. (2009), with the only drawback that it does not contain any information about the local geometry of points. SHOT and S3CD do incorporate local geometry, but are not widely used yet in point cloud data, and NARF and 3D-SIFT algorithms are designed to detect key points for object recognition, and are not used as features for every point in a point cloud dataset. Regardless, extracting 3D features for object recognition is still ongoing research.

2.3.3. Challenges and opportunities in semantic segmentation and classification

As pointed out by Tang et al. (2010), quite a few methods for modeling and recognition rely on hand-coded knowledge and are encoded in a set of rules such as "walls are vertical" and "walls intersect with floors and ceilings". These approaches break down when used for slightly different environments, which is very common in the AEC industry where rules are not

universal. Weinmann et al. (2015) further supports this notion, stating that classifiers based on rule-based learning cannot compete with other classifiers, and that methods that learn from experience can address these limitations. Learning-based methods learn the rules from examples, but are still open for flexibility in these rules. For example, instead of encoding "walls are vertical", one could encode a probability distribution over the space of surface orientations. For a wall, this might be centered on vertical, but also allows for flexibility towards a surface being near-vertical. However, this brings additional computation costs, considerably in the testing phase (Tang et al., 2010).

Another possibility could be incorporating contextual information into such methods, to enhance the semantic segmentation process. As the findings of Xiong et al. (2013) indicate, their algorithm works well for their case study, but is not tested for a wide variety of environments. Also, their approach first detects walls before openings, but this process could be reversed by leveraging contextual relations between these objects. For example, the fact that a surface contains a door-like opening, would support the interpretation of the surface as a wall. Additionally, if a surface in a room is a wall, then the parallel surface in the adjacent room is also very likely to be a wall.

It also becomes evident that complex areas such as staircases, are harder to classify correctly, and elements that consist of small parts are harder to detect, because their surface features resemble features of noise surfaces as in the research by Bassier et al. (2016). Also, parameters as user input are focused on maximization of precision instead of recall. The algorithm would work better with recall maximization, because it is easier to remove false positive then to add false negatives. Additionally, Kim et al. (2017) support this statement, expressing that accuracy on small objects is lower than on large objects, due to large objects being easy to segment and containing more geometric information. Small objects are harder to detect when point clouds are incomplete (which is often the case) and are more susceptible to noise.

Weinmann et al. (2015) tested the different components of a method for segmentation and classification methods. In these methods, choosing the right number of neighbors is similar for the complete dataset. However, they state that the choice of a neighborhood definition and size, which is used for feature extraction, should be different based on point cloud data, point density and object class. Additionally, their experiment shows that classifiers are less prone to overfitting this way, resulting in a better classification input.

As point features for segmentation and classification features go, surface normal vectors and curvature estimations are somewhat basic in their representations of the geometry around a specific point. They are easy and fast to compute, but cannot capture much detail. Consequently, most scenes contain points with many similar feature values, reducing their informative characteristics. For this particular reason, Rusu (2009) and his related work Rusu et al. (2008) and Rusu et al. (2009), introduces point feature histograms as a substitute for conventional descriptors such as curvature and point normal vectors.

However, Hackel et al. (2017) contradicts this notion and states that in their experiment, using histogram-based descriptors only gives a marginally improved performance. They coin the idea to sidestep the feature extraction process as a whole, and focus on deep learning

implementations such as artificial neural networks (ANN). Nevertheless, even machine learning algorithms need features as becomes evident in section 2.4, and the fact that features were not able improve their results, could be explained by their use of a normalized height feature.

For speech and image understanding, convolutional neural networks (CNNs) have become the standard, but has only seen limited application in building semantics in other research. There is however quite some research on scene understanding on 2.5D range images and object detection in 3D point clouds (although not specific for the AEC industry), which integrates machine learning methods for incorporating context and searching for *semantics* within data. These will be explained and discussed in section 2.4.

2.4. Machine learning for object recognition

As becomes clear from section 2.3, workflows that contain hardcoded rulesets and the limited amount of research in point cloud data of indoor scenes, it seems that researchers are turning away from these methods. Now that methods involving machine learning, as explained in section 2.2.1, become more widely spread as these methods tend to perform quite well for object recognition tasks.

2.4.1. Deep learning – Convolutional neural networks

One of the most noticeable machine learning methods for object recognition is the use of convolution, mostly prominent as convolutional neural networks (CNNs). CNNs are mostly used for object detection and classification on 2D images, and they have become a very powerful tool. To give a simplification and an intuition on how CNNs work, the process of image classification and feature detection is given in Figure 14. A CNN incorporates a sliding window algorithm that detects patches which characterize parts (feature patches) of the image that belong to a certain class. All feature patches that are detected are inserted into the classifier (a neural network, elaborated further in section 3.2). Note that the neural network requires to have matrices as input, although conveniently, the raster of pixels that represent a 2D image are in fact a matrix of numbers (in this case intensity values).

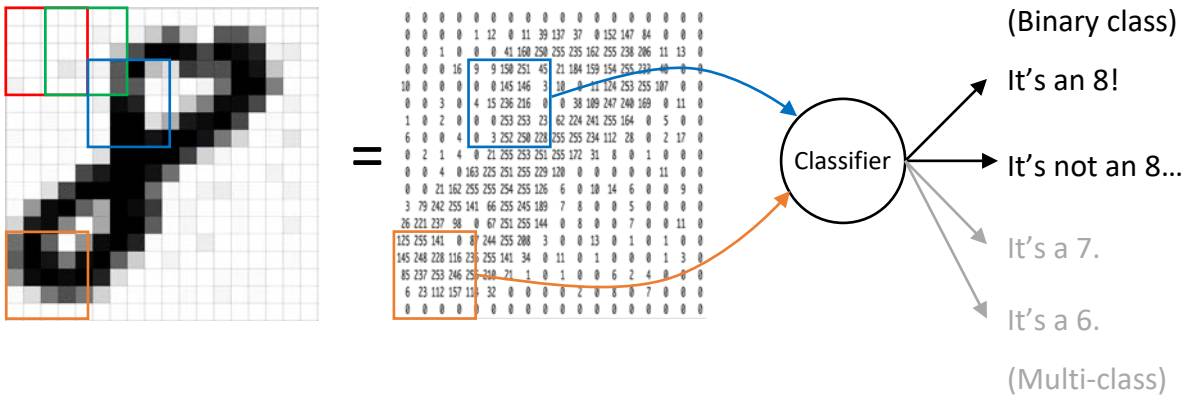


Figure 14: A simple overview of the process of Convolutional Neural Network classification (note that the process visualization is to give the reader an intuition, and therefore is oversimplified)

The advantage of using CNNs on 2D data, is the disadvantage of using CNNs on n-D data (where n ≥ 3). Reasons such as 3D point cloud data being an unstructured data type (as explained in section 2.1), and the computational complexity of CNNs by adding dimensions, causes the training of such models to be computationally infeasible. However, researchers have been creative bypassing these limitations.

There are several methods available on applying CNNs on 3D point cloud data such as proposed in the work of Maturana & Scherer (2015), Soares et al. (2015), Huang & You (2016), Pang & Neumann (2016), Engelcke et al. (2016) and Li et al. (2016).

Maturana & Scherer (2015) classify objects from a large scale, such as persons and cars, and use occupancy grids with a 3D-CNN for predicting these class labels. Point cloud data from 3D objects are transformed to 32 x 32 x 32 3D voxel grids, where each voxel is assigned a value between 0 (low occupancy) and 1 (high occupancy), decreasing computation time by decreasing the size of the data significantly, and circumventing the problem of empty space in 3D point cloud data.

More research has been done of using voxel data as input for Neural Networks, and can be found on the ModelNet leaderboard (http://modelnet.cs.princeton.edu/), although the leaderboard is for prominently for volumetric 3D (mesh) data and point cloud data is not discussed. To elaborate on the issue of using point cloud data, Huang & You (2016) discerns that all data needs to be pre-segmented before training and prediction on CNNs becomes possible. Similar to Maturana & Scherer (2015), they parse correctly classified point cloud data from outdoor scenes through an occupancy generator, and use it for training the 3D-CNN. For the testing phase, point cloud data is transformed to a voxelized grid, fed into the neural network which defines the voxel labels, and de-voxelizes the grid into a labeled point cloud, as represented in Figure 15.

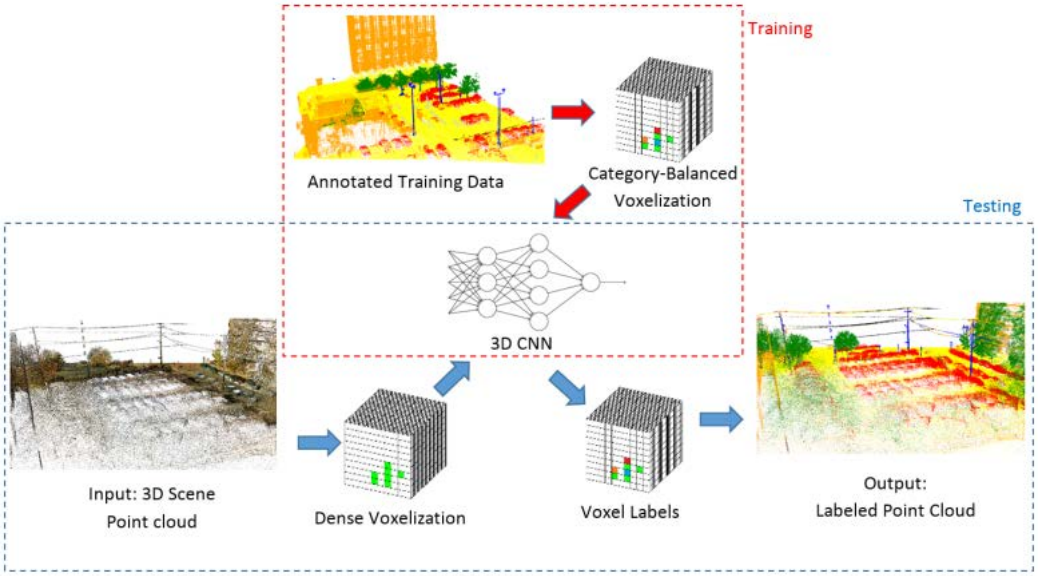


Figure 15: Framework for point cloud labeling by Huang & You (2016)

In more recent work, Zhao & Ilies (2017) classify 3D point cloud data of separate objects. They create their own database for deep learning by transforming 3D meshed object models to point cloud representations, including several feature descriptors such as Zernike descriptors to define shape and light field descriptors to look for similarities between objects through different viewing angles. Consequently, they transform the point clouds into a voxelized data representation and add noise for realism. Although previously mentioned methods using CNNs for classification achieve high results, none of them truly incorporates scene understanding and it is not obvious on how to extend these methods to do so. The research that relates the closest to using deep learning for point cloud indoor scene understanding, is done by using 2.5D range images such as proposed in the method by Soares et al. (2015) and Qi et al. (2016).

The method proposed by Soares et al. (2015) detect objects on tables from RGB-D data, and using different viewpoints combined with Viewpoint Feature Histograms (VFH) to train a 2D CNN. However, the research is very limited on small objects, such as cups and books, and is not appropriate for indoor scene understanding specific for the construction elements.

The models proposed by Qi et al. (2016) and Qi et al. (2017) uses the Stanford 3D semantic dataset, for training a CNN to classify points within a scene. These scenes comprise of a 1 x 1m 2.5D point cloud (2D + depth information) block, each containing 4096 points. They extend their CNN to classify objects and part segmentation from previous work, to be able to use the neural network for scene understanding. They address the issue of unordered point clouds by applying a symmetric function over the inputs and indicate that regular sorting point cloud data is not very robust, especially when point cloud datasets are large. Furthermore, they conducted a test by incorporating a Recursive Neural Network (RNN) to incorporate a certain form of sorting unsorted point cloud datasets, i.e. semantic relationships between points. Without going too much in-depth, RNNs are a type of neural networks with a certain 'memory' attribute, i.e. they are able to learn sequences in data, such as in speech or writing (hence

often applied in speech recognition and automatic text generation). The first problem that becomes evident when using RNNs is that points need to be arranged as a sequence of input vectors, without truly being a sequence as their order is randomized. RNNs do require order to learn well, and the second problem is that RNNs are built for sequences with small lengths, a small patch of point cloud data often exceeds a thousand points. Regardless, their model PointNet and PointNet++ seem to perform well on the ModelNet40 dataset, due to the incorporation of symmetric functions to retain local and global point information.

Most recent work is done by Li et al. (2018) that is very similar to the work of Qi et al. (2016) and Qi et al. (2017), however, their CNN is specifically designed for local point neighborhoods instead of a symmetric function for incorporating structure. For learning they construct local coordinate systems for every point and their neighbors, and therefore retaining the ability of using the xyz-coordinates that are discarded in most methods in literature. The idea is similar to that of the SHOT descriptor (although without the histograms) which represents a local neighborhood that incorporates local geometry, but is extracted within different stages of the point cloud classification process.

Their most value addition to training seems to be the introduction of their X-Conv Operator, which provides operations to the features used for classification of the classes in ModelNet40 in such a way that the features become more clearly distinguishable, as depicted in x. (t-SNE is a dimension-reduction algorithm that visualizes high dimensional data in 2D or 3D scatterplots)

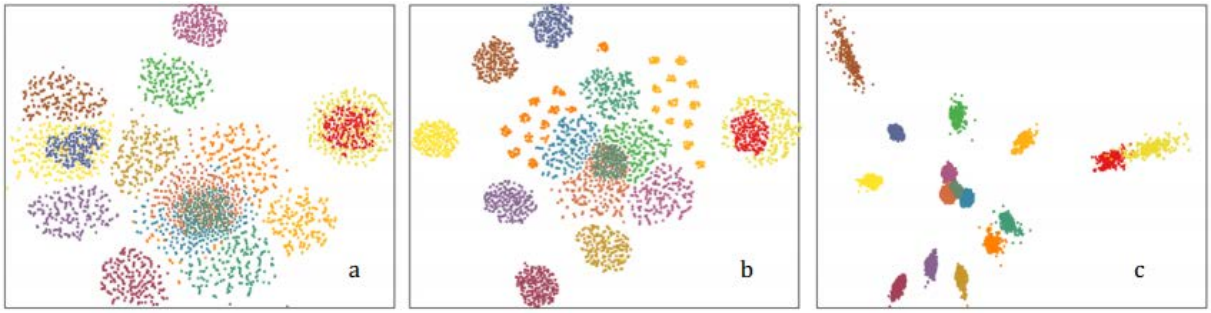


Figure 16: t-SNE plots of the increased distinguishability by the X-conv operator by Li et al. (2018), the process is shown from left to right

2.4.2. Other ways of integrating context by machine learning

Beside the incorporation of CNNs for scene understanding, it is worthy to mention that researchers have used other kinds of machine learning models, especially in times before the recent success of CNNs, to incorporate some way of context for point cloud data. Most of those methods have incorporated probabilistic graphical models, which have their roots in statistics.

A probabilistic graphical model is a network of nodes with similarities to neural networks. The nodes (that represent the variables) and their directional vectors (representing relationships) represent a probabilistic reasoning framework, with examples such as the Bayesian network, Conditional Random Fields and Markov Random Fields. For a set of input variables given (such as features in the case of point clouds) and their underlying (in)dependencies allow these models to perform classification tasks based on conditional distributions, and therefore to incorporate context.

While the methods that integrate context through probabilistic frameworks proposed by Shapolavov et al. (2010) and Najafi et al. (2014) work well for simple outdoor scenes, Rusu et al. (2009), Koppula (2011) and Anand et al. (2013) are able to use them to successfully capture indoor scene understanding from 2.5D depth images. The reason that probabilistic models work well as stated by Rusu et al. (2009) is that it outputs a confidence measure, allowing the user to set thresholds to suit the data that is to be classified. However, the drawback with probabilistic models becomes clear from the test cases by Koppula et al. (2011), where contextual relationships on a small scale between different objects (such as furniture), cannot always be determined. This is called context range, i.e. in some environments, tables and chairs may have high contextual relationships, but in others, they may not. Additionally in these methods, large point cloud datasets cannot be parsed efficiently, for which only small scenes are suitable.

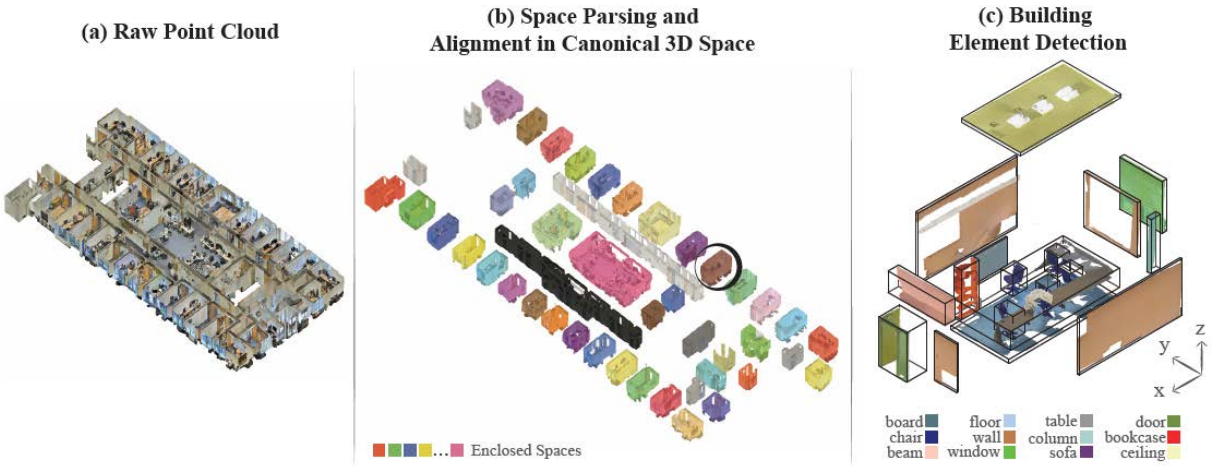


Figure 17: workflow of Stanford building parser for detecting building elements by Armenti et al. (2016)

Armeni et al. (2016) solve this problem in their large point cloud dataset parser, where large point clouds are segmented into small spaces, as explained in section 2.3.1. Due to the similarity of dimensions of these spaces, the method incorporates a normalized coordinate system that is similar for every individual segment, a *canonical* coordinate system, for which a CRF is learned to detect the contextual relationships between objects in these spaces, as shown in Figure 17.

Xiong et al. (2013) proposed a method in their study to recognize planar walls, ceilings and floors, and distinguishes these components from the rest of the data, which is classified as clutter. They use contextual information to distinguish relevant objects from clutter, through using the relationships between the planar surfaces of ceilings, walls, and floors. Therefore, the algorithm interprets multiple surfaces at once. The classification phase consists of four steps: (a) voxelization, (b) patch detection, (c) patch classification and (d) patch intersection and clutter removal, as visualized in Figure 18.

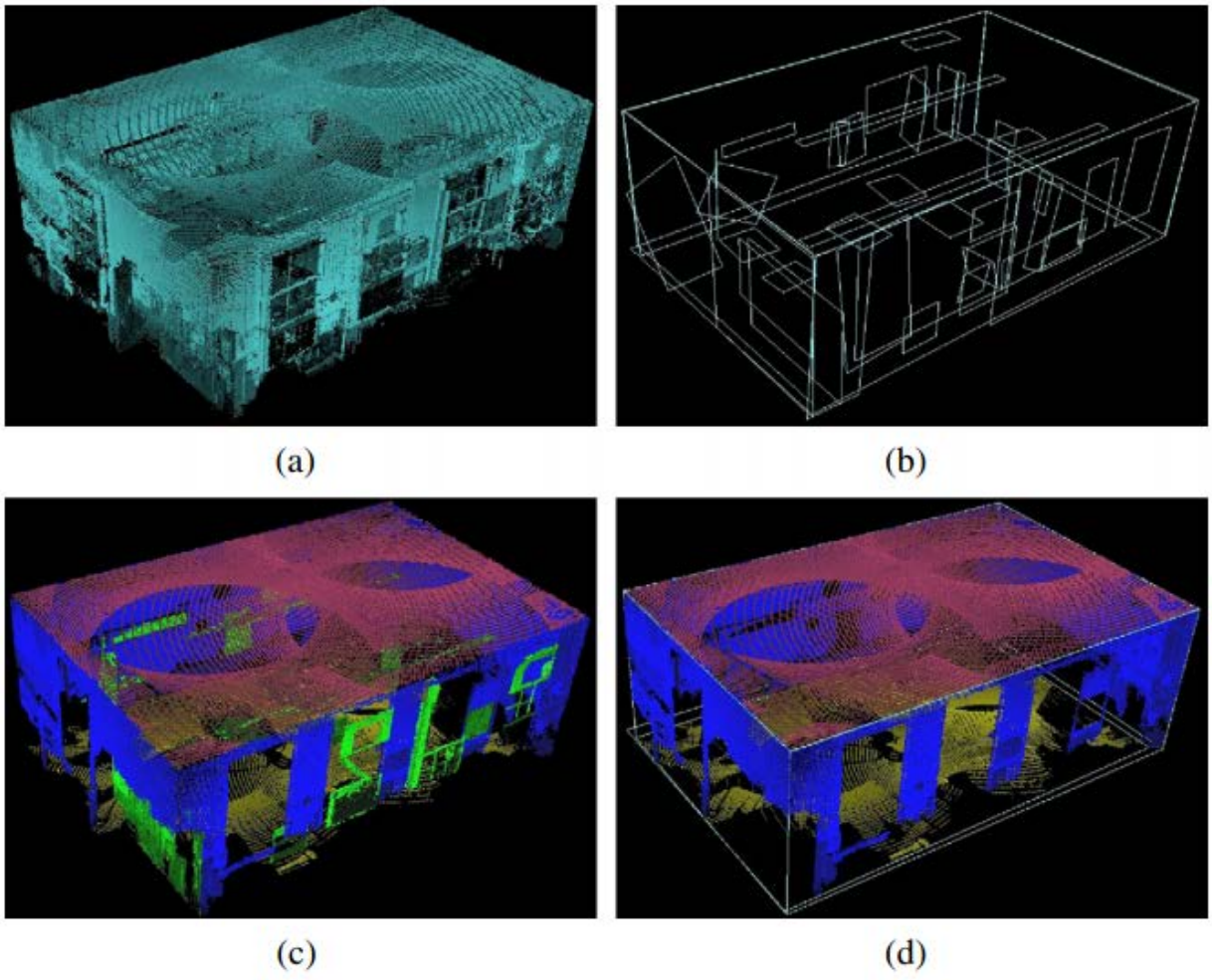


Figure 18: Workflow of context-based modeling algorithm by Xiong et al. (2013): (a) voxelization, (b) patch detection, (c) patch classification and (d) clutter removal

From the voxelized data, planar patches are found by a region growing algorithm that groups point neighbors with similar surface normal vectors. The planar patches are then classified through the use of context by using extracted features of neighboring patches through a

probabilistic classifier. These features entail context as describing the orthogonality or parallelism between different patches. Although the study seems to be robust against highly occluded environments, the amount of classes that are distinguished is very basic.

A very different and inventive approach in recent work is by Kim et al. (2017), who extend a simple point cloud scan with information from a thermal heat map for classifying objects through a probabilistic decision tree, that are otherwise difficult to distinguish such as persons, displays and lighting. As shown in Figure 19, objects that are difficult to distinguish with normal vectors, curvature or histograms, are classified correctly with a highly increased accuracy.

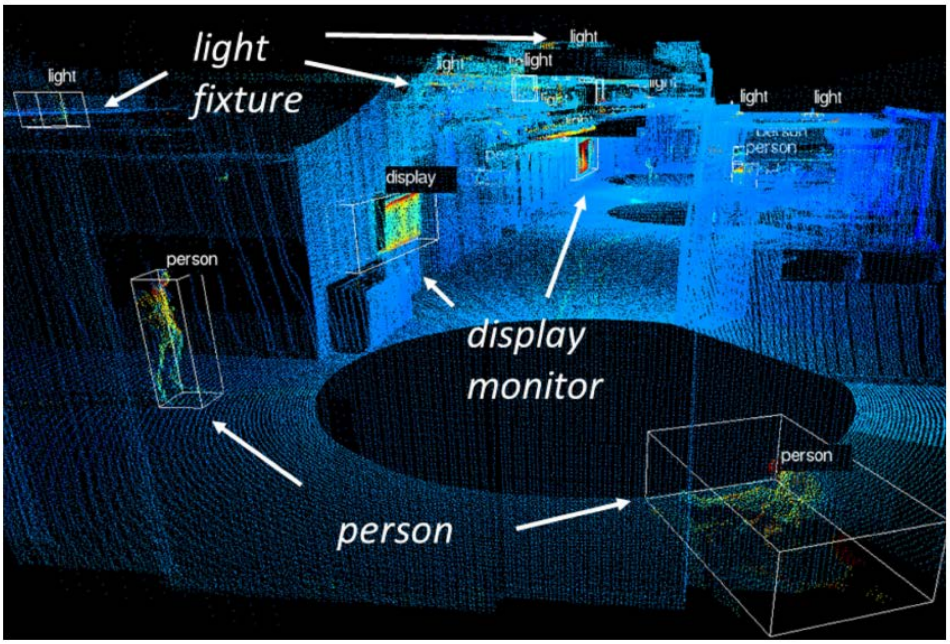


Figure 19: Object segmentation results using thermal and geometric information (by Kim et al. (2017))

Although not directly associated with point clouds, Zhang et al. (2014) study contextual understanding of panoramic scenes. The scenes consist of a whole room (acquired by 360° horizontal and 180° vertical FOV), allowing them to generate 3D bounding boxes of the rooms and objects inside, and to generate hypotheses based on contextual constraints and semantic information. Their workflow consists of estimating vanishing points (the points where perspective lines in 2D projections converge to) through HT, generating room layout and object hypotheses, and creating several whole-room scene hypotheses from the room-object hypothesis pool. The next step is whole-room sampling, and scoring the different rooms on several attributes. These whole-room hypotheses are used to train an SVM and to select the best fitting hypothesis. For visualizing the process, Figure 20 shows three columns, with the first column showing the input panorama, the second column showing the results of the intermediate steps, and the last column showing the final classification as a 3D representation. The research shows promising results for indoor scenes and indicates that using context is a very powerful tool.

Figure 20: Workflow of PanoContext by Zhang et al. (2014): first column is panorama input and output of object detection results, second column visualizes the hypotheses and the third column visualizes the results in 3D

Armeni et al. (2017) showed similar work to the study of Zhang et al. (2014), but use panoramic RGB-D point cloud data for indoor scene understanding. The proposed method allows for mutual registered models from the 2D, 2.5D and 3D domain. Their contribution to the field is that they propose an inventive method for registering models from different modalities and domains, allowing for a new set of learners to be developed for contextual scene understanding.

2.5. Limitations in literature and final conclusions

The literature shows that shows that point cloud data has a wide variety of applicability in the AEC and BIM industry, but the usage of raw point cloud data is rather limited. Conventional methods transform point cloud data into meshes for BIM model reconstruction or comparison, and into voxels for object recognition. There is quite some research conducted and many workflows have been proposed to automatically process point cloud data to increase its direct applicability. The key aspect of processing point cloud data is segmentation and classification of point or point clusters, which evidently seems to be the main challenge.

A significant amount of workflows to segment and classify point cloud data for the AEC industry incorporate manually designed rulesets, which are based on contextual logic such as "walls are vertical", "walls intersect with floor and ceilings" and "walls have a certain height". Approaches like these break down when they are used for slightly different environments which is increasingly common in the AEC industry, especially with significant advancements in the AEC industry such as parametric design. Additionally, occlusion and clutter pose serious problems in the segmentation and classification process, since they block important data points or are classified unintentionally as a certain class, and removing objects and furniture prior to scanning is impractical.

Several features have been introduced to avert these problems, and act as an extension to the point cloud data for increasing the accuracy in segmentation and classification methods. These descriptive features allow for more discriminative power, but using too much can unwillingly overfit models, thus decreasing their generalizability. Additionally, even though high-level features and statistical models are combined for machine learning, they seem to be tailored to certain scenes or specific classification problems, reducing their generalizability for changes in test data and reducing their usefulness for adopting them in BIM related classification problems.

With machine learning being re-introduced and becoming very popular again caused by increased computational power of CPUs and GPUs, researchers have turned to these kinds of methods to solve the automatic segmentation and classification problem. Especially in 2D applications, machine learning has seen significant advancements with the introduction of CNNs, but in turn, slowed down research in 3D applications as in computer vision and object recognition, which still seems to be in its infancy state. However more recently, deep learning has become increasingly popular to solve classification problems, especially for 3D volumetric models and 3D point clouds, discarding the need for heavy and complex statistical methods. While CNNs can be used for problems with 3 or more dimensions, computation time increases exponentially, thus researchers have come up with several tricks to circumvent this. These methods work for separate object classification, but are not directly extendable to indoor scenes, and depend heavily on preprocessing and transforming point cloud data. To conclude, context enabled deep learning methods on indoor scenes with specific classification labels, for the utilization within use cases regarding construction management, is a topic that is hardly investigated.

Another topic that is worthy to point out is the abundancy of user interaction within the segmentation or classification process in literature. Most, if not all literature incorporates workflows that rely heavily on their automated workflows and algorithms, and ignoring the process of manual enhancements. This makes sense in a theoretical point of view, as the accuracy of models provide a good comparison value, but regarding the AEC industry and buildings in specific, accuracy is a poor way of describing the classification of environments where elements share many similar attributes and where sometimes elements can converge into one another. The result is that user interaction is not taken into account, which would otherwise be a very powerful addition for increasing the quality of output a model can give.

3. Methodology

Based on the literature study and the research question from section 1.3, the following chapter describes the techniques that have been applied for the development of the interactive classification method. Section 3.12 will give an outline of the validation of the proposed method, by incorporating a set of test cases.

3.1. Introducing deep learning for point cloud classification for the AEC industry

From the literature, it is concluded that point cloud data is a valuable data type in the AEC industry, ranging from data storage in a data repository, 3D BIM model reconstruction to automation in process management and even object recognition for robots. The main bottleneck is the segmentation and classification of point cloud data, which is manually tedious and time consuming. Therefore, researchers have come up with various methods to automate this process.

Workflows proposed by researchers incorporate manually designed rulesets that follow a sequential hierarchy or contextual logic, although they seem to lack a certain generalizability, as they are tailored to solve a specific problem, i.e. in this case a point cloud scene. However, the AEC industry is known for being aberrant, especially the introduction of BIM and 3D CAD software allows engineers and architects to design complex parametric buildings. Using current inflexible workflows, in combination with problems such as occlusion, clutter and shadows, seem to be insufficient for fully automating the segmentation and classification process.

Now that machine learning, and more specifically ANNs and deep learning, have had a re-introduction due to increased computational power and an increasing number of easy-to-learn APIs for design and implementation, it is an interesting idea to explore how these kind of deep learning algorithms handle 3-dimensional point cloud data. There is already plenty of research conducted for CNNs for the 2D image classification problem, but only a small amount for 3D data. The research that is the most promising for classifying raw point cloud data, is the methods proposed by Li et al. (2018) and by Qi et al. (2016), which uses a complex CNN for classifying raw 2.5D point cloud data from an RGB-D sensor with only the (x, y, z) coordinates of points, RGB colors and normalized location to the room (ranging from 0 to 1) as input vector. Their model works well because convolutional models have been optimized through the years and given the fact that they use a huge annotated dataset of 271 rooms to train their algorithm.

In any case, the current methods basically make it impossible to pick a single point from the point cloud and classify it correctly through 'understanding' its' surroundings, and the shape and surface that it is part of. None of the methods use spatial context or information of what the most accurate class of a certain point in space would be.

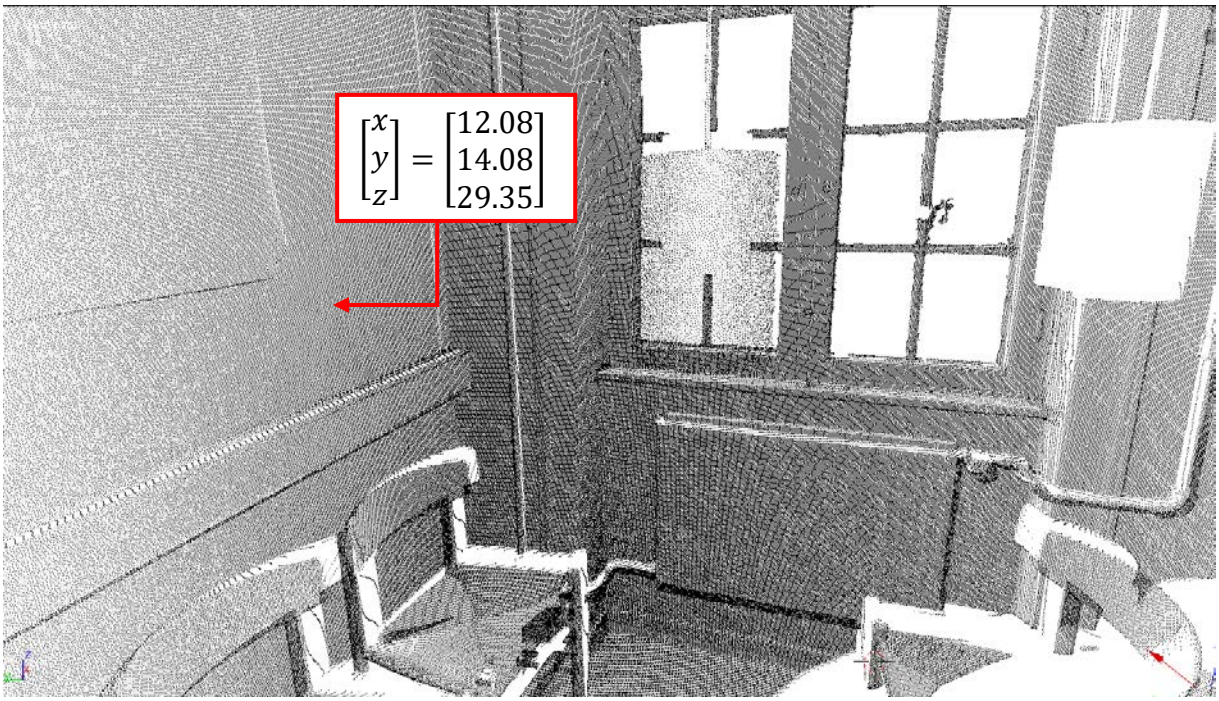


Figure 21: Point cloud scene example taken from the Byg72 dataset from the DURAARK online repository

To elaborate on this problem of incorporating *context*, let's consider the point that is appointed by the arrow in Figure 21. Humans would consider this point to be a wall point, but for computers this is not an evident fact, as the only information they 'see' is a range of numbers. This human recognition comes from a hierarchy of reasoning and recognizing abstract concepts. On a larger level of detail for example, the question "*what makes a [...insert construction element...], a [...insert construction element...]?*" is an interesting one for asking oneself to decide which cluster of points belong to a certain element. For example, a column has a cylindrical or horizontal cuboid-like shape, with a floor attached at the bottom, and a ceiling or beam attached at the top. Simultaneously, a wall has the floor attached to its bottom, and the ceiling to its top, and has the property of vertically segmenting empty spaces within the point clouds also known as rooms. This seems to be a trivial way of thinking for humans, but computers lack this abstractionism or cognitive ability.

ANNs seem to bear the closest resemblance to this human reasoning. The reason that ANNs are so popular is that they, beside the fact that they work well with large amounts of data, are essentially very simple. As Chollet (2017) states that in deep learning, everything is a vector, i.e. everything is a point in a geometric space, and each layer in a deep learning model operates geometric transformations to this data, and passes these transformed vectors to the next, until the final output vector. This chain of transformations can be described as a complex function, where all layers and neurons constituting that layer are updated after the final output vectors are compared to the desired output.

However, this simplicity has its limitations. Although it was previously mentioned that ANNs closely resemble human reasoning, the reality is that they do not, at least not entirely the

same way humans do. Figure 22 by Chollet (2017) clearly visualizes the part where machine learning models differ from human learning, where the key take-away is that machine learning models do not *understand* the task they perform.

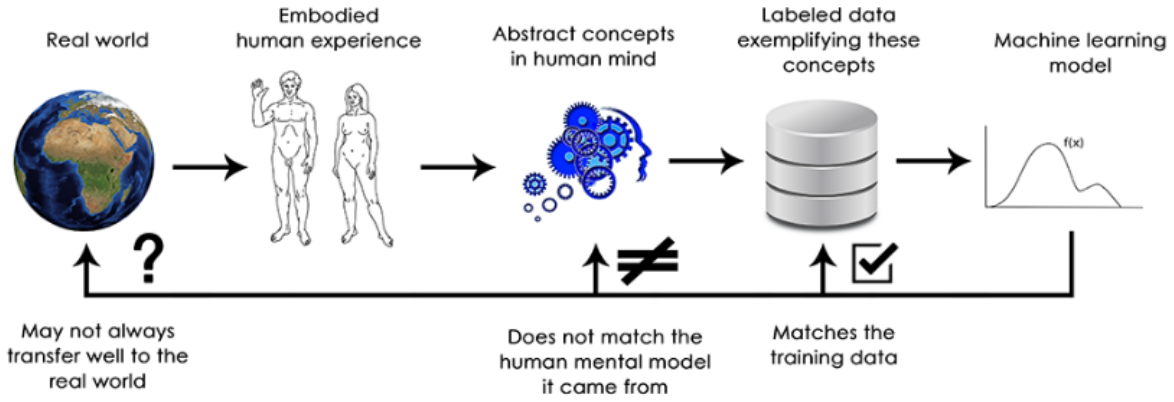


Figure 22: The difference between human learning and machine learning by Chollet (2017)

Regardless of this limitation, deep learning models do show promise, such as mentioned in the studies on deep neural nets on point clouds from section 2.4.1, as long as they are modelled with a certain scope in mind, and guided through the use of features to increase discriminative power within data. Combined with increased knowledge and, sophisticated and open-source APIs, the following method of this research will incorporate ANNs to automate the segmentation and classification process of point cloud data sets.

3.2. Method

The method used in this paper for the initial semantic classification of point cloud datasets is a Multi-layer perceptron that serves as a base model, which will be thoroughly explained in section 3.2.1 till section 3.2.5. To train the neural network, the point cloud dataset that is chosen will be described in section 3.3 and features will be extracted as described in section 3.4 and 3.5. The results of the trained model will be shown in section 3.7 and discussed in 3.8. The trained neural network will then be used as a base model to incorporate user interaction through the use of an interactive region-growing algorithm, such that point clouds can be classified where the user has more control over parameters within this classification task. The scripts created in Python including a detailed workflow of their utility can be found on GitHub. The link can be found in Appendix II.

3.2.1. The feed-forward neural network (standard sequential model)

The research approach incorporates a multi-layer perceptron (abbreviated as MLP, or also known as deep feed-forward neural network) as a method for solving the classification problem of point cloud data. Unlike CNNs or RNNs, an MLP is a more 'vanilla' type of neural

network, without convolution layers or the recurring capability of neurons in an RNN. An MLP consists of neurons (also called 'perceptrons' or 'nodes') that are ordered in different layers: the input layer as the first layer, the output layer as the final layer, and in between the hidden layers. A single neuron, such as represented in Figure 23.1 takes several inputs, and uses weights that express the importance of the input to compute the output. The neuron gives an output when the sum of the input multiplied by the weights is higher than a certain threshold. Figure 23.2 gives a more detailed overview of a single neuron, where $\sum_i w_i x_{ij} = net_j$ is used as input for the activation function where the threshold decides whether net_j is passed onto the next layer of neurons.

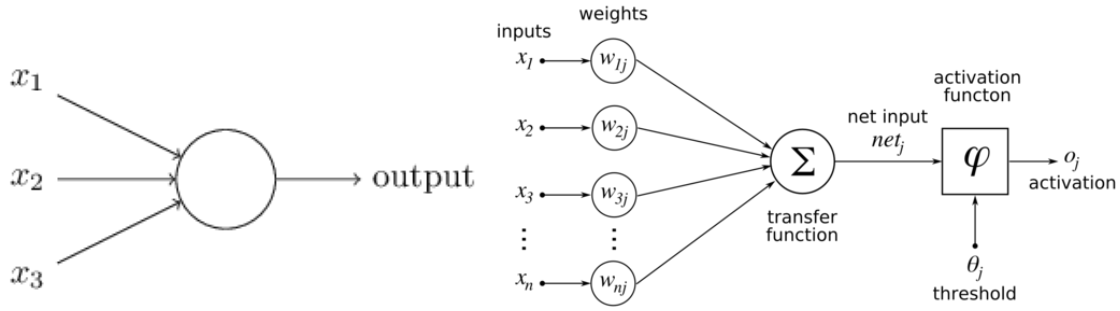


Figure 23: Schematic representation of a perceptron (left) and a multi-layered perceptron with weights (right)

This concept is similar to that of making a regular decision or classification, based on factors multiplied by their respective importance or weight to make the decision or classify the input into a certain category. Evidently, stacking multiple neurons such as represented in Figure 24 creates a method that can make more complex decisions.

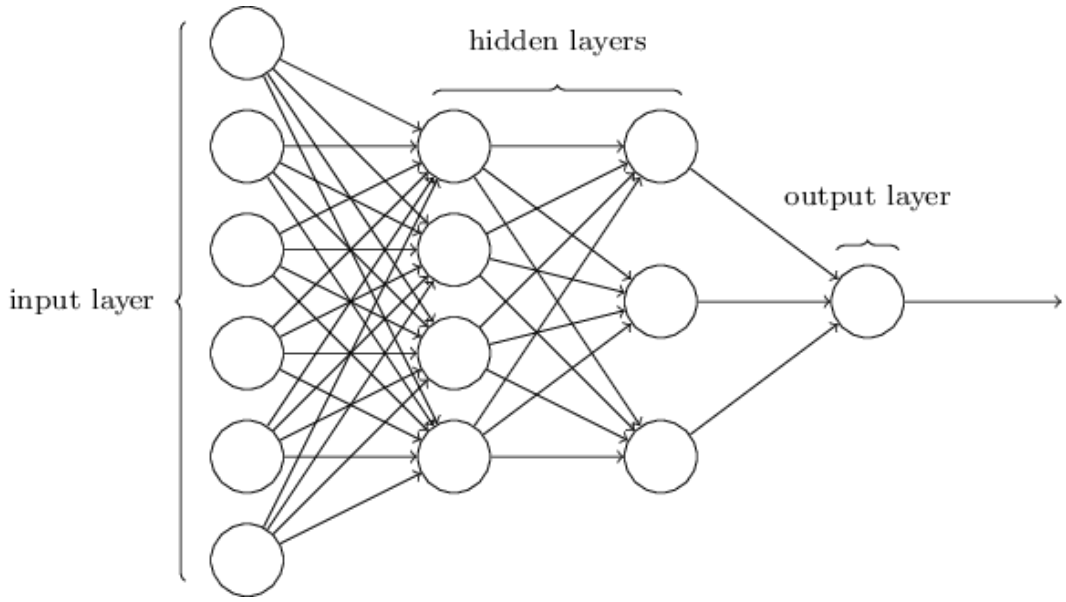


Figure 24: Example of a Multi-layer perceptron (MLP)

The parameters for designing MLPs, such as the number of nodes, layers, learning rates, the usage of regularization techniques, choosing the type of activation functions, etc. are all

dependent on the machine learning problem, the amount of noise in the dataset and the number of training cases. A brief explanation of these parameters including the design choices made for this methodology will be given in the next paragraphs.

3.2.2. Layers

The first layer of an MLP is the input layer. The conventional dimension of the input layer is equal to the amount of input variables, i.e. the dimension of the input vector. The hidden layer(s) of an MLP are responsible for the transformation of the input vector to gain the desired output. Unlike CNNs or RNNs, the amount of layers for an MLP is usually one or two maximum, since adding more layers does not increase the accuracy of the model. This is in contrast with CNNs which do benefit from using many convolutional layer to discover data signatures, and RNNs which benefit from more layers for memorizing sequenced data.

The final layer of an MLP is the output layer which is fully connected to the neurons from the last hidden layer. In order to estimate in which output category the input data belongs to, an activation function is used to transform the data from the last hidden layer into readable and sensible output, such as a probability vector or class prediction. For a multi-classification problem, the number of neurons in the output layer is equal to the amount of classification categories.

3.2.3. Activation functions

A neuron within a neural network receives signals from other neurons, where the weight that belongs to this single neuron gets multiplied by the incoming signal. Then, if the signal is still strong enough to pass the threshold, then it is propagated further onto the next layer of the neural network. This threshold is defined by an activation function, either a linear function or non-linear function.

The first types of deep learning neural networks used the sigmoid function $\sigma(x) = \frac{1}{1+ e^{-x}}$ or the tanh function $anh(x) = \frac{e^{x}- e^{-x}}{e^{x}+ e^{-x}}$, because it returns a non-linear function of the neural network, which is desirable if the classification problem is also non-linear. However, the problem with these functions is that they cause very slow learning during backpropagation, since the weights of the neurons will be updated with minuscule values. This issue is called the vanishing gradient, and therefore the proposed method uses the Rectified Linear Unit (ReLU) as the activation function for the deep layers and the input layer. This activation function is expressed by $f(x) = \max(0, x)$, where the input x < 0 gives output 0 and x > 0 gives output x. This activation function rapidly increases learning, but can cause the algorithm to update the weights in a way they may die off and not pass any signal anymore. Therefore, the learning rate needs to be chosen carefully, as too much will blow up the neuron where $x \to \infty$, and too small will slow down the learning process tremendously.

For the output layer, a different activation function is used, based on the problem that the neural network is supposed to solve. In this case, the neural network is used to solve a multi-

class classification problem, which uses the softmax activation function. This function calculates the probabilities of each class in the output layer, within the range of 0 to 1 and where the sum of all the probabilities equals to 1 (categorical distribution). Take note that the output of the layers containing ReLU activation functions can become quite large as they have no upper bound. To get a clear solution to the classification problem, the output layer only applies a transformation to the output of the last hidden layer, such that the final output of the neural network is a probability vector where the largest probability will be used to predict the class or category an input vector belongs to.

3.2.4. Backpropagation and optimization

Feed-forward neural nets learn through supervised learning, where pairs of the input and output are fed into the network, so that the neural net learns the relationship between input and output. This comparison is called backpropagation, where a cost function is used to determine the weights of a neuron to minimize the error. The most known cost function is the Mean Squared Error (MSE) in statistics, but for a multi-class classification problem categorical cross-entropy is the most suitable cost function, defined by:

$$C = \frac{1}{n}\sum_{x}\sum_{j}\left[y_j \ln a_j^L + \left(1 - y_j\right)\ln(1 - a_j^L)\right]$$

Where $y = \{y_1, y_2, \dots, y_j\}$ are the desired output values at the output neurons, and $a^L = \{a_1^L, a_2^L, \dots, a_j^L\}$ are the actual output values where j denotes the individual neurons in output layer L. Now, instead of feeding data in the direction from the input layer to the output layer (feed forward), the difference of the actual output vs the desired output (the error) is back-propagated into the network in the opposite direction, such that the weights of the neurons can be updated accordingly, before the next batch of input data is presented to the MLP.

3.2.5. Overfitting, dropout and regularization

One of the major issues of ANNs in general is the occurrence of overfitting. Overfitting occurs when the neural network is very accurate on training and validation data, but very inaccurate when external data is presented to the model, i.e. the error on training data is very small but is very large on new situations, which means the network has memorized the training examples and is not able to generalize to new data. Overfitting is caused when the neural network is trained with too many epochs, when the input data contains too many features, or when the dataset is too small to create a model with good generalization abilities. Fortunately, there are several techniques to prevent this situation, such as early stopping, adding dropout, and adding regularization.

Dropout is a method introduced by Srivastava et al. (2014) to prevent ANNs from overfitting by dropping a certain percentage of the neurons from the neural network, in a random way for every raining iteration. Additionally, dropout prevents neurons to become inactive or to 'die off', which is caused by using ReLU as an activation function. The percentage of dropout regularization is arbitrary, but ranges mainly between 10% - 40% for the best results.

Weight regularization is another simple method to prevent ANNs from overfitting and to prevent the weights becoming too large. This method uses the L2 parameter norm, also known as weight decay, which adds an additional parameter to the cost function, in the expression:

$$C = \frac{1}{n}\sum_{x}\sum_{j}\left[y_j \ln a_j^L + \left(1 - y_j\right)\ln(1 - a_j^L)\right] + \frac{\lambda}{2n}\sum_{w} w^2$$

Where w is the sum of squares of all weights in the ANN and λ is the parameter between 0 and 1, where a λ close to zero focusses on minimizing the cost function, and a λ close to 1 focusses on keeping the weights small. The important take-away of using weight decay is that the input with a heavy weight becomes too dominant over other inputs, and that subtleties in the data that would otherwise increase the accuracy on external data, become ignored.

3.3. Dataset selection

For training the ANN, and for testing and experimenting, the Byg72 point cloud data set from CITA University (DK) is used, which is an open-access point cloud data set downloaded from the DURAARK dataset repository. The dataset contains 22 laser scans from a single floor from the architecture department of the CITA University in Denmark. It contains 8 floors and two connected hallways in total, and it contains a significant amount of occlusion and clutter. The dataset is chosen because there are no pre-classified point cloud data sets of indoor scenes that are publicly available, at least not as a 3D representation (the only dataset that has the most resemblance is the one used in PointNet by Qi et al. (2016), although it is a collection of manually classified 2.5D scans represented in a 1m by 1m frame of 4096 points). Additionally, the Byg72 dataset is relatively small in size which is a necessity due to time constraints of this thesis, and because of its significantly occluded interior environments, adding realism to the problem that the interior of buildings are not uniform, and that machine learning applications need to cope with this problem. A visual representation of this dataset is given in Figure 25.

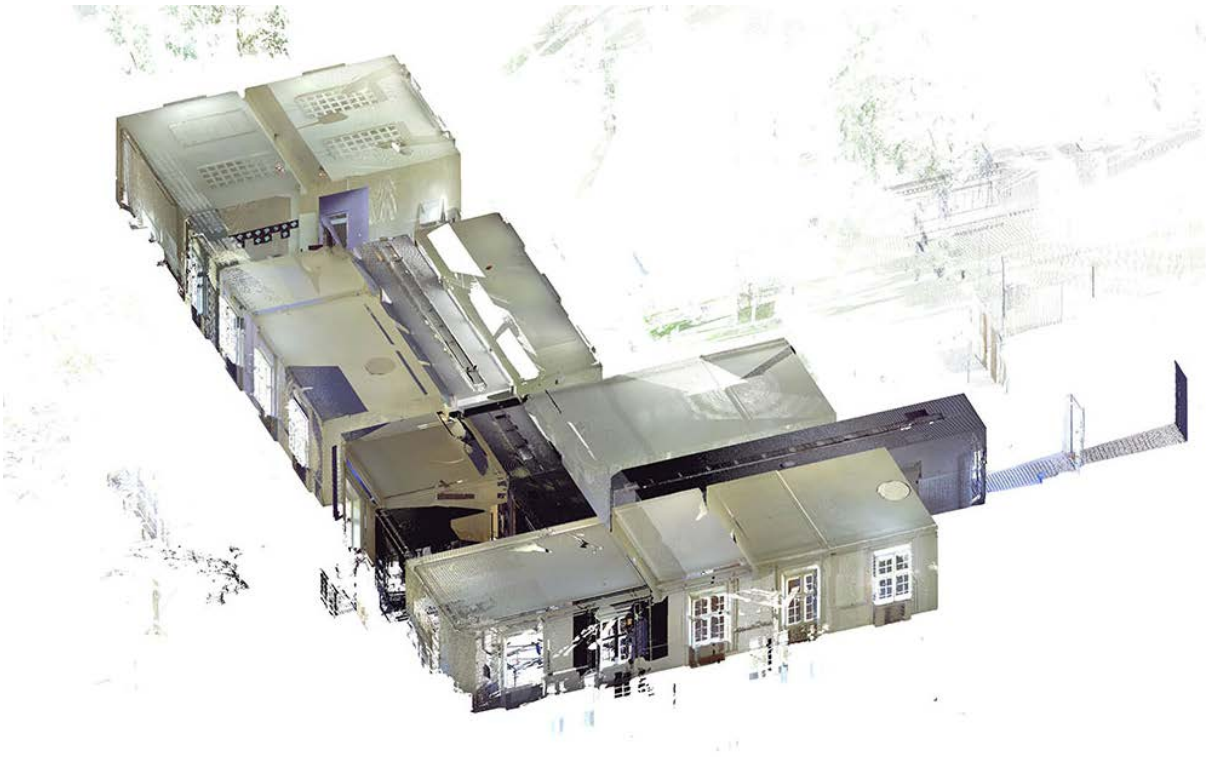


Figure 25: Visual representation of the complete Byg72 point cloud dataset from the DURAARK online repository (http://duraark.eu/data-repository/)

The dataset has been subsampled to .ply format to a total of approximately 55 million data points, allowing the .ply files to be imported into Blender for manual classification into 10 categories: beams, ceilings, chairs, columns, doors, floors, tables, walls, windows and finally occlusion. Consequently, after manual classification, the amount of data points of each scene is exported with a factor 0.1 due to the computational complexity of feature extraction (which would otherwise be infeasible).

3.4. Tools and implementation

The first tool used to retrieve and immediately sub-sample the point cloud data is CloudCompare, a powerful tool to efficiently visualize large point cloud datasets, including a wide variety of libraries that allows users to import and export point cloud data in many formats. The tool used for manual segmentation and classification of all 22 point cloud scenes is Blender. Note that when .ply format is imported into blender, all information except for the xyz coordinates is lost (such as RGB data, intensity values, etc.), due to limitations of the importer and Blender's inability to represent vertices with color information.

Besides some feature extraction algorithms, all experiments, algorithm set-up and computations are done in Python 3.5, because it allows for a better compatibility since this version is similar to the Python API of Blender v2.79 that is used for pre-classification and final visualization, allowing the algorithms and machine learning models to be integrated in

Blender's Python API. Additionally for this project, the following list of Python dependencies and packages have been used:

- Numpy 1.13.1
- Matplotlib 2.0.2
- H5py 2.7.1
- Pandas 0.20.3
- Pyflann3 1.8.4.1
- Scikit-learn 0.19.0

Creating an ANN from scratch is a very complex and tedious task, and for this reason, there are many frameworks available with each having their own pros and cons. The choice of framework for implementing ANNs in Python is Keras with TensorFlow-GPU as back-end API. Keras is a high-level neural network API, released in March 2015 that runs on a lower-level neural network API such as Theano, Tensorflow and CNTK. The reason for using Keras is that it is easy to learn and simple to use for researchers without going too much into detail of ANN architectures, or requiring much knowledge of all parameters ANNs can contain. This allows researchers from other departments than computer and data science to quickly set up simple ANNs to solve their machine learning problems.

Tensorflow is an open-source library for fast numerical computing (matrix multiplication, since that is what training ANNs essentially is), specifically tailored for deep learning neural networks. It has been initially developed by the Google Brain team for internal use, but has been publicly released as open-source in November 2015.

For this project, TensorFlow is the primary choice as the neural network back-end API, due to several reasons. The first reason is that further development on Theano has officially been dropped per September 2017. The second reason is that TensorFlow (although Keras is used as front-end) has a much larger community support compared to Microsoft's CNTK. The final reason for using TensorFlow is the additional availability of computational graph visualization through Tensorboard. This allows users to effectively visualize and analyze their ANNs.

As previously mentioned, an ANN learns through feed-forwarding input vectors (or tensors, hence the name TensorFlow), transforming these vectors by adding weights and biases, comparing the output vectors with the desired output vectors, and finally back-propagating this information to update the weights of all neurons in the network. In essence, all these processes are matrix multiplications that increase in computational complexity when ANNs become deeper. Fortunately, GPUs are very effective at matrix multiplications due to their specific hardware, and it is for this reason that deep learning has seen tremendous growth in research and applications in recent years.

TensorFlow support GPU acceleration when the API is run on a system with an NVIDIA GPU that has a CUDA compute capability of 3.0 or higher, including CUDA Toolkit 8.0 and cuDNN 5.1 installed. TensorFlow-gpu is used for this project with the aforementioned set-up.

The last tool used in this project is Point Cloud Library (PCL) for FPFH feature extraction. As mentioned in section 2.2.3.3, FPFH is a histogram based feature for point clouds that describes the local geometry around a points in a point cloud data set.

3.5. Data preparation and feature extraction

Before the point cloud data is ready to use for machine learning, it needs to be preprocessed in such a way that it is suitable as input or training the ANN. The pre-processing stages are visualized in Figure 26. Since the original dataset contains approximately 400 million data points, it is impractical to load into Blender to classify the data, to extract features or to present it to a machine learning algorithm. Therefore, the data is first subsampled through CloudCompare with the spatial method with 0.05m as a minimum space between points. This reduces the total dataset to 55.4 million points. This data is imported to Blender to classify it into either of the 10 categories. Thereafter, the classified data is subsampled another time when exported to further reduce the dimensionality by a factor 10 to a total of 5.54 million points. The distribution of data points between classes in the final dataset is shown in Table 4.

Figure 26: Pre-processing stages prior to ANN training

Now that the data has been classified, it is ready to extract features from it. The reason for classifying the data prior to feature extraction, is that Blender can only import XYZ data from .ply file types, thus losing any other meaningful information. The features that are extracted are point normal vectors and curvature with 11 neighbors as a parameter for the nearest neighbor search, and FPFH for every point within the dataset with a radius of 0.25m. The radius is based on the study done by Rusu et al. (2009) as they state that a radius between 0.2m and 0.35m give the best results for indoor scenes. All features are extracted by PCL 1.8.1 according to the same methods as described as in section 2.2.3.3.

Table 4: Distribution of classes in dataset

Category	Number of points
Beam	88219
Ceiling	775497
Chair	60325
Column	70925
Door	117978
Floor	391757
Occlusion	563295
Table	219975
Wall	1223944
Window	180559
Total	3692474

To use the algorithm for calculating the normal vectors, curvature and FPFH, the scenes are converted from .ply to .pcd, which is the standard file type to use within PCL, since it allows for the incorporation of histograms and other descriptors that other point cloud file types do not support.

3.6. Training and validating the base model

The final data that is used for training the ANN consists of 37 parameters; 3 FPFH histograms consisting of 11 bins each, the normal vector with its respective uvw coordinates, and curvature. Additionally, the data contains the category classification (1) beam, (2) ceiling, (3) chair, (4) column, (5) door, (6) floor, (7) occlusion, (8) table, (9) wall and (10) window. Before the classification parameter can be used for training the ANN, the classes are converted to a binary class matrix, similar to converting the class to dummy variables. As indicated in section 2.2.3, the xyz-coordinates are not used for training, validation or testing, as these values are not pose invariant and even a slight change in location due to scaling or rotation would give erroneous classification results.

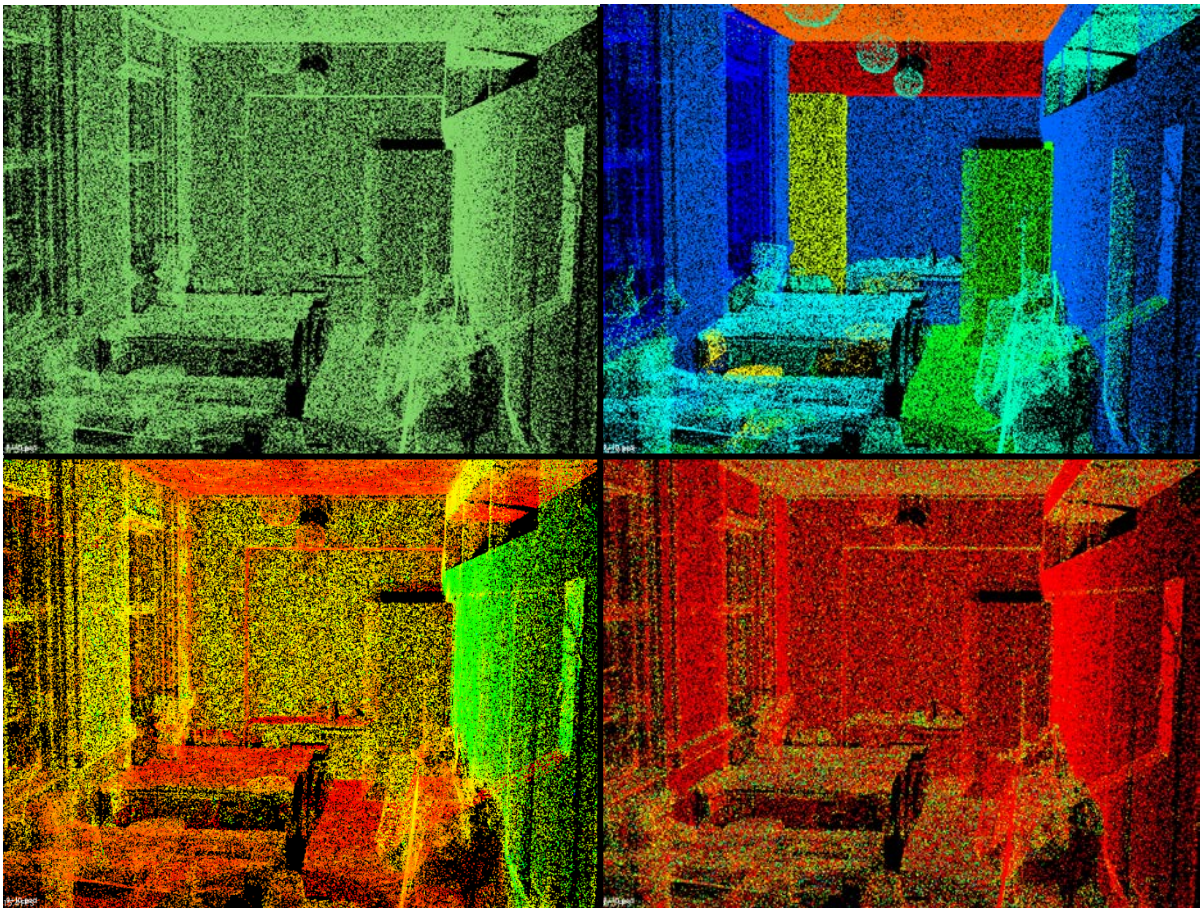


Figure 27: Scene from the pre-processed Byg72 point cloud dataset; (1) un-classified view, (2) classified point cloud data, (3) FPFH desciptor and (4) curvature

As shown in Table 1, the dataset contains a class imbalance, i.e. some classes are over-represented compared to other classes. The class imbalance needs to be solved, as training the ANN without balancing will result in the ANN learning too much on the classes that are over represented, such as walls and ceiling in this case. This is called the accuracy paradox, where the ANN will maximize accuracy without taking into account precision. Therefore, in this case where the used dataset consists of 33% wall points and 21% ceiling points, the ANN could classify everything either walls or ceiling (which are clearly distinguishable as explained later on), and therefore acquiring an accuracy of 54%. However, the goal of the ANN is to find smaller objects such as chairs and windows as well.

There are several techniques to combat class imbalance, such as gathering more data, re-sampling the current dataset or using a penalized model. Although gathering more data for this research is definitely desirable, class imbalance would still be occurring given the fact that point clouds of indoor scenes are very likely to contain more data points of elements that are more common or that have larger surfaces, such as walls and ceilings. Re-sampling the data would either mean to leave data out of the sample, however due to the data sample already being quite small for this research, excluding more data points would possibly result in a loss of valuable information to train the ANN properly. This issue could be solved by subsampling the dataset multiple times such that all data points are represented at least once, where after an ensemble of ANN classifiers (for each subsample training a separate ANN) and combining the output of these ANNs produces a unanimous output. However, creating such a method is not within the scope of this thesis.

The proposed method of this thesis to solve the problem of data imbalance, is by using a penalization technique. The weight for every class is calculated by:

$$w_{class(i)} = \frac{\sum_{i=1}^{n} m}{m_i}$$

Where $w_{class(i)}$ is the weight for class , n is the amount of classes and m is the amount of data points within class i. Table 5 shows the resulting weights for each class, which are used to penalize the misclassifications that belong to the class with a larger amount of data points. A lower weight will result in a greater penalty, whereas a smaller weight results in a smaller penalty.

Table 5: class weights for ANN training

Beam	Ceiling	Chair	Column	Door	Floor	Occlusion	Table	Wall	Window
13.87	1.58	20.29	17.26	10.37	3.12	2.17	5.56	1.0	6.78

Now that the issue of class imbalance has been resolved, the ANN is trained with the architecture schematically represented in Figure 28. The full schema of the ANN's architecture can be found in Appendix I.

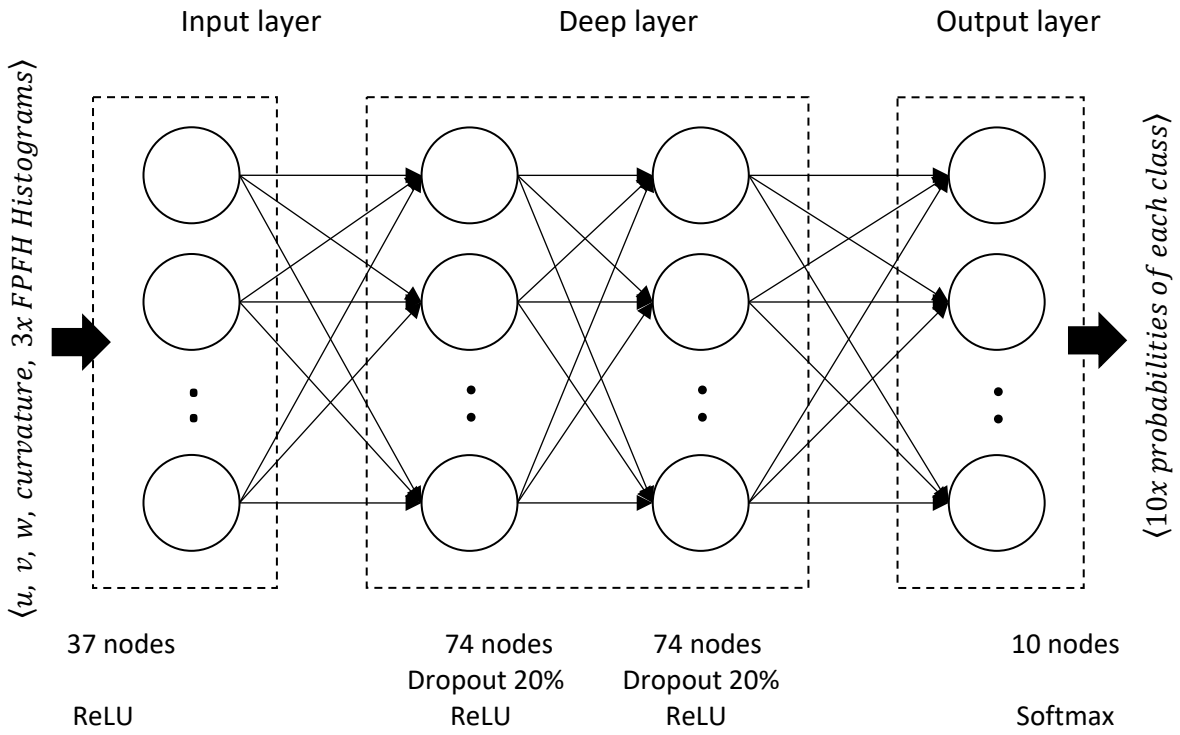


Figure 28: ANN architecture

The model has two deep layers, as from tests it became evident that adding more layers would not result in increased performance, and even showed that there was an increase in overfitting. To further reduce the chance and impact of overfitting, both deep layers have a dropout rate of 20% and use the ReLU activation function as described in section 3.2.3. Additionally, the amount of nodes in these deep layers is twice the size of the amount of input nodes, since more nodes do not increase the models' accuracy noticeably. The input layer consists of 37 nodes, as the input vector consists of 37 parameters (it is a requirement that the input layer has the same shape as the input vector), and the output layer consists of 10 nodes which is equal to the amount of class categories that are to be distinguished. The process behind the transformation from the input layer to the output layer has already been explained in section 3.2, but to recap, the output layer uses the softmax function to output a set of probabilities for each corresponding class prediction (10 in total), where the datapoint will be allocated to the class category with the highest probability value.

However while training, the training algorithm requires feedback if it is training properly on the training set, i.e. to validate if it is classifying properly according to the weights and features it is currently learning. Therefore, a portion of the training dataset is separated and used as validation dataset, to be able to provide the necessary feedback and to allow backpropagation to operate, as described in section 3.2.4. The percentage of data that is used for validation is called the validation split, and ranges often between 15%-33%. A few test have been done for different validation splits, but a percentage of 25% gave the best balance between prediction variance and validation accuracy. An overview of the total divisions made from the Byg72 point cloud dataset are represented in Figure 29. The fact is that the bigger the validation set,

the less data is seen by the model that can be used for training (and thus learning) but it will have a smaller variance in predictions. On the other hand a smaller validation set will provide the model with more data to learn, but with a higher variance (and thus a poorer generalizability on external data).

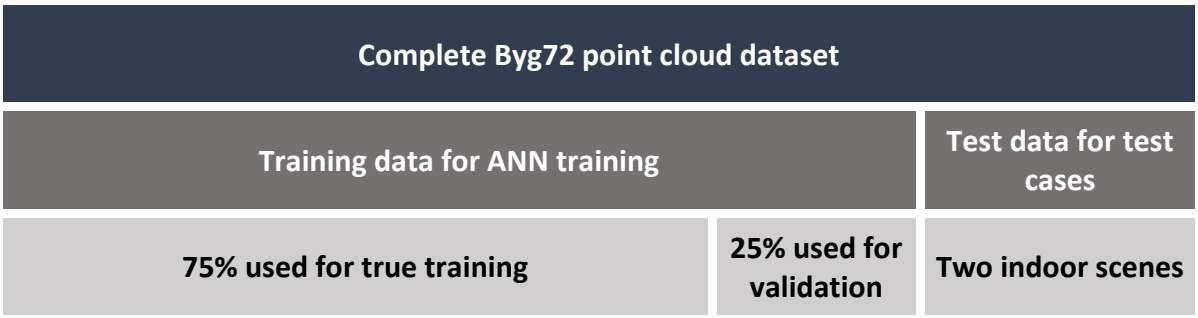


Figure 29: Overview of divisions made from the complete Byg72 point cloud dataset for training and testing

The validation accuracy is approximately 40%, and while it gives a good first impression of the result, it is not a good indicator of the performance of a model, and it is impractical to use for improving and refining a multi-classification model. To gain a better insight of the performance of an ANN, precision and recall are introduced. Precision is the ratio between the data points that are correctly classified (true positives) and the data points that have been falsely classified (false positives). This ratio is calculated by:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Recall is the ratio between the data points that the ANN correctly classifies and the total amount of data points of that respective class (true positives and false negatives), and is calculated by:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Finally, the F1-score measures the balance between precision as recall, since a good classifier should (typically) minimize wrong classifications, but also maximize correct classifications for a good trade-off, and is calculated by:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Table 6 gives an overview of the calculated precision, recall and f1 values that indicate the performance of the model for each individual class. From the values in the table, it becomes clear that the performance of the ANN for classifying tables and occlusion is quite worrisome. The ANN has problems finding or recognizing these classes, due to a low recall score, suggesting that the model has not learned enough features to recognize data points that

belong to these classes. On the other hand, the precision ratios of the beam, chair and door class are also low, suggesting that the ANN classifies too many data points as beam, chair or door, while in fact they are not. This could imply that the ANN learns features from these classes, which are shared with other classes. For example, for beams and doors this makes sense, as doors share the same geometrical properties with walls, beams share the same properties with ceiling and walls, and chairs might share similar properties with occlusion.

Table 6: Neural network model precision, recall & f1-score

Category	Precision	Recall	F1-score	Number of points
Beam	0.22	0.44	0.29	88219
Ceiling	0.70	0.51	0.59	775497
Chair	0.16	0.68	0.26	60325
Column	0.22	0.58	0.31	70925
Door	0.18	0.67	0.29	117978
Floor	0.39	0.69	0.50	391757
Occlusion	0.45	0.11	0.17	563295
Table	0.20	0.07	0.10	219975
Wall	0.83	0.63	0.71	1223944
Window	0.46	0.57	0.51	180559
Avg / Total	0.58	0.49	0.50	3692474

To give a better insight into misclassifications within classes, a confusion matrix summarizes the true and predicted labels for each class. Figure 30 represents the confusion matrix for the trained ANN in this research, which contains the false negatives left from the green line, the true positives (i.e. the correct classifications) between the green and red line, and the false positives right from the red line.

From the confusion matrix, it becomes clear that there is a lot of mislabeling between ceiling, floor and table. This can be attributed to the fact that these classes share similarities between normal vector orientation, curvature (flat surface) and the normal vector orientations of neighboring points. Unfortunately, the ANN does not take the spatial relationship between classes into account, such as “points of a table class are always above the floor” and “point of the ceiling class are always above the floor and tables”. Indeed, the xyz-coordinates that contain the spatial information to learn this ‘logic’ have been left out due to the fact that the ANN would learn the wrong values, as xyz coordinates are not universal, and therefore the ANN would not be able to generalize properly on external data.

Additionally, there are many predictions of the door class, which should have been either classified as occlusion, table, wall or window. This means that the ANN finds signatures in the data, which are similar between these classes.

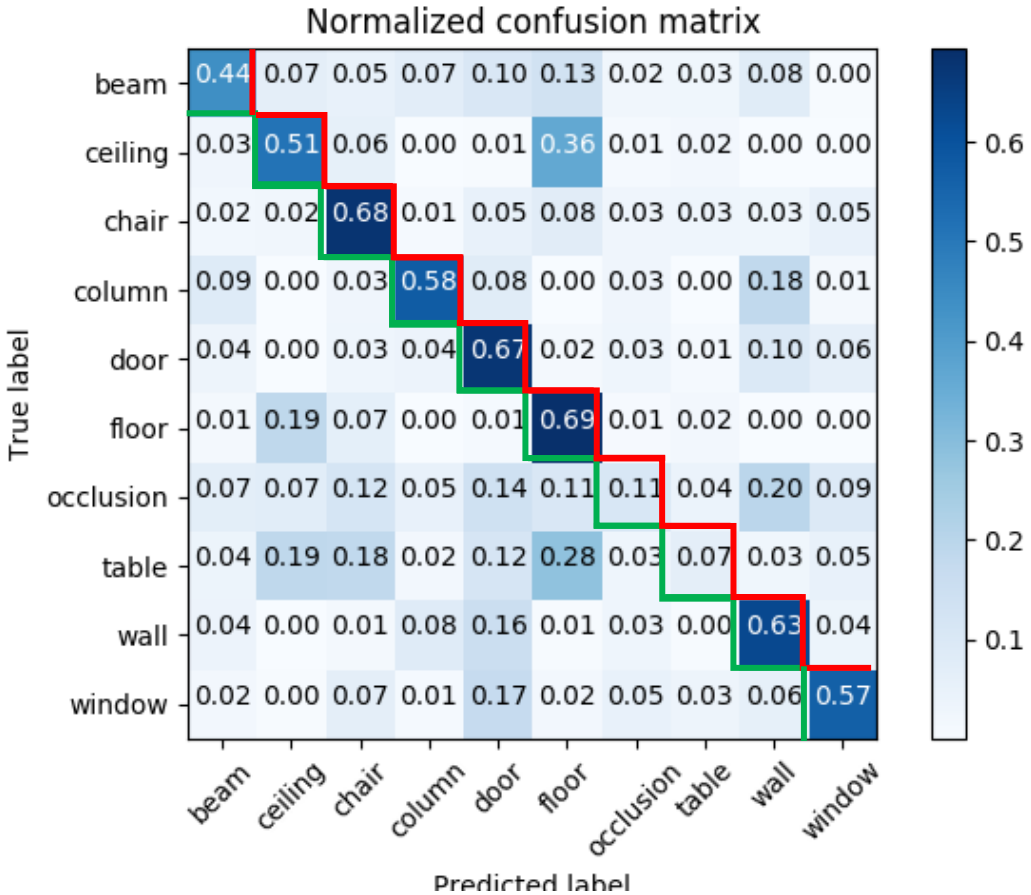


Figure 30: Confusion matrix of the trained ANN

3.7. Testing and results

Although section 3.6 gives an outline of how the ANN performs on validation data, the true performance and generalizability of an ANN can only be measured by using the model on test data. Alongside the dataset used for testing, two scenes have been left out for training and will be used to test the model. The results of the models' performance on these tests will be analyzed in this section.

The first test scene is an office or work space that is quite similar to the spaces that have been used for training the ANN. It contains clearly distinguishable floor, ceiling, a few windows and a few walls, and has some chairs and tables in the scene. Running the data through the ANN gives an overall classification accuracy of 33%, which is not too far off from the 40% of the validation data accuracy. However, to provide a better insight to analyze the result, a confusion matrix is drawn, shown in Figure 31.

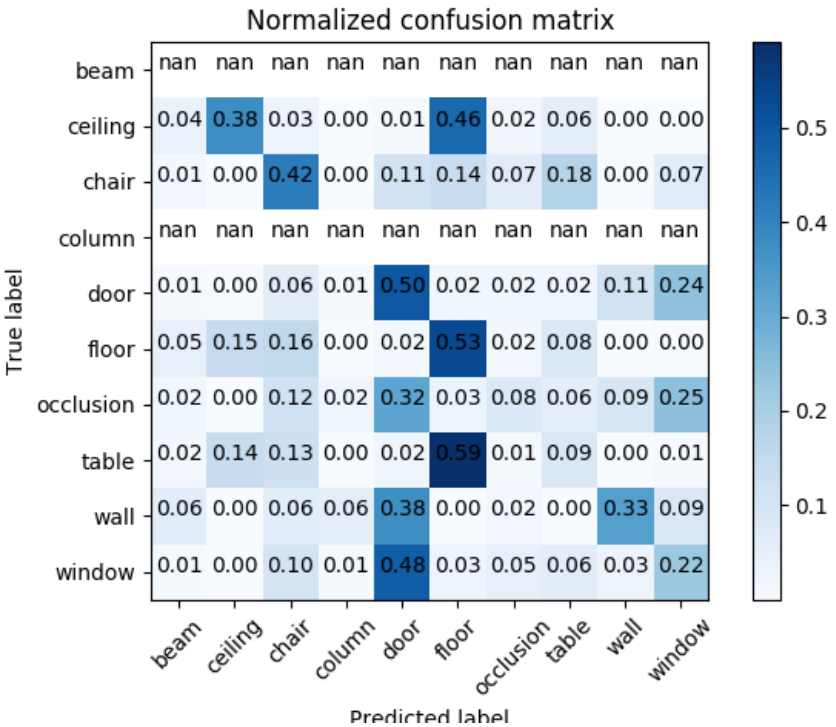


Figure 31: Confusion matrix for the first test scene. The NaN values indicate that the beam and column class are not present in this scene.

From the confusion matrix in Figure 31, it becomes clear that the ANN misclassifies certain classes to a considerable extent. Points from the ceiling and table class are repeatedly misclassified as either floor, window or occlusion, and in a similar fashion, walls are repeatedly misclassified as doors. Furthermore, doors and occlusion are often misclassified as window. To further analyze these misclassifications, the point cloud model is plotted with the true class labels (Figure 32.1), the (mis)classified labels done by the ANN (Figure 32.2), and two highlights from the labeling process that give a more detailed view of the occurrence of correctly classified and misclassified data points (Figure 32.3 and Figure 32.4). For these figures, the correctly classified points are colored in blue, while misclassified points are colored in red.

From Figure 32.2 it becomes clear that the ANN is not able to distinguish the table from the rest of the data, even though the table points represent a clearly distinguishable table as it is captured almost in full by the laser scanner, and given the fact that there are no objects on the table to provide clutter of any sort. An interesting observation can be made on the classification on chairs however, as the result from Figure 32.4 shows that the ANN is able to distinguish chairs (the lounge chair and the office chair) which differ in shape from the chairs that have been used for training the model (simple wooden classroom chairs).

Moreover, the door including the doorframe seems to be classified very well, although from further analysis, it seems that large parts of the wall that lies in the same plane of the doorframe are classified as door, which is supported by the amount of false positives from the confusion matrix in Figure 31. Therefore, it is not clear if the ANN has actually learned to

distinguish the doorframe correctly, or that it coincidentally annotated the entire plane as a door through erroneous learning, and that the door just happens to be part of that plane.

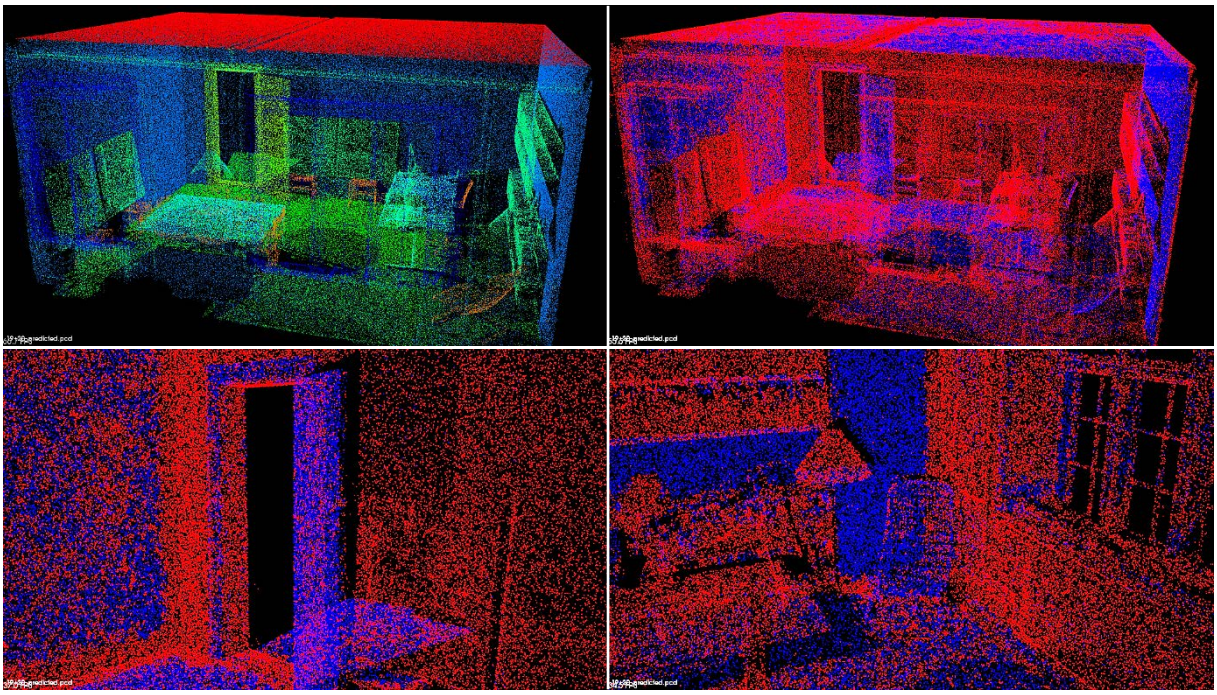


Figure 32: (1) True class labeling of the first test scene; (2) Class labeling done by the ANN; (3) Classification highlight result by the ANN of the door and (4) Classification highlight result for a corner with a chair (blue=correct label, red=incorrect label)

The second test scene is a small kitchen with a dining table and a few chairs, which differs quite a bit from the scenes that have been used for training the ANN. It is for this reason used as a test set to discover more effectively what the ANN has actually learned, and if it is able to generalize on data that is different. The ANN gives an overall accuracy of 17% on this test set, for which a better overview is given in the confusion matrix in Figure 33. This confusion matrix shows that the ANN is quite able to recognize data points that belong to the floor class, but also shows that it erroneously predicts the ceiling and table as a floor class.

Similar to the first test scene, the confusion matrix in Figure 33 shows that a great amount of points have been classified as door, despite the fact that this scene does not contain any points that belong to the door class. This suggests that the ANN has learned false signatures in the data for making a good distinction between the wall and door class, or that the pre-processed data contains variables that appear to be too similar for the ANN to distinguish.

From Figure 34.3, it becomes clear that the ANN is able to distinguish windows to a certain extent, especially the muntin bars (or window grilles) in the window seem to have signatures that are distinguishable from other classes. Figure 34.4 shows the high accuracy on the floor class within this test scene.

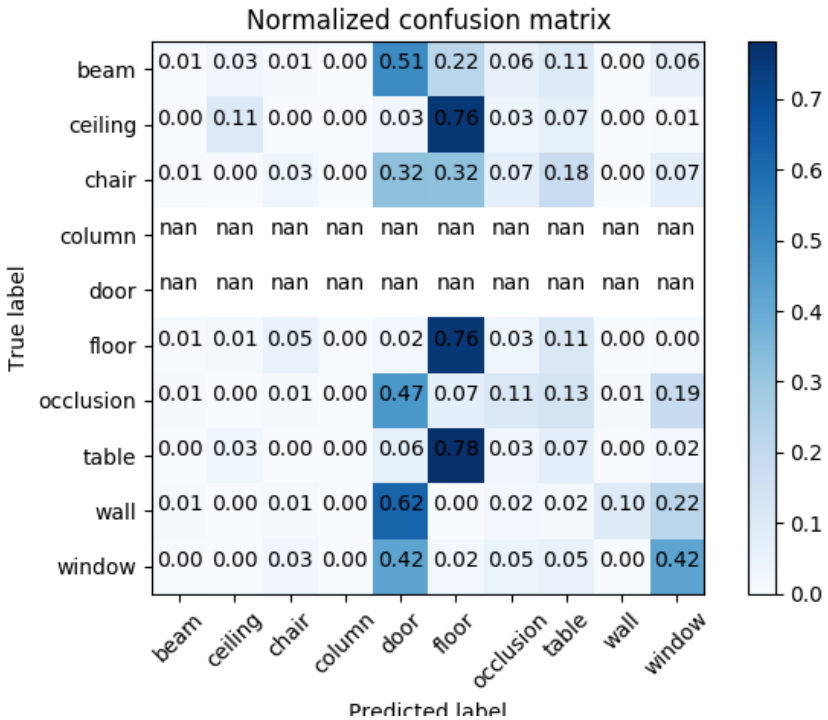


Figure 33: Confusion matrix for the second test scene. The NaN values indicate that the column and door class are not present in this scene

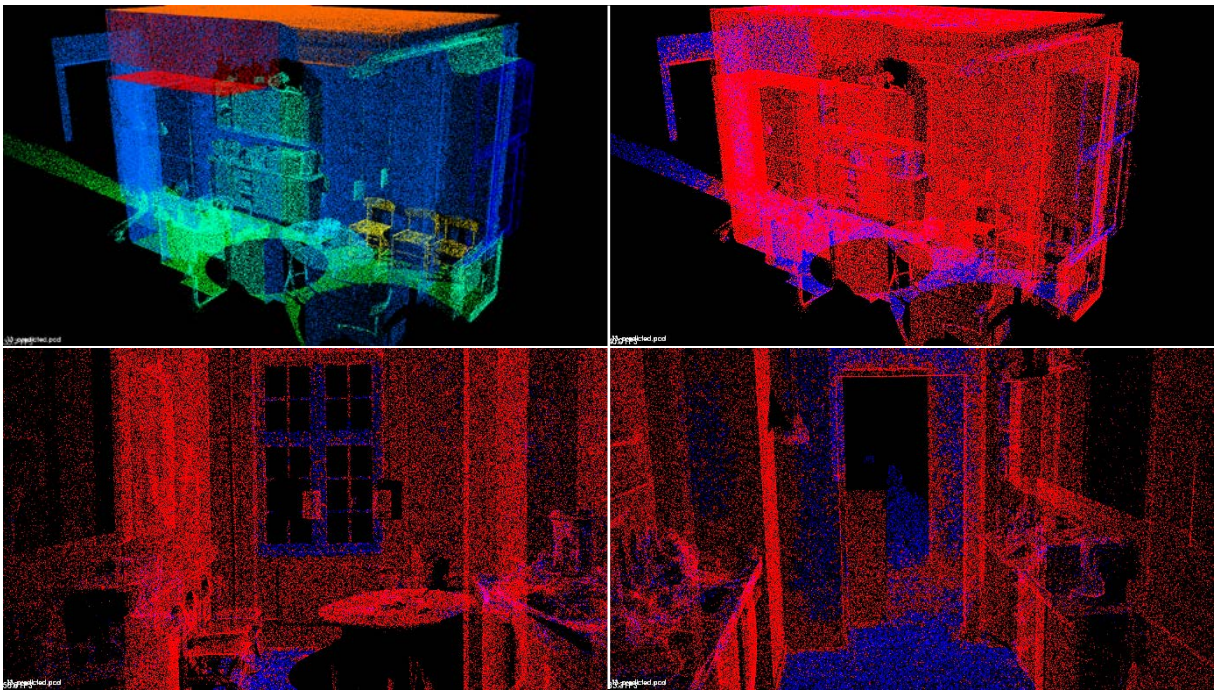


Figure 34: (1) True class labeling of the second test scene; (2) Class labeling done by the ANN; (3) Classification highlight result by the ANN of the window and (4) Classification highlight result for the floor and doorway (blue=correct label, red=incorrect label)

3.8. Discussion of the base model

A known issue of ANNs is that the classifier represents a kind of black box model, as the model provides no clear indication of why it classifies data the way it does, and it is hard to discover what the model has learned and what it has not. However, the point cloud data contains rich information to be visualized which can give substantial insight into the problems that occurred during learning and testing.

The most noticeable error the base model produced was the missclassified wall in the first test case as seen in Figure 35.1, while the surrounding walls of the room were correctly classified (notice the red wall cluster on the right, in comparison the the correctly classified blue wall cluster on the left in Figure 35.1). From the comparison of Figure 35.2 which shows the true class labels of the point cloud data, and Figure 35.3 which shows the predicted class labels of the point cloud data, it becomes clear that the wall section has been classified (almost entirely) as a door. Analyzing the same scene for the curvature values (Figure 35.4) and FPFH values (Figure 35.5), it becomes clear that both curvature and FPFH values are different for the missclassified wall section.

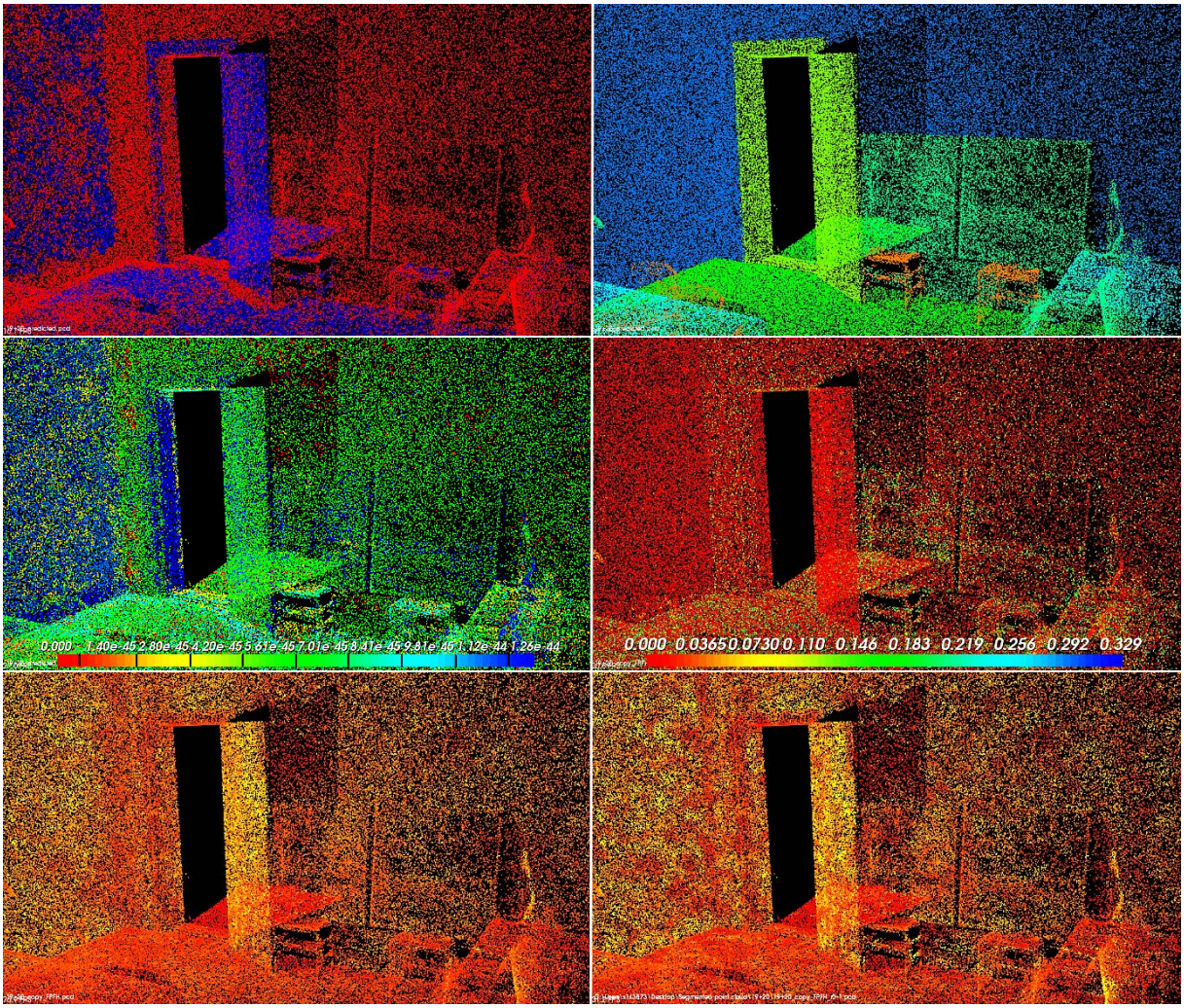


Figure 35: Analysis of missclassified wall by base model: (1) blue is correctly classified, red is missclassified; (2) true class labels; (3) predicted class labels; (4) curvature; (5) FPFH radius = 0.25m; (6) FPFH radius = 0.1m

It seems that the surface of the misclassified wall is not entirely flat as the curvature is very noisy (notice the yellow and green data points on the wall in Figure 35.4, which indicate significant change in point normal vector orientations), opposed to the curvature of correctly classified walls. The same can be examined from the FPFH feature in Figure 35.5, which can be attributed to the fact that FPFH is largely linked to curvature through the normal vectors and their orientations. However, the variation in FPFH could also be attributed to the presence of the occlusion near the wall (the paintings on the chairs that lean against the wall). To investigate this, the same scene has been plotted with an FPFH radius of 0.1m (Figure 35.6), which should ensure that less occlusion data points are taken into account due to the smaller radius for neighbor selection. Therefore, the angular variations ought to be smaller as less noisy data is taken into account. However, decreasing the FPFH radius does not seem to give a significant change, and even shows that the correctly classified walls would contain noisier FPFH values (note the noisy patches on the left wall in Figure 35.6 compared to the same wall in Figure 35.5).

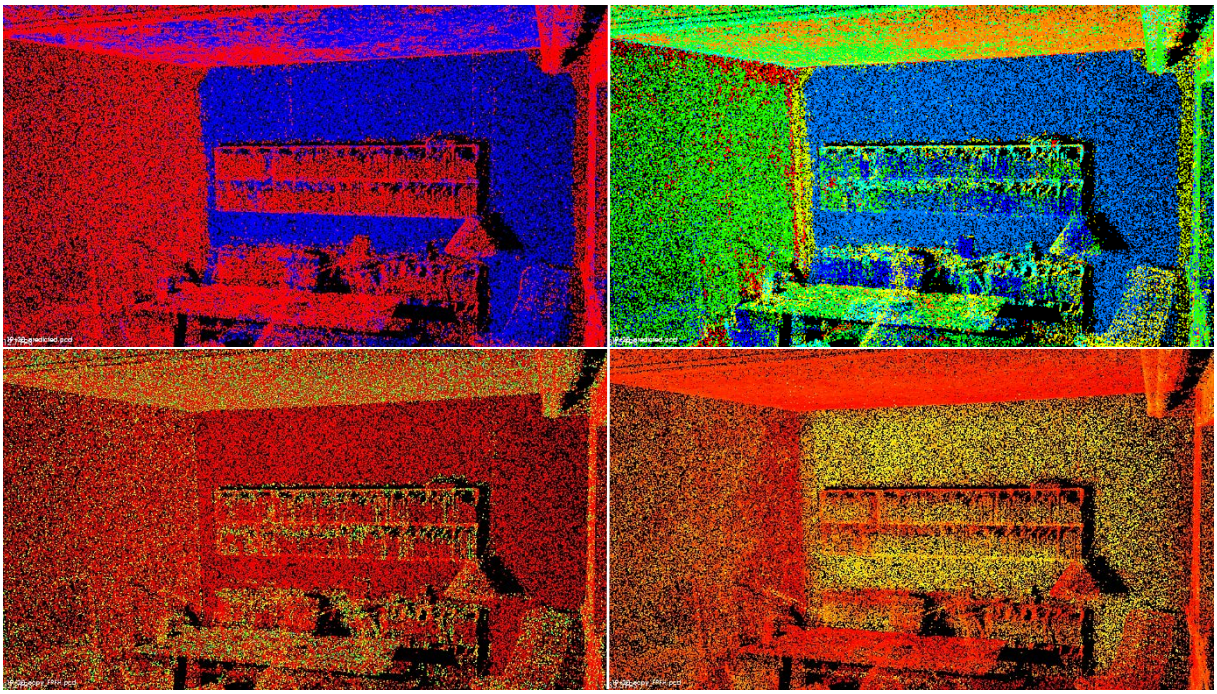


Figure 36: Analysis of missclassified wall by base model: (1) missclassified (red) vs correctly classified (blue); (2) predicted labels; (3) curvature; (4) FPFH radius = 0.25m

For further indication on how sensitive the ANN seems to be to the flatness of walls for the classification of walls in specific, notice Figure 36.1 where the entire wall has been (almost entirely) correctly classified, even including a significant amount of correctly classified wall points between the books on the bookshelf on the same wall. This seems to suggest that FPFH can be robust to low amounts of noise, but this is contradicted on the issue explained for Figure 35. Therefore, it can be concluded that the ANN has learned to classify walls correctly based on their flatness, resulting in walls that show imperfections are not classified correctly.

Another issue that is very apparent, as was already visible and mentioned in the training phase, is the incorrect distinction between floors, tables and ceilings. A small test has been conducted by using a normalized height feature, similar to that proposed in the work of Hackel et al. (2016), where the z-coordinate is transformed to two additional features as:

$$H_{above} = Z_{max} - Z \; and;$$

$$H_{below} = Z - Z_{min}$$

which would indicate a clear distinction where $H_{ceiling} > H_{table} > H_{floor}$. The trained ANN is then able to make a clear distinction between those three classes where the accuracy for ceiling, floor and table is 95%, 99% and 71% respectively. However, it is not clear if the ANN has correctly learned that $H_{ceiling} > H_{table} > H_{floor}$, or that it would output erroneous classifications when the height of the ceiling is increased (and thus increasing the H_{above}). Most presumably it has learned the latter, and is therefore not generalizable. Furthermore, the ANN would be unusable as it relies on the 'logic' that ceilings are always above tables, and tables are always above floors, which is not the case when multiple floors are stacked ontop of each other, or when the tables are not always above floors, with examples seen such as in cinemas or lecture rooms.

Another issue that became apparent during manual classification prior to ANN training, was the interpretation of certain elements and their corresponding categories that might have impeded the learning process. To give some examples, some classified tables that otherwise would be more appropriately classified as occlusion, closed doors that are classified as walls, but in reality are actually doors and the decision if paintings on walls would either belong to the wall or occlusion category. These issues are dependent on the interpretation, experience and goals of the person that manually prepares and classifies the point cloud dataset.

To extend on the issue of category interpretation, it is important to set the level of detail one wants to classify based on the goals for solving the classification problem. For example in the dataset prepared for training the ANN, the doors classified as such include the respective doorframes, but could also be interpreted as walls, or even distinguished separately as a 'doorframe category. Another example is the misclassification of certain window parts in both test cases, where the sills are often misclassified as chair and the window head is often misclassified as table. Apparently these 'sub-elements' show some overlapping features, but there is no literature or theory that explores the issue of choosing the right level of detail to solve this issue. Therefore, this task is entirely up to the user.

To expand on the issues in a general sense about the trained ANN as a base model including the respective results of both test cases, it seems that the ANN is unable to 'learn' context as it initially was expected to do. However, when zoomed into how the ANN is trained (as a multi-layer perceptron) and how it is used as a classifier, it becomes clear that the model predicts the class for every point independently. It seems the features calculated for every single point do not contain the geometrical information that is required for mapping surrounding geometry. Although FPFH proves to be a sound foundation for classification, it contains two significant limitations. A significant limitation is the absence of a distance metric for neighboring points. While the distant metric for nearest neighbors is used for the calculation

of the histograms, it is not stored into a separate histogram that describes the distance metric specifically for the point neighborhood. This is related to the fact that FPFH contains no spatial geometric information of the point neighborhood, unlike features such as SHOT or 3DSC.

To visualize this limitation, Figure 37.1 shows a MDS (Multi-dimensional scaling) plot and Figure 37.2 shows a t-SNE (t-distributed stochastic neighbor embedding) plot, which has been made of the training dataset. Both MDS and t-SNE are developed for visualizing similarity in a dataset and for visualizing dimensionality reduction, and in this case, show the distinguishability of the category classes. These plots confirm the erroneous outputs given from the ANN, as it becomes clear that classes show significant overlap, and that that relying on FPFH, normal vector orientation and curvature is often not sufficient.

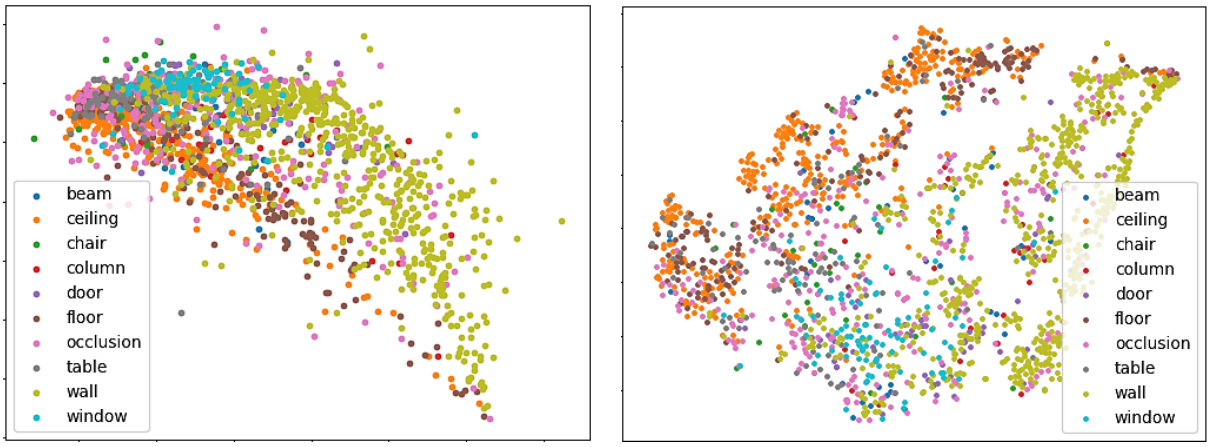


Figure 37: Visualization of class distinguishability by (1) Multi-dimensional scaling (MDS) and (2) t-distributed stochastic neighbor embedding (t-SNE)

3.9. Incorporating region growing and user interaction for increased accuracy

From the discussion in section 3.8, it becomes clear that points are classified individually based on the features they contain. Now let's consider that certain patches of a classified wall contain random door points. These door points are very unlikely to belong to the door class due to the fact that their surrounding neighborhood contain many wall points. Furthermore, given the issue of the correct distinction between ceiling, table and floor points, it is very unlikely that large patches of the ceiling class would contain table or floor points.

Secondly, as became evident from the literature review, construction elements are ambiguous where for example walls devolve into column or beams. Conventional region growing on normal vectors or curvature would not result in a proper distinction between these elements. When point cloud data would be pre-processed and classified by an ANN, resulting in probability tables for each data point, it establishes the possibility for new region growing methods, as more diversity in thresholds can be introduced.

Finally, beside the pre-processing of data and tweaking a few parameters of the ANN, user interaction is limited and the output values and the quality of these values is static.

Considering the previously mentioned problems of individual classified data points, and the ambiguity of construction elements, it is desirable to have some sort of user interaction with the data for more precise and effective classification results. The solution for these issues proposed in this method is incorporating the technique of region growing, which is an iterative process that examines neighboring points of the initial seed points, and determines whether these points should be added to the region, based on threshold parameters.

The ANN from section 3.6 gives for each data point a class probability table as output, such as exemplified in Table 8 and Table 9. Algorithm 1 gives an overview of the process that is used in this research regarding region growing. Given a list of seed points S within point cloud P that includes xyz-coordinates for nearest neighbor search and for every point $P(i)$ a probability table of classifications ρ_{class}, the threshold θ is calculated that includes an upper bound and lower bound. To calculate the upper and lower bound, the $mean$ and standard deviation is calculated for every $\rho_{class} \in S$. The upper and lower bound are defined by adding or subtracting the standard deviation $std(\rho_{class})$ multiplied by a factor m, from the $mean(\rho_{class})$. When the mean and std. are low initially, some class probabilities will fall above the upper threshold, even though it is desired that they are added to the region. In that case, the upper bound should be raised to a sufficient amount.

The list of selected seed points are inserted into the queue Q, which will pass these points onto the nearest neighborhood search function, and check if these points are within the threshold, so that they can be added to the final region list. The points that are taken from the queue, and passed to the neighborhood search, are tracked by storing them in an additional set $[Seen]$, so that the same points are not considered twice for the nearest neighbor search and added to the queue for the next iteration (to guarantee that the algorithm terminates). The library used for the neighbor search is FLANN (Fast Library for Approximate Nearest Neighbors) by Muja & Lowe (2009) and the algorithm used is k-means. The seed selection and visualization is done in Blender v2.79.

Algorithm 1: Region growing

Input

Point cloud = P

Probabilities = ρ_{class}

Neighbor finding function = FLANN with k-means

Threshold = $\theta(\rho_{class})$ where:

$$Threshold\ \theta(\rho_{class}): \begin{Bmatrix} upper\ bound = mean(\rho_{class}) + m * std(\rho_{class}) \\ lower\ bound = mean(\rho_{class}) - m * std(\rho_{class}) \end{Bmatrix}$$

List point seed = S

Output

Region list

Initialize

Q ← ∅

Seen ← ∅

Region ← ∅

For i in S **do**

    Q += i

While Q ≠ ∅ **do**

    If i ∉ seen **then**

        Q -= i

        Seen += i

        If i < $\theta(p_{class})$ **then**

            Region += i

            nns = flann(for i in P)

            For i in nns **then**

                If i ∉ seen **then**

                    Seen += i

Return region

3.10. Results and discussion of incorporating region growing

3.10.1. Test case wall – easily distinguishable elements

Consider the scene of Figure 32.3 where the wall has been incorrectly classified, and where the wall clearly contains noisy misclassifications within correctly classified wall patches and noisy misclassifications near the edges of the floor and corner. To test the proposed method from section 3.9, a seed region is selected (Figure 38) and inserted into the algorithm. The mean and standard deviations for every class probability of the total point seed list, is given in Table 7. For the threshold, only the wall class is considered, as it is the desired class selection. However, multiple thresholds with multiple classes and multiple factors for std. multiplication can be utilized to tailor the region growing algorithm to the user's preferences or goals. Choosing the right factor value is a matter of trial and error, due to the fact that the probabilities do not need to be Gaussian distributed. For this test, a multiplication factor of 2.1 gave the best results for selecting the wall points within the point cloud scene, the first scene from testing the ANN in section 3.7. Figure 39 shows the results of the conducted test.

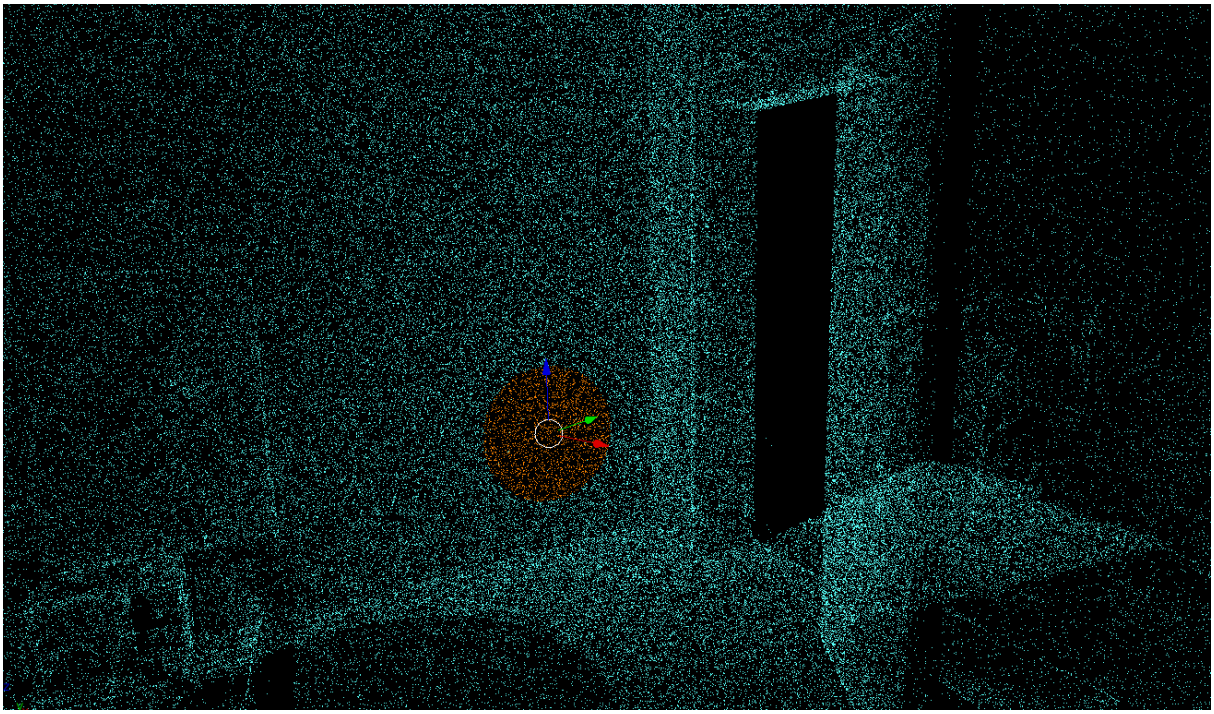


Figure 38: Selected seed region (orange) on wall

Table 7: Mean and standard deviation for all class probabilities in selected seed region (0=beam, 1=ceiling, 2=chair, 3=column, 4=door, 5=floor, 6=occlusion, 7=table, 8 = wall, 9=window)

	Class probabilities									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
mean	0,011	0,001	0,204	0,009	0,149	0,002	0,179	0,051	0,292	0,104
Std.	0,021	0,007	0,088	0,012	0,110	0,011	0,039	0,018	0,113	0,092

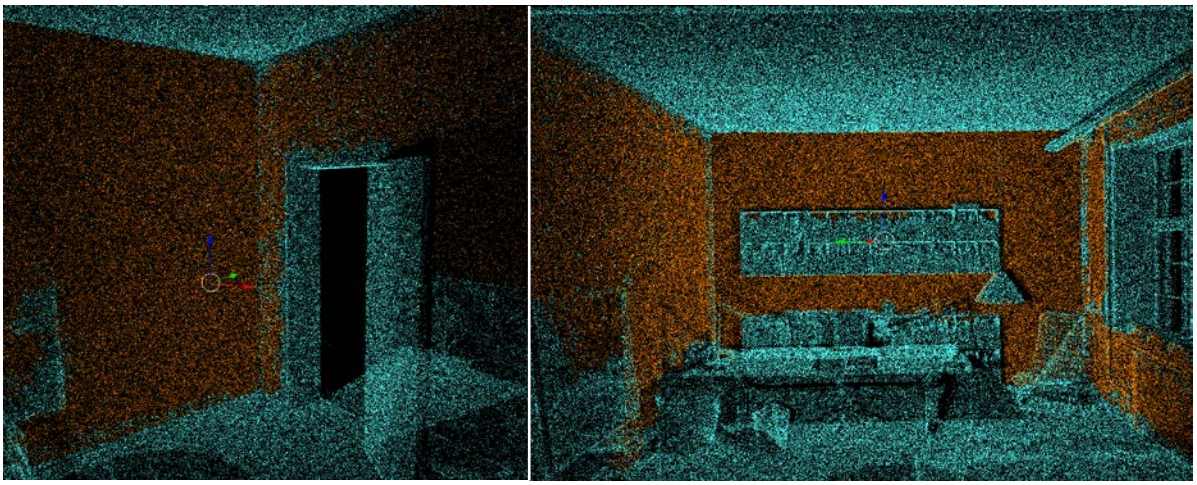


Figure 39: Example of region growing algorithm for quickly selecting all walls with a single seed region

Table 8: Example selection from region growing algorithm output (0=beam, 1=ceiling, 2=chair, 3=column, 4=door, 5=floor, 6=occlusion, 7=table, 8 = wall, 9=window)

Coordinates			**Class probabilities**										**Class**
x	**y**	**z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Pred.**
…													
-2,65	17,62	30,12	0,04	0,00	0,12	0,01	0,43	0,01	0,11	0,09	0,09	0,11	4
-2,65	17,61	30,12	0,02	0,04	0,55	0,00	0,03	0,09	0,08	0,15	0,01	0,03	2
-2,65	17,60	30,13	0,05	0,00	0,11	0,02	0,38	0,00	0,12	0,09	0,14	0,09	4
-2,66	17,58	30,14	0,00	0,00	0,31	0,00	0,09	0,00	0,18	0,05	0,23	0,13	2
-2,65	17,62	30,13	0,09	0,00	0,05	0,03	0,51	0,01	0,09	0,08	0,10	0,04	4
-2,65	17,61	30,12	0,01	0,00	0,09	0,01	0,25	0,00	0,14	0,06	0,05	0,38	9
…													

Table 9: Points erroneously left out after region growing algorithm (which should have been included) (0=beam, 1=ceiling, 2=chair, 3=column, 4=door, 5=floor, 6=occlusion, 7=table, 8 = wall, 9=window)

Coordinates			**Class probabilities**										**Class**
x	**y**	**z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Pred.**
…													
1,31	16,32	29,67	0,00	0,00	0,10	0,00	0,08	0,00	0,11	0,03	0,04	0,64	9
1,30	16,32	29,69	0,03	0,00	0,07	0,01	0,38	0,00	0,11	0,07	0,04	0,28	4
1,30	16,33	29,69	0,03	0,00	0,04	0,01	0,54	0,00	0,09	0,07	0,04	0,16	4
1,31	16,32	29,67	0,01	0,00	0,06	0,01	0,27	0,00	0,12	0,06	0,04	0,43	9
1,30	16,33	29,69	0,04	0,00	0,07	0,02	0,41	0,00	0,11	0,07	0,05	0,23	4
1,31	16,32	29,67	0,01	0,00	0,15	0,00	0,19	0,00	0,11	0,07	0,03	0,43	9
…													

Further analysis, and also Figure 39 show that there are still points left outside the seed region, which should be included. Table 9 shows that these points contain probability thresholds of the wall class (column 8) that are below the lower bound of the calculated threshold (0,0547). This can be attributed to the performance of the ANN, and thus could also be solved by improving the precision of the ANN, or by using a wider variety of thresholds allowing other class probability to be considered in the region growing algorithm. However, this could also in turn introduce more problems than it initially would solve.

This is mainly the issue when certain ‘jumps’ are made in the nearest neighbor search. These jumps are unexpected nearest neighbors (when e.g. the amount of nearest neighbors would be high) that should initially not be included, from which new regions within the threshold are constructed. This would further impede a ‘clean’ selection of points. Additionally, it could be possible that some points are skipped due to the use of FLANN as it approximates (although still accurately) nearest neighbors instead of using true distances.

3.10.2. Test case chair – extraordinary misclassified points

Another test case is performed on a chair that has been entirely misclassified by the ANN from section 3.7 represented in Figure 34.3. The same chair and the selected seed patch for region growing is depicted in Figure 40, including the predicted labels of the selected seed region in Table 10 and the mean and standard deviation of every predicted class probability in Table 11. It becomes clear that the selected region contains no predicted chair points, and therefore region-growing becomes problematic. Due to the fact that the underlying probabilities within the selected region for 'chair' (as seen in Table 11) are considerably low, the region growing algorithm will select all neighboring points (with k-NN = 11) that are above this probability when there is no upper bound threshold. For cases that are initially well classified by the ANN, omitting the upper bound threshold will make sure that all categories that have very high and specific class probabilities will be included into the region.

In any case, with a k-NN = 11 and threshold of 1.1 std., it seems that at least a small region can be grown within the sub-optimal chair category probabilities, as visualized in Figure 41. Therefore, when a poorly selection has been made by the user, and/or when the selection has been poorly classified by the ANN, the region-growing algorithm will also output poor results.

Table 10: Predicted labels of selected seed patch

Predicted label of selection	Amount
Floor	170
Table	19
Occlusion	6
Ceiling	2
Total	197

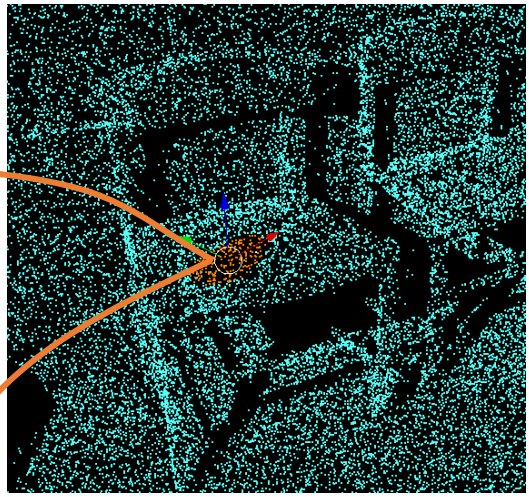


Figure 40: Seed selection for region growing on a misclassified chair

Table 11: Mean and std. of soft probabilities for predicted seed patch

	Class	Mean probability	Std. probability
1	Beam	0.068	0.017
2	Ceiling	0.186	0.061
3	Chair	0.044	0.027
4	Column	0.001	0.001
5	Door	0.030	0.031
6	Floor	0.309	0.064
7	Occlusion	0.139	0.041
8	Table	0.206	0.027
9	Wall	0.006	0.004
10	Window	0.010	0.012

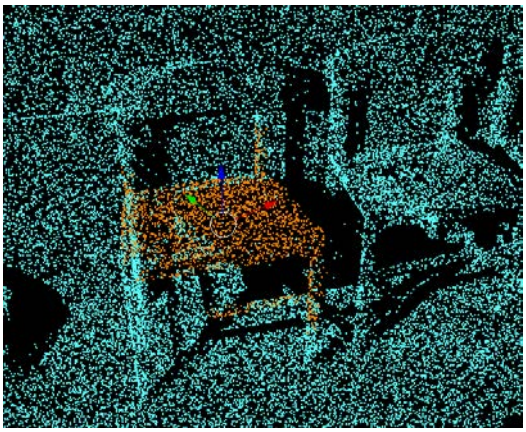


Figure 41: Performed region growing on chair

To analyze why the chairs as seen in Figure 40 and Figure 41 have been poorly classified by the ANN, Figure 42 presents a comparison between the (1) FPFH values of the chairs of the trained ANN model, and (2) the FPFH values belonging to the chairs within the test scene. The explainable reason for the difference in FPFH values is the proximity of the wall behind the chairs in the test scene in Figure 42.2, which results in different angular variations and thus skewed FPFH calculations.

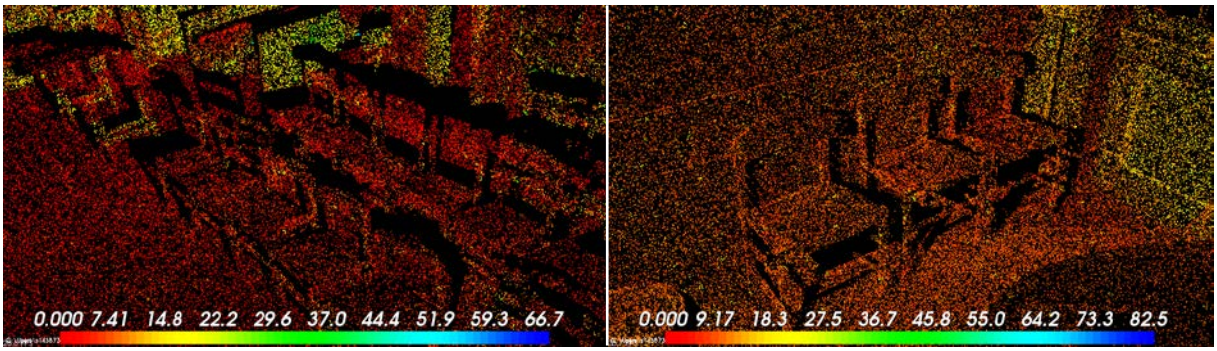


Figure 42: FPFH color representations (1) FPFH data used for training the ANN vs (2) FPFH data used for testing

However, FPFH contains three separate histograms and the colors in Figure 42 shows that there is a difference in values, and although the wall behind the chairs in Figure 42.2 could be a plausible explanation, it is still not specific on why they are different. To analyze the difference in class distinguishability for FPFH values between the training and test data, two t-SNE plots are depicted in Figure 43. However for the chair class, this comparison does not present a clear difference as chair points are scattered across for both training and test data.

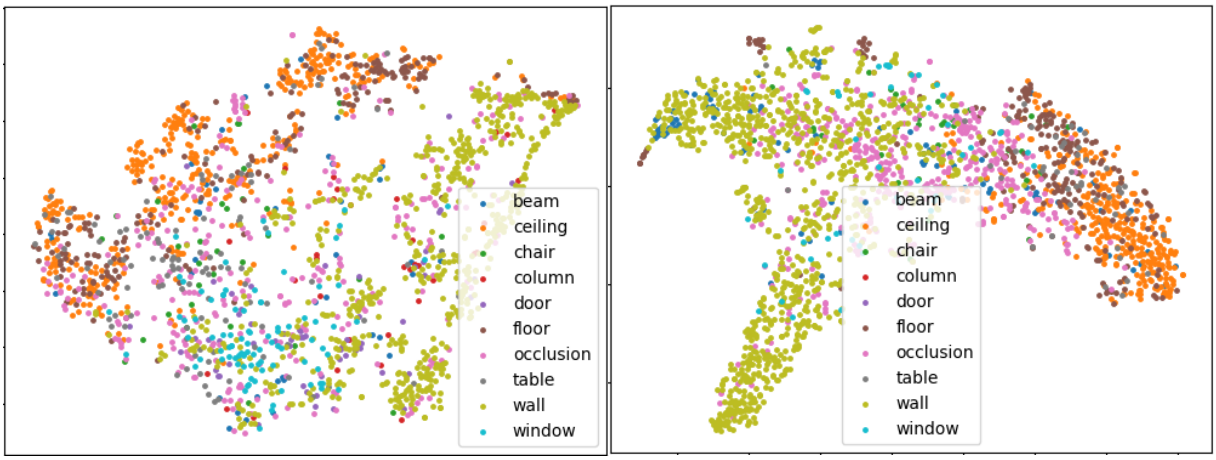


Figure 43: Comparison t-SNE visualization of class distinguishability of (1) training data and (2) test data

However, Figure 44 gives a more clearly comparison in FPFH values of the manually labeled chairs in the training and testing dataset, and it becomes apparent that the calculated FPFH values for testing are indeed different than the values than the ANN has learned, explaining its misclassification.

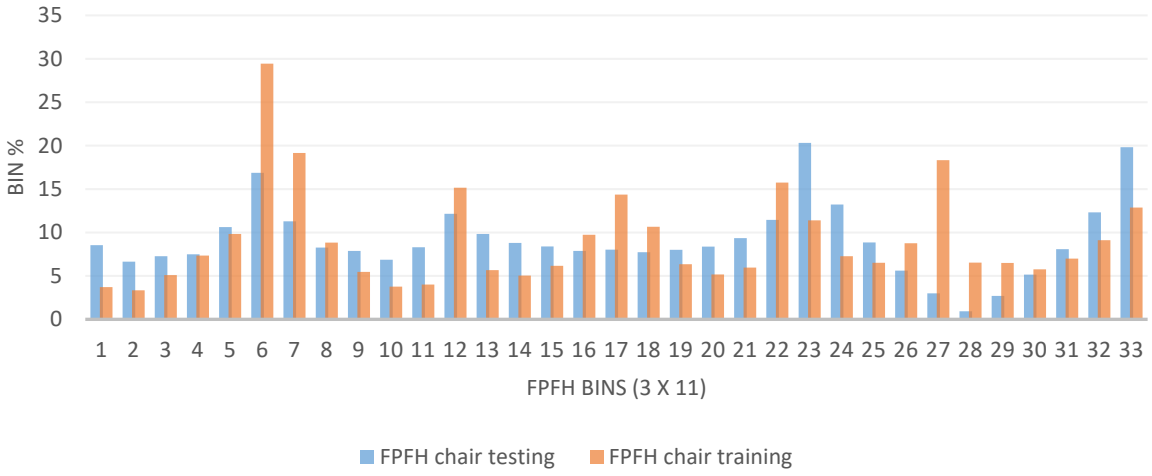


Figure 44: Comparison of average FPFH values of a single chair in the training dataset (not near any walls) and a single chair in the testing dataset (standing against a wall).

3.10.3. Test case window – sensitivity of k-NN parameter

Another test case is done to analyze the influence of the nearest neighbor (k-NN) parameter when region growing. Figure 45.1 shows the initial seed selection on a window frame (the patch of points within the red circle) that will be used for selecting the entire window through the region-growing algorithm. Figure 45 shows a comparison of the k-NN parameter, where $k = 5, 7\ or\ 11$. The inside of the window head contains high probabilities for the table class and the inside of the window sill contains high probabilities for the chair class, and are therefore classified as such. The underlying soft probabilities of the window class of these points is substantially low, and therefore are not within the region after region-growing. Increasing the std. factor results in a selection of points through 'jumps' that are far from being window points, and it is therefore undesirable to increase this factor.

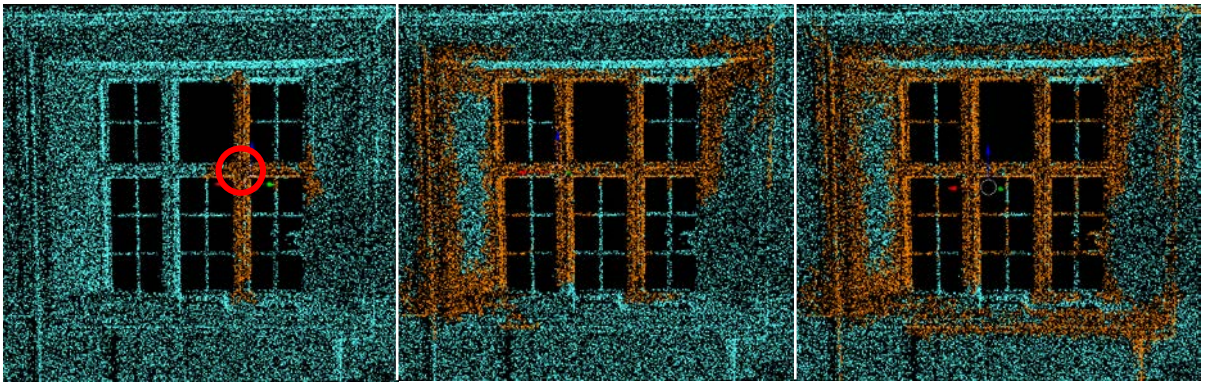


Figure 45: Comparison of sensitivity of k-NN nearest neighbors value: (1) k-NN = 5; (2) k-NN = 7; (3) k-NN = 11

3.10.4. Test case table – issue of missing points

Another limitation of the region-growing algorithm becomes apparent when objects contain missing points, as can be seen in Figure 46. An initial seed selection has been made for selecting the entire table, for which the region-growing algorithm has been used. The resulting region seems to have skipped points which are clearly table points, but has included some small objects that are on top of the table (such as a few cups or glasses). However, the legs of the table (marked by the red circle in Figure 46.2) could have not been included into the region, as there is a substantial amount of missing points which is caused during the point cloud acquisition process. However, these points have been further analyzed from which it becomes clear that they are predicted as the door class, and from the analysis of their underlying soft probabilities it is observed that the probabilities of being a table are substantially low, that they would not have been included into the region even when the issue of missing points was non-existent.

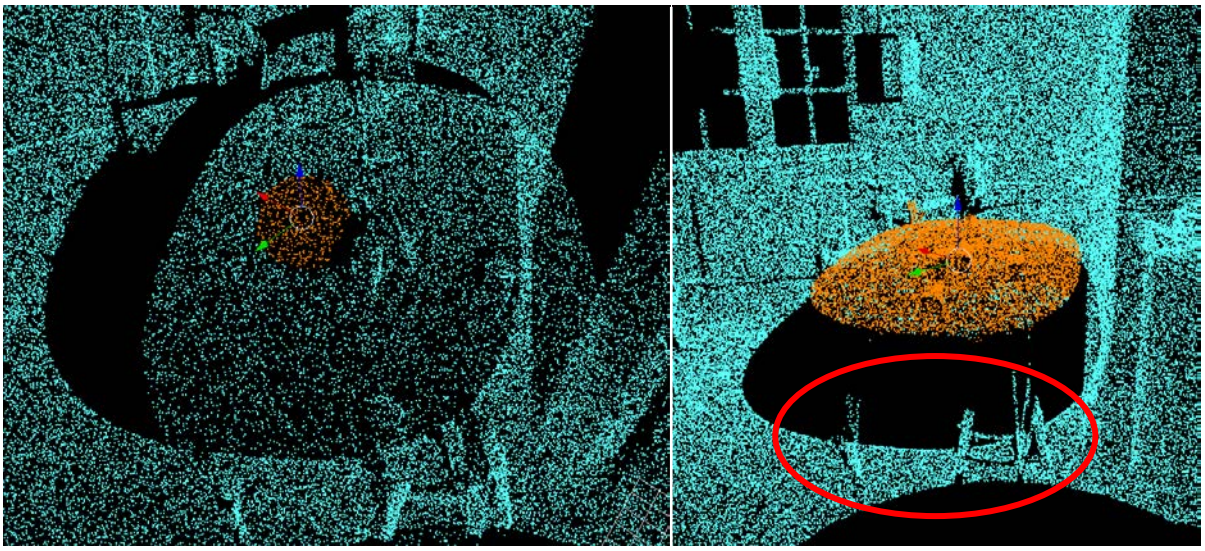


Figure 46: The issue of missing points through point cloud acquisition

3.10.5. Test case occlusion – influence of proper seed selection

The importance of proper initial seed selection became already apparent to a certain degree in section 3.10.2. A proper seed selection is vital to good results from the region-growing algorithm, as it uses the mean and std. of the underlying soft class probabilities of the selected group of points. A clear example is given in Figure 47.1 where an initial seed selection is made to select the kitchen counter as occlusion (which is the category it is manually classified as), and the resulting region from region-growing in Figure 47.2. The initial seed selection contains points that contain decent values for the probability of being classified as occlusion, and therefore resulting in acceptable mean and std. values from which a new region can be grown.

The contrary is true for the example given in Figure 47.3, where a seed selection has been made for region growing to select the kitchen counter as occlusion. The selected seed region does not contain any points classified as occlusion, and includes underlying probabilities for

occlusion that are very small. This will result in a low threshold for region-rowing on the underlying occlusion probabilities for neighboring points, and thus, a poor output from the region-growing algorithm, as seen in Figure 47.4. The kitchen cabinets draw substantial resemblance with the door and wall class and have been classified as such by the ANN, which has been observed by analyzing sample patches belonging to these cabinets.

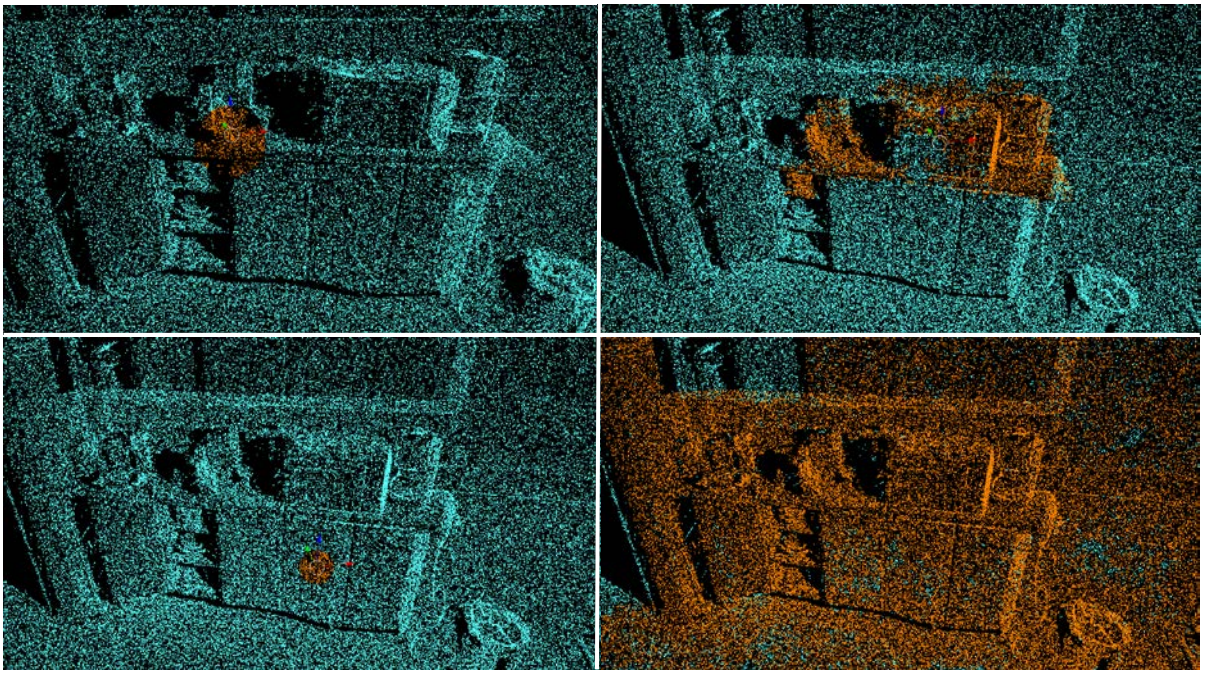


Figure 47: different region-growing outputs by different seed point selection

3.10.6. Proposed extension to region-growing

An extension to the region growing algorithm, would be the introduction of de-selecting points from the output of the region growing algorithm proposed in section 3.9. Instead of region growing, a selection of points could be made within the grown region output, and inserted into an algorithm that removes these points, including their nearest neighbors, within a certain threshold (either the same, or a different threshold). To give an example, consider the region growing result from before, as visualized in Figure 39. The scene contains a painting on a chair leaning against the wall, and therefore the surface of this object has a certain angle relative to the wall. Therefore, the point normal direction is slightly different, and could be exploited by selecting a group of points on this object, retrieve point normal information, and 'region-shrink' until all neighboring points are deselected that contain a certain normal vector direction. To give a better understanding, the process is visualized in Figure 48. However, take note that the above is conceptual, and is not developed in this thesis.

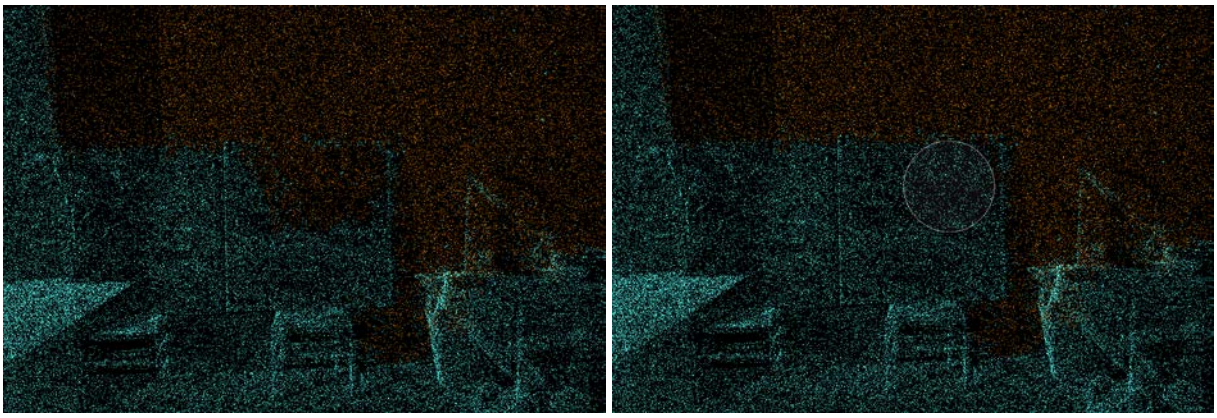


Figure 48: The conceptual process of region de-growing or 'region-shrinking'

3.11. Improving the base model - further research

From the results and discussion of region-rowing in section 3.10, it becomes clear that the quality of the output is heavily dependent on the quality of the underlying base model (the ANN), and dependent on the user input on the parameters of the region-growing algorithm. From the analyses made in section 3.7, 3.8 and 3.9, it becomes clear that the ANN relies heavily on the FPFH values for classification. However as already mentioned in section 3.8, while FPFH stores information about the angular variations of neighboring points within a user-specified radius, it does not store the geometrical location of where these angular variations are present. This issue is elaborated on in the next section.

3.11.1. Inclusion of surrounding geometry

The SHOT feature has already been briefly mentioned in section 2.2.3, but after development and testing, the case of including surrounding geometry into a feature as additional semantic information for point cloud data has become very significant. The SHOT descriptor (abbreviation for Signature of histograms of orientations) has been proposed by Salti et al. (2014), who make a distinction between the nature of histogram-based features in their research.

Histogram based features are based on user-specified point neighborhoods, also known as local reference frames. A local reference frame refers to the point neighborhood and their respective values (such as location or normal vector orientations) that is based on a local coordinate system tailored to that of the respective feature. Salti et al. (2014) make a distinction between signatures and histograms as seen in Figure 49.1. They define a histogram as a range of bins in which trait values are stored according to a value count, and therefore losing the coordinate values that belong to the neighboring points. On the other hand, signatures keep their local coordinate values to a certain extend by dividing coordinate ranges into bins, and storing the trait values in these bins accordingly. For a descriptor that can describe local surrounding geometry, the histograms are therefore three dimensional. The structure of SHOT is visualized in Figure 49.2.

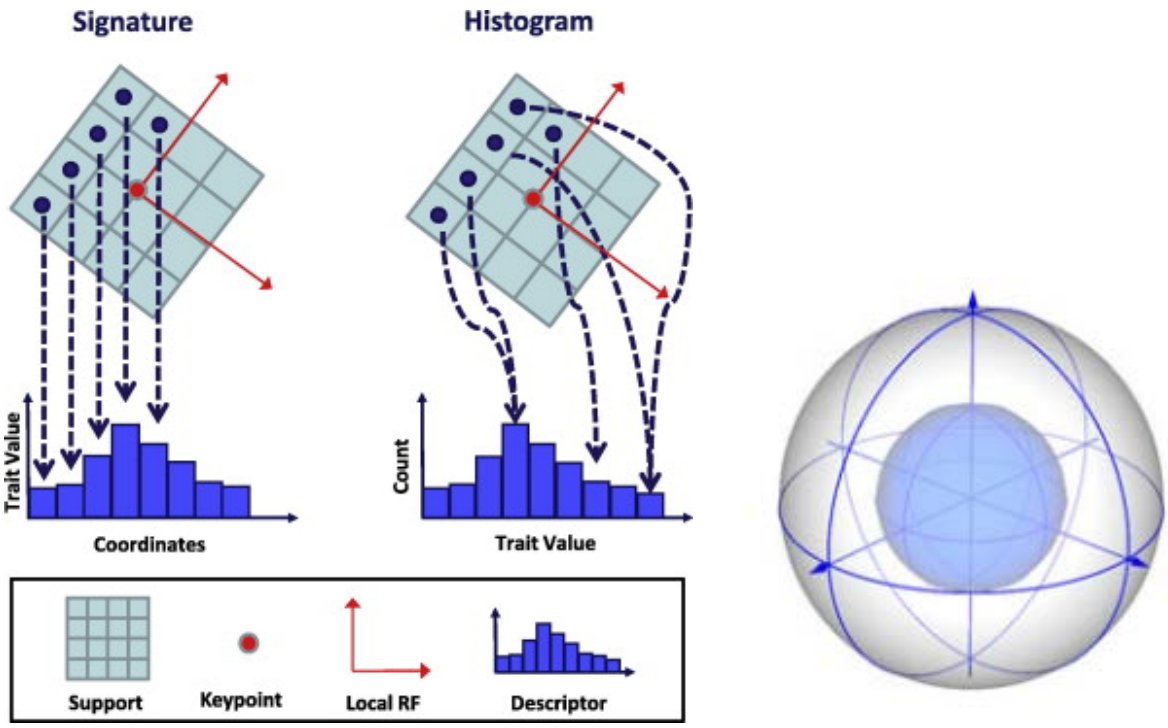


Figure 49: (1) Distinction between signatures and histograms as feature descriptor and (2) SHOT signature structure

However, the current SHOT feature algorithm relies on volumetric (mesh) information, and therefore could not directly be used for point clouds. In any case, the addition of local geometric information to every point within a point cloud dataset, could potentially result in a better base model that can perform classifications with increased quality. In turn, this would result in better region-growing results, and requiring less trial-and-error for selecting parameter values by the user.

3.11.2. Ensemble learning for increasing base model performance

Ensemble learning is a method that is occasionally used in related literature which incorporates multiple learning algorithms to increase the quality of the output, instead of relying on a single learning algorithm. The ensemble learning method that potentially could increase the base model proposed in this research would be stacking (constructing a meta-learner).

There is a wide variety of machine learning algorithms for classification, where some types contain disadvantages which are otherwise the advantages of other types of algorithms and vice versa. In other words, some models may perform well for solving part A of the classification problem, but perform poor on part B of the classification problem. However, another type of machine learning algorithm might perform well on part B put poorly on part A. In these cases, part A and B can either refer to being certain parts of a dataset, or either a chain in the sequence of the meta-learner that is constructed. Both models can be combined to balance out their disadvantages. An example regarding the research in this thesis for

incorporating stacking could be to create separate learners for separate tasks; a learner that can properly distinguish ceilings, tables and floors, and a particular learner for distinguishing walls, doors and windows. Both learners are combined through the meta-learner, by using another machine algorithm to combine them. Therefore, using different machine learning algorithms for different category classifications.

The idea is to increase the distinguishability of data, and for the sake of visualization to increase the space between class categories on the t-SNE scatterplots such as exemplified in Figure 16 in section 2.4.2 and Figure 37 in section 3.8. This is what Li et al. (2018) and Qi et al. (2017) do to increase the accuracy of their machine learning models. Li et al. (2018) use a small combined MLP and CNN on local point neighborhoods which enriches the feature vector with spatially-local information. Afterwards, this output it inserted into a regular CNN for further learning. In other words, the local data is enriched first to make categories more distinguishable and therefore results in better classification performance.

3.12. Discussion

The idea of using context for the multi-classification of point cloud datasets is still on-going in research overall, and in the case of this research, it has not been able to exploit context extensively for the use within the proposed neural network as machine learning model. However, the part of user interaction that forms an important extension to the original idea of the research presents the user with a tool to exploit the output of the proposed neural network for manual classification of point cloud datasets, and to incorporate some manual context and thus manually enriching the dataset with semantic information to a certain extent.

Current literature focusses heavily on increasing the performance of machine learning models, and in particular, deep neural networks used for object recognition. However, the current solutions that are proposed aim to increase the performance by incorporating very complex data pre-processing, data transformations and neural network structures, that even researchers acknowledge that they do not understand what their proposed methods exactly do (see: Li et al. (2018)). Besides complexity, neural networks are still being regarded as a black-box model that adds to the difficulty for increasing a model's performance. Additionally, most methods in literature still seem to be tailored and designed with the ModelNet40 leaderboard in consideration, and end where the model gives a good accuracy rate as output.

This research steps away entirely from competition to propose a method that pre-processes point cloud data through a neural network, and aims to go beyond where other research ends. The machine learning model creates an output that assists the user for selecting and classifying the point cloud data according to their goals. Therefore, the user is the pivot in this process for which the machine learning model only provides a sound foundation, therefore adding a significant extend of user interaction.

Regardless, an improvement of the base model would result in better classification results, including better output results that can be exploited by the user, and allowing him or her to

select and classify more precisely, but retaining their freedom for interpretation and goals in mind. Even though for the selection through region-growing based on the soft-probability outputs from the ANN, which is unique in this way, the proposed method aims to act as a cornerstone for more research and development on classifying point cloud data with machine learning models and user interaction. Especially given the results and limitations of the current proposed region-growing algorithm for selection, there is definitely room for improvement.

To address a more holistic matter, to set-up and run a deep neural network is a relatively easy task, especially with APIs such as those used in this research, the model set-up of a simple deep neural network takes only around 25 lines of code(!) in python, although this is without the preprocessing of the dataset. As Chollet (2017) also mentions, the idea behind deep learning is not difficult, it just requires sufficient quantitative data to be able to learn. Indeed, the premise of deep learning is that it *just works* and that its' applicability seems nearly endless. However, this is definitely not the case as any task that requires reasoning (as humans do, e.g. programming, long-term planning and generalization through abstraction) is currently out of reach for deep learning. This is because deep learning essentially comes down to being a simple chain of geometric transformations of input data until the desired output is achieved, no matter how many layers the model contains or how many learners are stacked. This is an inconvenient limitation of deep learning, as the ANN proposed in this research cannot be used for tasks that are slightly different than which it was designed for.

Additionally while deep learning looks simple from the outside, breaking away from the standard 'cookbook' models that are found in the books and internet, and optimizing it for more difficult problems, such as presented in this research or those presented in current State-of-the-Art literature, is a tedious and difficult task. Especially given the fact that theoretically, there is a wide variety of models that are able to solve the same problem and even give the same output, it is infeasible to test and compare all of these models. To conclude, it is not hard to set-up a deep neural network that works, however optimizing it to do certain tasks that require more than being able to perform adequately is a field of research on its own.

4. Conclusion

The following chapter will give a critical evaluation of the research and development conducted in this thesis, and will do so by reflecting holistically on the research questions defined in section 1.3, addressing the societal relevance of this research within the AEC industry, and finally give recommendations for further research and development.

4.1. Scientific relevance

This research proposes a new method for the classification of point cloud datasets. This is done by using machine learning as a fundamental sub-stage for classification, and providing region-growing as an extension to incorporate user interaction for point cloud data classification specifically for the AEC industry, which keeps the freedom and flexibility that emphasizes on the user's interpretation and goals to do so. From an extensive literature study, it became clear that the use of point cloud data in the AEC industry is increasing, as the nowadays widespread application of BIM gives an impulse to new ideas and concepts to automate the construction process. It seems that point cloud data can provide a significant added value and extension to current BIM applications and management practices in the AEC industry, with notable examples such as: the Scan-to-BIM process, progress monitoring by comparing the as-built and as-planned situation of a building under construction, discovering critical structural deviations and keeping a data repository for facility management purposes.

However, conventional ways of transforming point cloud datasets into volumetric models for these previously described purposes is a tedious process and can often be omitted by recognizing construction objects and elements and classifying them accordingly. This is a contemporary issue that is studied in the field of computer vision and object recognition, often incorporating machine learning and deep learning to automate the process. However, current proposed methods are very limited with regard to the irregularity and ambiguity of construction elements, struggle with the incorporation of context and semantic relationships between objects, and are often static and inflexible to be truly useful for users without a background in Computer Science or Mathematics.

The objective of this research was to develop a hybrid method for segmenting and classifying point cloud datasets for the AEC industry, that both incorporates machine learning and user interaction to exploit their advantages and to balance out their disadvantages. This has been done by using a deep neural network for proposing an initial distribution of category classifications for every point within the point cloud dataset. These distributions of classifications act as an additional set of thresholds, for which a region-growing algorithm has been designed that allows the user to exploit this information. The region-growing algorithm therefore functions as a smart selection tool for quickly segmenting and classifying point cloud datasets.

Even though the general idea is supposed to break away from current research in the field of object recognition, act as an initiator within the AEC industry, and proves to have a significant amount of potential, there are quite some limitations. The first notable limitation is that of the performance of the underlying machine learning model. The application of deep learning for object recognition and appropriate classification seems to be a very difficult task, even when used as an underlay that acts as an assistant for easier manual selection and classification. The original intention for using a multi-layer perceptron for the object recognition and classification task was that it would be able to incorporate some sort of contextual information, i.e. unfold semantic relationships between construction elements and other objects. However, this was not the case. Additionally, the issue is that the performance of the proposed neural network directly influences the quality of region-growing. To address this issue, there are near unlimited ways for increasing the performance of the base model, most notable being the use of stacking multiple neural network types. However, as also became evident in literature, doing so adds tremendous complexity to these models, and results in poor understanding of how these models work and how they can be further improved or tailored to tasks that are slightly different.

The second limitation is regarded to practicality, and became evident in the test cases performed for region-growing. Initial seed selection is vital to the quality of the output by the region-growing algorithm and the optimization of parameters for setting proper thresholds is a matter of trial-and-error. Furthermore, while Blender is a convenient tool for creating research prototypes as it embraces the possibility of scripting, it is inconvenient for handling large point cloud datasets and for visualizing their additional attributes (such as color information or color maps for attribute values). This became already apparent near the start of the research due to how Blender essentially handles *vertices* (i.e. data points), but especially near the end during the test cases for region-growing, as indices of data points shift when segments and classifications are made. These test cases only proved the feasibility of the concept, but there is still a lot of work to be done for the proposed method to be truly practical and to directly implement it in open-source software such as Blender.

4.2. Societal relevance

To extend on the topic of practicality, there are many companies that specialize themselves in a wide variety of BIM related software, ranging from modelling software to specialized management software, all focused with the purpose of offering solutions for improving the building process, from idea till completion and usage. Therefore, the relationship with Computer Science and Information Systems draws closer, and the field of construction management is shifting heavily into the digitalized world.

This research also aims to close the gap between the field of Construction Management and Computer Science, and succeeds to do so by combining automation from Computer Science and the practicality that is necessary from a Construction Management point of view, into a hybrid method that is supposed to be understandable by construction management practitioners. More specifically, many of the use cases for point cloud datasets from literature

and their proposed solutions often have one goal in common: act as a forerunner and initiator to give an impulse in further (commercial) development to facilitate the building process. Likewise, this research aims to offer the same.

4.3. Recommendations

Based off the conclusions in section 4.1, there are several recommendations that can be formed for improving the method proposed in this research. The first and most straightforward one is improving the base model, the deep neural network, by adding additional feature descriptors to increase class distinguishability and to add some form of spatial or geometrical context, e.g. by adding SHOT or 3DSC and run additional tests to evaluate their descriptive power. Furthermore, additional tests should be done by using the same point cloud dataset (Byg72 from the DURAARK online dataset repository) with current State-of-the-Art methods in literature, such as PointNet++ by Qi et al. (2017) and PointCNN by Li et al. (2018), from which comparisons can be made to evaluate the true performance of the proposed method in this research. These State-of-the-Art methods might potentially serve as a better base model for increased region-growing performance and user interaction. Additionally, an interesting concept would be to incorporate machine learning model stacking, e.g. develop a model that performs well on discerning tables, floors and ceilings and a model that performs well on discerning wall, doors and windows, and combining these models for better accuracy.

Regarding the region-growing algorithm and workflow proposed in this research, it is recommended that more testing should be done for the proposed region-growing algorithm, as it is still a trial-and-error approach instead of being straightforward, and the region-growing algorithm should be extended so that it can overcome he issues that are currently at hand when segmenting and classifying point clusters in Blender, i.e. the point cloud index list shifts when clusters are taken out. Furthermore, the region-growing algorithm could be extended to incorporate region de-growing for selecting points more precisely and Blender should be extended to make the software more suitable for working with point cloud datasets. Especially, incorporating the possibility to store additional information to datasets beside the xyz-values, e.g. normal vector information, category, color, features, etc., which Blender currently not supports and therefore being a very inconvenient limitation.

Regarding broader applications for which this study acts as a foundation, it is recommended to expand research and development of methods towards automated processes for point cloud datasets in the AEC industry, which could greatly benefit the sector, such as Scan-to-BIM, and comparing the as-built and as-designed situation of a building for progress monitoring and checking for structural deficiencies.

Concerning research in machine learning and deep learning, hybrid methods are severely under represented. For practicality and the extendibility of these concepts to other industries, machine learning model implications potentially benefit by including a significant amount of user interaction, both during and after model training. To elaborate on this issue, there exists a necessity to get rid of the ‘magic black box’ aura around deep learning and deep neural

networks in particular. Although this is easier said than done, it could greatly benefit many fields, including object recognition, for optimizing and proposing new methods that can incorporate and exploit contextual or semantic information.

References

Anand, A., Koppula, H. S., Joachims, T., & Saxena, A. (2013). Contextually guided semantic labeling and search for three-dimensional point clouds. *The International Journal of Robotics Research*, *32*(1), 19–34. https://doi.org/10.1177/0278364912461538

Armeni, I., Sax, S., Zamir, A. R., & Savarese, S. (2017). Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv:1702.01105 [Cs]*. Retrieved from http://arxiv.org/abs/1702.01105

Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, *13*(2), 111–122. https://doi.org/10.1016/0031-3203(81)90009-1

Bassier, M., Vergauwen, M., & Van Genechten, B. (2016). *Automated Semantic Labelling of 3D Vector Models for Scan-to-BIM*. https://doi.org/10.5176/2301-394X_ACE16.83

Belsky, M., Sacks, R., & Brilakis, I. (2016). Semantic Enrichment for Building Information Modeling. *Computer-Aided Civil and Infrastructure Engineering*, *31*(4), 261–274. https://doi.org/10.1111/mice.12128

Bosché, F., Ahmed, M., Turkan, Y., Haas, C. T., & Haas, R. (2015). The value of integrating Scan-to-BIM and Scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components. *Automation in Construction*, *49*(Part B), 201–213. https://doi.org/10.1016/j.autcon.2014.05.014

Del Giudice, M., & Osello, A. (2013). BIM for cultural heritage. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *40*, 225–9.

Dong, P., & Guo, H. (2012). A framework for automated assessment of post-earthquake building damage using geospatial data. *International Journal of Remote Sensing*, *33*(1), 81–100. https://doi.org/10.1080/01431161.2011.582188

Ebrahim, M. (2014). 3D LASER SCANNERS: HISTORY, APPLICATIONS, AND FUTURE. https://doi.org/10.13140/2.1.3331.3284

Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., & Posner, I. (2016). Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. *ArXiv:1609.06666 [Cs]*. Retrieved from http://arxiv.org/abs/1609.06666

Fischler, M. A., & Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, *24*(6), 381–395. https://doi.org/10.1145/358669.358692

Gao, T., Akinci, B., Ergan, S., & Garrett, J. (2015). An approach to combine progressively captured point clouds for BIM update. *Advanced Engineering Informatics*, *29*(4), 1001–1012. https://doi.org/10.1016/j.aei.2015.08.005

Golparvar-Fard, M., Peña-Mora, F., & Savarese, S. (2009). D4AR–a 4-dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication. *Journal of Information Technology in Construction*, *14*(13), 129–153.

Golparvar-Fard, M., Peña-Mora, F., & Savarese, S. (2015). Automated Progress Monitoring Using Unordered Daily Construction Photographs and IFC-Based Building Information Models. *Journal of Computing in Civil Engineering*, *29*(1), 04014025. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000205

Grilli, E., Menna, F., & Remondino, F. (2017). A review of point clouds segmentation and classification algorithms (Vol. XLII-2/W3). https://doi.org/10.5194/isprs-archives-XLII-2-W3-339-2017

Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., & Pollefeys, M. (2017). Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark. *ArXiv:1704.03847 [Cs]*. Retrieved from http://arxiv.org/abs/1704.03847

Hackel, T., Wegner, J. D., & Schindler, K. (2016). Fast semantic segmentation of 3D point clouds with strongly varying density. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, *3*(3).

Han, K. K., Cline, D., & Golparvar-Fard, M. (2015). Formalized knowledge of construction sequencing for visual monitoring of work-in-progress via incomplete point clouds and low-LoD 4D BIMs. *Advanced Engineering Informatics*, *29*(4), 889–901. https://doi.org/10.1016/j.aei.2015.10.006

Holz, D., Holzer, S., Rusu, R. B., & Behnke, S. (2011). Real-time plane segmentation using RGB-D cameras. In *Robot Soccer World Cup* (pp. 306–317). Springer.

Huang, J., & You, S. (2016). Point cloud labeling using 3D Convolutional Neural Network. In *2016 23rd International Conference on Pattern Recognition (ICPR)* (pp. 2670–2675). https://doi.org/10.1109/ICPR.2016.7900038

Jakob Beetz, Thomas Krijnen, & Raoul Wessel. (2015). *Point Cloud schema extension for the IFC model.pdf* (Project deliverable). DURAARK.

Kim, C., Son, H., & Kim, C. (2013). Automated construction progress measurement using a 4D building information model and 3D data. *Automation in Construction*, *31*(Supplement C), 75–82. https://doi.org/10.1016/j.autcon.2012.11.041

Kim, P., Chen, J., & Cho, Y. K. (2017). Robotic sensing and object recognition from thermal-mapped point clouds. *International Journal of Intelligent Robotics and Applications*, *1*(3), 243–254. https://doi.org/10.1007/s41315-017-0023-9

Krijnen, T., & Beetz, J. (2017). An IFC schema extension and binary serialization format to efficiently integrate point cloud data into building models. *Advanced Engineering Informatics*, *33*, 473–490. https://doi.org/10.1016/j.aei.2017.03.008

Li, Y., Bu, R., Sun, M., & Chen, B. (2018). PointCNN. *ArXiv:1801.07791 [Cs]*. Retrieved from http://arxiv.org/abs/1801.07791

Li, Y., Pirk, S., Su, H., Qi, C. R., & Guibas, L. J. (2016). FPNN: Field Probing Neural Networks for 3D Data. *ArXiv:1605.06240 [Cs]*. Retrieved from http://arxiv.org/abs/1605.06240

Macher, H., Landes, T., & Grussenmeyer, P. (2017). *From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings* (Vol. 7). https://doi.org/10.3390/app7101030

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Presented at the Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, The Regents of the University of California. Retrieved from https://projecteuclid.org/euclid.bsmsp/1200512992

Maturana, D., & Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 922–928). https://doi.org/10.1109/IROS.2015.7353481

Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, *2*(331–340), 10.

Navon, R., & Shpatnitsky, Y. (2005). Field Experiments in Automated Monitoring of Road Construction. *Journal of Construction Engineering and Management-Asce - J CONSTR ENG MANAGE-ASCE*, *131*. https://doi.org/10.1061/(ASCE)0733-9364(2005)131:4(487)

Pang, G., & Neumann, U. (2016). 3D point cloud object detection with multi-view convolutional neural network. In *Pattern Recognition (ICPR), 2016 23rd International Conference on* (pp. 585–590). IEEE.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2016). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *ArXiv:1612.00593 [Cs]*. Retrieved from http://arxiv.org/abs/1612.00593

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *ArXiv:1706.02413 [Cs]*. Retrieved from http://arxiv.org/abs/1706.02413

Salti, S., Tombari, F., & Di Stefano, L. (2014). SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, *125*, 251–264.

Shapovalov, R., Velizhev, A., & Barinova, O. (2010). Non-associative markov networks for 3D point cloud classification. Presented at the ISPRS, France. Retrieved from http://www.isprs.org/proceedings/XXXVIII/part3/a/pdf/103_XXXVIII-part3A.pdf

Soares, M. B., Barros, P., Parisi, G. I., & Wermter, S. (2015). Learning objects from rgb-d sensors using point cloud-based neural networks. In *23th European Symposium on Artifical Neural Networks, Computational Intelligence and Machine Learning. Bruges* (pp. 439–444).

Tang, P., Huber, D., Akinci, B., Lipman, R., & Lytle, A. (2010). Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction*, *19*, 829–843. https://doi.org/10.1016/j.autcon.2010.06.007

Turkan, Y., Bosche, F., Haas, C. T., & Haas, R. (2012). Automated progress tracking using 4D schedule and 3D sensing technologies. *Automation in Construction*, *22*(Supplement C), 414–421. https://doi.org/10.1016/j.autcon.2011.10.003

Vo, A.-V., Truong-Hong, L., Laefer, D. F., & Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *104*(Supplement C), 88–100. https://doi.org/10.1016/j.isprsjprs.2015.01.011

Volk, R., Stengel, J., & Schultmann, F. (2014). Building Information Modeling (BIM) for existing buildings — Literature review and future needs. *Automation in Construction*, *38*(Supplement C), 109–127. https://doi.org/10.1016/j.autcon.2013.10.023

Wang, J., Sun, W., Shou, W., Wang, X., Wu, C., Chong, H.-Y., … Sun, C. (2015). Integrating BIM and LiDAR for Real-Time Construction Quality Control. *Journal of Intelligent & Robotic Systems*, *79*(3–4), 417–432. https://doi.org/10.1007/s10846-014-0116-8

Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, *105*, 286–304. https://doi.org/10.1016/j.isprsjprs.2015.01.016

Xiong, X., Adan, A., Akinci, B., & Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, *31*(Supplement C), 325–337. https://doi.org/10.1016/j.autcon.2012.10.006

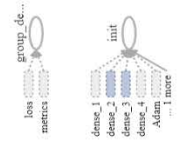
Zeibak-Shini, R., Sacks, R., Ma, L., & Filin, S. (2016). Towards generation of as-damaged BIM models using laser-scanning and as-built BIM: First estimate of as-damaged locations of reinforced concrete frame members in masonry infill structures. *Advanced Engineering Informatics*, *30*(3), 312–326. https://doi.org/10.1016/j.aei.2016.04.001

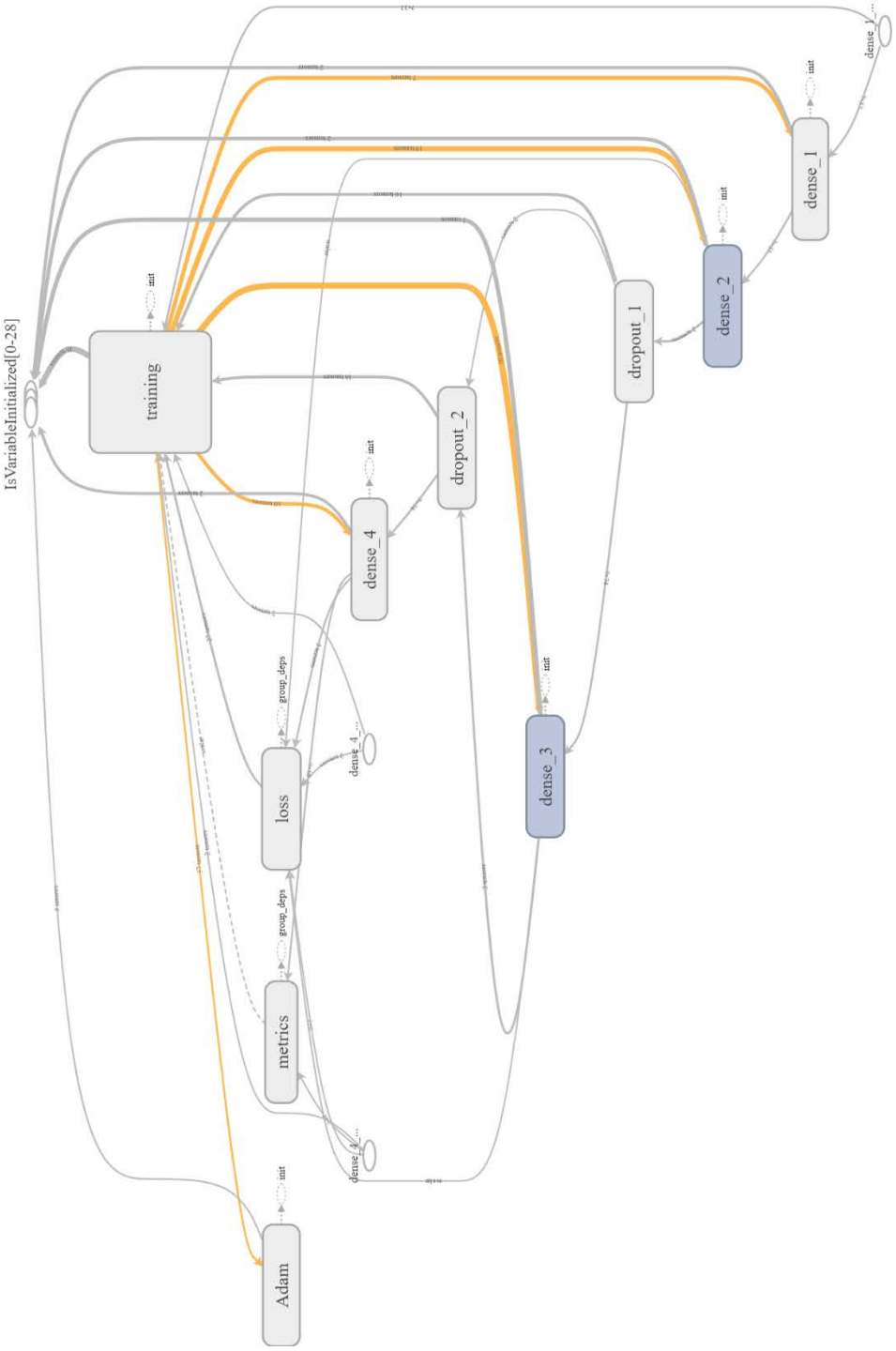
Zhang, X., Bakis, N., Lukins, T. C., Ibrahim, Y. M., Wu, S., Kagioglou, M., … Trucco, E. (2009). Automating progress measurement of construction projects. *Automation in Construction*, *18*(3), 294–301. https://doi.org/10.1016/j.autcon.2008.09.004

Zhang, Y., Song, S., Tan, P., & Xiao, J. (2014). Panocontext: A whole-room 3d context model for panoramic scene understanding. In *European Conference on Computer Vision* (pp. 668–686). Springer.

Zhao, X., & Ilieş, H. T. (2017). Learned 3D shape descriptors for classifying 3D point cloud models. *Computer-Aided Design and Applications*, *14*(4), 507–515. https://doi.org/10.1080/16864360.2016.1257192

Appendix I

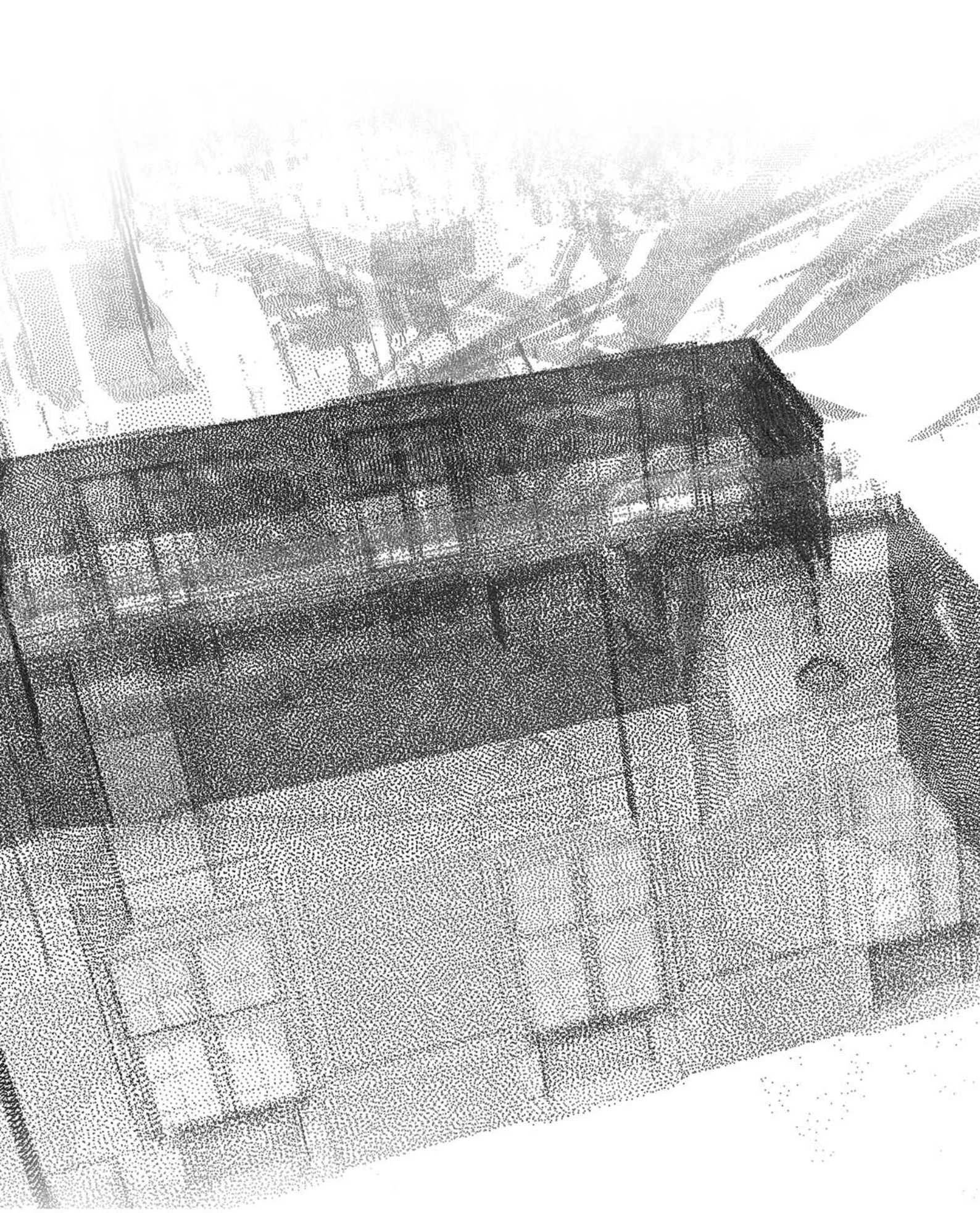

group_de...
loss
metrics
init
dense_1
dense_2
dense_3
dense_4
Adam
... 1 more


IsVariableInitialized[0-28]
training
init
dropout_1
dropout_2
dense_1
dense_2
dense_3
dense_4
loss
metrics
Adam
group_deps

Appendix II

A detailed workflow of data parsing and preparation and python scripts can be found at:

https://github.com/guidoport/Graduation-Project



TU/e Technische Universiteit Eindhoven University of Technology