Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
*** * ***
Université de Carthage
*** * ***
**Institut National des Sciences
Appliquées et de Technologie**

Towards Automated Construction Modeling: Semantic Segmentation and Classification of Point Clouds into BIM Objects

- Presented by : Malek Zitouni

- Host Company : BIM IN TOUCH

- Internship Period : from 01/08/2024 to 31/08/2024

- Academic Year : 2024-2025

Introduction

- ***The documentation is a research study on the automation of Building Information Modeling (BIM) creation directly from scan data, known as SCAN TO BIM***

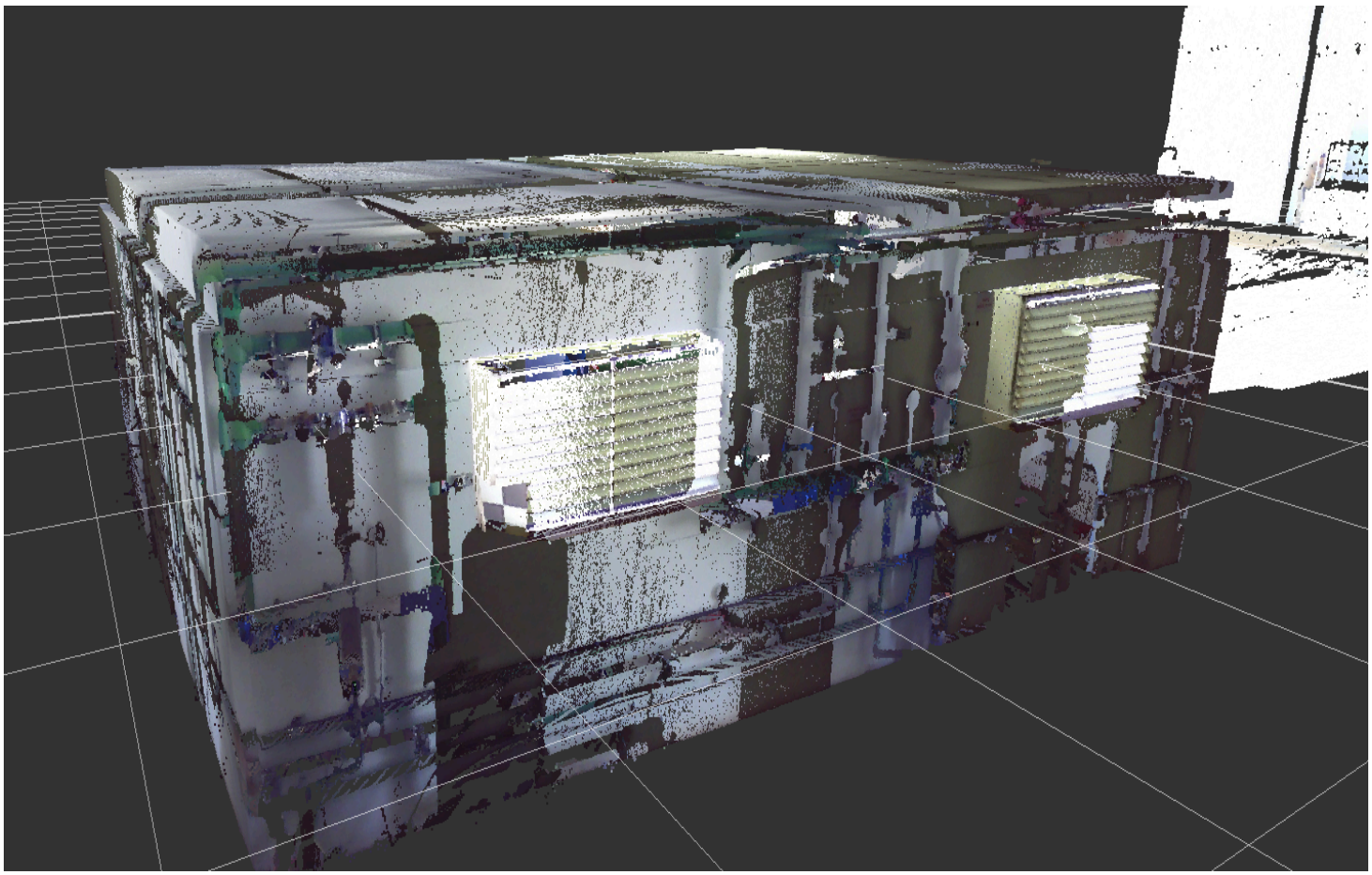


Figure1 : Scan data known as Point Cloud

Book 1: State of research in automatic as-built modeling

Building Information Models (BIMs) are digital representations of facilities that encode all the relevant information about their life cycle from construction to demolition, including, but not limited to, 3D design drawings, schedule, material characteristics, costs, and safety specifications. BIMs gained a general acceptance within the construction industry, as they are expected to provide significant cost savings and improved productivity in construction projects

As-Designed BIM (AD BIM)

1. **Purpose and Creation:**
    - **Purpose: The primary goal of AD BIM is to represent the design intent. It's created to visualize and simulate the project before it is built. This helps stakeholders understand how the final structure is expected to look and function.**
    - **Creation: AD BIM is developed during the design phase of a project. It is based on architectural, structural, and MEP (mechanical, electrical, and**

plumbing) design plans and specifications. Designers and engineers use software tools to model the proposed building according to these plans.

As-Built BIM (AB BIM)

1. **Purpose and Creation:**
    - **Purpose: AB BIM reflects the actual conditions of the building once construction is complete. Its primary purpose is to provide an accurate record of what was actually built, including any changes made during construction.**
    - **Creation: Created after construction is completed. It involves capturing data through methods such as laser scanning, photogrammetry, or manual measurements to document the as-built conditions. The model is updated to include any deviations from the original design due to modifications, errors, or unforeseen issues during construction.**

- **Creating an AB BIM requires two major steps: data collection,to capture the as-built conditions, and data modeling, to generate compact, but rich representations readily understandable by other processes.**

- **The 3D reconstruction field, related to Computer Vision and/or 3D laser scanning techniques, filled the technology gap, offering off-the-shelf tools for generating 3D models of scenes. Consisting, basically, of a set of 3D points endowed with 3D Cartesian coordinates and possibly color information, these models that will be denoted from now on as point clouds, are useful for visualization or augmented reality purposes. However, the data modeling side continues to be deficient, and the problem of converting the raw point cloud into a semantically rich BIM model is far from being settled. The commercial and academic tools available at this point to perform this conversion require extensive human intervention, making them expensive and error-prone ,automatically generating AB BIMs from point clouds is a key objective for the industrial, academic, and governmental parties involved in the Architectural/Engineering/Construction and Facility Management industry (AEC/FM).**
- **As-built modeling has a large overlap with the Geometry Processing field, but the input is restricted to point clouds of infrastructure assets, and not any generic object, simplifying the problem to rigid shape**

analysis . However, the huge amount of data to be processed (more than one hundred million points in some cases) and the high number of elements in the scene require efficient methods, and therefore generic Geometry Processing tools can not be straightforwardly applied. Considering the data to be modeled, as-built modeling acts like a zoom in into the urban modeling problem. Naturally, the approaches proposed in the two areas bear many resemblances. However, as-built modeling must operate at a finer level of detail, requiring more complex elements to be detected and recognised. But even so, due to the prior information available on the shape and distribution of the scene elements, the object recognition within as-built model ling is a simpler problem than the generic object recognition encountered in Computer Vision. Still, the common lack of distinct features and the non-discriminant shape of the elements typically encountered in facility scenes, generate ambiguities and often cause Computer Vision tools to fail . Being closely related to several research fields, good interdisciplinary communication and joint research

efforts are essential for the state-of-the-art in as built modeling to progress.

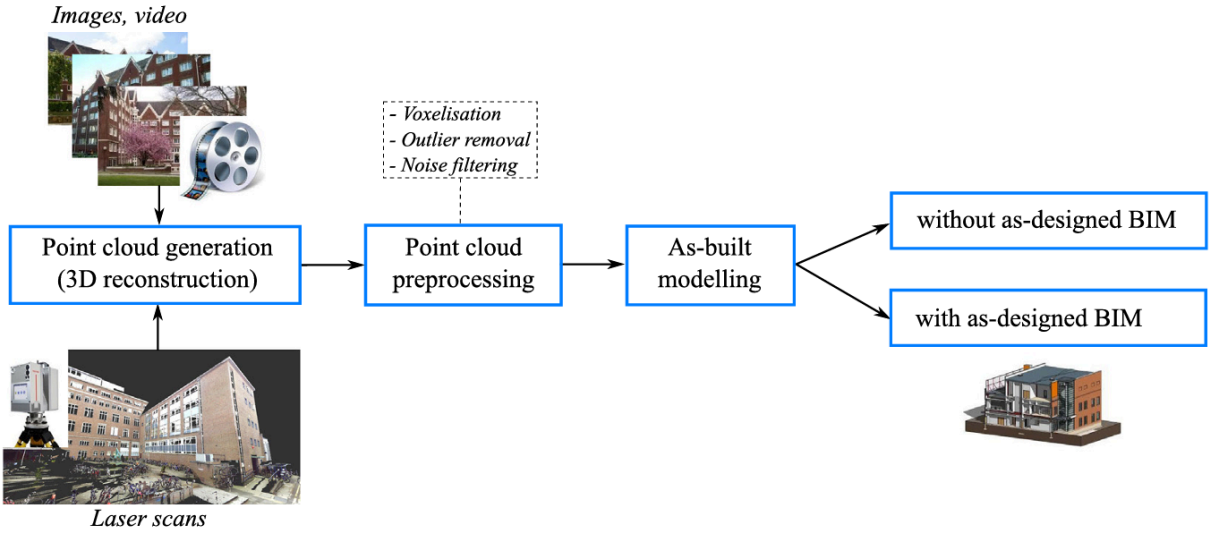


Fig. 1. As-built modelling process.

- **Generation of 3D point clouds :**

  The input of the as-built modeling process is a 3D point cloud, usually obtained through a 3D reconstruction method, which takes its data either from imaging (camera) systems , or from (time-of-flight, phase-shift) laser scanners . Regardless of the data source, the core of the 3D reconstruction process consists of a registration procedure aiming to bring different sets of data into the same

coordinate system. The differences, however, reside in the deformations induced by each system, and the features available to recover them. Both cameras and laser-scanners need to be moved around the scene to ensure a good coverage, hence in both cases, rigid deformations (translation, rotation) must be considered.

- **3D point cloud preprocessing :**

  The measurement accuracy of the point cloud data has a crucial influence on the accuracy of the as-built modeling process. Pre-processing the input data to remove outliers, reduce noise, and compensate for missing data, has become a prerequisite in as-built modeling. In the case of very large point clouds, an initial segmentation into smaller parts can be performed, which are processed in parallel . The alternative is to downsample the point cloud, e.g. through voxelization: partition the point cloud according to a 3D grid, and replace all the points within a cell with the centroid of the cell. Note that voxelization can also be used to gain adjacency information between the points , useful for density estimation , or later during modeling . For memory efficiency, tree structures are used to store the voxelized point cloud

Book 2 : Automatic BIM component extraction from point clouds of existing buildings for sustainability applications

Abstract: Building information models (BIMs) are increasingly being applied throughout a building's lifecycle for various applications, such as progressive construction monitoring and defect detection, building renovation, energy simulation, and building system analysis in the Architectural, Engineering, Construction, and Facility Management (AEC/FM) domains. In conventional approaches, as-is BIM is primarily manually created from point clouds, which is labor-intensive, costly, and time consuming. This paper proposes a method for automatically extracting building geometries from unorganized point clouds. The collected raw data undergo data downsizing, boundary detection, and building component categorization, resulting in the building components being recognized as individual objects and their visualization as polygons. The results of tests conducted on three collected as-is building data to validate the technical feasibility and evaluate the performance of the proposed method indicate that it can simplify and accelerate the as-is building model from the point cloud creation process.

-Although much progress has been made in automating visual data workflows, construction progress assessment, defect management, and building performance simulation are not fully attuned to the benefits of automatic as-is geometric creation during the design, construction, and facility management stages. Point clouds are increasingly being used as 3D as-is geometric representations of buildings. An as-is point cloud, obtained from a laser scanner or from the processing of massive photographs,

comprises millions of individual points, each having its own 3D relative coordinate information. The primary objective of our study is to provide a preliminary solution that automatically and rapidly creates 3D geometric model of existing buildings from point cloud data that can be further utilized for building performance simulation applications. We collected point cloud data from a laser scanner system and processed it to recognize and distinguish different building envelope components (such as windows, doors, walls, and roof) as individual objects for utilization in applications.There are two major methods of achieving point cloud data, processing massive photographs or using a laser scanner. Golparvar-Fard et al. introduced an image-based as-built modeling technique based on computing from the images themselves, the photographer's locations and orientations, and a sparse 3D geometric representation of the as-built scene using daily progress photographs. The major advantages of photogrammetric systems include fast data collection rates (tens to hundreds of 1024 × 1024 pixel frames per second), and acquisition of rich color and textural information about workspace objects for appearance based object recognition. However, this method has a number of limitations: Different lighting and weather conditions make it difficult to use time lapse photography for performing consistent image analysis at occluded and dynamic site conditions . Further, the geometry of the area will be overlooked if common features from multiple images cannot be found.

Category		Advantages	Limitations
Commercial software programs	Leica CloudWorx	• Providing a point cloud processing plug-in capability for many of the most popular CAD systems • Semi-automatic pipe center line creation • Designed for various applications	• Need manual point cloud segmentation and modeling object category selection
	Intergraph Smart 3D for Plants	• Enabling user for comparison of as-designed model vs. as-is point cloud	• Mainly for industrial application • Need manually creating a pipe center line for pipe modeling
	Autodesk Plant 3D	• More easily performing whole-project review. • Customized standard equipment library • Externally referencing or parametrically creating structural elements • Combine 3D scanning, imaging and position data	• Model spec-driven pipelines and components semi-automatically or manually • Mainly for modeling pipe, equipment, and structure
	Kubit PointSense	• Semi-automatically generating 2D plans from 3D laser scanner data • Designed for various applications • Combine 3D scanning, imaging and position data	• Two manually selected reference points needed for pipeline modeling • Openings (windows, doors) need to be manually created in the 2D plans.
	AVEVA Laser Model InterfaceTM	• Accurately positioning and clash checking new 3D design in situ within an existing installation	• Need manual point cloud segmentation and modeling object category selection • Limited for industrial application
	Trimble RealWorks	• Combine 3D scanning, imaging and position data • Semi-automatically modeling pipes	• Need manual point cloud segmentation and modeling object category selection
	ClearEdge3D	• Semi-automated feature extraction • Deep integration with Revit, AutoCAD, PDMS and other platforms • Designed for various applications	• Need manually segmenting point cloud and selecting object category for object modeling
Academic research efforts		• Automatically modeling for indoor environment • Modeling regular shape	• Difficult to accurately model curved shapes • Incomplete building envelope components recognition

Fig. 1. Literature review of the current as-is BIM recognition techniques.

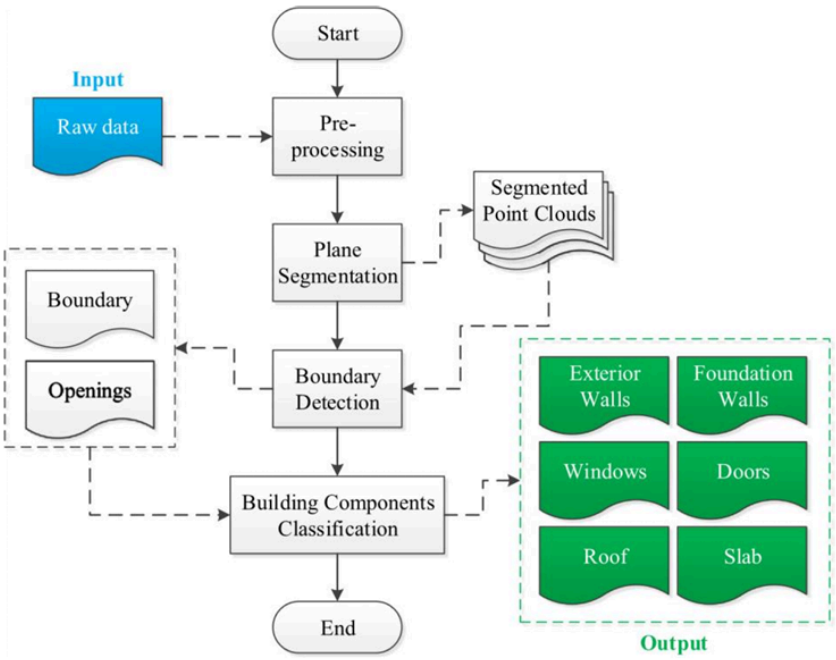


Fig. 2. Flowchart for our proposed method.

- **Recent research efforts towards as-is BIM recognition from point clouds Manually creating 3D models from point clouds is a labor-intensive and time-consuming process. Many commercial software programs or plug-ins have been developed to accelerate this manual process. For example, Leica CloudWorx is able to automatically create a pipe center line based on manually selected pipe, and then the pipe can be manually created follow**
- **ing the center line; Intergraph Smart 3D for Plants can automatically model pipes after user identifying the scanned piping axis of symmetry; Autodesk Plant 3D and Kubit PointSense Plant enable user for manually choosing two points from an image of a pipe, and the corresponding 3D point cloud between the two points can be automatically located and modeled; Kubit PointSense Building can automatically generate 2D building plan (wall, floor, ceiling) from 3D laser scanner data, but with manual openings (window, door) creation;and AVEVA Laser Model Interface™ , Trimble RealWorks and ClearEdge3D are designed to automatically create 3D model by manually segmenting the point cloud and choose the corresponding catalogs for each segment of point cloud Some research efforts have been made on automatic building modeling from point cloud to assist building facility management and performance analysis. Pu and Vosselman proposed a knowledge based method for reconstructing**

building models from laser scanner data, in which they extract the features and the outline of the building and make the geometric model of the building based on several assumptions because only facades on the street side are scanned. Xiong et al. proposed a context based modeling algorithm for creating semantic 3D as-is building models of the interior of buildings. Their context-based modeling algorithm was able to identify and model the main visible structural components of an indoor environment, but could not recognize components with irregular shapes that are frequently seen from the exterior of the building envelope. Hinks introduced a point-based voxelization method to automatically transform point cloud data into solid models for computational modeling. This method can directly convert point data into solid models based on volumetric subdivision, however, limits on 2D building façade modeling. It is also difficult to accurately model curved shapes, such as arched windows. Díaz-Vilariño et al. presented a semantic as-built 3D modeling method for shading analysis on buildings. In this method, only wall, ceiling, and floor components can be recognized and segmented. Complete building envelope component recognition is essential for building performance analysis. As a result, automated and efficient extraction of building envelope geometric information is a challenging emerging topic to be solved

Book 3 : Scan to BIM MappingProcess Description for Building Representation in 3D GIS

Abstract: This paper introduces a novel approach for mapping process description with Scan data to Building Information Modeling (BIM) in a 3D Geographic Information System (GIS). The methodology focuses on automatically generating building mass and facade information on the GIS platform using Point Cloud Data (PCD) of Airborne Laser Scanning (ALS). Advanced scanning techniques capture detailed geometry from the physical site and generate high-resolution point clouds, which are processed to create 3D models for GIS integration. The critical contribution of this research lies in a scalable Scan to BIM mapping process, which can be used for generating building footprints and masses, including attributes, on 3D GIS. The resulting integrated BIM–GIS dataset provides an accurate building mass, facade information, facility asset management, and architectural design and facilitates improved decision-making in urban planning based on enhanced visualization, analysis, and simulation. This study suggests a flexible Scan to BIM mapping process description based on use cases, including algorithms. Through prototype development, a

case study demonstrates the effectiveness of the process approach, the automatic generation of BIM on a 3D GIS platform, and reducing the manual efforts. The proposed method automatically creates DEM, SHP, GeoJSON, IFC, and coordinate system information from scan data and can effectively map building objects in 3D GIS (A system for 3D geospatial mapping. ArcGIS is powerful 3D mapping software that enables industries to get more from their 3D data.

A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. GIS can show many different kinds of data on one map, such as streets, buildings, and vegetation. This enables people to more easily see, analyze, and understand patterns and relationships.

- **Capturing the digital data of existing building assets and converting them from BIM to GIS is one of the main challenges in the Scan to BIM mapping process for BIM–GIS integration . Building masses, including attributes, are critical components that must be accurately represented in BIM and GIS datasets. The conversion process of this dataset should be scalable and adjustable according to user requirements and processed based on**

the GIS coordinate system. PCD processing between these systems in Scan to BIM mapping is time-consuming and error-prone, because it is based on manual processes that are not fully integrated , often leading to discrepancies and data inconsistencies. Since it is difficult to reuse the algorithm parameter values used in the manually incompletely integrated Scan to BIM mapping process, these problems are repeated in similar projects. This paper proposes a flexible Scan to BIM mapping description (SBMD) approach in 3D GIS. This study utilizes ALS technology to capture detailed geometric information from the physical site. The captured PCD is then processed to create highly accurate 3D models that are the basis for integration with GIS.

Article : University Science Building As-Built Revit Model

- **What used to take days to process can now be completed in hours**
- **3DIS wanted to streamline the process of converting laser-scanned data into a usable format for 3D modeling, with the goal of reducing overall project time and costs.**

Automating this process was extremely important for both meeting the project's tight deadlines, and helping the company handle more jobs simultaneously

- **3D Imaging Services specializes in as-built documentation for the AEC and Plant industries. The company uses both traditional survey equipment and 3D laser scanning to capture spatial information from sites and convert them into dynamic 2D drawings, 3D models or Building Information Models for a diverse market of clients.**

Book 4 :Quality Control for Autodesk® Revit® MEP Models

Now that your Autodesk Revit software implementation is progressing nicely, have you considered what your next step will be? What about quality control checks of the Revit model itself? No, we're not talking about validating design—we're talking about the quality of the model itself. Are the elements correct? Are your company's standards being followed? This class will demonstrate approaches to checking a model for quality that can be used in any company and using the

Autodesk® Revit® Model Review plug-in for more automated checking

Book 5 : Google Compute Engine

-Compute Engine offers many advantages: leading-edge hardware, upgraded regularly and automatically; virtually unlimited capacity to grow or shrink a business on demand; a flexible charging model; an army of experts maintaining computing and networking resources; and the ability to host your resources in a global network engineered for security and performance. Google Compute Engine is about giving you access to the world's most advanced network of data centers—the computing resources that power Google itself. Practically speaking, this means providing APIs, command-line tools and web user interfaces to use Google's computing and networking resources At a high level, Compute Engine instances, networks, and storage are all owned by a Compute Engine project. A Compute Engine project is essentially a named collection of information about your application and acts as a container for your Compute Engine resources. Any Compute Engine resources that you create, such as instances, networks, and disks, are associated with, and contained in, your

Compute Engine project. The API offers a way to interact with Compute Engine components and resources programmatically

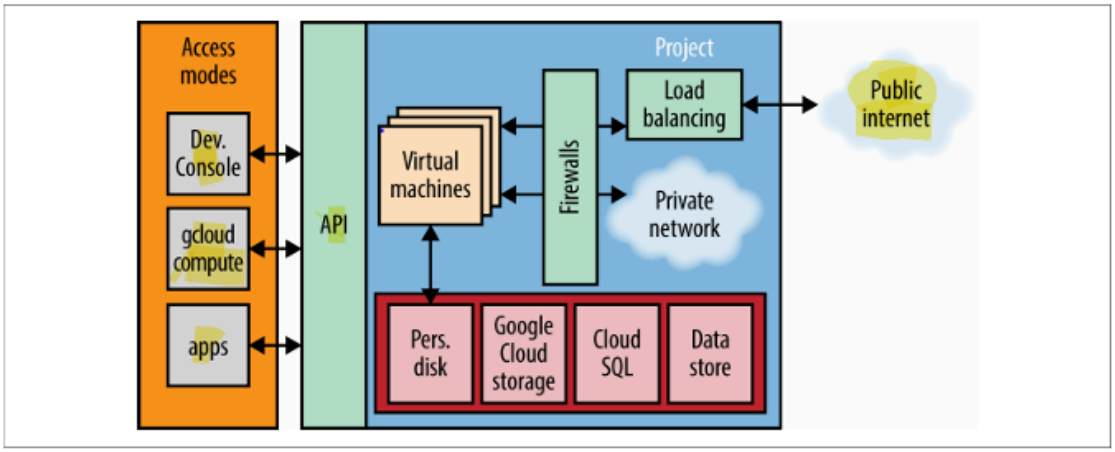


Figure P-2. Overview of Google Compute Engine's components

- Google Compute Engine is a service that provides virtual machines (VMs) that run on Google's infrastructure. You can create VMs with a variety of configurations using a number of available operating systems. The instance's data is stored and maintained on persistent block storage that is replicated for redundancy and persists beyond the life cycle of the VM. Network access can be configured to allow your virtual machines to talk to each other, the Internet, or your own private network. Google Compute Engine provides a variety of tools you can use to interact with and manage your Compute Engine instances and configurations; for example, you can start and stop instances, attach disk storage, and configure network access using each of these access points.

- To start working with Google Compute Engine, you first need to create a Compute Engine project in the Developers Console. A Compute Engine project is a collection of information about your application and acts as a container for your Compute Engine resources and configurations. Disks, firewalls, networks, and instances are all associated with and contained within a single project. Billing is applied to a project based on the amount of resources used

-"Can view" gives the team member read access to your project settings and Compute Engine resources. "Can edit" gives the team member write access to your project settings and Compute Engine resources (including adding and deleting resources). "Is owner" gives the team member all edit permissions, plus the ability to manage project settings and team members.

- For example, disk resources provide storage to your instance, and network resources route traffic to and from your instance. Resources available for instance configuration include: Image The base software for a hosted virtual machine, including the operating system and all associated system and application software Disk A disk provides storage for your Compute Engine instances Snapshot A copy of the contents of an existing persistent disk (i.e., an image copy) Network Set of rules defining how an instance interacts with other instances, other networks, and the Internet ,Firewall A single rule defining how

an instance can accept incoming traffic1 Route A table which determines how traffic destined for a certain IP should be handled Address A static IP address for your instances Machine type A hardware configuration dictating the number of cores and available memory for a given instance All resources are owned by a single project. Within that project, resources are available at different hierarchical levels (a.k.a. scopes). Currently, there are three scopes: global, regional, and zonal. Resources within the global scope are available to resources within the same project (in other words, they are globally available within that project). Resources within the regional scope are available to other resources within the same region. Finally, resources within the zonal scope are available to other resources within the same zone. The current breakdown of resource levels is as follows: Global resources Image, snapshot, network, firewall, route Regional resources Address Zone resources Instance, machine type, disk

- Cloud SDK The Cloud SDK (https://cloud.google.com/sdk/) contains multiple tools and libraries that allow you to interact with Google Cloud Platform. For example, the Cloud SDK contains gsutil, a command-line tool for managing your Cloud Storage buckets and objects, and bq, a command-line tool for working with and querying your BigQuery datasets. Similar to gsutil and bq, the Cloud SDK also contains a tool called gcloud compute that provides a command-line interface for managing

your Compute Engine resources. Similar to the UI, gcloud compute allows you to start or stop instances, add or remove persistent disks, and create or destroy firewalls. gcloud compute is useful for learning how Compute Engine works and also for writing bash scripts to control your Compute Engine resources.

-Google API libraries. Google has developed a convenient set of libraries that simplify both the OAuth 2.0 flow and the HTTP requests to all of Google's APIs, including the Compute Engine API. The libraries are available in a variety of languages, such as Java, Python, JavaScript, and Ruby. The libraries are available from the Google APIs Client Library page in the Compute Engine online documentation (https://develop ers.google.com/compute/docs/api/libraries). My first Compute Engine application. Let's take a look at how to use the Python Google API Client Library to make an HTTP request to the Compute Engine API to retrieve information about your project.

- Instances The core capability provided by Google Compute Engine is an instance, also called a virtual machine (VM). An instance is a simulated computer, running in a partitioned environment on a real server in one of Google's data centers. From the user's point of view, an instance behaves like a dedicated physical computer, with its own operating system,

storage devices, network adapters, and so on. Virtualization is the process of mapping a virtual machine's environment and resources onto services provided by real hardware. The software managing the virtualization service is called a hypervisor or virtual machine manager. There are many popular virtualization systems in use today. Google Compute Engine uses the Linux Kernel-based Virtual Machine (KVM) software. The KVM hypervisor runs in a standard Linux environment, which means that virtualized and non virtualized workloads can run side by side on the same physical hardware. This eliminates the need to manage a separate pool of resources dedicated to the Compute Engine product—the same hardware and software stack used to serve Google search, Gmail, Maps, and other Google services can also provide Compute Engine virtual machines. In this chapter, we'll learn about Compute Engine instances and their attributes. Along the way, we'll learn how to create, access, and delete instances via three differ ent access modes: • The Developers Console Web UI • The gcloud command-line tool • The Google Python client library

- the VM Instances tab, which shows you all instances currently active in any state in your project, along with a summary of the most useful information about each instance: Name The name given to the instance when it was created Zone The location in which the instance is situated Disk The name of the instance's boot persistent disk (we'll cover persistent disks in Chapter 3,

but for now just think of this as the instance's boot drive) Network The name of the virtual network associated with the instance External IP The instance's externally accessible IP address Connect Provides three ways to remotely log in to the instance

Book 7 : Labeled Indoor Point Cloud Dataset for BIM Related Applications

Abstract:

BIM(buildinginformationmodelling)hasgainedwideracceptancei ntheAEC(architecture, engineering, and construction) industry. Conversion from 3D point cloud data to vector BIM data remains a challenging and labour-intensive process, but particularly relevant during various stages of a project lifecycle. While the challenges associated with processing very large 3D point cloud datasets are widely known, there is a pressing need for intelligent geometric feature extraction and reconstruction algorithms for automated point cloud processing. Compared to outdoor scene reconstruction, indoor scenes are challenging since they usually contain high amounts of clutter. This dataset comprises the indoor point cloud

obtained by scanning four different rooms (including a hallway): two office workspaces, a workshop, and a laboratory including a water tank. The scanned space is located at the Electrical and Computer Engineering department of the Faculty of Engineering of the University of Porto. The dataset is fully labeled, containing major structural elements like walls, floor, ceiling, windows, and doors, as well as furniture, movable objects, clutter, and scanning noise. The dataset also contains an as-built BIM that can be used as a reference, making it suitable for being usedinScan-to-BIM andScan-vs-BIMapplications. For demonstration purposes, a Scan-vs-BIM change detection application is described, detailing each of the main data processing steps.

-Indoor environment 3D reconstruction presents specific challenges due to the particularly irregular layout of the buildings and the existence of clutter and occlusions [7]. For example, inefficient scan location planning causing deficient coverage may lead to the acquisition of insufficient data about elements such as walls, floors, and windows during data collection, which can greatly affect the performance of the reconstruction algorithms. Additionally, this lack of data may hinder the determination of the topological relationships between the indoor spaces

-Despite this growing awareness, few datasets adequate for research and development Despite this growing awareness, few datasets adequate for research and development of Scan-to-BIM and Scan-vs-BIM methodologies can be found that are of Scan-to-BIM and Scan-vs-BIM methodologies can be found that are publicly accessible and well documented. The EU FP7 funded Durable Architectural Knowledge (DURAARK) project was a three-year project that aimed to develop methods for the long-term preservation of building data, including 3D point clouds and BIM models as well as the related metadata, knowledge, and Web data. The project covered a wide range of processes and methods related to the domain of architectural 3D content.

-An interesting 3D reconstruction benchmarking initiative held since 2017, named the ISPRS benchmark on indoor modelling, aimed at enabling a direct comparison of different methods for generating 3D indoor models from point cloud data by providing a public benchmark dataset and an evaluation framework. After the initiative ended, the analysis of the results reported by the authors [11] indicated that the reconstructed methods displayed different performances across the datasets, demonstrating the importance of having distinct datasets to assess performance, with different features and degrees of complexity. Interestingly, they also state that the generated

models displayed higher correctness if points corresponding to clutter are classified and filtered in a pre-processing step before geometric feature extraction. Additionally, the presence of clutter in the point cloud wasidentified as one of the main challenges in automated indoor modelling as clutter (1) can easily be confused with structural elements and (2) occludes the mentioned elements, causing missing data in the point cloud. Unfortunately, after these results were published the benchmarks datasets webpage went offline [12]. It is understandable that this would happen knowing that the purpose of the dataset was to benchmark the reconstruction algorithms and the study with the results had already been published. Other reason that contributed to this was, as in the previous case, the decision to host the data on a non-specialized self-hosted repository. Anotherdataset is hosted in a non-specialized repository for data storage (e.g., GitHub) andpresentsacompletedatasetdescription, but doesnotprovidealabelledpointcloud[13]. Overall, even if the unavailable datasets were in fact available, they are not labelled, thus not contributing to the development of CV techniques which enable object classi f ication and efficient clutter removal. The complex interactions between objects, clutter, and occlusions create difficulties for feature extraction algorithms especially in indoor environments, therefore making this dataset valuable to the CV community

-This data descriptor contributes to the goal of having a reference dataset by presenting an indoor mapping dataset and information about the major elements present in the scene like walls, floor, ceiling, windows, and doors, as well as furniture, movable objects, and clutter

-Validating Scan-to-BIM and Scan-vs-BIM algorithms: These algorithms involve processing of the acquired point clouds in order to extract structural features for modeling purposes. As mentioned, there is a growing interest in automated reconstruction techniques and datasets including high-quality, high-resolution point clouds and corresponding BIM models are required for validating them for benchmarking purposes.

- BIM can be regarded as a process responsible for generating and managing build ing information during its entire lifecycle. The diverse information contained in a BIM model together with its 3D representation ability makes it possible to enhance the design, construction, and maintenance phases of the corresponding building. One key factor that enables a successful BIM implementation is the existence of accurate information in the BIM model. However, the contained information is often inaccurate, out of date, or even missing. Modern sensing technologies such as 3D laser scanning, have been widely adopted in the AECdomainasameantocapture this information. In essence, two BIM related processes use the

acquired point clouds to tackle these issues: Scan-to-BIM: the process of transforming point cloud data into actual BIM models; Scan-vs.-BIM: the process of comparing point cloud data representing an as-built building to its as-designed BIM model in order to identify differences.

Table 2. Point cloud column variables.

Variable	Description
Point X coordinate (m)	Coordinate of a point in the X axis
Point Y coordinate (m)	Coordinate of a point in the Y axis
Point Z coordinate (m)	Coordinate of a point in the Z axis
Point colour (R)	Red colour intensity (0–255)
Point colour (G)	Green colour intensity (0–255)
Point colour (B)	Blue colour intensity (0–255)
Intensity	Return strength of the laser beam (0–1)
Label	Point label

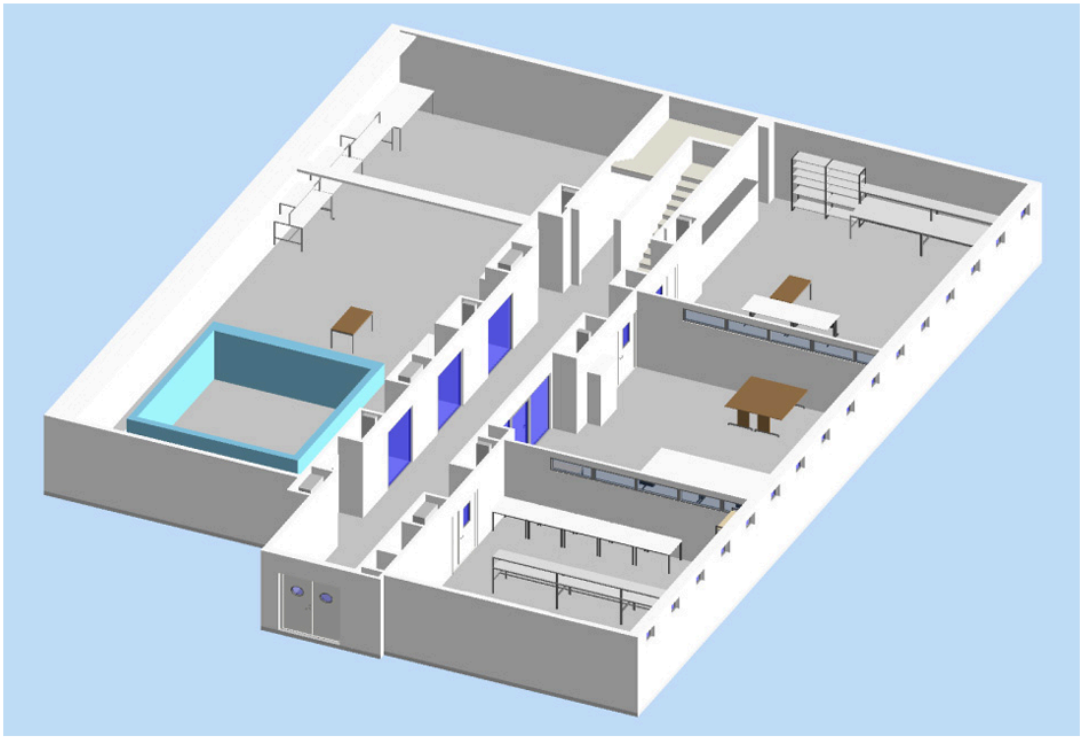


Figure 4. BIM model of the spaces.

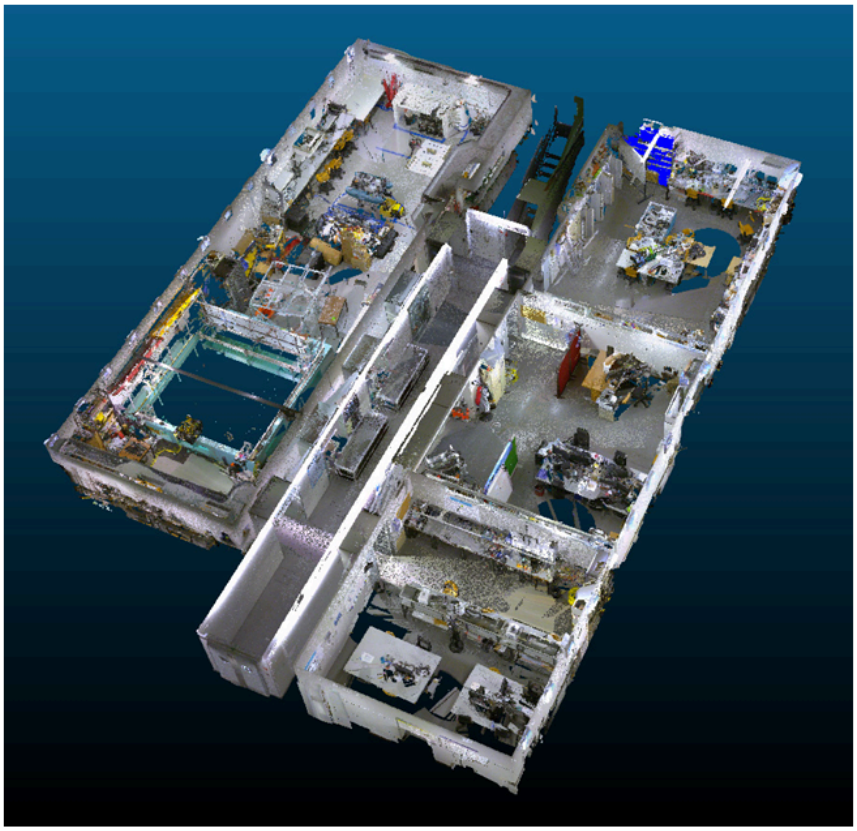


Figure 3. The point cloud acquired in one of INESC TEC robotics research laboratories.

-Construction quality can typically be assessed by comparing the as-planned BIM model to the acquired point cloud, obtained by scanning the target structure. In the literature, this comparison is performed by registering the point cloud to the BIM model (since they do not have the same coordinate reference frame), and detecting differences such as the existence of certain elements in the BIM which are not covered by the point cloud or the existence of geometric features in the point cloud which are not present in the BIM.

Book 8 : Automatic creation of semantically rich 3D building models from laser scanner data

In the Architecture, Engineering, and Construction (AEC) domain, semantically rich 3D information models are increasingly used throughout a facility's life cycle for diverse applications, such as planning renovations, space usage planning, and managing building maintenance. These models, which are known as building information models (BIMs), are often constructed using dense, three dimensional (3D) point measurements obtained from laser scanners. Laser Scanners can rapidly capture the “as-is” conditions of a facility,which may differ significantly from the design drawings. Currently, the conversion from laser scan data to BIM is primarily a

manual operation, and it is labor-intensive and can be error-prone. This paper presents a method to automatically convert the raw 3d Point data from a laser scanner positioned at multiple locations throughout a facility into a compact, semantically rich information model. Our algorithm is capable of identifying and modeling the main visible structural components of an indoor environment (walls, floors, ceilings, windows, and doorways) despite the presence of significant clutter and occlusion, which occur frequently in natural indoor environments. Our method begins by extracting planar patches from a voxelized version of the input point cloud. The algorithm learns the unique features of different types of surfaces and the contextual relationships between them and uses this knowledge to automatically label patches as walls, ceilings, or floors. Then, we perform a detailed analysis of the recognized surfaces to locate openings, such as windows and doorways. This process uses visibility reasoning to fuse measurements from different scan locations and to identify occluded regions and holes in the surface. Next, we use a learning algorithm to intelligently estimate the shape of window and doorway openings even when partially occluded.Finally, occluded surface regions are filled in using 3D inpainting algorithm.We Evaluated The Method on a large, highly cluttered data set of a building with forty separate rooms.

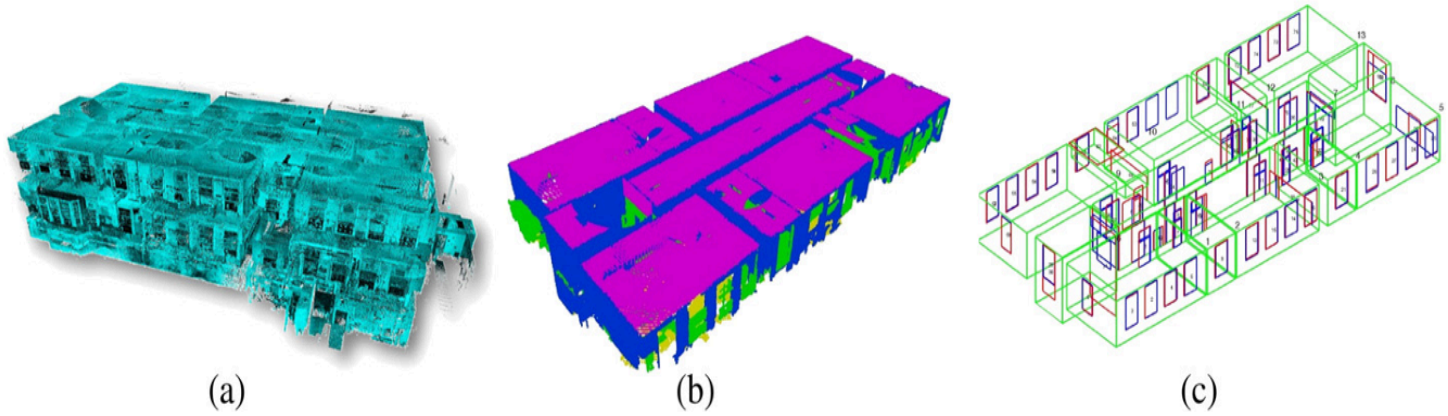


Fig. 2. Algorithm overview. Given a registered point cloud of a facility (a), each room is analyzed in two phases. First, context-based modeling (b) recognizes and models key structural components – walls (blue), floors (yellow), ceilings (magenta), and clutter (green). Detailed surface modeling (c) detects and models openings (red rectangles) from windows, doors, and built-in cabinets and fills in occluding regions on each wall, ceiling and floor surface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Book 9 : AUTOMATED SEMANTIC MODELING OF BUILDING INTERIORS FROM IMAGES AND DERIVED POINT CLOUDS BASED ON DEEP LEARNING METHODS

In this paper, we present an improved approach of enriching photogrammetric point clouds with semantic information extracted from images to enable a later automation of BIM modelling. Based on the DeepLabv3+ architecture, we use Semantic Segmentation of images to extract building components and objects of interiors. During the photogrammetric reconstruction, we project the segmented

categories into the point cloud. Any interpolations that occur during this process are corrected automatically and we achieve a mIoU of 51.9 % in the classified point cloud. Based on the semantic information, we align the point cloud, correct the scale and extract further information. Our investigation confirms that utilizing photogrammetry and Deep Learning to generate a semantically enriched point cloud of interiors achieves good results. The combined extraction of geometric and semantic information yields a high potential for automated BIM model reconstruction.

-The digitalisation of the building sector is progressing steadily and, with Building Information Modeling (BIM), is taking the step from two-dimensional plans on paper to comprehensive, three-dimensional digital building models. These BIM models are the central element and represent the entire life cycle of a building, from planning and operation to demolition. In addition to the three-dimensional component and object geometries, they also contain all relevant semantic information. However, the introduction of BIM is currently taking place virtually only in the planning of new buildings. Due to the very high complexity of manual data acquisition and processing, the recording of already existing buildings as BIM models has been a minor topic of interest so far. Developing an automatic extraction of the necessary information out of measurement data yields a high potential at simplifying the creation of such

“As-Build” or “As Is” models and thus the possibility to make them widely available. Creating these models requires three-dimensional data. We consider photogrammetry as a great method to not only capture and reconstruct buildings as point clouds of high quality but also extract further semantic information out of these images. Especially the categories of objects need to be available, as these are the foundation of every model. In this paper, we present our approach to providing both semantic and geometric information of an interior room in a classified point cloud in an automated process.

-Overview : Images are the main component of our approach because of the massive amount of information contained in them. They are the input for the photogrammetric point cloud generation and the extraction of objects based on Deep Learning methods.

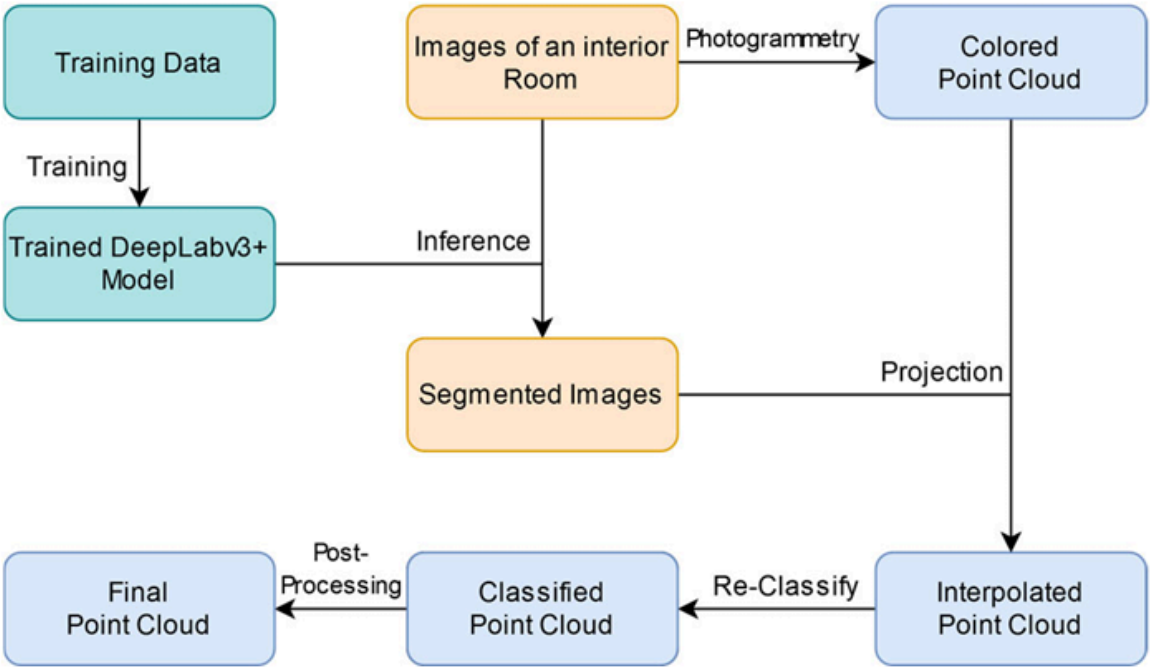


Figure 1. Overview of our workflow. Images of an interior room are the input for inference with our trained neural network and the photogrammetric point cloud generation. An interpolated point cloud is produced by projection of the segmented images. This point cloud is reclassified and used as the input in the final post-processing step.

-We use DeepLabv3+ as the architecture of our neural network to segment components and objects of interiors visible in the images at pixel-level. The training of the model is conducted using manually segmented ground truth data. The trained model then is used for inference on the images of an exemplary room. The base images also are used to generate a point cloud using photogrammetry. Afterwards, the category information

stored in the RGB-values of the segmented images is transferred in the point cloud by projection based on the determined camera parameters. Ideally, the result would have been a classified point cloud, but because of interpolation, there is no clear assignment of category colours to the points. Therefore, an additional step is taken to reclassify them. By using the clearly classified point cloud as input for further post processing, we are able to automatically correct the rotation and scale, as well as extract additional information like floor, ceiling and wall planes. The presented steps are described in more detail below. 3.2 Semantic Segmentation of Interiors A comprehensive and high-quality segmentation of all the important building components and objects in the images is of essential importance for a complete reconstruction of an existing building. For this purpose, the object categories were expanded to a total of 25 components and objects important for a great variety of interiors. Even though "Wall" is a major component of interiors, it had to be dismissed from the segmentation classes of the neural network because it had a negative influence on training. Therefore, the model was trained for extracting the remaining 24 categories and a "Background" class of unclassified objects. As these were to be segmented with Deep Learning out of images, a new training data set was created. It is based on approximately 300 images and the corresponding manually segmented ground truth

annotations. Using data augmentation, the training dataset was expanded to almost 18,000 unique images.

Room forming components	Connecting components	Fixed objects of interest		Movable objects of interest
Floor	Door	Light switch	Socket	Poster
Wall	Window	Lamp	Pillar	Bookshelf
Ceiling		Heater	Pipe	Carpet
		Stairs	Railing	Cabinet
		Sink	Toilet	Chair
		Cable trunking	Fire alarm	Table
		Fire alarm siren		Fire extinguisher

Table 1. Overview over the 25 categories of interiors we are interested in. The category "Wall" had to be excluded from the segmentation classes as it couldn't be segmented by the neural network. Our model was trained for the remaining 24 classes plus a "Background" class of unidentified objects.

Figure 2. Figure 2: Exemplary results of using our trained DeepLabv3+ model for inferencing the images of an interior room.

-In this paper, we were able to verify that the combination of photogrammetry and Deep Learning is a solid approach to generate a semantically enriched point cloud of interiors. The combined extraction of geometric and semantic information based on segmentation with DeepLabv3+ and projection into the photogrammetric point cloud achieves good results. In consequence, the components and objects can be differentiated very well in the point cloud. The reached mIoU of 51.9 % for the classified point cloud confirms the good quality of this approach. Additional important information essential for a BIM model can be extracted by analyzing and post-processing the point cloud. We are confident that we will be able to improve the results by further optimizing the methods used. In the future, it is planned to extend this approach to the combination and joint processing of several rooms as well as including mobile laser scanners. With the methods presented by us, we have created a solid basis for the acquisition and modeling of semantic and geometric information of interiors for BIM models towards their automated reconstruction.

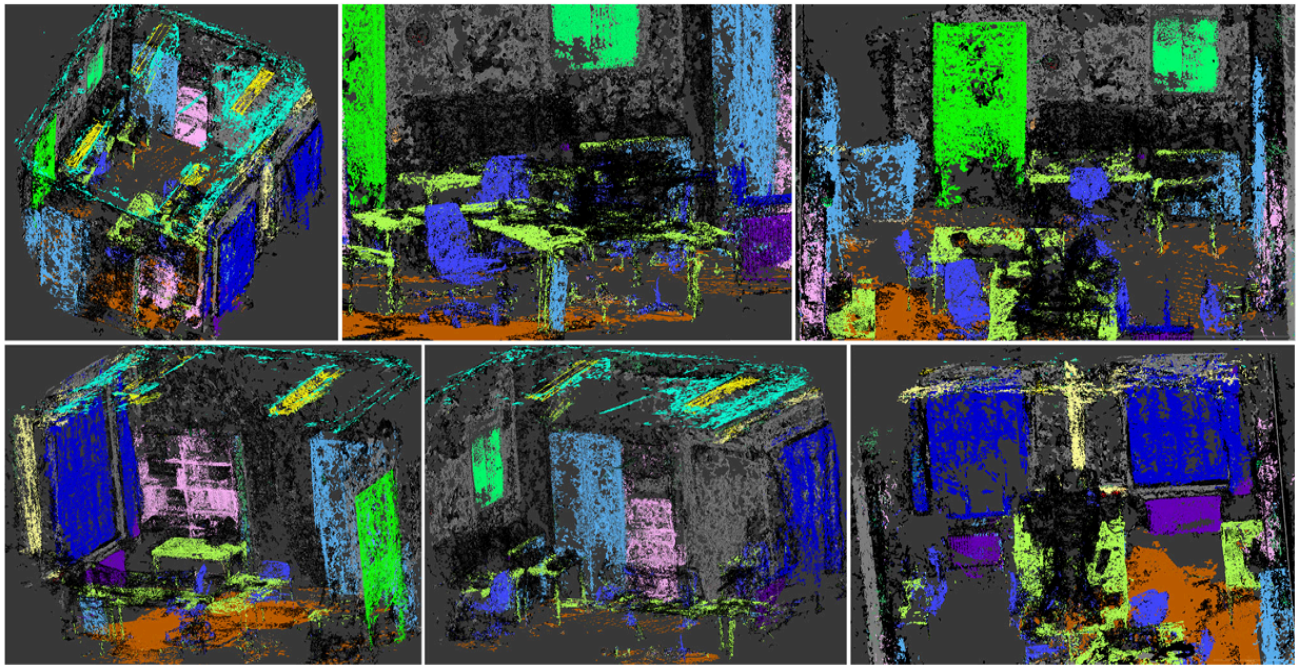


Figure 6. Different views of the post-processed final point cloud. Rotation and scale are corrected. Extracted wall point are visible in the point cloud in light grey colour.

Book 10 : Exploiting Inductive Bias in Transformer for Point Cloud Classification and Segmentation

-Point cloud is a digitized data of the surface on a real three-dimensional object, containing geometry coordinates, and sometimes, also having attribute information. The original point cloud data can be obtained by equipment such as lidar or depth camera. Because of its irregularity and disorder, new challenges have been brought to point cloud processing.

- We propose an efficient Inductive Bias-aided Transformer (IBT). Inductive Bias is indeed like the local consistency of CNN filters, which is mentioned in many papers [18]–[20]. Broadly speaking, inductive bias encourages learning algorithms to prioritize solutions with certain properties. IBT mainly consists of three crucial modules. The Relative Position Encoding module is mainly to explicitly encode the spatial layout of 3D points, which plays a crucial role in shape analysis. Existing methods [5], [9], [21] usually concatenate location information and features into features for learning, but are not ideal in capturing meaningful ensemble patches. Therefore, the advantage of our design module is to first explicitly encode the point position, and then combine the output with the feature of the point to further enhance it, so that each point can observe its local geometry, thus finally enriching the entire network. This lays a good foundation for subsequent deep feature learning. Attentive Feature Pooling in the follow-up design is to update the features of the center point, focusing on deep relational learning. we design two branches to fully integrate deep-level features, including maximum pooling to obtain the most prominent features of neighboring points and attention pooling to automatically select the most important features. In order to establish the connection between local and global features, we pass the learned local features through the sigmoid function to obtain the weight coefficient of each channel for

each point, and integrate them into the Locality Aware Transformer for global feature learning. We stack three layers of IBTtoobtainsemanticinformationfromcoarse-to fineforbetterclassificationandsegmentation.Wehavedone extensiveexperimentsontheModelNet40[2],ScanObjectNN [3]andShapeNetPart [4]datasets to confirm its effectiveness. Our Main Contributions Are Summarized As Follows: WedesignaneffectiveInductiveBias-aidedTransformer layertodeeplyintegratelocalattentionwithglobalatten tion.Theinductivebiasrefersspecificallytothelocality used to model local visual structure. WedesignRelativePositionEncodingandAttentiveFea turePoolingtoobtainthemostcriticalandrepresentative local features, and improve the traditional transformer with locality-based channel interaction. By incorporating the above modules into the design of the network architecture,ourmethodsurpassesprevious best performingwork in point cloud classification, and yields the best IoU performance for most of object categories,demonstrating very competitive segmentation performance.

-Direct processing on raw points will not damage original geometry information. PointNet [1]was the first of its kind to advance this type of research and achieved good results. It mainlyusesMLPforeachpointindependentlyandaggregates the global featureswith a symmetric function. Subsequent

development work can be roughly divided into three categories:MLP convolution-based, graph convolution-based andTransformer-based methods. MLPorConvolution basedMethods. It extracts features through FLO or convolution.SincePointNet [1] ignores the interconnection between points, the subsequently proposed PointNet++ [5] uses a multi-scale structure to effectively capture local features. In Addition, therearesomestudieson howtoperformconvolutiononpoints.For Example,PointConv [6] introducespointdensityintotheconvolutionkernel.This algorithmalsoproposesastrategyfor effectivelycalculating weight functions toincreasenetworkdepthand performance. Graph-basedMethods. It updates features by constructing local or global graphs. Dynamic graph-basedDGCNN [9] is good at capturing geometric structural features from structured local graphs. In addition, DeepGCN[11] connects all points as a global graph structure, introduces residual connectionandconvolutionintographconvolutionalnetwork (GCN),whichfinallybuildsaGCNasdeepasCNN. Transformer-basedMethods. It pays more attention to important parts by computing different coefficients between points.For Example,PCT[12]obtains deeper semantic information by stacking several global offset attention layers. PT [13]usesattentiontoobtainsemanticinformationofdifferent resolutionsvia theprocessof progressivedownsampling. In

addition, PointANSL[14] employs the attention mechanism to handle noisy point cloud data via adaptive sampling.

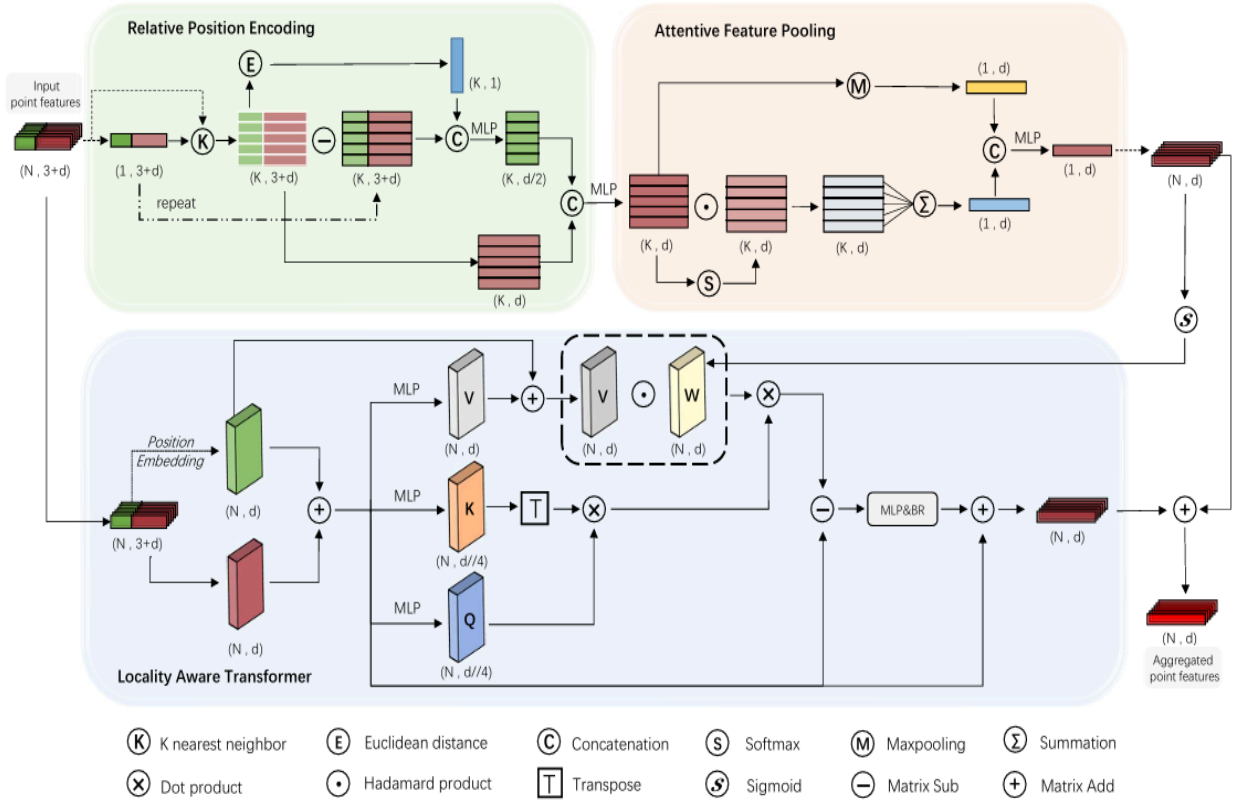


Fig. 1. The proposed Inductive Bias-aided Transformer. The top-left shows the relative position encoding module for extracting position information, and the top-right illustrates the procedure of fusion of two types of feature pooling. The bottom panel shows how the improved transformer utilizes local features to enhance self-attention mechanism.

Book 11 :A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS

Today 3D models and point clouds are very popular being currently used in several fields, shared through the internet and even accessed on mobile phones. Despite their broad availability, there is still a relevant need of methods, preferably automatic, to provide 3D data with meaningful attributes that characterize and provide significance to the objects represented in 3D. Segmentation is the process of grouping point clouds into multiple homogeneous regions with similar properties whereas classification is the step that labels these regions. The main goal of this paper is to analyze the most popular methodologies and algorithms to segment and classify 3D point clouds. Strong and weak points of the different solutions presented in literature or implemented in commercial software will be listed and shortly explained. For some algorithms, the results of the segmentation and classification are shown using real examples at different scales in the Cultural Heritage field. Finally, open issues and research topics will be discussed.

- **A first attempt to group segmentation methods follows the works of Sapkota (2008) and Nguyen (2013) and a schematic representation is shown in Figure 2. 2.1 Edge-based segmentation As described by Rabbani et al. (2006), edge-based segmentation algorithms have two main stages: (i) edge detection to outline the borders of different regions and (2) grouping of points inside the boundaries to deliver the final segments. Edges in a given depth map are defined by the points where changes in the local surface properties exceed a given threshold. The mostly used local surface properties are normals, gradients, principal curvatures or higher order derivatives. Methods based on edge based segmentation techniques are reported by Bhanu et al. (1986), Sappa and Devy (2001), Wani and Arabnia (2003). Although such methods allow a fast segmentation, they may produce not accurate results in case of noise and uneven density of point clouds, situations that commonly occur in point cloud data. In 3D space, such methods often detect disconnected edges making the identification of closed segments difficult without a filling or interpretation procedure (Castillo et al., 2013).**

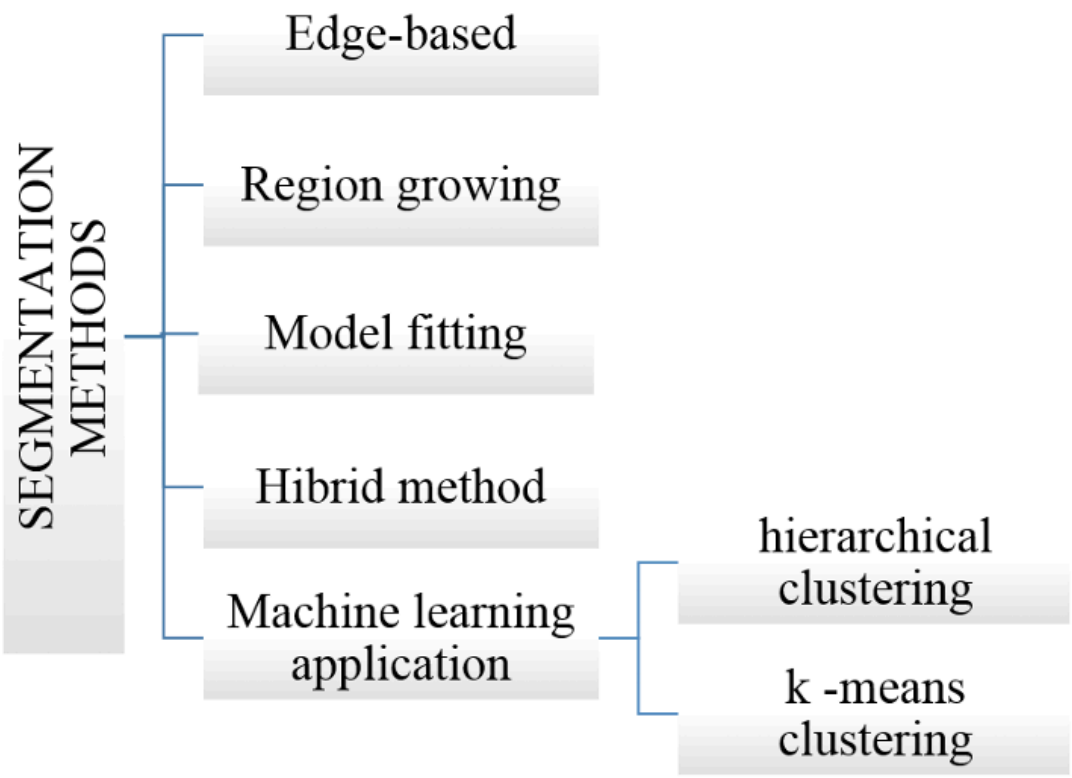


Figure 2: Synthetic representation of the segmentation methods.

-These methods start from one or more points (seed points) featuring specific characteristics and then grow around neighboring points with similar characteristics, such as surface orientation, curvature, etc. (Rabbani et al., 2006; Jagannathan and Miller, 2007). Region-based methods can be divided into:

Bottom-up approaches: they start from some seed points and grow the segments on the basis of given similarity criteria. Seeded region approaches are highly dependent on selected seed points. Inaccurate selection of seed points will affect the

segmentation process and can cause under- or over-segmentation results. Top-down approaches: they start by assigning all points to one group and then fit a single surface to it. Where and how to subdivide unseeded-region remain the main difficulty of these methods. Region-based algorithms includes two steps: identification of the seed points based on the curvature of each point and growing them based on predefined criteria such as proximity of points and planarity of surfaces. The initial algorithm was introduced by Besl et al. (1988) and then several variations were presented in the literature. The region growing method proposed by Vosselman et al. (2004) has introduced the use of color properties beside geometrical criteria. Surface normal and curvature constraints were widely used to find the smoothly connected areas (Klasing et al., 2009; Belton and Lichti, 2006) whereas Xiao et al. (2013) proposed to use sub window as the growth unit. Ackermann and Troisi (2010) used a region growing approach to segment planar pitched roofs in 3D point clouds for automatic 3D modeling of buildings. Anh-Vu Vo et al. (2015) presented an octree-based region growing approach for a fast surface patch segmentation of urban environment 3D point clouds. A collection of region growing algorithms is available in the Point Cloud Library (http://pointclouds.org). Figure 2 shows the results of a segmentation done by a region growing algorithm implemented

in the pcl::RegionGrowing class. The purpose of the algorithm is to merge/join similar points and deliver a set of clusters with points belonging to the same smooth surface.

-Segmentation based on hierarchical clustering :These methods compute representative measures (features) for each (3D) point, based e.g. on geometrical and radiometric characteristics: point position, locally estimated surface normals, residuals of best fitting surface procedures, points reflectance, etc. They usually create a hierarchical decomposition of a dataset by iteratively splitting the dataset into smaller subsets until each subset consists of only one object (Ng and Han, 1994). Xiaohu Lu et al. (2016) recently presented a novel hierarchical clustering algorithm which clusters any dimensional data and can be applied to mobile mapping, aerial and terrestrial point clouds.

-CLASSIFICATION: Once a point cloud has been segmented, each segment (group) of points can be labeled with a class thus to give some semantic to the segment (hence point cloud classification is often called semantic segmentation or point labeling). Point cloud (or mesh) classification is gaining interest and becoming a very active field of research (Weinmann et al., 2013; Guo et al., 2014; Niemeyer et al., 2014; Schmidt et al., 2014; Weinmann et al., 2014; Xu et al., 2014; Hackel et al., 2016). The class labeling procedure is normally achieved following three different approaches: a supervised approach, where semantic

categories are learned from a dataset of annotated data and the trained model is used to provide a semantic classification of the entire dataset. A large amount of annotated data is normally mandatory to train the model. an unsupervised approach, where the data is automatically partitioned into segments based on a user-provided parameterization of the algorithm. No annotations are requested but the outcome might not be aligned with the user's intention. an interactive approach, where the user is actively involved in the segmentation/classification loop by guiding the extraction of segments via feedback signals. This requires a large effort from the user side but it could adapt and improve the segmentation result based on the user's feedback

Book 13 :Enhancing Deep Learning-based BIM Element Classification via Data Augmentation and Semantic Segmentation

-A critical aspect of BIM is the capability to embody semantic information about its element constituents. To be interoperable, such information needs to conform to the Industry Foundation Classes (IFC) standards and protocols. Artificial intelligence approaches have been explored as a way to verify the semantic integrity of BIM to IFC mappings by learning the geometric features of individual BIM elements. The authors through

previous studies also investigated the use of geometric deep learning to automatically classify individual BIM element classes. However, such efforts were limited in the number of training data and restricted to subtypes of BIM elements. This study has significantly expanded the training set, to include a total of 46,746 elements representing 13 types of BIM elements. The magnitude of the data set is considered to be the first of this kind. Furthermore, Conditional Random Fields as Recurrent Neural Networks (CRF-RNN), a deep learning algorithm for semantic segmentation, was deployed to enhance the quality of individual input images. Deploying the dataset and segmentation improved the performance of previous model (Multi View CNN) by 4.37% and achieved an overall performance of 95.38%.

-Building Information Modeling (BIM) has become the mainstream medium for integrating and disseminating information during the project life cycle of construction projects. Multiple stakeholders today employ a variety of specialized BIM tools tailored to their various needs in the design, engineering, and construction management of their projects. The Industrial Foundation Classes (IFC), a neutral and open data format, is a critical component in ensuring interoperability within the BIM centric work process The absence of such a standard would require local data protocols

for each and every pair of applications, quickly making BIM based project execution intractable. Working under the IFC protocol requires BIM elements, relationships, and their properties to be represented in conformance to its standards. However, due to the lack of logical rigidity of the IFC schema, IFC model instances are prone to misrepresentations and misinterpretations, resulting in a lack of semantic integrity [4]. Such issues continue to be addressed in the domain of 'semantic enrichment', which reasons about the relations and meaning implicit in the geometry and topology of BIM models to check and rectify semantic inaccuracies. In particular, a subset of these studies investigated ways to check the correct mapping of individual BIM elements to their corresponding IFC entities [1-3, 12, 16]. For example, [12] formalized sets of inference rules to check mappings, and subsequently automate inaccurate associations. More recently, artificial intelligence approaches have been employed as an alternate approach to check the integrity of BIM-element to IFC-entity mappings. This approach has been conducted by extracting geometric features existing in the IFC or using 2D image or 3D shape information of each element for learning model training. [3] conducted a study to classify space in the BIM model using geometric features-based machine learning, and demonstrated that the classification accuracy of machine learning approach was superior to the inference rule-based approach. [11] used images extracted from

BIM models to classify building types, and [6] classified BIM furnishing elements using 2D CNN. The authors also explored the use of different machine learning and deep learning models to determine their applicability [7-9]. Mainly, we trained learning models based on the geometric features of individual BIM elements. In particular, we attained promising results from incorporating Multi-View CNN(MVCNN), a geometric deep learning model that learns from multiple panoramic images of a 3D artifact to learn and distinguish its shape [7].

Book 14 :MeshNet: Mesh Neural Network for 3D Shape Representation

-Meshis an important and powerful type of data for 3D shapes and widely studied in the field of computer vision and computer graphics. Regarding the task of 3D shape representation, there have been extensive research efforts concentrating on how to represent 3D shapes well using volumetric grid, multi-view and point cloud. However, there is little effort on using mesh data in recent years, due to the complexity and irregularity of mesh data. In this paper, we propose a mesh neural network, named MeshNet, to learn 3D shape representation from mesh data. In this method, face-unit and feature splitting are introduced, and

a general architecture with available and effective blocks are proposed. In this way, MeshNet is able to solve the complexity and irregularity problem of mesh and conduct 3D shape representation well. We have applied the proposed MeshNet method in the applications of 3D shape classification and retrieval. Experimental results and comparisons with the state-of-the-art methods demonstrate that the proposed MeshNet can achieve satisfying 3D shape classification and retrieval performance, which indicates the effectiveness of the proposed method on 3D shape representation.

- Three-dimensional (3D) shape representation is one of the most fundamental topics in the field of computer vision and computer graphics. In recent years, with the increasing applications in 3D shapes, extensive efforts (Wu et al. 2015; Chang et al. 2015) have been concentrated on 3D shape representation and proposed methods are successfully applied for different tasks, such as classification and retrieval. For 3D shapes, there are several popular types of data, including volumetric grid, multi-view, point cloud and mesh. With the success of deep learning methods in computer vision, many neural network methods have been introduced to conduct 3D shape representation using volumetric grid (Wu et al. 2015; Maturana and Scherer 2015), multi-view (Su et al. 2015) and point cloud (Qi et al. 2017a). PointNet (Qi et al. 2017a) proposes to learn on point cloud directly and solves the disorder problem

with per-point Multi-Layer Perceptron (MLP) and a symmetry function

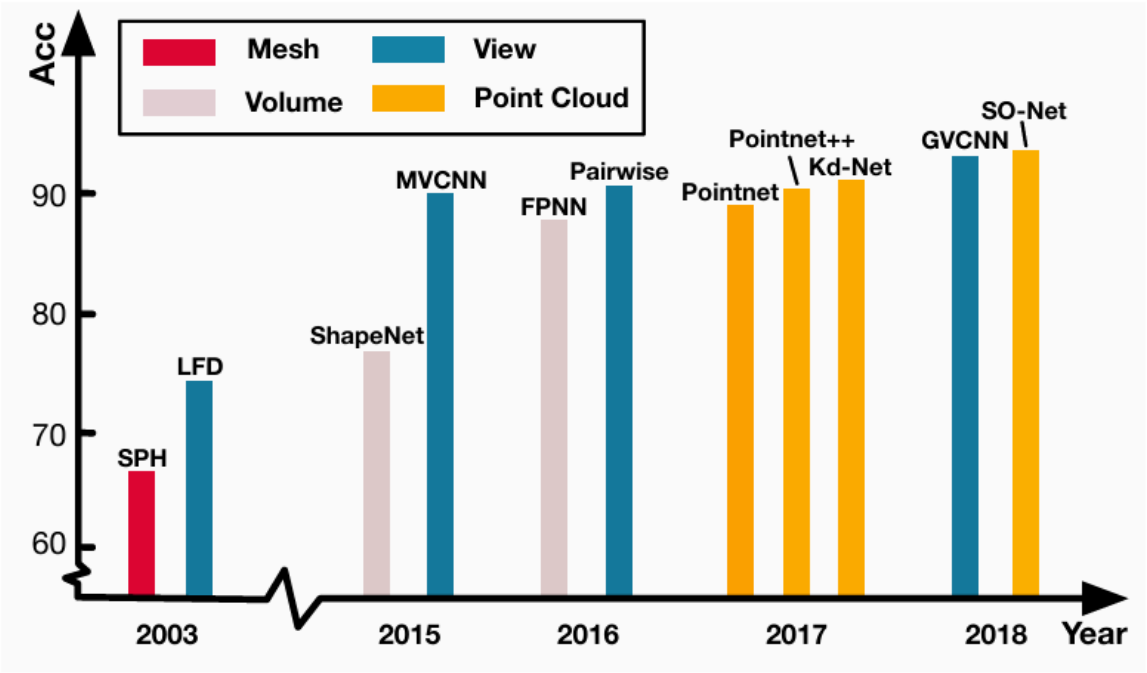


Figure 1: **The developing history of 3D shape representation using different types of data.** The X-axis indicates the proposed time of each method, and the Y-axis indicates the classification accuracy.

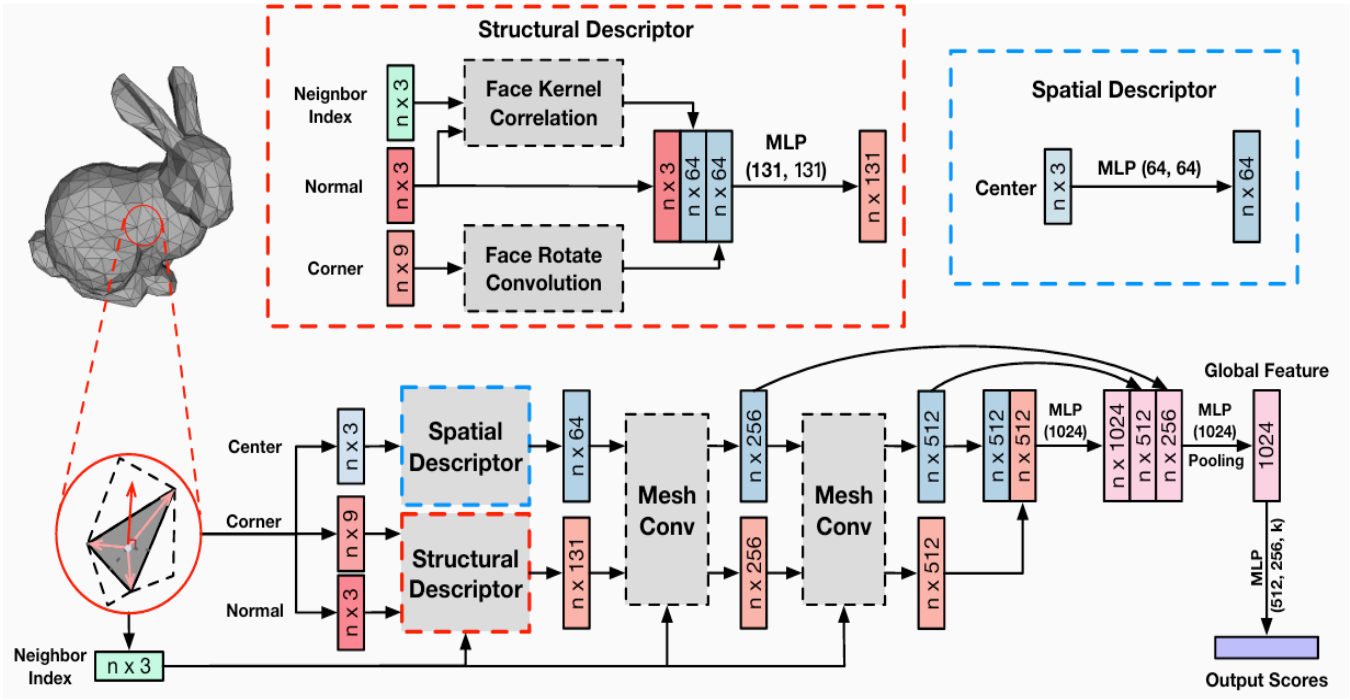


Figure 2: **The architecture of MeshNet.** The input is a list of faces with initial values, which are fed into the spatial and structural descriptors to generate initial spatial and structural features. The features are then aggregated with neighboring information in the mesh convolution blocks labeled as "Mesh Conv", and fed into a pooling function to output the global feature used for further tasks. Multi-layer-perceptron is labeled as "mlp" with the numbers in the parentheses indicating the dimension of the hidden layers and output layer.

- **Deep Learning Methods for 3D Shape Representation**

  - With the construction of large-scale 3D model datasets, numerous deep descriptors of 3D shapes are proposed. Based on different types of data, these methods can be categorized into four types.Voxel-based method. 3DShapeNets (Wu et al. 2015) and VoxNet (Maturana and Scherer 2015) propose to learn on volumetric grids, which partition the space into regular cubes. However, they

introduce extra computation cost due to the sparsity of data, which restricts them to be applied on more complex data. Field probing neural networks (FPNN) (Li et al. 2016), Vote3D (Wang and Posner 2015) and Octree-based convolutional neural network (OCNN) (Wang et al. 2017) address the sparsity problem, while they are still restricted with input getting larger. View-based method. Using 2D images of 3D shapes to represent them is proposed by Multi-view convolutional neural networks (MVCNN) (Su et al. 2015), which aggregates 2D views from a loop around the object and applies 2D deep learning framework to them. Group-view convolutional neural networks (CNN) (Feng et al. 2018) proposes a hierarchical framework, which divides views into different groups with different weights to generate a more discrimi native descriptor for a 3D shape. This type of method also expensively adds the computation cost and is hard to be applied for tasks in larger scenes.

Point-based method. Due to the irregularity of data, point cloud is not suitable for previous frameworks. PointNet (Qi et al. 2017b) solves this problem with per-point processes and a symmetry function, while it ignores the local informa tion of points. PointNet++ (Qi et al. 2017b) adds aggregation with neighbors to solve this problem. Self-organizing net work (SO-Net) (Li, Chen, and Lee 2018),

kernel correlation network (KCNet) (Shen et al.) and PointSIFT (Jiang, Wu, and Lu 2018) develop more detailed approaches for captur ing local structures with nearest neighbors. Kd-Net (Klokov and Lempitsky 2017)proposes another approach to solve the irregularity problem using k-d tree. Fusion method. These methods learn on multiple types of data and fusion the features of them together. FusionNet (Hegde and Zadeh 2016) uses the volumetric grid and multi view for classification. Point-view network (PVNet) (You et al. 2018) proposes the embedding attention fusion to exploit both point cloud data and multi-view data.

Method	Modality	Acc (%)	mAP (%)
3DShapeNets (Wu et al. 2015)	volume	77.3	49.2
VoxNet (Maturana and Scherer 2015)	volume	83.0	-
FPNN (Li et al. 2016)	volume	88.4	-
LFD (Chen et al. 2003)	view	75.5	40.9
MVCNN (Su et al. 2015)	view	90.1	79.5
Pairwise (Johns, Leutenegger, and Davison 2016)	view	90.7	-
PointNet (Qi et al. 2017a)	point	89.2	-
PointNet++ (Qi et al. 2017b)	point	90.7	-
Kd-Net (Klokov and Lempitsky 2017)	point	91.8	-
KCNet (Shen et al.)	point	91.0	-
SPH (Kazhdan, Funkhouser, and Rusinkiewicz 2003)	mesh	68.2	33.3
MeshNet	mesh	91.9	81.9

Table 1: Classification and retrieval results on ModelNet40.

Book 15 : PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Point cloud is an important type of geometric data structure. Due to its irregular format, most researchers transform such data to regular 3D voxel grids or collections of images. This, however, renders data unnecessarily voluminous and causes issues. In this paper, we design a novel type of neural network that directly consumes point clouds, which well respects the permutation invariance of points in the input. Our network, named PointNet, pro vides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing. Though simple, PointNet is highly efficient and effective. Empirically, it shows strong performance on par or even better than state of the art. Theoretically, we provide analysis towards understanding of what the network has learnt and why the network is robust with respect to input perturbation and corruption

- In this paper we explore deep learning architectures capable of reasoning about 3D geometric data such as point clouds or meshes. Typical convolutional architectures require highly regular input data formats, like those of image grids or 3D voxels, in order to perform weight sharing and other kernel

optimizations. Since point clouds or meshes are not in a regular format, most researchers typically transform such data to regular 3D voxel grids or collections of images (e.g, views) before feeding them to a deep net architecture. This data representation transformation, however, renders the resulting data unnecessarily voluminous — while also introducing quantization artifacts that can obscure natural invariances of the data. For this reason we focus on a different input representation for 3D geometry using simply point clouds– and name our resulting deep nets PointNets. Point clouds are simple and unified structures that avoid the combinatorial irregularities and complexities of meshes, and thus are easier to learn from. The PointNet, however still has to respect the fact that a point cloud is just a set of points and therefore invariant to permutations of its members, necessitating certain symmetrizations in the net computation. Further invariances to rigid motions also need to be considered. Our PointNet is a unified architecture that directly takes point clouds as input and outputs either class labels for the entire input or per point segment/part labels for each point of the input. The basic architecture of our network is surprisingly simple as in the initial stages each point is processed identically and independently. In the basic setting each point is represented by just its three coordinates (xy z). Additional dimensions may be added by computing normals and other local or global features.

Key to our approach is the use of a single symmetric function, max pooling. Effectively the network learns a set of optimization functions/criteria that select interesting or informative points of the point cloud and encode the reason for their selection. The final fully connected layers of the network aggregate these learnt optimal values into the global descriptor for the entire shape as mentioned above (shape classification) or are used to predict per point labels (shape segmentation). Our input format is easy to apply rigid or affine transformations to, as each point transforms independently. Thus we can add a data-dependent spatial transformer network that attempts to canonicalize the data before the PointNet processes them, so as to further improve the results.

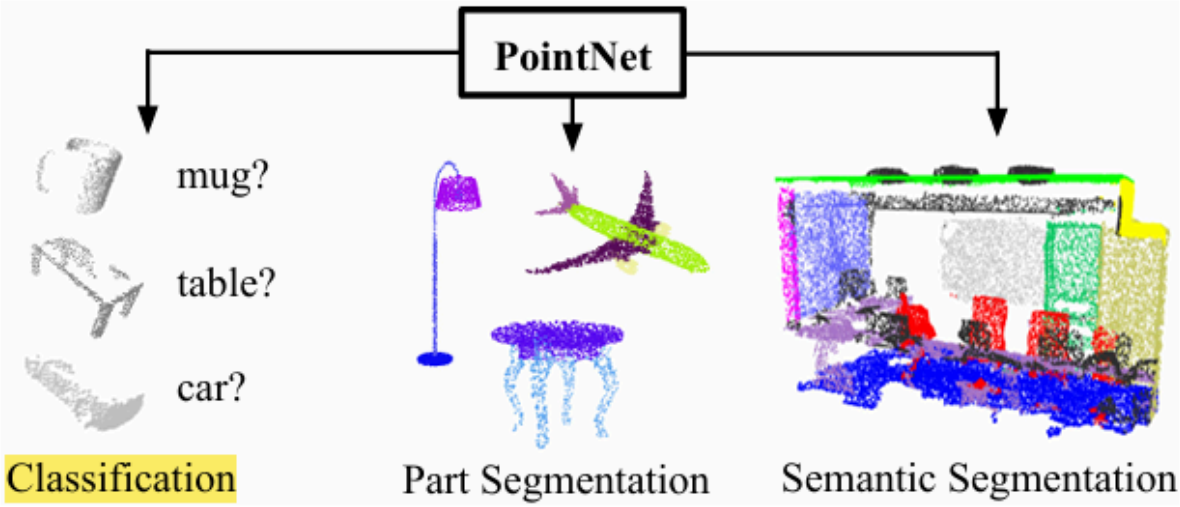


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

We provide both a theoretical analysis and an experimental evaluation of our approach. We show that our network can approximate any set function that is continuous. More interestingly, it turns out that our network learns to summarize an input point cloud by a sparse set of key points, which roughly corresponds to the skeleton of objects according to visualization. The theoretical analysis provides an understanding why our PointNet is highly robust to small perturbation of input points as well as to corruption through point insertion (outliers) or deletion (missing data). On a number of benchmark datasets ranging from shape

classification, part segmentation to scene segmentation, we experimentally compare our PointNet with state-of the-art approaches based upon multi-view and volumetric representations. Under a unified architecture, not only is our PointNet much faster in speed, but it also exhibits strong performance on par or even better than state of the art. The key contributions of our work are as follows: We design a novel deep net architecture suitable for consuming unordered point sets in 3D; We show how such a net can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks; We provide thorough empirical and theoretical analysis on the stability and efficiency of our method; We illustrate the 3D features computed by the selected neurons in the net and develop intuitive explanations for its performance. The problem of processing unordered sets by neural nets is a very general and fundamental problem– we expect that our ideas can be transferred to other domains as well.

-Point Cloud Features Most existing features for point cloud are handcrafted towards specific tasks. Point features often encode certain statistical properties of points and are designed to be invariant to certain transformations, which are typically classified as intrinsic [2, 24, 3] or extrinsic [20, 19, 14, 10, 5]. They can also be categorized as local features and global features. For a specific task, it is not trivial to find the optimal feature combination. DeepLearningon3DData

3Ddatahasmultiplepopular representations, leading to various approaches for learning. Volumetric CNNs: [28, 17, 18] are the pioneers applying 3D convolutional neural networks on voxelized shapes. However, volumetric representation is constrained by its resolution due to data sparsity and computation cost of 3D convolution. FPNN [13] and Vote3D [26] proposed special methods to deal with the sparsity problem; however, their operations are still on sparse volumes, it's challenging for them to process very large point clouds. Multiview CNNs: [23, 18] have tried to render 3D point cloud or shapes into 2D images and then apply 2D conv nets to classify them. With well engineered image CNNs, this line of methods have achieved dominating performance on shape classification and retrieval tasks [21]. However, it's nontrivial to extend them to scene understanding or other 3D tasks such as point classification and shape completion. Spectral CNNs: Some latest works [4, 16] use spectral CNNs on meshes. However, these methods are currently constrained on manifold meshes such as organic objects and it's not obvious how to extend them to non-isometric shapes such as furniture. Feature-based DNNs: [6, 8] firstly convert the 3D data into a vector, by extracting traditional shape features and then use a fully connected net to classify the shape. We think they are constrained by the representation power of the features extracted. DeepLearningonUnorderedSets Fromadatastructure point of

view, a point cloud is an unordered set of vectors. While most works in deep learning focus on regular input representations like sequences (in speech and language processing), images and volumes (video or 3D data), not much work has been done in deep learning on point sets. One recent work from Oriol Vinyals et al [25] looks into this problem. They use a read-process-write network with an attention mechanism to consume unordered input sets and show that their network has the ability to sort numbers. However, since their work focuses on generic sets and NLP applications, there lacks the role of geometry in the sets.

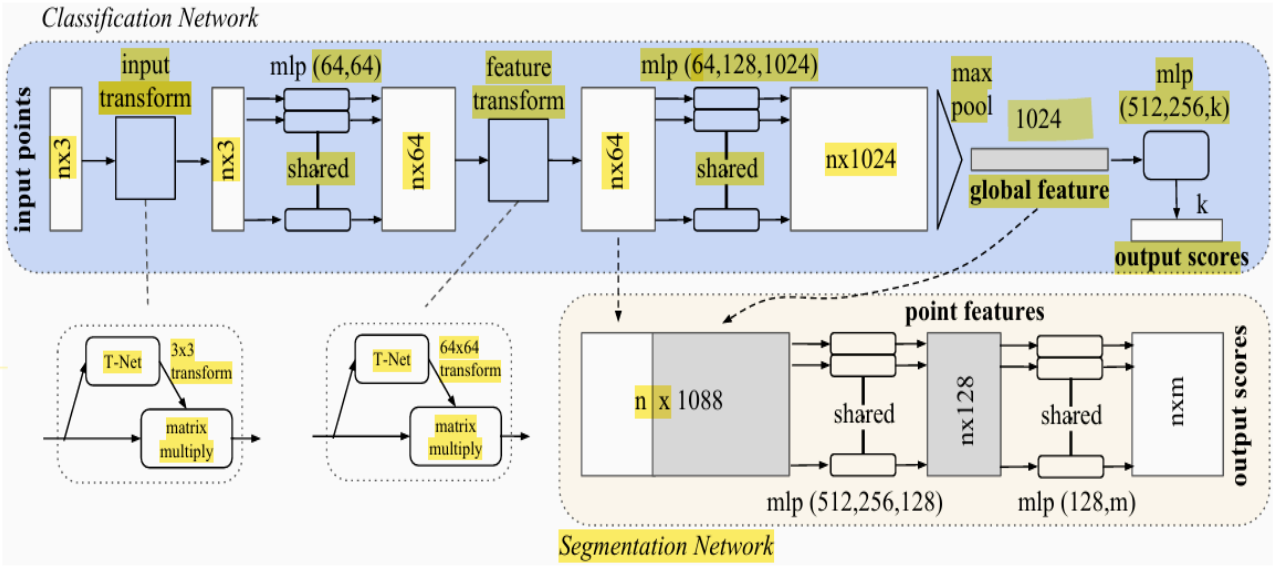


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

- **Properties of Point Sets in Rn Our input is a subset of points from an Euclidean space. It has three main properties: Unordered. Unlike pixel arrays in images or voxel arrays in volumetric grids, point cloud is a set of points without specific order. In other words, a network that consumes N 3D point sets needs to be invariant to N! permutations of the input set in data feeding order. Interaction among points. The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures. Invariance under transformations. As a geometric object, the learned representation of the point set should be invariant to certain transformations. For example, rotating and translating points all together should not modify the global point cloud category nor the segmentation of the points.**
- **Symmetry Function for Unordered Input In order to make a model invariant to input permutation, three strategies exist: 1) sort input into a canonical order; 2) treat the input as a sequence to train an RNN, but augment the training data by all kinds of permutations; 3) use a simple symmetric function to aggregate the information from each point. Here, a symmetric function takes n vectors as**

input and outputs a new vector that is invariant to the input order. For example, + and operators are symmetric binary functions.

Book 15 : Rule-Based Scan-to-BIM Mapping Pipeline in the Plumbing System

- The number of scan-to-BIM projects that convert scanned data into Building Information Modeling (BIM) for facility management applications in the Mechanical, Electrical and Plumbing (MEP) fields has been increasing. This conversion features an application purpose-oriented process, so the Scan-to-BIM work parameters to be applied vary in each project. Inevitably, a modeler manually adjusts the BIM modeling parameters according to the application purpose, and repeats the Scan-to-BIM process until the desired result is achieved. This repetitive manual process has adverse consequences for project productivity and quality. If the Scan-to-BIM process can be formalized based on predefined rules, the repetitive process in various cases can be automated by re-adjusting only the parameters. In addition, the predefined rule-based Scan-to-BIM pipeline can be stored and reused as a library. This study proposes a rule-based Scan-to-BIM Mapping

Pipeline to support application-oriented Scan-to-BIM process automation, variability and reusability. The application target of the proposed pipeline method is the plumbing system that occupies a large number of MEPs. The proposed method was implemented using an automatic generation algorithm, and its effectiveness was verified.

- Introduction Recently, for the purpose of facility maintenance, there are increasing cases of introducing reverse engineering technology, whichhasbeenmainlystudiedinthemanufacturingfield,into the construction field. Since many construction projects are placing orders for BIM, interest in Scan-to-BIM has naturally increased. Scan-to-BIM work has different work parameters depending on the purpose of the project. Inevitably, a BIM modeler manually adjusts the modeling parameters according to the application purpose and performs the Scan-to-BIM process. This process generates repetitive manual work according to the project, leading to work productivity and quality problems. If the Scan-to-BIMprocesscanbeformalizedandautomatedbasedonpredefinedrules,thisprocess can be conveniently reused depending on the purpose of the application. This study proposes a method of customizing and recycling the process by formalizing the Scan-to-BIM workflow according to the application purpose, and slightly changing the work parameter. To this end, this study proposes a rule-based Scan-to-BIM Mapping pipeline

processing method. The proposed method limits the scope of the plumbing system that occupies the majority in MEP .

- The concept of a pipeline refers to a structure or a set of data processing elements in which the output of the prior processing step is connected to the input of the next. These connected pipeline steps become a standardized process that can be easily replicated for other projects with only minor modifications in regular and formal parameters. Since multiple predefined pipelines can be executed collaboratively, performance can be much improved by processing a large amount of data in parallel. If a pipeline well determines a standardized input/output format, parameters, and logistics for execution steps, reusability rates can be improved by modifying the parameters of the predefined pipeline according to the purpose of use. This can be a meaningful way to increase productivity in Scan-to-BIM projects with many repetitive tasks. Figure 1 shows a conceptual diagram of the Scan-to-BIM Mapping Pipeline proposed in this study . It is important to analyze a Scan-to-BIM use-case scenario and define its processing steps in designing a pipeline structure. Through this, elements constituting a process can be mapped to the pipeline components. Each step in the process transforms the data through avarietyoftools, techniques, and manual interventions, and sends it to the next step. At this stage, a modeler determines steps for automation and analyzes what data

inputs/outputs and operation parameters are required for the pipeline structure. Data input/output types and operation parameters can be defined as generalizable rules. Scan-to-BIM rules can be stored in a library through predefined pipelines and parameters, allowing them to be reused in similar use-cases. Scenario analysis of Scan-to-BIM use-cases, process component definition, data input/output format, and parameter normalization studies is discussed in the following subsections

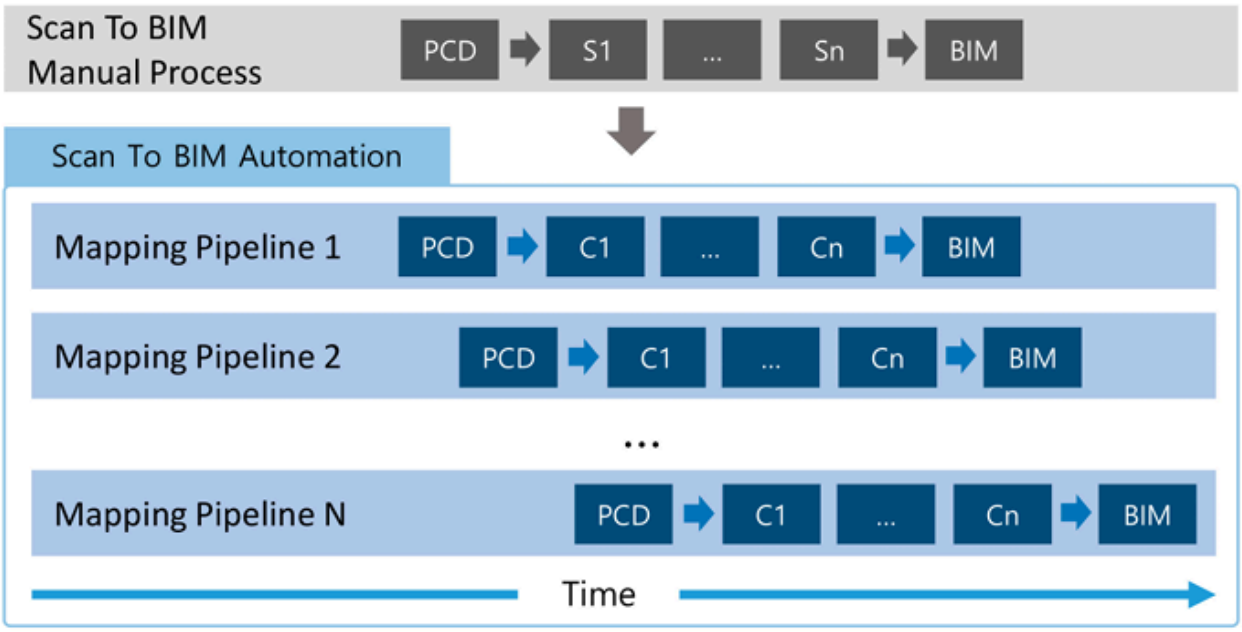


Figure 1. Scan-to-BIM Mapping Pipeline Concept Diagram.

-AScan-to-BIM project for MEP with Plumbing System was performed to validate the proposed rule-based Scan-to-BIM Mapping Pipeline method. The target of the scan is the MEP

facility of the K-Institute's Zero Energy Building. The number of scanned point clouds is 41,790,655, and the capacity is 1,504,463,580 bytes. For reference, the point cloud is composed of coordinate values, intensity, RGB values, etc. as follows.

PCD volume = PCDcount(pointsize + RGBsize)

-The main elements that compose the Scan-to-BIM process are as follows :

S1. Field trip and planning

S2. Target installation for matching

S3. Field scanning

S4. Data matching

S5. Filtering

S6. Segmentation

S7. Modeling

S8. Scan-to-BIM Quality Check Report

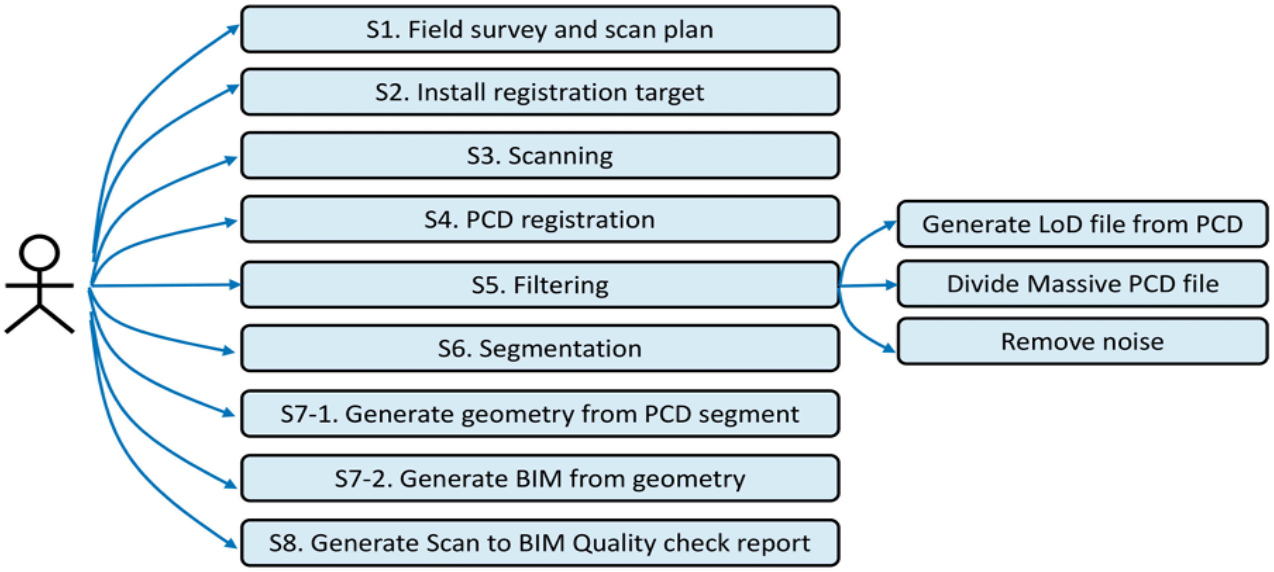


Figure 2. Use-case diagram.

- **Pipeline Component Design and Input/Output Data Normalization :**

  In order to design the structure of the Scan-to-BIM pipeline, it is necessary to analyze the input/output data items and types related to each step. This study formalized component and scan data input/output types based on the use-case scenario analyzed in Section 4.2. Figure 3 shows the relationship and input/output data items for each step of the Scan-to-BIM pipeline design.

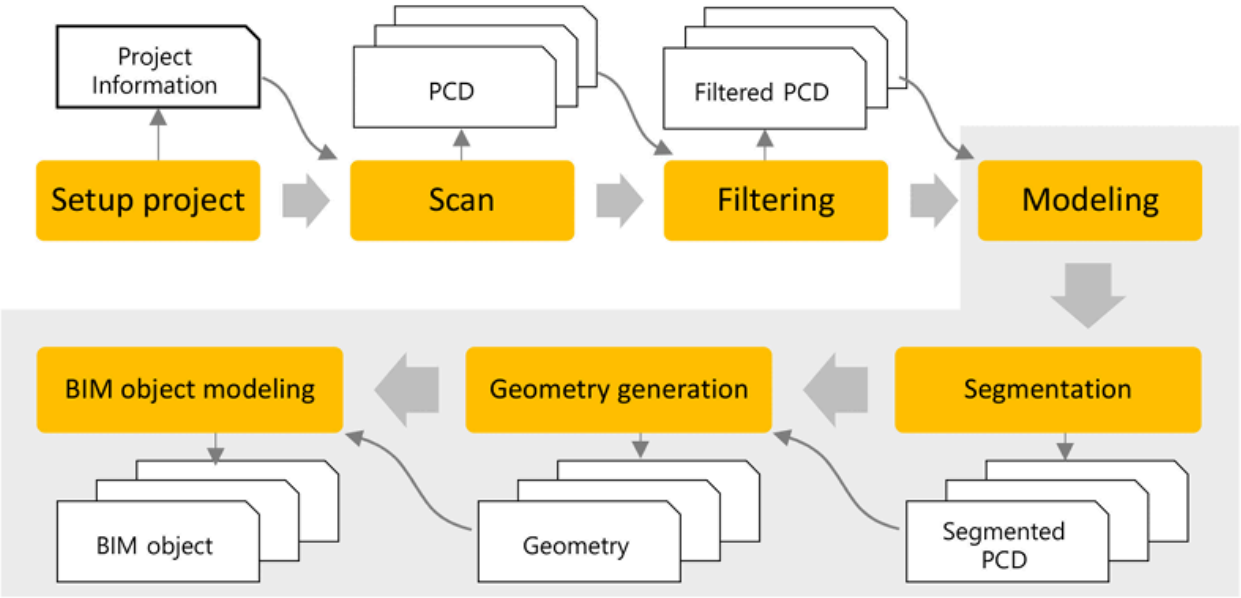


Figure 3. Scan-to-BIM Pipeline Component and Input/output Data.

- When the Scan-to-BIM use-case scenario is generalized, each step of the process can be divided into Scan-to-BIM project setting, scanning, filtering, modeling, segmentation, shape creation, and BIM object modeling steps. In every step, the scanned PCD is received and the PCD is processed until a BIM object is obtained. Each step has a specific type of data item and an input/output relationship. For example, in the point cloud filtering step, a PCD file is input based on a working parameter related to point cloud noise removal, and the filtered PCD file is generated. The geometry generation step yields geometry by receiving segmented PCD. Table 1 summarizes the pipeline components and input/output data derivation. Pipeline data

management becomes convenient if the connection relationship between pipeline components and input/output data items can be structured. By grouping input/output data items related to components, independent reuse is possible for each component. Taking this into account, the relationship between component and data item is structured in a hierarchy as shown in Figure 4

Table 1. Scan-to-BIM pipeline component and input/output data derivation (* = multiple).

Scan-to-BIM Pipeline Component	Data Item	Data Format
C1. Setup project	D1. Project information file	{name, description}
C2. Scan	D2. PCD file	{x, y, z, I, RGB} *
C3. Grid generation	C3. PCD file	{grid.ID, x, y, z, I, RGB} *
C4. Level of Detail (LoD)	D4. PCD file	PCD files
C5. Filtering	D5. Filtered PCD file	PCD files
C6. Segmentation	D6. Segmented PCD	{segment.ID, x, y, z, I, RGB}
C7. Geometry generation	D7. Geometry data	{geometry.ID, type, dimension *} Dimension = {name, value}
C8. BIM object generation	D8. BIM object data	{BIM.object.ID, type, dimension *, property*} Property = {name, value}

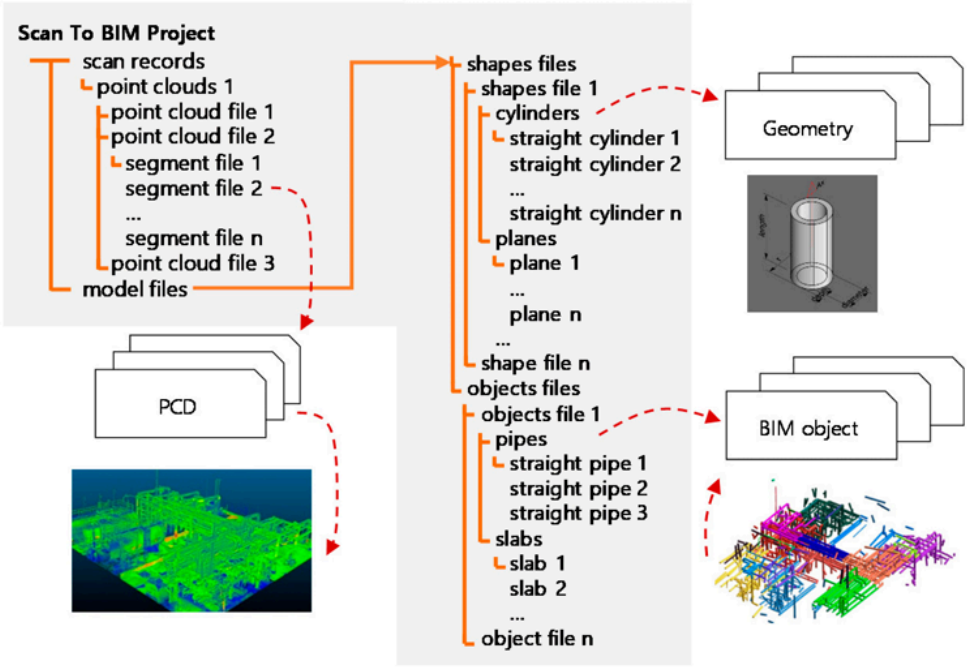


Figure 4. Scan-to-BIM Pipeline Component Dataset Hierarchy Structure Definition.

- Pipeline Component Parameter and Rule Design:

Scan-to-BIM Component Algorithm and Parameters To design Scan-to-BIM component operation parameters and rules, this section investigates the algorithms and parameters of each component in Table 1. If the working parameters can be extracted and ordered, the predefined Scan-to-BIM pipeline can be replicated with only a few parameter modifications. The parameter value must be able to be determined differently according to the purpose of Scan-to-BIM. In other words, Scan-to-BIM configuration with parameter setting is purpose-oriented. AsshowninFigure 5, the modeler sets parameters to suit the purpose or reuses them in the parameters library. This is because the dimensions and properties of the modeled BIM object are different depending on the use case.

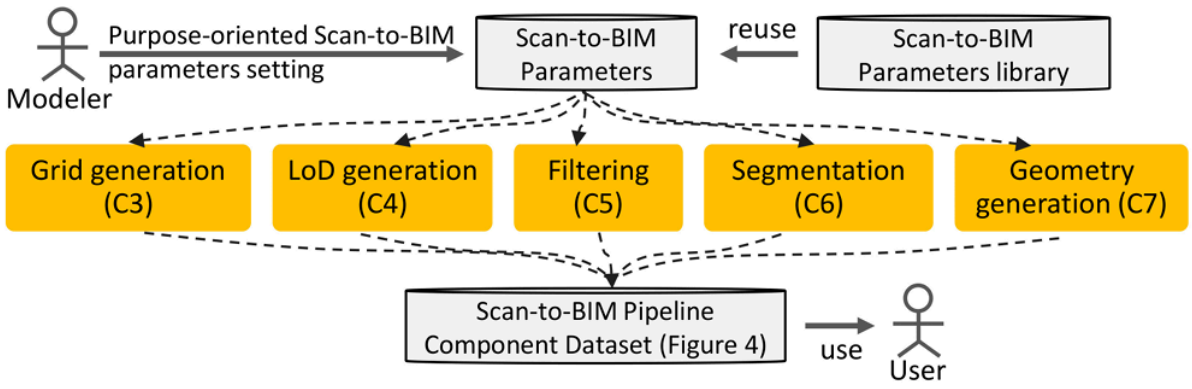


Figure 5. Purpose-oriented Scan-to-BIM Parameters Setting related to Pipeline Component.

-Grid Generation(C3), LoD(C4) Parameters

If a large amount of point cloud data is used for a Scan-to-BIM project, a processing problem on a computer arises. In general, the data size obtained by matching point groups obtained from LiDARis tens of millions to more than billions of points. Not all projects need this high-precision point cloud. In general, point clouds are used by converting precision according to the purpose. Even after converting the LoD of the point cloud to a low level, if the computer is not capable of processing the data, the point cloud should be divided into regions. Grids are used for segmentation. The grid size can be defined by specifying the x, y, and z axis spacing in the 3D coordinate system

parameter grid = {size x, size y, size z}

LoD processing transforms the input point group with a density equal to or lower than the specified number of points.

LoD can be expressed as the number of points in a specific area like resolution. LoD processing generally uses a quadtree, octree spatial indexing technique to divide a space and calculates the average point and center point of the point group of the divided area. The related parameters are as follows

parameterLoD = {size, point count}

- **Filtering(C5) Parameters :**

  When Unwanted areas or point clouds are acquired during the scan step, the process of removing these noise data directly affects the quality of the subsequent work. Automatic filtering is a method to remove the region after determining the region to be removed using a clustering algorithm such as k-NN (k-nearest neighbor). When a small number of point groups are separated by more than a certain distance from the dense point group, or when point group data outside the region of interest are acquired, these point groups will be removed. Equations (4)–(6) express parameters for these point groups.

$$parameter_{filter} = \left\{filter_{k-NN},\ filter_{bounding\_box}\right\}$$

$$filter_{k-NN} = \left\{distance,\ point_{\min\_count}\right\}$$

$$filter_{bounding\_box} = \left\{inner \,|\, outer,\ x1,\ y1,\ z1,\ x2,\ y2,\ z2\right\}$$

- **Segmentation(C6), Geometry Generation(C7) Parameters:**

Segmentation is processed in two stages. The first step is to perform segmentation based on the point where the curvature is sharply divided, and the second step is to calculate a point group with parameters that fit the pipe shape. The segmentation uses a region growing algorithm based on curvature similarity. This method employs the local surface normal and connectivity of the point group as shown in the following equation.

$$||n0n1|| > \cos(th)$$

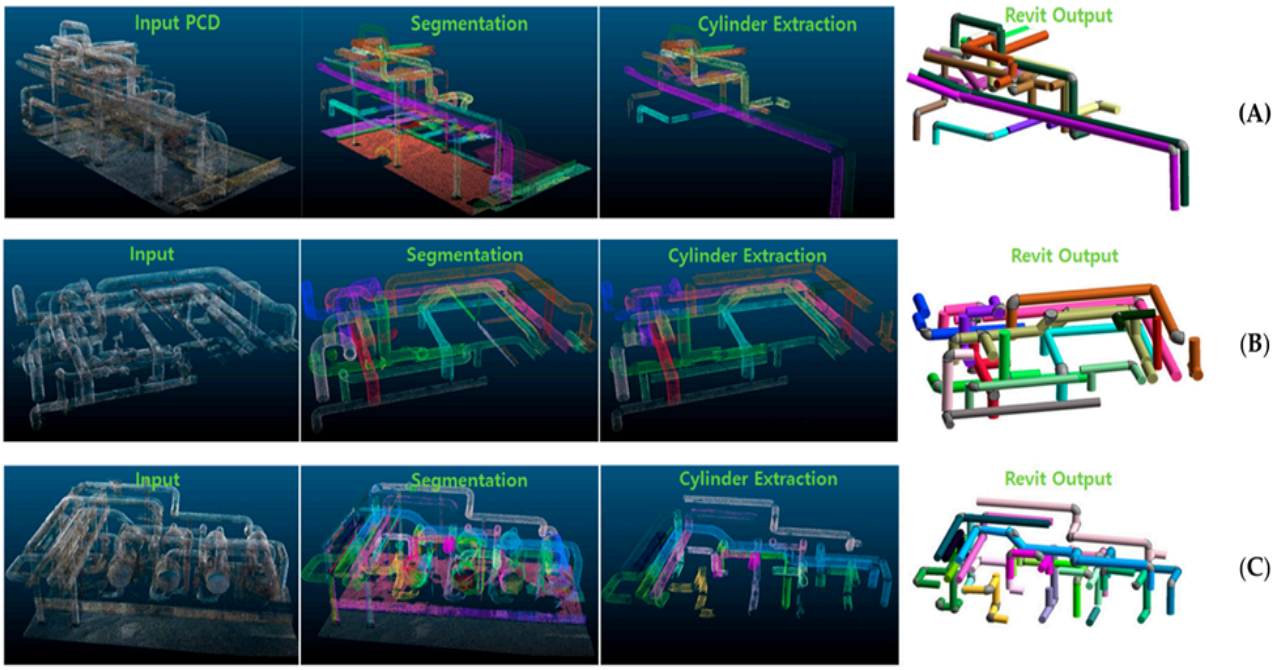


Figure 10. Scan-to-BIM Mapping Pipeline Results (**A**,**B**,**C** cases).

Book 16 : Automated Semantic Labeling of 3D Vector Models for Scan-to-BIM

-With the increasing popularity of Building Information modeling (BIM), the demand for accurate as-built models of existing buildings is rising. However, the manual creation of these models is labor intensive and error prone. Therefore, automation of the process is a must. One of the key factors in the automated Scan-to-BIM process is the labeling of the data for further reconstruction. Currently, semantic labeling is still ongoing research. This paper presents a flexible method to automatically label highly cluttered vector models of existing buildings. In our proposed method, a reasoning framework is used that exploits geometric and contextual information. A major advantage to our approach is that our algorithm can label both cluttered environments and large data sets very efficiently. Unlike other solutions, this allows us to label entire buildings at once. In addition, the implementation of our algorithm and the platform we use allows for flexible data processing, visualization of the results and improvement of the labeling process. Our work covers the entire labeling phase and allows the user to label data sets with a minimal amount of effort.

- The popularity of intelligent three dimensional data models like Building Information Modelling (BIM) is rapidly increasing. Most commonly, these models are created during the designing phase of a structure to support the construction process. Afterwards, stakeholders can employ the BIM model for a wide variety of applications such as facility management, energy performance analysis, project planning, etc [1], [2], [3]. However, the BIM model created during the designing phase often deviates from the actual conditions. Therefore, the need exists for as-built BIM models, where the as-design models are updated to as-built conditions. Experiencing the advantages of BIM, the industry now looks to implement as-built models for existing buildings. With no prior BIM available, these models have to be created from scratch. In general, as-built models are created from point clouds. Commonly, a terrestrial laser scanner is employed to acquire scans from different locations. The individual data sets are then aligned using cloud-based or target-based registration techniques [4]. Once a complete point cloud is acquired, the data is exported to a modeling software and BIM objects are fitted onto the points. This process is titled

Scan-to-BIM. However, the creation of such a model is labour intensive and time consuming. One of the main problems is the amount of manual labour required in the modelling process. Therefore, the industry would greatly benefit from automation

in the Scan-to-BIM process [5], [6], [7]. Our goal is to develop methods to automate this workflow.

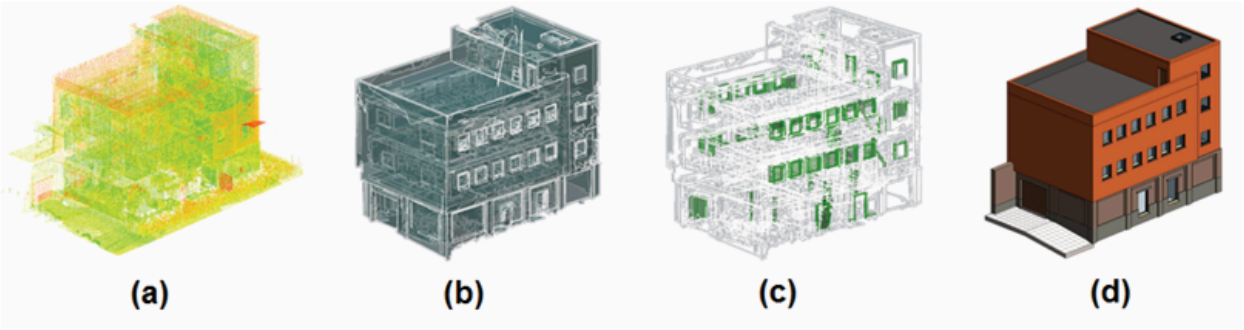


Fig. 1. Overview of intermediate results of the Scan-To-BIM Workflow: From left to right, Point Cloud (a), Vector model (b), labelled Vector model (c) and as-built BIM model (d).

- **Modelling Process:**

  The Scan-to-BIM process can be divided into several phases [8]. Some intermediate results of the different stages are shown in Fig. 1. First, the point cloud is segmented into different clusters. Many solutions has been presented to tackle this problem [9], [10], [11]. Once the clusters are identified, primitives or meshes are fitted onto the data. While complex surfaces provide a more accurate approximation of the as-built conditions, planar surfaces are generally employed to model the structural elements of buildings. Commercial software like Pointfuse [12] already

provide fully automated plane reconstruction. In the robotics industry, vector models are often considered as the deliverable of the reconstruction process [13], [14], [15]. However, in the case of Scan-to-BIM, intelligence should be added to the model. This phase is titled ”Semantic Labeling”. The vector model is processed by reasoning frameworks that provide the individual surfaces with labels such as walls, floors, ceilings, etc. Several ap-proaches have been presented using heuristic or probabilistic techniques [16], [17], [18]. However, most of these solutions only handle small scale data e.g. a single room. Our method provides a reasoning framework that will work on any scale of data, even entire buildings. Considering complete structures allows us to consider some contextual information that is not available in smaller data sets. Following the semantic labeling, the individual surfaces can be grouped and used as a basis to reconstruct the as-built BIM.

-Existing buildings:

The focus of this research is on the reconstruction of existing buildings e.g. hospitals, office buildings, schools, houses etc. It is important to know that the data sets of these structures have varying properties. Some of the key factors for the semantic labelling are: a) Noise: The goal of

our research is provide an algo rithm that works for realistic data. This means that furniture and clutter (such as small objects, persons, etc) will be present in the data sets. This introduces a high degree of confusion in labelling since some furniture can show great resemblance to structural elements. E.g. a built-in closet can be mistaken for a wall or several adjacent tables can be seen as a floor. b) Varying zones: Typical for real structures is the wide variety of zones inside the building. They contain staircases, attics or other unusual spaces. The characteristics of these zones deviate from regular rooms which causes problems in the labeling process. c) Type of data set: Depending on project deliverables and methodology, the type of point cloud data differs. Some data sets only contain a portion of a structure while others consist out of the entire building and the surroundings. Common examples of different types of data sets are multi storey and single-storey: The former represents a data set that contains the entire structure (both interior and exterior across all floors). The latter only contains a part of the building, consisting of one floor. The use of single-storey data sets is a common strategy in the Scan-to-BIM industry since treating each floor as a separate project has computational advantages. However, the data characteristics of both types are inherently

different. For instance, single-storey data sets do not contain some of the contextual information present in multi-storey data sets. This proves problematic for reasoning algorithms. E.g. a potential ceiling cannot not have a floor above in a single storey data set. Therefore, filters employing this information are meaningless. Even worse, some filters even have a counter-productive impact on the labeling if they expect information that is not present in the data.

- Automation:

Traditional, manual modeling relies on the user to interpret the point cloud. In this case, the segmentation of the data, the labeling and the primitive fitting is purely done visually. It is up to the operator to identify the type of object and where to place it. Automation of the Scan-to-BIM process looks to aid the user in the different phases. Two approaches are currently being developed: Fully Automated Reconstruction and Assisted Manual Reconstruction. The emphasis of the former is on the creation of an initial proposal of the objects, effectively emoving all user input. Commercial software such as Edge wise [19] focus on this strategy. On the other hand, the latter strives to provide the user with a set of tools that facilitate the modelling process. Software pursuing this course are FARO Pointsense [20] and Leica Cloudworxs [21]. While the Fully

Automated Reconstruction process is faster, it is limited to generic objects, and thus can only provide an initial solution. The Assisted Manual Reconstruction can aid the operator to a further extent but requires additional labour. We believe a promising solution is to merge the two approaches. For example, automated algorithms could provide an initial proposal during each stage of the Scan-to-BIM process. In addition, the user would be provided with a set of tools to easily update or modify the proposals before continuing. The goal of our research is to automatically provide the user with such an initial BIM model on a flexible adjustment platform. This article describes our recent work on the automation of the Scan-to-BIM process. More specifically, we address the problem of semantic labeling. Our method provides a reasoning framework that labels elements that frequently appear in typical existing buildings. It identifies floors, ceilings, roofs, walls, windows and doors. The output of our algorithm is a labeled data set that can be easily adjusted by the user for further reconstruction.

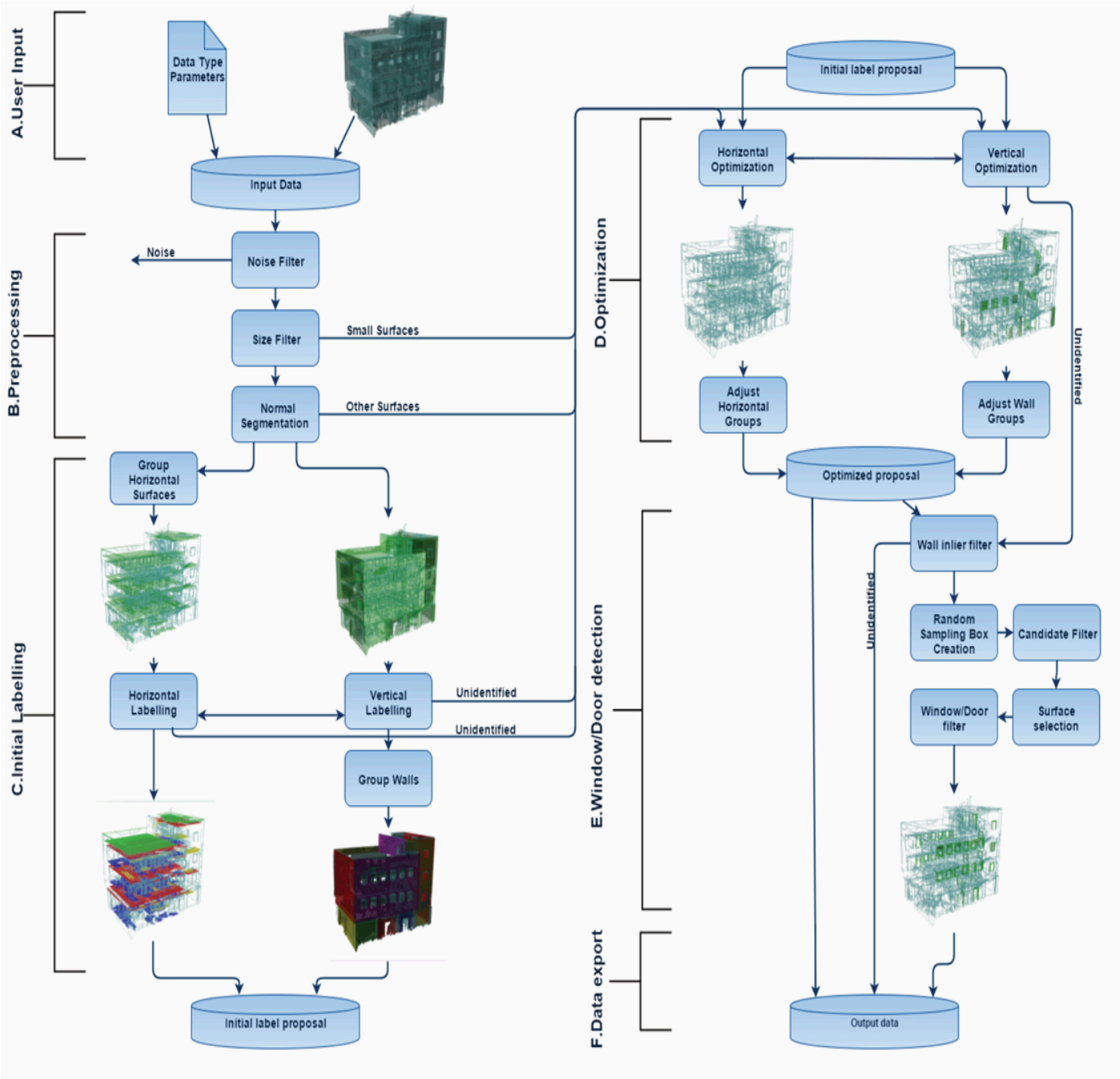


Fig. 2. Workflow diagram of our algorithm. From top left to bottom right the following steps are depicted: User Input (A), Preprocessing (B), Initial labeling (C), Optimization (D), Window/Door detection (E) and Data Export (F).

-Semantic labeling is being investigated from multiple points of view. A lot of work is performed in the area of computer vision, where labeling is often utilized for object recognition. One of the popular approaches is the use of neural networks such as Conditional Random Fields [22], [23]. By connecting several nodes into a graph, probabilistic reasoning allows likelihood maximization of the different labels for the nodes. Researchers employing this approach have published promising results for small scale building scenes [5], [16], [17], [24], [25], [26]. However, these reasoning algorithms require extensive learning and are computationally expensive. Other approaches present algorithms that employ geometric and contextual rules [18], [27], [28], [29]. Typically, they encode ”features”, which represents a characteristic of the candidate object. These features are used to specify a set of rules. E.g. a candidate surface with a large area will have a higher probability of being a structural element. Geometric features found in publications contain normals, dimensions, distance to bounding box, area, height, point density, aspect ratio, etc. Contextual features can consist of associative or non associative relationships [30]. Common features employed are coplanarity, convexity, proximity, geometric similarity, topol-ogy, texture similarity, etc. Most works only employ some of these rules to identify the zone specific elements of the data set. Our context-based labelling algorithm extrapolates these rules to full scale buildings.

Furthermore, different sets of weights for the rules are predefined for varying zones so that the user no longer has to train the algorithm. The input data differs between varying approaches. Some researchers directly segment the point cloud [13] while others prefer to work with primitives such as surfaces. Although working directly on the point clouds can be more accurate, it also introduces a higher computational cost and uncertainty in the process. In our work, we use primitives as a basis because of the computational advantages. Some research has been performed on the automated reconstruction of rooms based on labeling [31], [32]. The graph based techniques in these works provide promising labeling results, but requires a watertight mesh. However, the creation of such a mesh in a highly cluttered and occluded environment is challenging. Another point of view is the use of prior knowledge. [33], [34] employ contextual information on existing plans to extract building elements. However, these plans often lack consistency making them hard to interpret. Semantic labeling has also been a major topic in outdoor scenes. Facade and window opening detection algorithms have been successfully used on mobile mapping data [35]. In aerial applications, building extraction has seen major breakthroughs in the past several years [29], [36], [37], [38], [39]. However, these algorithms only employ exterior information. Our algorithm

exploits both indoor and outdoor data to find accurate walls, roofs and openings.

- In this paper we consider flexible labeling of planar surfaces for the creation of BIM models for existing buildings. Our approach consists of a weighted reasoning framework that employs geometric and contextual rules to label the data set. Our algorithm is implemented in Grasshopper, an open source Rhinoceros plug-in which is a platform for visual scripting of object based program languages. The algorithm pipeline shown in Fig. 2 consists of six steps. First, before the actual processing, the user chooses a set of parameters in respect to the data. Second, preprocessing of the data is performed for noise reduction and segmentation. Third, the initial labeling is computed. Fourth, an optimization step is performed to enhance the labeling. The fifth step allows the operator to automatically find window and door surfaces. Finally, the sixth step exports the labeled surfaces to Rhinoceros, where the data can be updated by the user. As an input, our algorithm accepts any set of flattened meshes. This geometry can be derived from multiple reconstruction software. The example input meshes used in this paper are produced by Pointfuse [12]. This software calculates flat triangular meshes to represent the planes in the point cloud.

- **User Input:**

The platform we currently employ is Rhinoceros. The input meshes are loaded into the software and visualized. Before any calculations are performed, the user has the opportunity to evaluate the data and choose appropriate settings in the Grasshopper interface: First, the user defines which part of the data should be processed. Second, the user indicates which type of data the algorithm should be expecting (E.g. multi storey). Varying types of data have different preset parameters for the contextual and geometric filters. These parameters are previously trained by example data. An advantage of our approach is that our algorithm allows flexible zone and parameter set selection. While an entire project can be loaded into the algorithm at once, it is possible to select only a part of the data at a time and feed it to the labeling algorithm with a specific set of parameters. This way, the user maintains control of the labeling process. Also, the user can easily create and train new sets of parameters for other types of data

-Preprocessing:

At the beginning of the preprocessing step, the data is im ported from Rhinoceros into the Grasshopper plug-in. There, planar surfaces are extracted from the meshes based on the exterior boundaries of the meshes. The advantage of the planar surface

representation is that the same geometric data can be handled more efficiently. Following, two dimension filters are applied: One for noise reduction and another to segment the large from the small surfaces. By splitting the data set by size, we allow for course to fine labeling. The noise dimension threshold is set to 0.7m for the major axis. The segmentation filter threshold considers surfaces to be large if both axes are larger than 1m or the major axis exceeds 2.5m. Next, the large surfaces are divided by their normal with an angular threshold of 15◦into Horizontal Surfaces and Vertical Surfaces. Surfaces with a normal between 15◦and 85◦are considered as Other Surfaces. These surfaces are not labeled because of their unpredictable characteristics and low occurrence (less than 1% of data set)

TABLE I
GEOMETRIC AND CONTEXTUAL FEATURES.

Labels	Geometric features	Contextual features
Floor Ceiling Roof Furniture	Area Dimensions Boundary Normal	Wall proximity Wall height similarity Horizontal proximity Extrusion collision Room edge proximity
Wall	Area Dimensions Boundary Normal	Ceiling proximity Floor proximity Coplanarity Perpendicularity Wall proximity Room edge proximity
Window Door	Diagonal length Diagonal angle Dimensions	Floor proximity Wall inlier
Threshold	**0.70**	**0.70**

- **Window/Door detection :**

After the optimization step, our algorithm employs the wall information as a basis to find windows and doors. First, a spatial filter is applied on the remaining data to isolate the surfaces that are located inside the walls. For single-faced walls, a local search area is defined to locate nearby surfaces. The data extracted from both filters serve as the candidate surfaces for the window and door detection step. Second, conditional random sampling is applied to locate candidate bounding boxes per wall. As

minimal sample, one vertical and one horizontal surface is iteratively selected at random from each wall to create a bounding box. The process is conditioned so that only boxes are created that have similar dimensions as the initial samples. After the candidates are created, a reasoning framework is applied using length of diagonal, angle of diagonal, similar centroid occurrences and surface inliers to filter the bounding boxes. Surfaces having more than 30% overlap with a filtered bounding box are withheld and grouped. Finally, windows are separated from doors based on dimensions and floor information.

- In this paper we explained and demonstrated a flexible automated labeling framework for existing buildings for the Scan-to-BIM pipeline. In our proposed method, a reasoning framework is employed that exploits geometric and contextual information. A major advantage to our approach is that our algorithm can label both cluttered environments and large data sets very efficiently. Unlike other solutions, this allows us to label entire buildings at once. Furthermore, the implementation of our algorithm and the platform we use allows for flexible data processing, visualization of the results and improvement of the labeling process. Our work covers the entire labeling

phase and allows the user to complete the labeling with a minimal amount of effort. Overall, the experimental data showed that our algorithm is able to label the general structural elements in common areas with high precision and recall values. However, in more complex areas such as staircases, the features of the surfaces differ from the features in common areas. This proves problematic for the surface labeling. Also, elements that consist of small parts are harder to detect because their surface features resemble features of noise surfaces. For these types of data our algorithm will underperform. Currently our approach relies on user input for zone and parameter selection. Work is being performed on the automation of this input, further reducing the users effort. Currently, the parameters in the algorithm are focussed on the maximization of precision instead of recall. However, our approach would benefit from recall maximization because of the flexible adjustment platform. It is easier for the user to remove false positives than to add false negatives. In order to maximize recall, the contribution of each parameter should be known. Therefore, further work is focussed on key parameter identification and recall optimization using machine learning techniques.

TABLE III
INDIVIDUAL LABEL PERFORMANCE OF "HOUSE" DATA SET.

	Noise	Floor	Ceiling	Roof	Wall	Window	Door	Unidentified	**Overall**
Total Surfaces	999	14	17	9	98	95	84	673	**1989**
Total Elements	-	5	8	3	50	7	12	-	**85**
Surface Precision [%]	-	100	93.3	100	90.6	70.2	63.6	-	**86.3**
Element Precision [%]	-	100	87.5	100	88	55.6	50.0	-	**79.7**
Surface Recall [%]	-	85.7	88.2	100	76.5	83.6	13.1	-	**74.5**
Element Recall [%]	-	80	77.8	100	100	81.3	16.7	-	**76.0**
Computation time [s]	1.6	1.7			6.4	10.3		-	**20.0**

Book 17 : Contextually Guided Semantic Labeling and Search for 3D Point Clouds

RGB-D cameras, which give an RGB image to gether with depths, are becoming increasingly popular for robotic perception. In this paper, we address the task of detecting commonly found objects in the 3D point cloud of indoor scenes obtained from such cameras. Our method uses a graphical model that captures various features and contextual relations, including the local visual appearance and shape cues, object co-occurrence relationships and geometric relationships. With a large number of object classes and relations, the model's parsimony becomes important and we address that by using

multiple types of edge potentials. We train the model using a maximum-margin learning approach. In our experiments over a total of 52 3D scenes of homes and offices (composed from about 550 views), we get a performance of 84.06% and 73.38% in labeling office and home scenes respectively for 17 object classes each. We also present a method for a robot to search for an object using the learned model and the contextual information available from the current labelings of the scene. We applied this algorithm successfully on a mobile robot for the task of finding 12 object classes in 10 different offices and achieved a precision of 97.56% with 78.43% recall

- Inexpensive RGB-D sensors that augment an RGB image with depth data have recently become widely available. These cameras are increasingly becoming the de-facto stan dard for perception for many robots. At the same time, years of research on SLAM (Simultaneous Localization and Mapping) has now made it possible to merge multiple RGB D images into a single point cloud, easily providing an approximate 3D model of a complete indoor scene (i.e., a room). In this paper, we explore how this move from part-of scene 2D images to full-scene 3D point clouds can improve the richness of models for object labeling. In the past, a significant amount of work has been done in semantic labeling of 2D images (Murphy et al., 2003; Hoiem et al., 2006; Heitz and Koller, 2008; Felzenszwalb et al., 2008; Collet et al., 2011). However, a lot of valu able information

about the 3D shape and geometric layout of objects is lost when a 2D image is formed from the corresponding 3D world. A classifier that has access to a full 3D model can access important geometric properties in addition to the local shape and appearance of an object. For example, many objects occur in characteristic relative geometric configurations (e.g., a monitor is almost always on a table), and many objects consist of visually distinct parts that occur in a certain relative configuration. More generally, a 3D model makes it possible to reason about a variety of 3D properties such as 3D distances, volume and local convexity. Some recent works (Hoiem et al., 2006; Divvala et al., 2009) attempt to first infer the geometric layout from 2D images for improving object detection. However, the inferred geometric layout is not accurate enough to give significant improvement. Other recent work (Xiong and Huber, 2010) considers labeling a scene using a single 3D view (i.e., a 2.5D representation). In our work, we first use SLAM in order to compose multiple views from a Microsoft Kinect RGB D sensor together into one 3D point cloud, providing each RGBpixel with an absolute 3D location in the scene. We then (over-)segment the scene and predict semantic labels for each segment (see Fig. 2). We not only predict coarse classes like in (Xiong and Huber, 2010; Anguelov et al., 2005) (i.e., wall, ground, ceiling, building), but also label individual objects (e.g., printer, keyboard,

monitor). Furthermore, we model rich relational information beyond an associative coupling of labels (Anguelov et al., 2005).

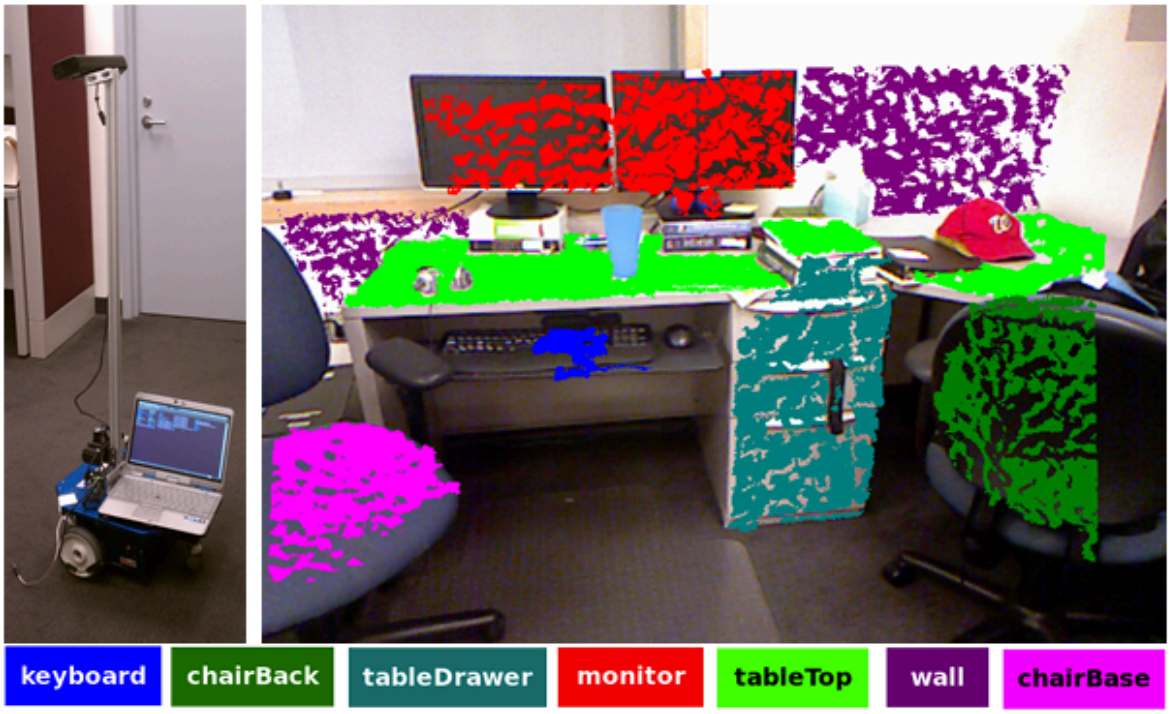


Fig. 1. (Left) Cornell's Blue robot mounted with an RGB-D camera (Microsoft Kinect). (Right) Predicted labeling of a scene.

Book 18 : Semantic Segmentation of 3D point Clouds

---Deep Learning for 3D point clouds :

- **Capturing 3D worlds**

-3D data is crucial for robotics, autonomous vehicles, 3D scale models, virtual reality etc... Can be computed from images: stereo, SfM, SLAM (cheap, not precise),LiDAR (expensive, precise),Can be fixed, mobile, aerial, drone-embarked , =>Produces a 3D point cloud,Large acquisition: n typically in the 10000000s

- LiDAR is getting cheaper :100k$ in a few years. 2k$ Also coming: solid state LiDAR (cheap, fast and resilient), single photon LiDAR (unmatched acquisition density). Major industrial application: autonomous driving, virtual models, land survey... Also to come: major advances in automatic analysis of 3D data,Rapid progress in hardware and methodology + major applications = a booming field.

Analysis of 3D point clouds

- **Classification**: classify the point cloud among class set $\mathcal{K}$:

$$P \mapsto \mathcal{K}$$

- **Partition**: cluster the point cloud in C parts/object:

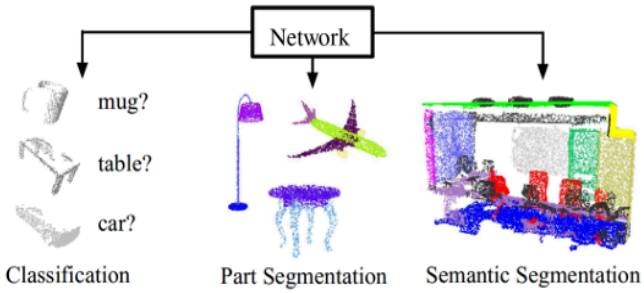
$$P_i \mapsto [1, \cdots, C]$$

- **Semantic Segmentation**: classify each point of a point cloud between K classes:

$$P_i \mapsto [1, \cdots, K]$$

- **Instance Segmentation**: cluster the point cloud into semantically characterized objects:

$$P_i \mapsto [1, \cdots, C]$$
$$[1, \cdots, C] \mapsto [1, \cdots, K]$$



credit: Qi et. al. 2017a

- **What makes 3D analysis so hard is the considerable amount of data , Lack of grid-structure , Permutation-invariance , Sparsity,highly variable density.- Acquisition artifacts.- Occlusions.**

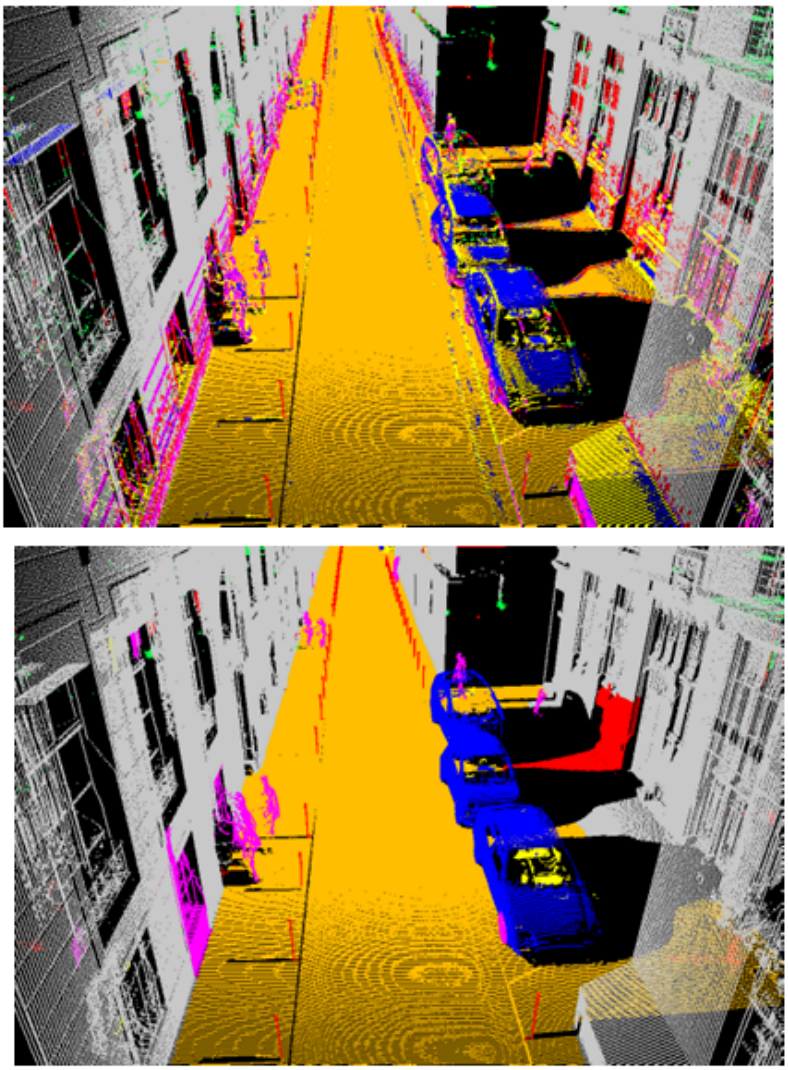
- **Traditional Approaches :**

  *** Pointwise classification**

  Step 1: compute point features based on neighborhood

  Step 2: classification (RF, SVM, etc...)

  Step 3: smoothing to increase spatial regularity (with CRFs, MRFs, graph-structured optimization, etc...)



- **First Deep_learning Approaches :**

  <u>*Image-Based Methods :</u>

  -A simple observation:

  CNNs work great for images. Can we use images for 3D? SnapNet:

  - surface reconstruction- virtual snapshots- semantic segmentation of resulting images with CNNs- project prediction back to p.c.

  <u>*Voxel-Based Methods :</u>

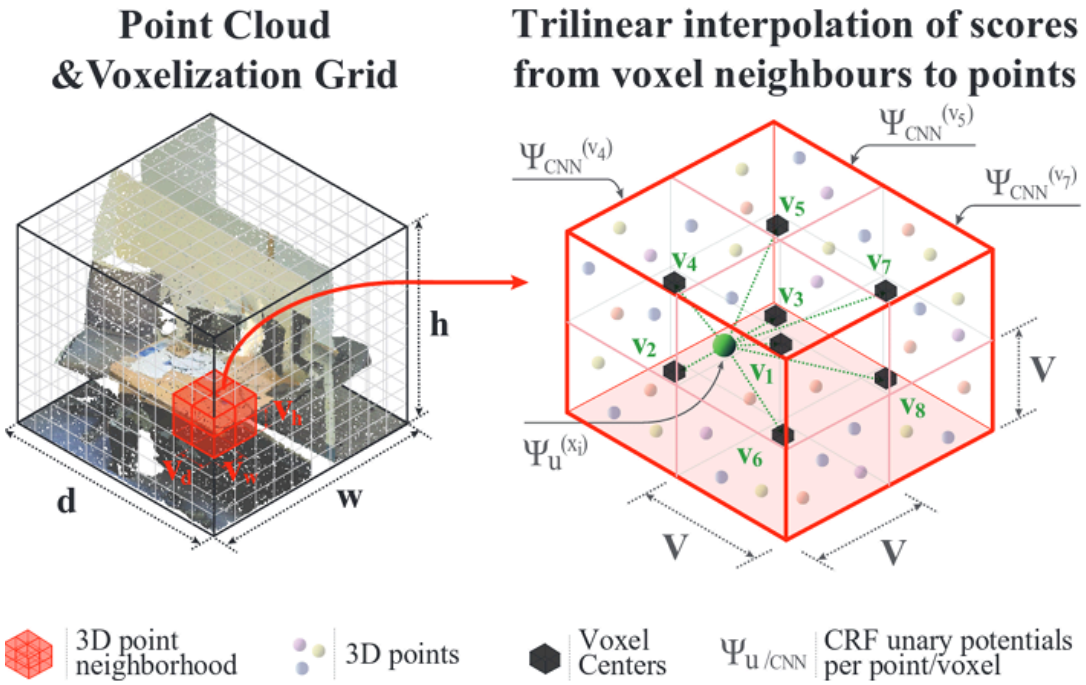
  - Idea: generalize 2D convolutions to regular 3D grids Voxelization + 3D convNets

  Problem: inefficient representation, loss of invariance, costly (cubic)

  Idea 1: OctNet, OctTree based approach

  Idea 2: SegCloud, large voxels, subvoxel predictions with CRFs.

  Idea 3: SplatNet, sparse convolutions with hashmaps.



***3D Convolution-Based Methods:**

Idea: generalize 2D convolutions to 3D point clouds as unordered data.

Tangent Convolution: 2D convolution in the tangent space of each point.

PointCNN :-convolutions: generalized convolutions for unordered inputs.

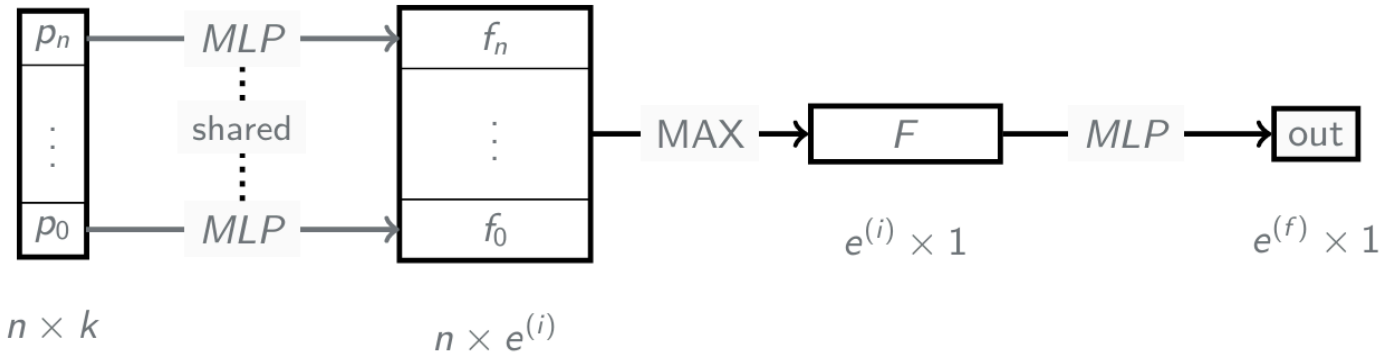
Principle: the network learns how to permute ordered inputs The invariance is learnt!

***PointNet :**

-A fundamental constraint: inputs are invariant by permutation

Solution: process points independently, apply permutation-invariant pooling, process this feature with a MLP.

n: number of points, k size of observations, e(i) size of intermediary embeddings, e(f) size of output



***Graph-Neural Network :**

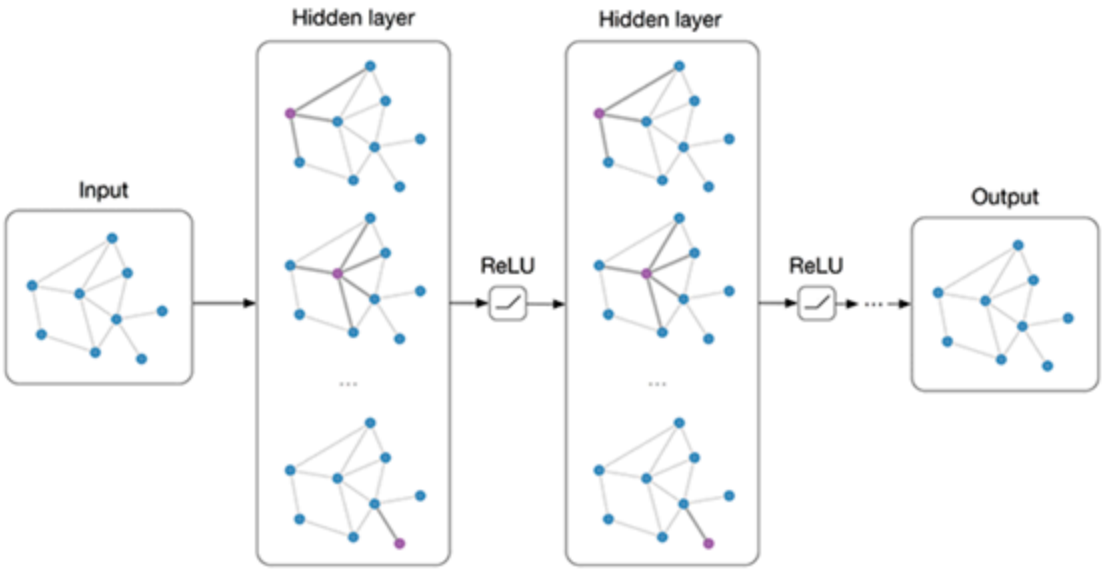
Generalize convolutions to the general graph setting. For example: k-nearest neighbors graph of 3D points. Idea: Each point maintain a hidden state hi influenced by its neighbors

GNN Qi2017: an iterative message-passing algorithm using a mapping f and a RNN g:

$$h_i^{(t+1)} = g\left(\sum_{j \to i} f(h_i^t), h_i^t\right)$$

ECC Simonovski2017 messages are conditioned by edge features:

$$h_i^{(t+1)} = g(\sum_{j \to i} \Theta_{i,j} \odot h_i^t, h_i^t)$$



***Scaling_Segmentation:**

<u>Problem</u>: best approaches are very memory-hungry and the data volumes are huge.

Previous methods only worked with a few thousands points.

Naive strategies:

- Aggressive subsampling: loses a lot of information.

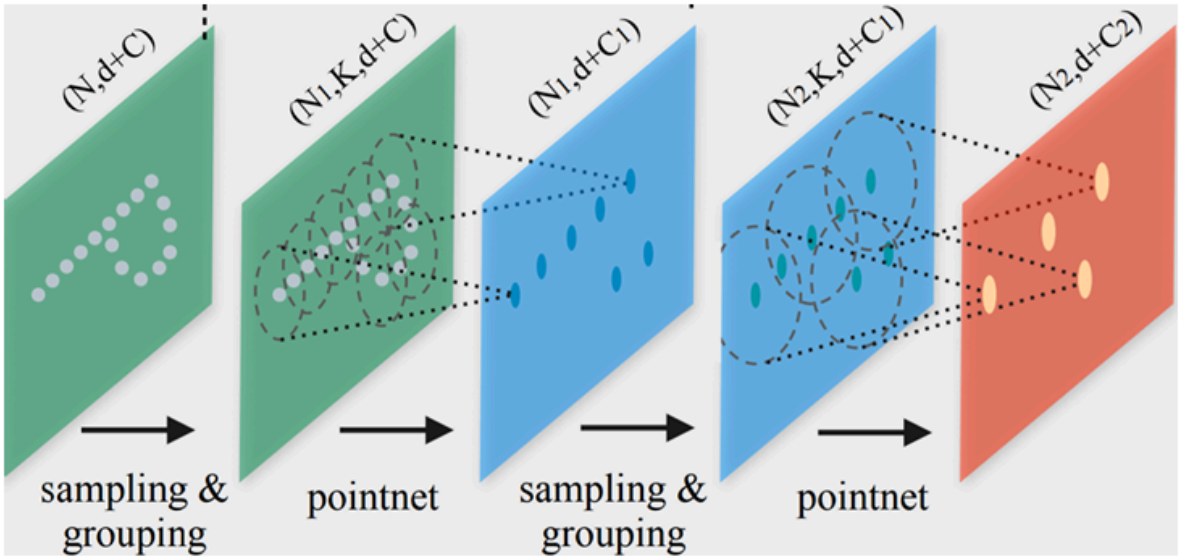
- Sliding windows: loses the global structure.

***PointNet ++:**

- Pyramid structure for multi-scale feature extraction.

-From local to global with increasingly abstract features.

-Still required to process millions of points.

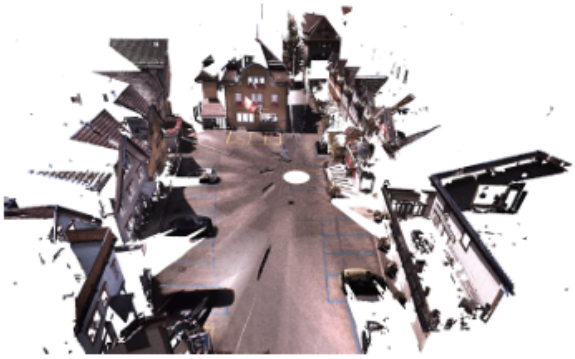


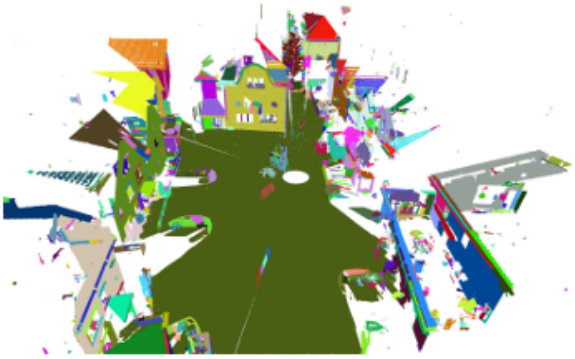
***SuperPoint-Graph :**

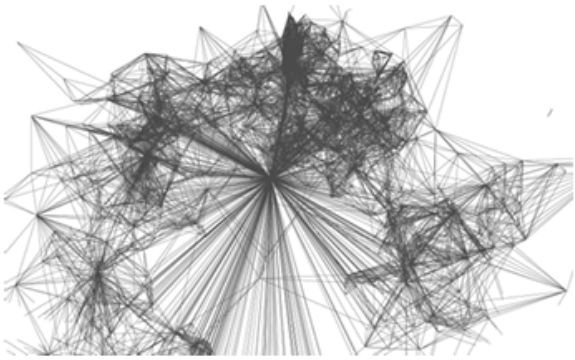
Observation: n points >>n objects.

Partition scene into superpoints with simple shapes.

Only a few superpoints, context leveraging with powerful graph methods.







***Pipeline:**

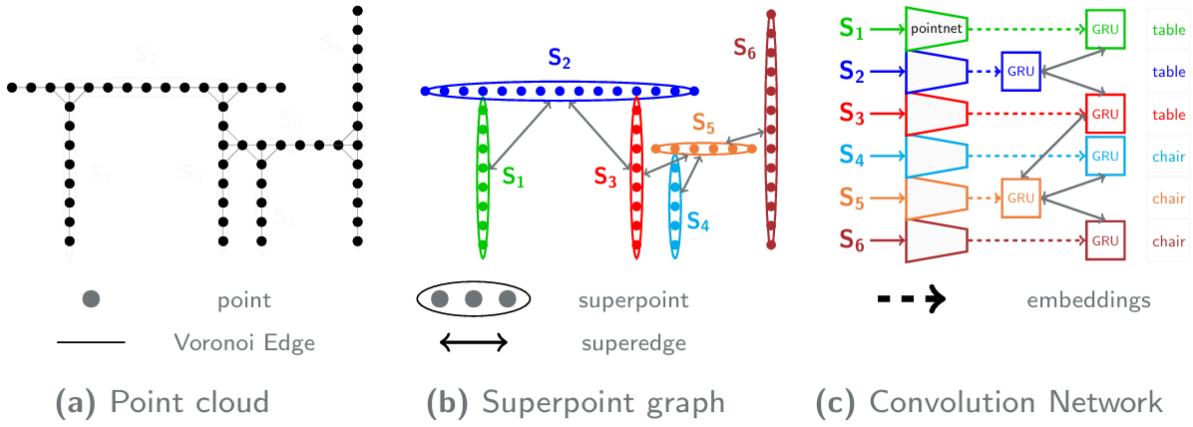
Semantic segmentation down to 3 sub-problems:

- <u>Geometric Partition</u> : into simple shapes.

Complexity: very high (clouds of 108 points) Algorithm: 0-cut pursuit

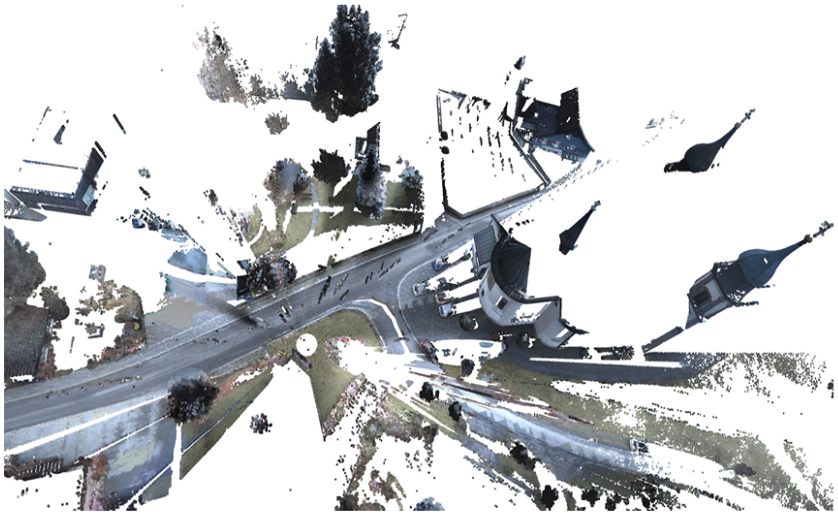
- <u>Superpoint embedding</u>: learning shape descriptors
Complexity: low (subsampling to 128 points 1000 points)
Algorithm: PointNet

- <u>Contextual Segmentation:</u> using the global structure
Complexity: very low (superpoint graph 1000 sp) Algorithm: ECC with Gated Recurrent Unit (GRU)

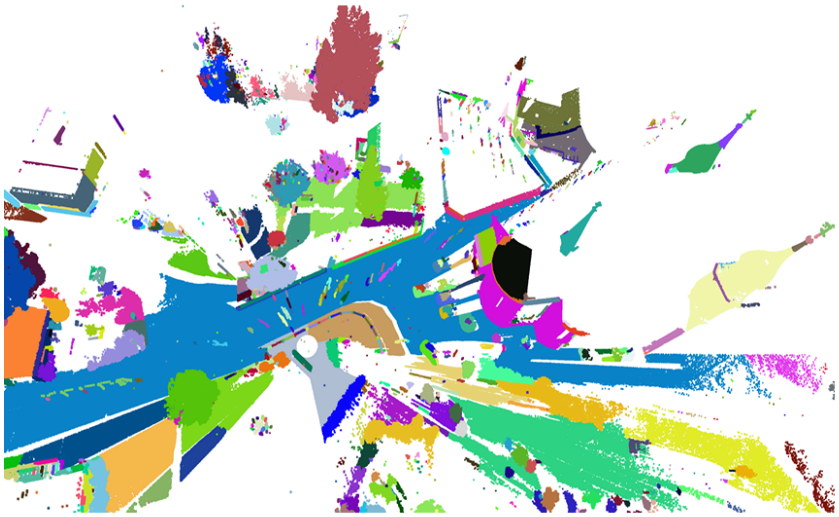


***Qualitative Results: Semantic3D**

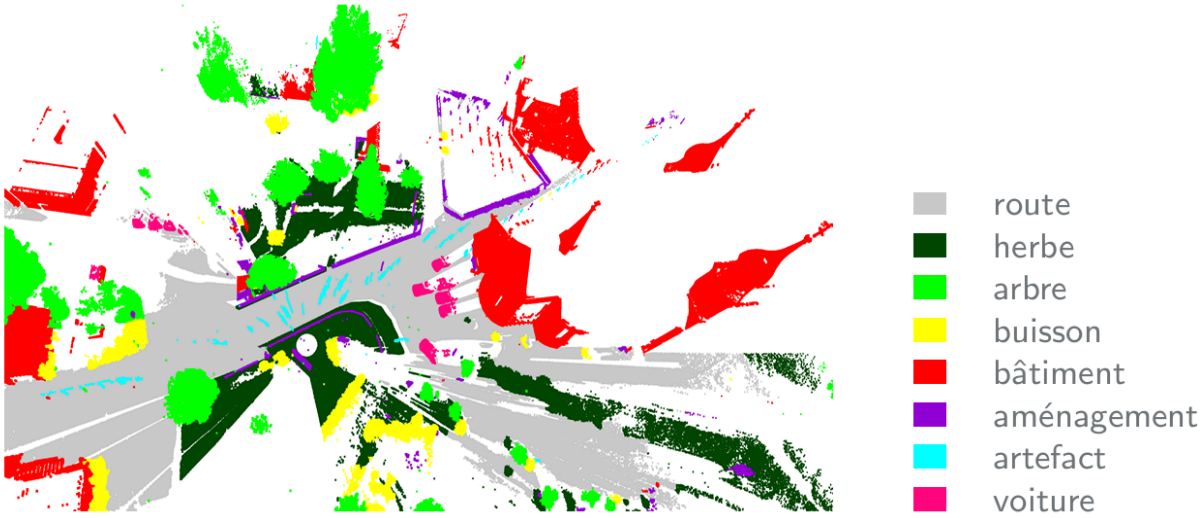
Semantic3D: 3 billions points over 30 clouds



route
herbe
arbre
buisson
bâtiment
aménagement
artefact
voiture



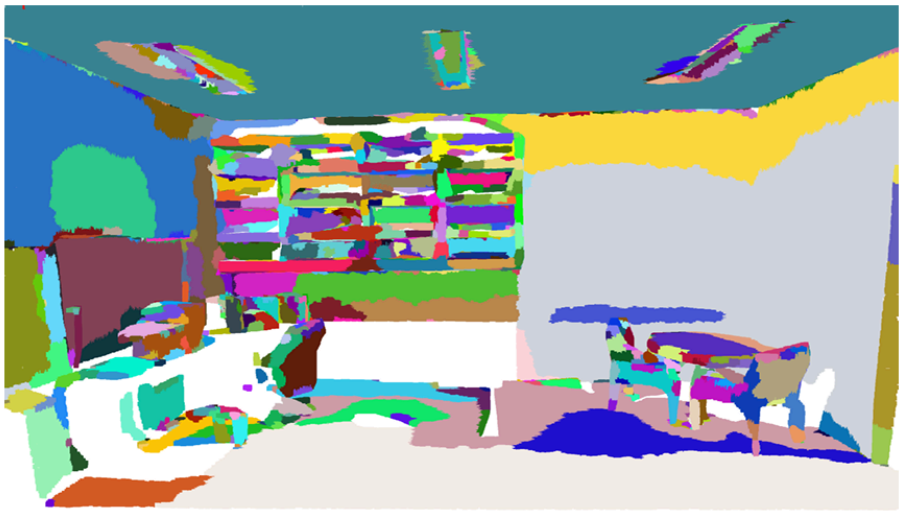
route
herbe
arbre
buisson
bâtiment
aménagement
artefact
voiture

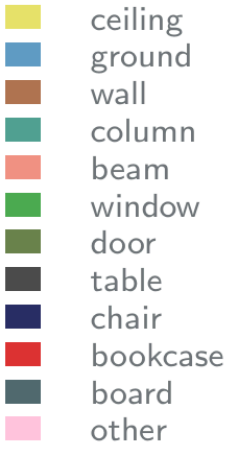


Methode	OA	mIoU	road	grass	tree	bush	build-ing	hard-scape	arti-fact	cars
reduced test set: 78 699 329 points										
TMLC-MSR	86.2	54.2	89.8	74.5	53.7	26.8	88.8	18.9	36.4	44.7
DeePr3SS	88.9	58.5	85.6	83.2	74.2	32.4	89.7	18.5	25.1	59.2
SnapNet	88.6	59.1	82.0	77.3	79.7	22.9	91.1	18.4	37.3	64.4
SegCloud	88.1	61.3	83.9	66.0	86.0	40.5	91.1	30.9	27.5	64.3
SPG (Ours)	**94.0**	**73.2**	**97.4**	**92.6**	**87.9**	**44.0**	**93.2**	**31.0**	**63.5**	**76.2**
full test set: 2 091 952 018 points										
TMLC-MS	85.0	49.4	91.1	69.5	32.8	21.6	87.6	25.9	11.3	55.3
SnapNet	91.0	67.4	89.6	**79.5**	74.8	56.1	90.9	36.5	34.3	77.2
SPG (Ours)	**92.9**	**76.2**	**91.5**	75.6	**78.3**	**71.7**	**94.4**	**56.8**	**52.9**	**88.4**

*Qualitative Results: S3DIS:

Indoor, 3 buildings, 6 stories, 200+ rooms, 600 000 000+ points





***Superpoint Partition:**

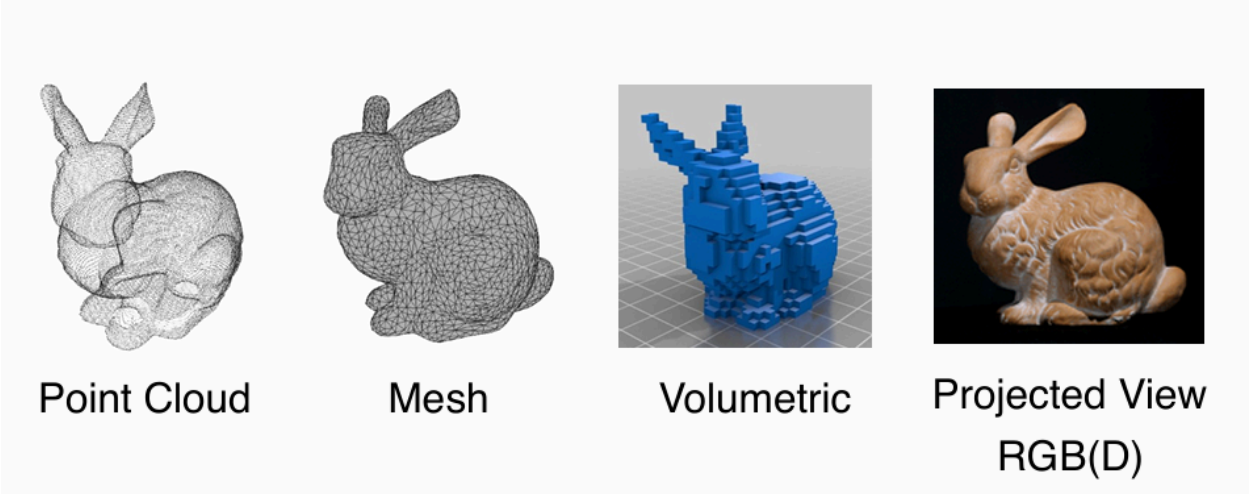
$$f^\star = \underset{f \in \mathbb{R}^{C \times m}}{\arg\min} \sum_{i \in C} \|f_i - e_i\|^2 + \sum_{(i,j) \in E} w_{i,j} \left[f_i \neq f_j\right],$$

- $e \in \mathbb{R}^{C \times m}$: handcrafted descriptors of the local geometry/radiometry

Superpoints: connected components of a piecewise constant approximation of e structured by an adjacency graph.

Problem: any errors made in the partition will carry in the prediction...

Book 18: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation



Point cloud is **converted to other representations** before it's fed to a deep neural network

Conversion	Deep Net
Voxelization	3D CNN
Projection/Rendering	2D CNN
Feature extraction	Fully Connected

Research Question: Can we achieve effective feature learning directly on point clouds?

- End-to-end learning for scattered, unordered point data Unified framework for various tasks

-Challenges:

–Unordered point set as input: Model needs to be invariant to N! permutations.

– Invariance under geometric transformations :Point cloud rotations should not alter classification results.

- **Unordered Input : Point cloud: N orderless points, each represented by a D dim vector**

  => Model needs to be invariant to N! permutations

***** Permutation Invariance : Symmetric Function**

$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

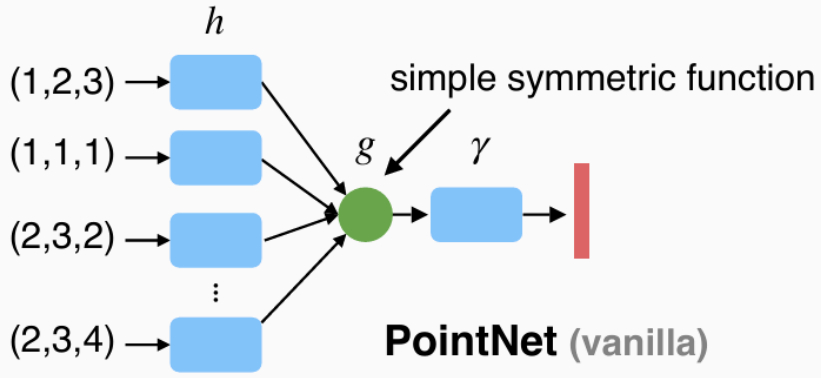
Examples:

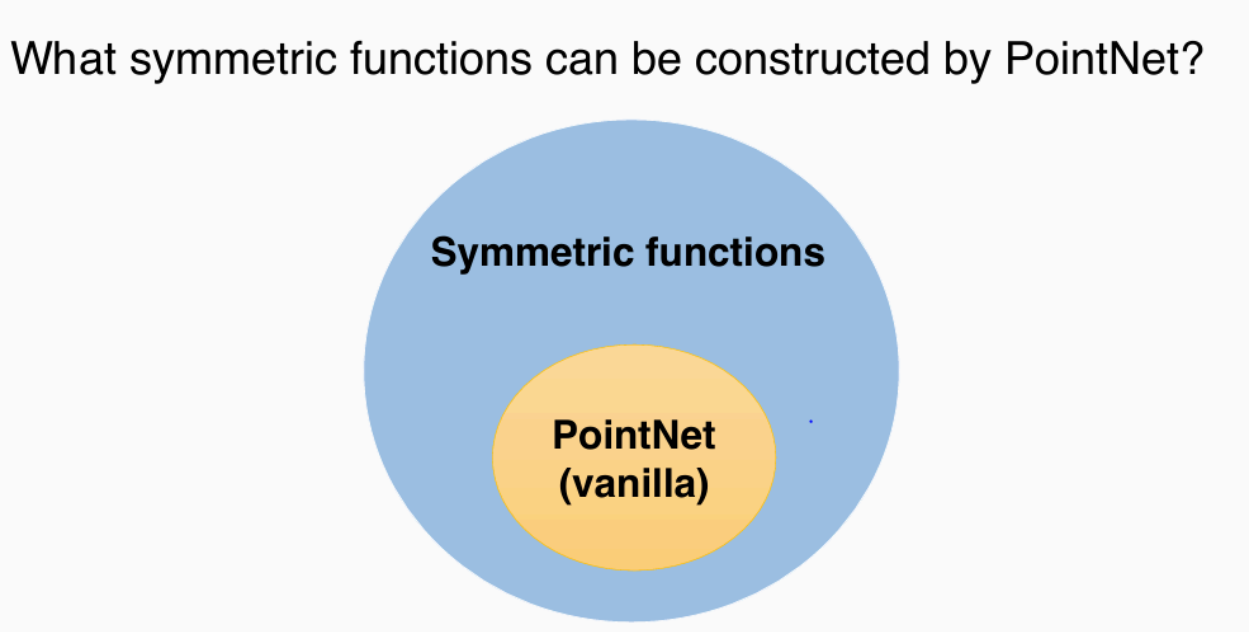
$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$
$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$
$$\ldots$$

How can we construct a family of symmetric functions by neural networks?

Observe:

$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric





***Universal Set Function Approximator :**

Theorem:

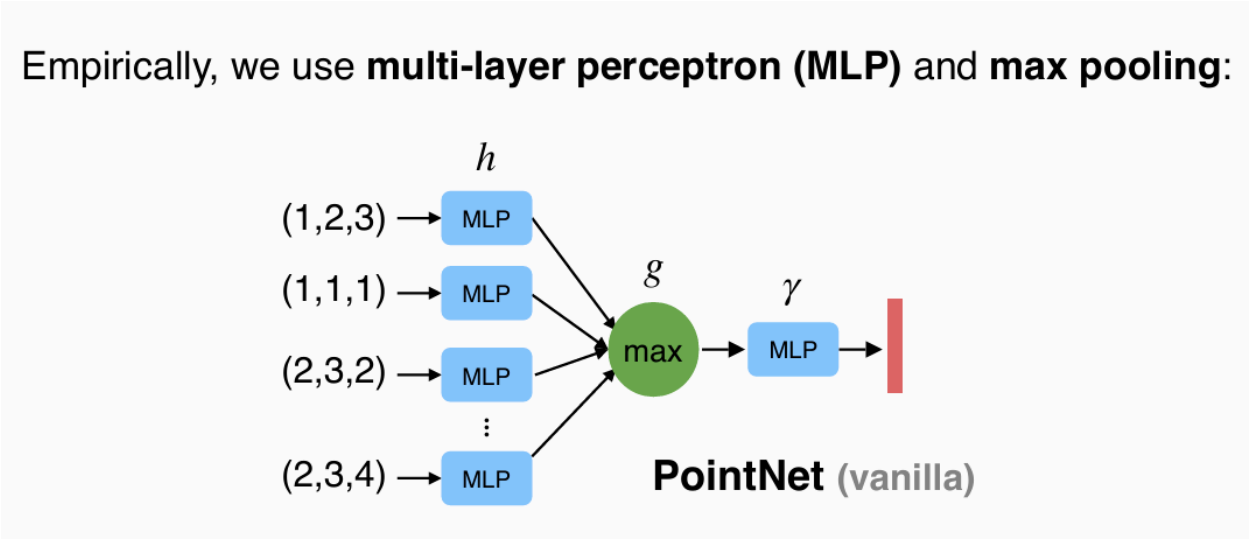
A Hausdorff continuous symmetric function $f : 2^{\mathcal{X}} \rightarrow \mathbb{R}$ can be arbitrarily approximated by PointNet.

$$\left| f(S) - \gamma \left(\underset{x_i \in S}{\mathrm{MAX}} \{ h(x_i) \} \right) \right| < \epsilon$$

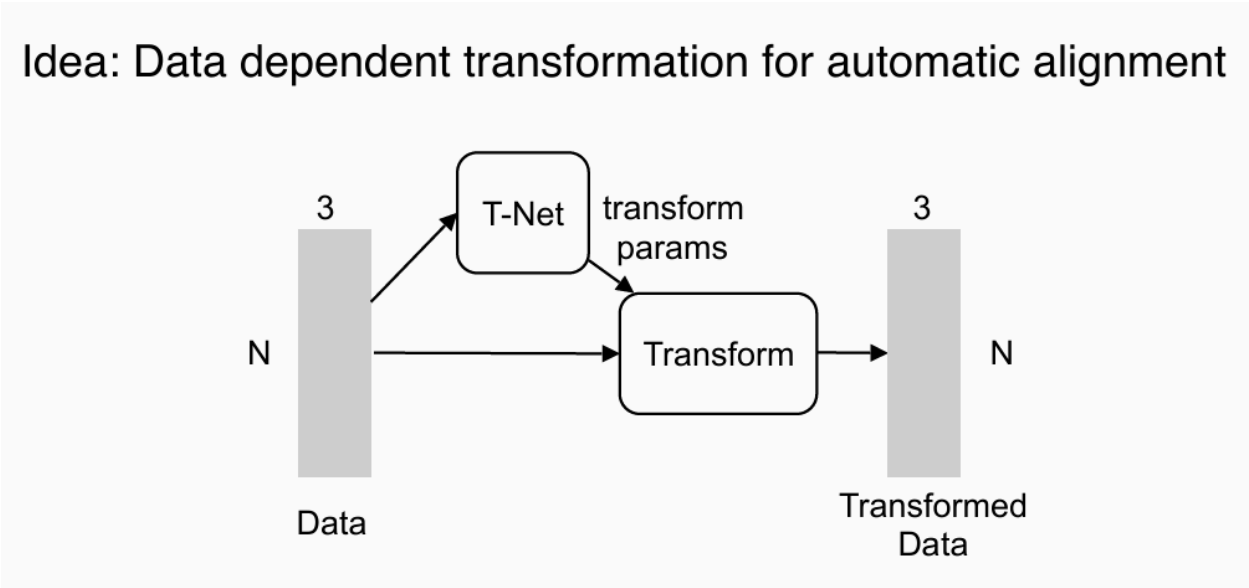
$S \subseteq \mathbb{R}^d$

PointNet (vanilla)

***Basic PointNet Architecture**

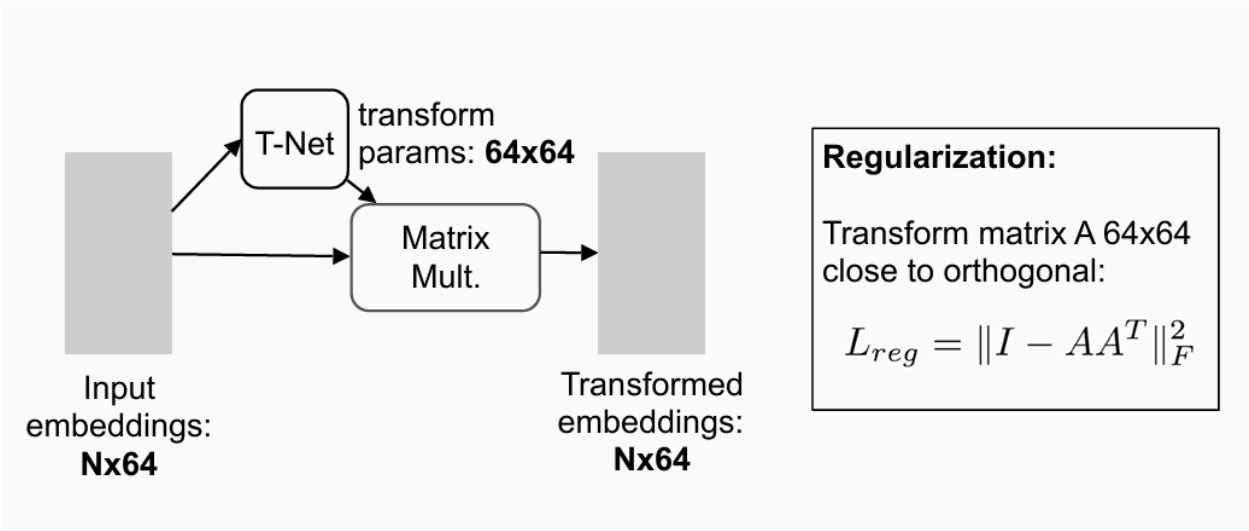


***Invariance under geometric transformations :Point cloud rotations should not alter classification results.**

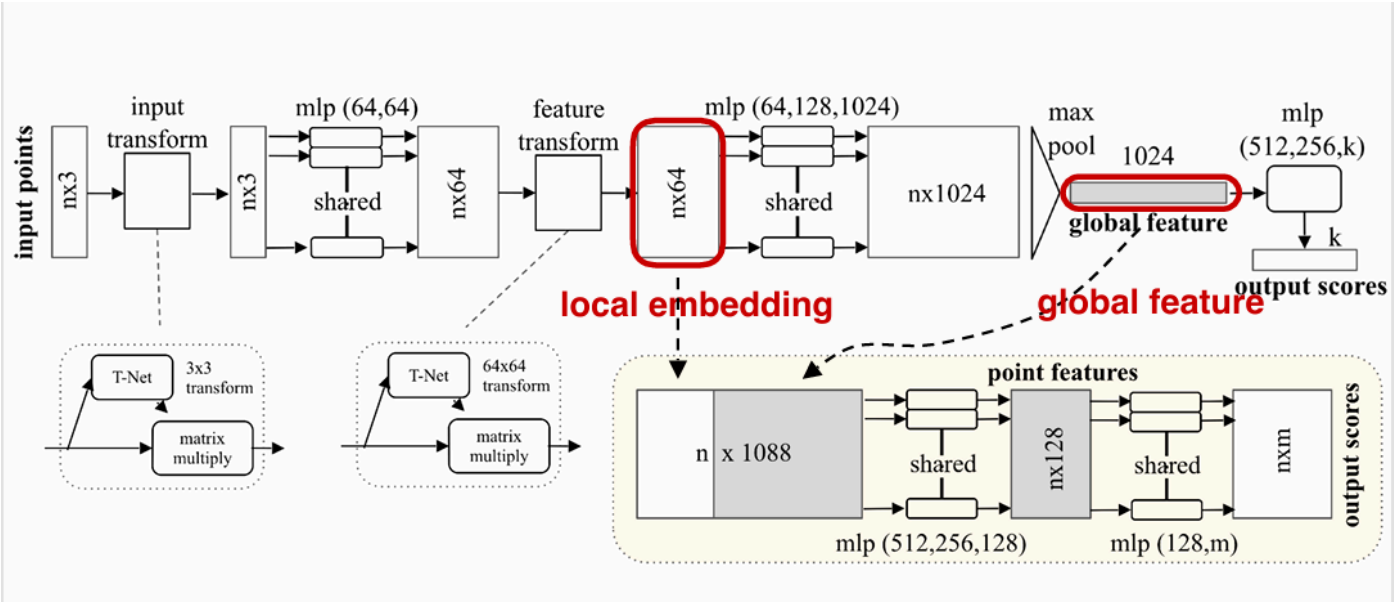


- The transformation is just matrix multiplication!

-Embedding Space Alignment :



***PointNet Classification Network:**



***Results on Object Classification**

		input	#views	accuracy avg. class	accuracy overall
	SPH [12]	mesh	-	68.2	
3D CNNs	3DShapeNets [29]	volume	1	77.3	84.7
	VoxNet [18]	volume	12	83.0	85.9
	Subvolume [19]	volume	20	86.0	**89.2**
	LFD [29]	image	10	75.5	-
	MVCNN [24]	image	80	**90.1**	-
	Ours baseline	point	-	72.6	77.4
	Ours PointNet	point	1	86.2	**89.2**

dataset: ModelNet40; metric: 40-class classification accuracy (%)

Book 19 : ArchShapesNet : a novel dataset for benchmarking architectural building information modeling element classification algorithms

-Abstract : Recent studies in the domain of semantic enrichment have employed artificial intelligence (AI) approaches to distinguish and classify building information modeling (BIM) elements to check their conformance with open standard data formats. Training AI algorithms requires the development of well-balanced and robust datasets of BIM elements.However,collection is difficult as sources are limited

to existing models and sample libraries. This study developed a parametric augmentation approach to create synthetic copies of BIM elements,and thus rapidly supplement manually collected samples.The approach was used to create ArchShapesNet,a dataset consisting of 11 common architectural elements with an equal size of 4,000 samples per class. Two multi-view convolutional neural networks (CNN), a geometric deep learning algorithm, were trained and tested separately on ArchShapesNet and an initial dataset with sample imbalances. Results showed significant improvement in the accuracy and F1 scores, providing evidence of the utility of ArchShapesNet. The size and scope of the dataset are considered to be the first of their kind and provide a benchmark for testing the semantic integrity of BIM models. The augmentation approach also provides a general framework to create custom datasets for different specialties in the Architectural Engineering and Construction industry.

-Introduction Building Information Modeling(BIM)provides a collaborative form for integrating and sharing information throughout the life cycle of construction projects. Designers, engineers, and contrac tors todayutilizeamultitudeofspecialtysoftwaretomeetvarious needs ranging from design quality checks, conflict detection, and cost and scheduling analysis, to more in-depth analyses such as building code compliance,and structural and energy

simulations. The Industrial Foundation Classes (IFC), a neutral and open data format, is critical in ensuring interoperability among these BIM applications. However, due to the burden of encapsulating diverse concepts and entities across multiple disciplines, the IFC schema is highly complex and exposed to redundant representations. Consequently, the lack of logical rigidness in describing model elements and their relationships makes IFC-based exchanges unpredictable (Eastman et al., 2009). "Semantic enrichment" encompasses the research conducted to check and correct the semantic integrity of the associations between BIM and their IFC representations (Belsky et al., 2016). A subset of these studies investigated ways to check the correct mapping of individual BIM elements to their corresponding IFC entities (Bassier et al., 2017;Maet al., 2017; Bloch & Sacks, 2018). Duetoit Open-ended architecture,theIFC doesn't mandate that such element-to-IFC "mappings" are indeed legitimate.

- While initial studies focused on defining sets of inference rules to check and update these mappings(Eastmanetal.,2009;Pauwels et al., 2011; Sacks et al., 2017), more recent works have explored the use of artificial intelligence (AI) approaches, purporting to their generality and scalability (Bloch & Sacks,2018; Wu & Zhang, 2019; Ying & Lee, 2019). The authors also explored their applicability by experimenting with different machine and deep learning algorithms to classify BIM elements based on their

geometric features (Koo & Shin, 2018;Kooet al., 2021a). Of these, promising results were attained using multi-view CNN (MVCNN), a geometric deep learning model that employs multiple panoramic images of a three-dimensional (3D) artifact from different perspectives to learn and distinguish its shape (Su et al., 2015;Kooet al., 2019). Despite its relatively high performance, MVCNN was still limited in classifying specific BIM elements correctly. These errors were in part attributed to the limited size of the training data, as well as the imbalance in the number of samples per class. These issues were previously unavoidable: The dataset was originally developed by collecting elements from existing BIM models and open-source libraries, and thus was dependent on their availability. This study focused on developing approaches to create "synthetic" BIM elements to increase the sample sizes and even out the number of samples per class. Specifically, we developed procedures to augment the existing dataset by parameterizing the dimensions of the original BIM elements and altering these parameters to create modified copies of them. Using this process, an augmented dataset termed "ArchShapesNet", was developed consisting of an equal 4,000 samples for 11 types of BIM element

Book 20 : PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

- Few prior works study deep learning on point sets. PointNet [20] is a pioneer in this direction. However, by design PointNet does not capture local structures induced by the metric space points live in, limiting its ability to recognize fine-grained patterns and generalizability to complex scenes. In this work, we introduce a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. By exploiting metric space distances, our network is able to learn local features with increasing contextual scales. With further observation that point sets are usually sampled with varying densities, which results in greatly decreased performance for networks trained on uniform densities, we propose novel set learning layers to adaptively combine features from multiple scales. Experiments show that our network called PointNet++ is able to learn deep point set features efficiently and robustly. In particular, results significantly better than state-of-the-art have been obtained on challenging benchmarks of 3D point clouds.

-1 Introduction Weare interested in analyzing geometric point sets which are collections of points in a Euclidean space. A

particularly important type of geometric point set is point cloud captured by 3D scanners, e.g., from appropriately equipped autonomous vehicles. As a set, such data has to be invariant to permutations of its members. In addition, the distance metric defines local neighborhoods that may exhibit different properties. For example, the density and other attributes of points may not be uniform across different locations — in 3D scanning the density variability can come from perspective effects, radial density variations, motion, etc. Few prior works study deep learning on point sets. PointNet [20] is a pioneering effort that directly processes point sets. The basic idea of PointNet is to learn a spatial encoding of each point and then aggregate all individual point features to a global point cloud signature. By its design, PointNet does not capture local structure induced by the metric. However, exploiting local structure has proven to be important for the success of convolutional architectures. A CNN takes data defined on regular grids as the input and is able to progressively capture features at increasingly larger scales along a multi-resolution hierarchy. At lower levels neurons have smaller receptive fields whereas at higher levels they have larger receptive fields. The ability to abstract local patterns along the hierarchy allows better generalizability to unseen cases. We Introduce a hierarchical neural network, named as PointNet++, to process a set of points sampled in a metric space in a hierarchical fashion.

The general idea of PointNet++ is simple. We first partition the set of points into overlapping local regions by the distance metric of the underlying space. Similar to CNNs, we extract local features capturing fine geometric structures from small neighborhoods; such local features are further grouped into larger units and processed to produce higher level features. This process is repeated until we obtain the features of the whole point set. The design of PointNet++ has to address two issues: how to generate the partitioning of the point set, and how to abstract sets of points or local features through a local feature learner. The two issues are correlated because the partitioning of the point set has to produce common structures across partitions, so that weights of local feature learners can be shared, as in the convolutional setting. We choose our local feature learner to be PointNet. As demonstrated in that work, PointNet is an effective architecture to process an unordered set of points for semantic feature extraction. In addition, this architecture is robust to input data corruption. As a basic building block, PointNet abstracts sets of local points or features into higher level representations. In this view, PointNet++ applies PointNet recursively on a nested partitioning of the input set. One issue that still remains is how to generate overlapping partitioning of a point set. Each partition is defined as a neighborhood ball in the underlying Euclidean space, whose parameters include centroid location

and scale. To evenly cover the whole set, the centroids are selected among input points set by a farthest point sampling (FPS) algorithm. Compared with volumetric CNNs that scan the space with fixed strides, our local receptive fields are dependent on both the input data and the metric, and thus more efficient and effective. Figure 1: Visualization of a scan captured from a Structure Sensor (left: RGB; right: point cloud). Deciding the appropriate scale of local neighborhood balls, however, is a more challenging yet intriguing problem, due to the entanglement of feature scale and non-uniformity of input point set. We assume that the input point set may have variable density at different areas, which is quite common in real data such as Structure Sensor scanning [18] (see Fig. 1). Our input point set is thus very different from CNN inputs which can be viewed as data defined on regular grids with uniform constant density. In CNNs, the counterpart to local partition scale is the size of kernels. [25] shows that using smaller kernels helps to improve the ability of CNNs. Our experiments on point set data, however, give counter evidence to this rule. Small neighborhood may consist of too few points due to sampling deficiency, which might be insufficient to allow PointNets to capture patterns robustly. A Significant contribution of our paper is that PointNet++ leverages neighborhoods at multiple scales to achieve both robustness and detail capture. Assisted with random input dropout during training, the network learns to

adaptively weight patterns detected at different scales and combine multi-scale features according to the input data. Experiments show that our PointNet++ is able to process point sets efficiently and robustly. In particular, results that are significantly better than state-of-the-art have been obtained on challenging benchmarks of 3D point clouds.

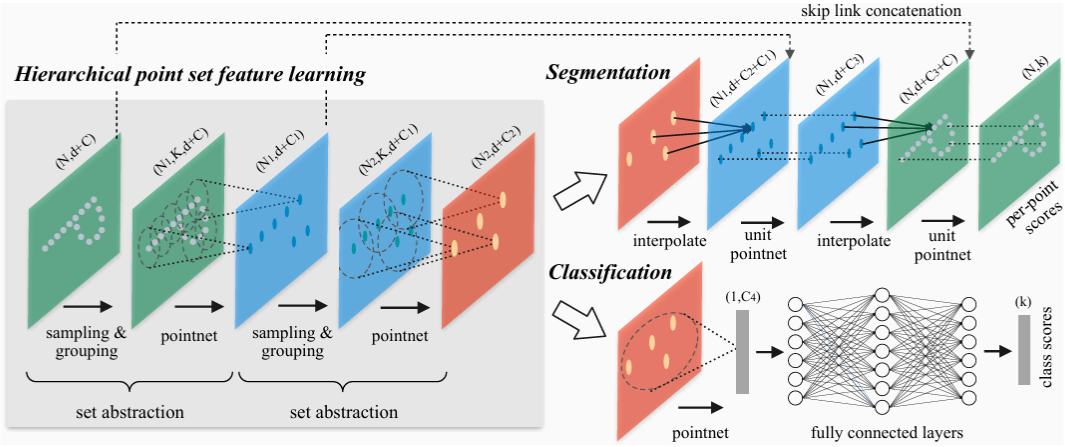


Figure 2: Illustration of our hierarchical feature learning architecture and its application for set segmentation and classification using points in 2D Euclidean space as an example. Single scale point grouping is visualized here. For details on density adaptive grouping, see Fig. 3

***Experiments :**

Datasets We evaluate on four datasets ranging from 2D objects (MNIST [11]), 3D objects (Model Net40 [31] rigid object, SHREC 15 [12] non-rigid object) to real 3D scenes (ScanNet [5]). Object classification is evaluated by accuracy. Semantic scene labeling

is evaluated by average voxel classification accuracy following [5]. We list below the experiment setting for each dataset:

- **MNIST: Images of handwritten digits with 60k training and 10^k testing samples.**
- **ModelNet40: CAD models of 40 categories (mostly man-made). We use the official split with 9,843 shapes for training and 2,468 for testing.**
- **SHREC 15: 1200 shapes from 50 categories. Each category contains 24 shapes which are mostly organic ones with various poses such as horses, cats, etc. We use five fold cross validation to acquire classification accuracy on this dataset.**
- **ScanNet: 1513 scanned and reconstructed indoor scenes. We follow the experiment setting in [5] and use 1201 scenes for training, 312 scenes for test.**

Book 20 : Learning Semantic Segmentation of Large-Scale Point Clouds with Random Sampling

- **Abstract:We study the problem of efficient semantic segmentation of large-scale 3D point clouds. By relying on expensive sampling techniques or computationally heavy pre/post-processing steps, most existing approaches are only able to be trained and operate over small-scale point clouds. In this paper, we introduce RandLA-Net, an efficient and lightweight neural architecture to directly infer per-point semantics for large-scale point clouds. The key to our approach is to use random point sampling instead of more complex point selection approaches. Although remarkably computation and memory efficient, random sampling can discard key features by chance. To overcome this, we introduce a novel local feature aggregation module to progressively increase the receptive field for each 3D point, thereby effectively preserving geometric details. Comparative experiments show that our RandLA-Net can process 1 million points in a single pass up to 200 faster than existing approaches. Moreover, extensive experiments on five large-scale point cloud**

datasets, including Semantic 3D, SemanticKITTI, Toronto 3D, NPM3D and S3DIS, demonstrate the state-of-the-art semantic segmentation performance of our RandLA-Net.

- **EFFICIENT semantic segmentation of large-scale 3D point clouds is a fundamental and essential capability for real-time intelligent systems, such as autonomous driving and augmented reality. A key challenge is that the raw point clouds acquired by depth sensors are typically ir regularly sampled, unstructured and unordered. Although deep convolutional networks show excellent performance in structured 2D computer vision tasks, they cannot be directly applied to this type of unstructured data. Recently, the pioneering work PointNet [4] has emerged as a promising approach for directly processing 3D point clouds. It learns per-point features using shared multilayer perceptrons (MLPs). This is computationally efficient but fails to capture wider context information for each point. To learn richer local structures, many dedicated neural modules have been subsequently and rapidly introduced. These modules can be generally categorized as: 1) neigh bouring feature pooling [1], [5], [6], [7], [8], 2) graph message passing [9], [10], [11], [12], [13], [14], [15], [16], 3) kernel based convolution [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], and 4) attention-based aggregation [28], [29], [30], [31]. Although these approaches achieve**

impressive results for object recognition and semantic segmentation, most of them are limited to extremely small 3D point clouds (e.g., 4k points or 1 1 meter blocks) and cannot be directly extended to larger point clouds (e.g., millions of points and up to 200 200 meters) without preprocessing steps such as block partition. The reasons for this limitation are three-fold. 1) The commonly used point-sampling methods of these networks are either computationally expensive or memory inefficient. For example, the widely employed farthest-point sampling [1] takes over 200 seconds to sample 10% of 1 million points. 2) Most existing local feature learners usually rely on computationally expensive kernelization or graph construction, thereby being unable to process a massive number of points. 3) For a large-scale point cloud, which usually consists of hundreds of objects, the existing local feature learners are either incapable of capturing complex structures, or do so inefficiently, due to their limited size of receptive fields. A handful of recent works have started to tackle the task of directly processing large-scale point clouds. SPG [2] preprocesses the large point clouds as super graphs before applying neural networks to learn per superpoint semantics. However, the preprocessing steps are too computationally heavy to be deployed in real-time applications. Both FCPN [32] and PCT [33] combine

voxelization and point level networks to process massive point clouds. However, they still partition the point clouds into small blocks for learning, resulting in the overall performance being sub optimal. In this paper, we aim to design a memory and computationally efficient neural architecture, which is able to directly process large-scale 3D point clouds in a single pass, without requiring pre/post-processing steps such as voxelization, block partitioning or graph construction. How ever, this task is extremely challenging as it requires: 1) a memory and computationally efficient sampling approach to progressively downsample large-scale point clouds to fit in the limits of current GPUs, and 2) an effective local feature learner to progressively increase the receptive field size to preserve complex geometric structures. To this end, we first systematically demonstrate that random sampling is a key enabler for deep neural networks to efficiently process large scale point clouds. However, random sampling can discard key information, especially for objects with sparse points. To counter the potentially detrimental impact of random sampling, we propose a new and efficient local feature aggregation module to capture complex local structures over progressively smaller point-sets. Amongst Existing sampling methods, farthest point sampling and inverse density sampling are the most frequently used for

small-scale point clouds [1], [7], [19], [34], [35]. As point sampling is such a fundamental step within these networks, we investigate the relative merits of different approaches in Section 3.2, where we see that the commonly used sampling methods limit scaling towards large point clouds, and act as a significant bottleneck to real-time processing. However, we identify random sampling as a more suitable strategy for large-scale point cloud processing as it is fast and scales efficiently. Random sampling is not without cost, because prominent point features may be dropped by chance and it cannot be used directly in existing networks without incurring a performance penalty. To overcome this issue, we design a new local feature aggregation module in Section 3.3, which is capable of effectively learn ing complex local structures by progressively increasing the receptive field size in each neural layer. In particular, for each 3D point, we firstly introduce a local spatial encoding (LocSE) unit to explicitly preserve local geometric structures. Secondly, we leverage attentive pooling to automatically keep the useful local features. Thirdly, we stack multiple LocSE units and attentive poolings as a dilated residual block, greatly increasing the effective receptive field for each point. Note that all these neural components are implemented as shared MLPs, and are therefore remarkably memory and

computational efficient. Overall, being built on the principles of simple random sampling and an effective local feature aggregator, our efficient neural architecture, named RandLA-Net, not only is up to 200 faster than existing approaches on large-scale point clouds, but also surpasses the state-of-the-art semantic segmentation methods on Semantic 3D [36], SemanticKITTI [3] and Toronto-3D [37] benchmarks. Figure 1 shows qualitative results of our approach. Our key contributions are: We analyze and compare existing sampling approaches, identifying random sampling as a suitable component for efficient learning on large-scale point clouds. We propose an effective local feature aggregation module to preserve complex local structures by progressively increasing the receptive field for each point. We demonstrate significant memory and computational gains over baselines, and surpass the state-of-the-art semantic segmentation methods on multiple large-scale benchmarks

- **RELATED WORK :**

  To extract features from 3D point clouds, traditional ap proaches usually rely on hand-crafted features [43], [44], [45], [46]. Recent learning based approaches [4], [47], [48] mainly include projection-based, voxel-based and point based schemes which are outlined here. (1) Projection and Voxel-Based Networks. To leverage the success of 2D CNNs,

many works [49], [50], [51], [52] project/flatten 3D point clouds onto 2D images to address the task of object detection. However, geometric details may be lost during projection. For example, the commonly used birds-eye-view projection can drop some points due to occlusion. Alternatively, several other approaches [53], [54], [55], [56], [57] use spherical projection to convert the LiDAR point clouds back to raw range images without dropping points. However, these methods usually suffer from blurry CNNoutputs and quantization errors in practice [53]. Point clouds can also be voxelized into dense 3D grids and then processed with powerful 3D CNNs [58], [59], [60]. However, their memory consumption and computational time are sig nificant if a high voxel grid resolution is required. This issue can be alleviated using Octree [61] or sparse-tensor [62], [63], but these advanced data structures and sophisticated operations are not always easily supported by existing tools, especially on GPUs. (2) Point Based Networks. Inspired by Point Net/PointNet++ [1], [4], many recent works introduced sophisticated neural modules to learn per-point local fea tures. These modules can be generally classified as 1) neigh bouring feature pooling [5], [6], [7], [8], 2) graph message passing [9], [10], [11], [12], [13], [14], [15], [64], 3) kernel based convolution [17], [18], [19], [20], [21], [22], [23], [24],

[65], and 4) attention-based aggregation [28], [29], [30], [31]. Although these networks have shown promising results on small point clouds, most of them cannot directly scale up to large scenarios due to their high computational and memory costs. Compared with them, our proposed RandLA-Net is distinguished in three ways: 1) it only relies on random sampling within the network, thereby requiring much less memory and computation; 2) the proposed local feature aggregator can obtain successively larger receptive fields by explicitly considering the local spatial relationship and point features, thus being more effective and robust for learning complex local patterns; 3) the entire network only consists of shared MLPs coupled with the simple random sampling, therefore being efficient for large-scale point clouds.

Method	Complexity[1]	Time (seconds)[2]	Description
FPS [1]	$\mathcal{O}(M^2N)$	200	Farthest Point Sampling iteratively returns a reordering of the metric space $\{p_1 \cdots p_m \cdots p_M\}$, such that each p_m is the farthest point from the first $m-1$ points. It has good coverage of points but high computational complexity.
IDIS [35]	$\mathcal{O}((K+N)logN)$	10	Inverse Density Importance Sampling reorders all N points according to the density of each point, after which the top M points are selected. The density is approximated by calculating the summation of the distances between the point and its nearest K points. It can control density, but sensitive to outliers and noise.
PDS [39]	$\mathcal{O}(MN)$	8	Poisson Disk Sampling samples points from a set of N points with blue noise characteristics i.e., points are sampled from a Poisson disk distribution, where all samples are at least a certain distance r apart.
RS	$\mathcal{O}(M)$	0.004	Random Sampling uniformly selects M points from the original N points, each point has the same probability to be selected. It is agnostic to the total number of input points, i.e., it is constant-time and hence is inherently scalable.
GS [40]	-	1200	Generator based Sampling learns to generate a small set of points to approximately represent the original point set. FPS matching is required during inference.

(3) Learning for Large-scale Point Clouds. SPG [2] preprocesses the large point clouds as superpoint graphs to learn per superpoint semantics. However, both the ge ometric partition and superpoint graph construction are computationally expensive. The recent FCPN [32] and PCT [33] apply both voxel-based and point-based networks to process the massive point clouds. Based on the assumption that points are sampled from locally Euclidean surfaces, TangentConv [66] firstly projects the local surface on the tangent plane and then operates on the projected geometry. Despite being able to process large-scale point clouds, it requires a relatively heavy preprocessing step to calculate the normal. In contrast, our

RandLA-Net is end-to-end trainable without requiring additional expensive operations.

(4) Sampling Methods for 3D Point Clouds. Existing point sampling approaches [1], [19], [34], [35], [40], [41] can be roughly classified into heuristic and learning-based methods. Farthest point sampling [1], [34] is the most commonly used heuristic sampling strategy in recent works. It iteratively samples the points most distant to the remaining subset from the entire point set. Groh et al. [35] use inverse density importance sub-sampling (IDIS) to preserve points with lower density. Hermosilla et al. [67] utilize Poisson disk sampling (PDS) to achieve a uniform distribution for the sampled points. For learning-based approaches, Yang et al. [30] introduce Gumbel subset sampling, an end-to end learnable and task-agnostic sampling method, to obtain better performance for downstream tasks. Dovrat et al [40] propose a generator network to directly generate a point set to approximate the original point set. However, all these methods are either ineffective or computationally heavy as evaluated in Section 3.2 and Table 1.

- **PROPOSED METHODS :**

-As illustrated in Figure 2, given a large-scale point cloud with millions of points which could span hundreds of meters, to process it with a deep neural network inevitably requires those points to be progressively and efficiently downsampled in each neural layer, without losing the useful point features. In our RandLA-Net, we propose to use the simple and fast approach of random sampling to greatly decrease point density, whilst applying a carefully designed local feature aggregator to retain prominent features. This allows the entire network to achieve an excellent trade-off between efficiency and effectiveness.

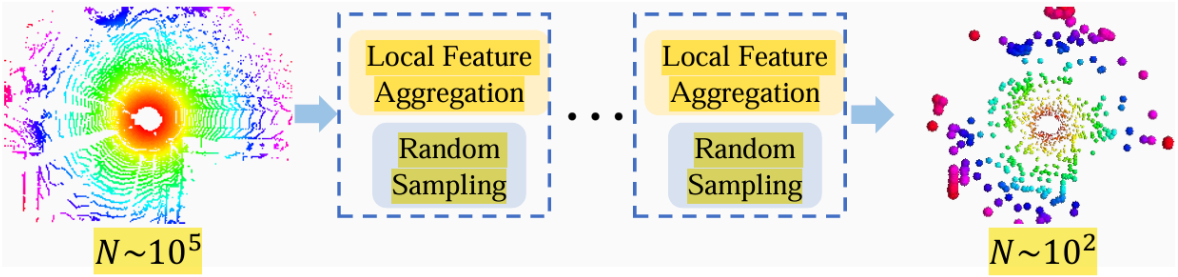


Fig. 2: In each layer of RandLA-Net, the large-scale point cloud is significantly downsampled, yet is capable of retaining features necessary for accurate segmentation.

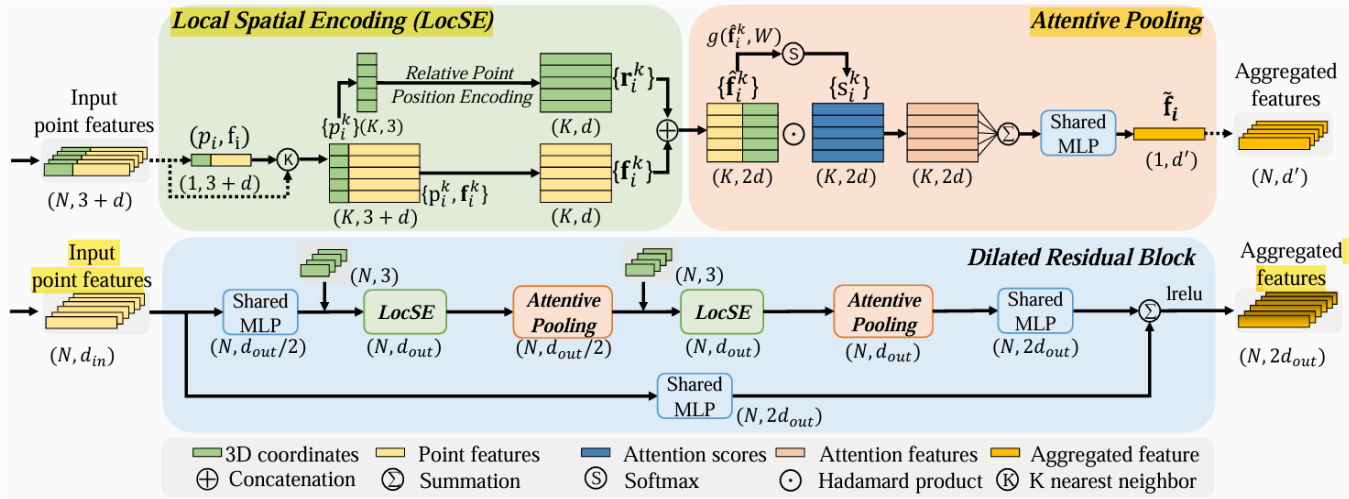


Fig. 3: The proposed local feature aggregation module. The top panel shows the local spatial encoding block that extracts features, and the attentive pooling mechanism that weights the important neighbouring features, based on the local context and geometry. The bottom panel shows how two of these components are chained together, to increase the receptive field size, within a residual block.

*** The Quest For Efficient Sampling:**

Overall,FPS, IDISandGS are too computationally expensive to be applied for large-scale point clouds. CRS approaches have nexcessivememoryfootprintan dOGS is hard to learn. PDS is relatively faster, but has worse performance, as shown in the ablation study.By Contrast, random sampling has the following two advantages: 1) it is remarkably computational efficient as it is agnostic to the total number of input points, 2) it does not require extra memory for computation.Therefore,we believe that random sampling is more suitable than other approaches to efficiently process large-scale point clouds. However, random sampling may result in many useful point features

beingdropped.To overcome it,we propose a powerful local feature aggregation module as presented in next section

*** LocalFeatureAggregation As shown in Figure3,our local feature aggregation module is applied to each 3D point in parallel and it consists of three neural units:**

1) local spatiale encoding (Loc S E),

2)attentive pooling

3)dilated residual block.

***LocalSpatialEncoding**

Given a point cloud P together with per-point features(e.g., rawRGB,or intermediate learned features),this local spatial encoding unit explicitly embeds the x-y-z coordinates of all neighboring points, such that the corresponding point features are always aware of their relative spatial locations. This Allows theLocSEunit to explicitly observe the local geometric patterns, thus eventually benefiting the entire network to effectively learn complex local structures. In particular,this unit includes the following steps: FindingNeighbouringPoints.Fortheithpoint, its neighboring points are firstly gathered by the simple K-nearest neighbors (KNN) algorithm for efficiency. TheKNNis based on point-wiseEuclideandistances. Relative Point Position Encoding:

Relative Point Position Encoding. For each of the nearest K points $\{p_i^1 \cdots p_i^k \cdots p_i^K\}$ of the center point p_i, we explicitly encode the relative point position as follows:

$$\mathbf{r}_i^k = MLP\Big(p_i \oplus p_i^k \oplus (p_i - p_i^k) \oplus ||p_i - p_i^k||\Big) \tag{1}$$

where p_i and p_i^k are the absolute x-y-z positions of points, $\oplus$ is the concatenation operation, and $||\cdot||$ calculates the Euclidean distance between the neighbouring and center points. It seems that $\mathbf{r}_i^k$ is encoded from redundant point positions. Interestingly, this tends to aid the network to learn local features and obtains good performance in practice.

Point Feature Augmentation. For each neighbouring point p_i^k, the encoded relative point positions $\mathbf{r}_i^k$ are concatenated with its corresponding point features $\mathbf{f}_i^k$, obtaining an augmented feature vector $\hat{\mathbf{f}}_i^k$. For simplicity, we keep the feature vectors $\mathbf{r}_i^k$ and $\mathbf{f}_i^k$ with the same dimension in the implementation, but they are flexible and can have different dimensions.

Eventually, the output of the LocSE unit is a new set of neighbouring features $\hat{\mathbf{F}}_i = \{\hat{\mathbf{f}}_i^1 \cdots \hat{\mathbf{f}}_i^k \cdots \hat{\mathbf{f}}_i^K\}$, which explicitly encodes the local geometric structures for the center point p_i. We notice that the recent work [68] also uses point positions to improve semantic segmentation. However, the positions are used to learn point scores in [68], while our LocSE explicitly encodes the relative positions to augment the neighbouring point features.

(2) Attentive Pooling

This neural unit is used to aggregate the set of neighbouring point features $\hat{\mathbf{F}}_i$. Existing works [1], [34] typically use max/mean pooling to hard integrate the neighbouring features, resulting in the majority of the information being lost. By contrast, we turn to the powerful attention mechanism to automatically learn important local features. In particular, inspired by [69], our attentive pooling unit consists of the following steps.

Computing Attention Scores. Given the set of local features $\hat{\mathbf{F}}_i = \{\hat{\mathbf{f}}_i^1 \cdots \hat{\mathbf{f}}_i^k \cdots \hat{\mathbf{f}}_i^K\}$, we design a shared function $g(\cdot)$ to learn a unique attention score for each feature. Essentially, the function $g(\cdot)$ consists of a shared MLP followed by $softmax$. It is formally defined as follows:

$$\mathbf{s}_i^k = g(\hat{\mathbf{f}}_i^k, \boldsymbol{W}) \tag{2}$$

where $\boldsymbol{W}$ is the learnable weights of a shared MLP.

Weighted Summation. The learned attention scores can be regarded as a soft mask which automatically selects the important features. Formally, these features are weighted summed as follows:

$$\tilde{\mathbf{f}}_i = \sum_{k=1}^{K} (\hat{\mathbf{f}}_i^k \odot \mathbf{s}_i^k) \tag{3}$$

where $\odot$ is the element-wise product. To summarize, given the input point cloud $\boldsymbol{P}$, for the i^{th} point p_i, our LocSE and Attentive Pooling units learn to aggregate the geometric patterns and features of its K nearest points, and finally generate an informative feature vector $\tilde{\mathbf{f}}_i$.

(3) Dilated Residual Block

Since the large point clouds are going to be substantially downsampled, it is desirable to significantly increase the receptive field for each point, such that the geometric details of input point clouds are more likely to be preserved, even if some points are dropped. As shown in Figure 3, inspired by the successful ResNet [70] and the effective dilated networks [71], we stack multiple LocSE and Attentive Pooling units to achieve the dilation of point receptive fields, and add a skip connection to achieve residual learning.

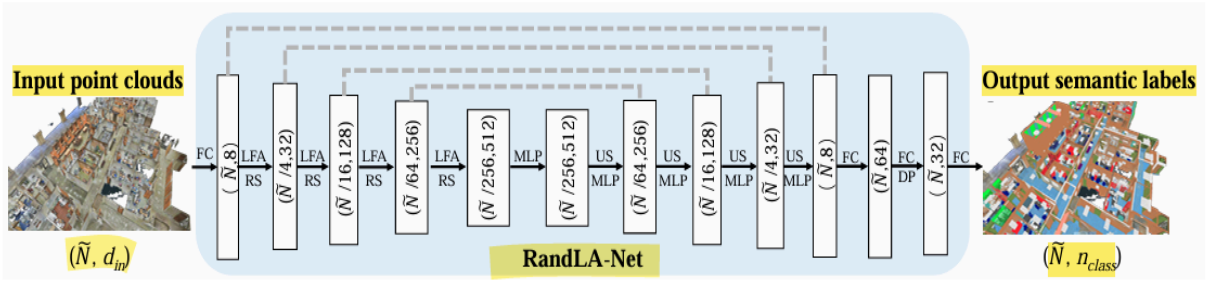


Fig. 4: The detailed architecture of our RandLA-Net. $(\tilde{N}, D)$ represents the number of points and feature dimension respectively. FC: Fully Connected layer, LFA: Local Feature Aggregation, RS: Random Sampling, MLP: shared Multi-Layer Perceptron, US: Up-sampling, DP: Dropout.

* Network Architecture :

Figure 4 shows the detailed architecture of RandLA-Net, which stacks multiple local feature aggregation modules and random sampling layers. The network follows the widely-used encoder-decoder architecture with skip connections. The input point cloud is first fed to a shared MLP layer to extract per-point features. Four encoding and decoding layers are then used to learn features for each point. At last, three fully-connected layers and a dropout layer are used to predict the semantic label of each point. Note that, our RandLA-Net can be wider and deeper by altering the number of layers, feature channels, and adjusting the sampling rate. However, larger models require additional computation and may more easily lead to overfitting. The details of each component are as follows:

Network Input: The input is a large-scale point cloud with a size of $\tilde{N} \times d_{in}$ (the batch dimension is dropped for simplicity), where $\tilde{N}$ is the total number of input points, d_{in} is the feature dimension of each input point. For both S3DIS [72] and Semantic3D [36] datasets, each point is represented by its 3D coordinates and color information (i.e., x-y-z-R-G-B), while each point of the SemanticKITTI

[3] dataset is only represented by 3D coordinates.

Encoding Layers: Four encoding layers are used in our network to progressively reduce the size of the point clouds and increase the per-point feature dimensions. Each encoding layer consists of a local feature aggregation module (Section 3.3) and a random sampling operation (Section 3.2). The point cloud is downsampled with a four-fold decimation ratio. In particular, only 25% of the point features are retained after each layer, i.e., $(\widetilde{N} \rightarrow \frac{\widetilde{N}}{4} \rightarrow \frac{\widetilde{N}}{16} \rightarrow \frac{\widetilde{N}}{64} \rightarrow \frac{\widetilde{N}}{256})$. Meanwhile, the per-point feature dimension is gradually increased each layer to preserve more information, i.e., $(8 \rightarrow 32 \rightarrow 128 \rightarrow 256 \rightarrow 512)$.

Decoding Layers: Four decoding layers are used after the encoding layers. For each layer in the decoder, we adopt the nearest-neighbor interpolation for efficiency and simplicity. In particular, the coordinates of all down-sampled points in each encoding layer are temporally stored for reference. For each query point in the decoding layers, we use the KNN algorithm to find the nearest neighboring point from the points of the previous layer. The features of the nearest point are copied to the target point. Subsequently, the upsampled feature maps are concatenated with the intermediate feature maps produced by encoding layers through skip connections, after which a shared MLP is applied to the concatenated feature vectors.

Final Semantic Prediction: The final semantic label of each point is obtained through three shared fully-connected layers $(\widetilde{N}, 64) \rightarrow (\widetilde{N}, 32) \rightarrow (\widetilde{N}, n_{class})$ and a dropout layer. The dropout ratio is 0.5.

Network Output: The output of RandLA-Net is the predicted semantics of all points, with a size of $\widetilde{N} \times n_{class}$, where n_{class} is the number of classes.

3.5 Implementation

We use the same network architecture for all the five large-scale open datasets, Semantic3D [36], SemanticKITTI [3], Toronto-3D [37], NPM3D [73], S3DIS [72]. The Adam optimizer [74] with default parameters is applied. The initial learning rate is set to 0.01 and decreases by 5% after each epoch. The network is trained for 100 epochs. We implement the KNN search in our framework based on the nanoflann[1] package, which leverages an efficient KD-Tree data structure for fast search and query. In addition, we also use OpenMP for better parallelization, and the number of nearest points K is set to 16. For several extremely large datasets such as Semantic3D [36], which have more than 10^6 or even 10^8 points in a single point cloud, we crop sub-clouds to feed into our RandLA-Net. During training, we sample a fixed number of points ($\sim 10^5$) from each point cloud as the input for parallelization. During testing, we iteratively infer several sub-clouds with overlaps to eventually cover all 3D points. Since many points have been inferred more than once, we follow [23] to use a simple voting scheme for better

performance. To alleviate the problem of class imbalance, we use weighted cross-entropy as the loss function. In particular, the weight of each class is determined by its inverse frequency in the training split. We do not use any explicit data augmentation techniques such as rotation, scaling, and translation during training. Note that, our random downsampling in each encoding layer can be regarded as implicit data augmentation. All experiments are conducted on an NVIDIA RTX2080Ti GPU.

Book 21 : IFCNet: A Benchmark Dataset for IFC Entity Classification

- Enhancing interoperability between domain-specific information modeling processes and, thus, software products for Building Information Modeling (BIM) is an important aspect to improve the lifecycle support of buildings and to facilitate the collaboration of the different disciplines across Architecture, Engineering, Construction and Operations (AECO). The Industry Foundation Classes (IFC) provide an open data exchange format for sharing information between these stakeholders. However, since the IFC standard has to cover a broad spectrum of concepts, it contains a large number of entities and is highly complex. Past studies have shown that

IFC-based exchanges of models are prone to an information loss due to reduction, simplification or interpretation when sharing data between multiple specialized software products (Bazjanac & Kiviniemi, 2007). One major issue is a potential mismapping between native BIM elements and IFC entities, which can arise through e.g. manual error during model creation or the reliance on default templates (Belsky, et al., 2016). Furthermore, CAD software products interpret specifications differently when processing in- and output data. When sharing BIM models with other teams, semantic integrity is a prerequisite for a seamless workflow and effective collaboration. Many specialized applications rely on accurate semantic information to perform their tasks, e.g. energy efficiency modeling (Schlueter & Thesseling, 2009; Ham & Golparvar-Fard, 2015) or code compliance checking (Eastman, et al., 2009). Inconsistent object classification has been identified to be a common interoperability issue between different BIM authoring software suites (Belsky et al., 2016; Lai & Deng, 2018). Researchers have started approaching this issue with methods from the area of machine and deep learning (Bloch & Sacks, 2018). These algorithms typically need labeled datasets to learn from. However, comprehensive and rich datasets in the domain of BIM and IFC are scarce, which makes the development and verification of such models difficult. In this work, the authors introduce a benchmark dataset of single-entity IFC files

covering a broad range of IFC classes. This dataset, named IFCNet1, should contribute to the standardization of performance .

-evaluations of future work in this domain. To evaluate the usefulness of IFCNet, three deep learning methods are trained to classify the entities and their performance is reviewed. The key contributions of this research paper can be summarized as follows: • A benchmark dataset for IFC entity classification, named IFCNet, is released. • The application of recent advances in the area of geometric deep learning to the classification of IFC elements is shown using three different approaches. • An evaluation of these deep learning methods is conducted to demonstrate the opportunities and challenges posed by IFCNet

3. The IFCNet Dataset

Figure 1: Example objects for each of the 20 classes of IFCNetCore.

To assemble IFCNet, around 1000 IFC models were collected from real-world projects, student works and online sources, such as the open IFC model repository of the university of Auckland (Dimyadi, et al., 2010). The models were created with different authoring software products,

2

Book 23 : Deep Learning for 3D Point Clouds: A Survey

Point cloud learning has lately attracted increasing attention due to its wide applications in many areas, such as computer vision, autonomous driving, and robotics. As a dominating technique in AI, deep learning has been successfully used to solve various 2d Vision problems. However, deep learning on point clouds is still in its infancy due to the unique challenges faced by the processing of point clouds with deep neural networks. Recently, deep learning on point clouds has become even more thriving, with numerous methods being proposed to address different problems in this area. To stimulate future research, this paper presents a comprehensive review of recent progress in deep learning methods for point clouds. It covers three major tasks, including 3D shape classification, 3D object detection and tracking, and 3D point cloud segmentation. It also presents comparative results on several publicly available datasets, together with insightful observations and inspiring future research directions.

-WITH the rapid development of 3D acquisition technologies, 3D sensors are becoming increasingly available and affordable, including various types of 3D scanners,

LiDARs,andRGB-D'Cameras(suchasKinect,RealSenseand Apple depth cameras) [1]. 3D data acquired by these sensors can provide rich geometric, shape and scale information [2], [3]. Complemented with 2D images, 3D data provides an opportunity for a better understanding of the surrounding environment for machines. 3D data has numerous applications in different areas, including autonomous driving, robotics, remote sensing, and medical treatment [4]. 3Ddatacan usually be represented with different formats, including depth images, point clouds, meshes, and volumetric grids. As a commonly used format,point cloud tion preserves the original geometric information in 3D space without any discretization. Therefore, it is the preferred representation for many scene understanding related applications

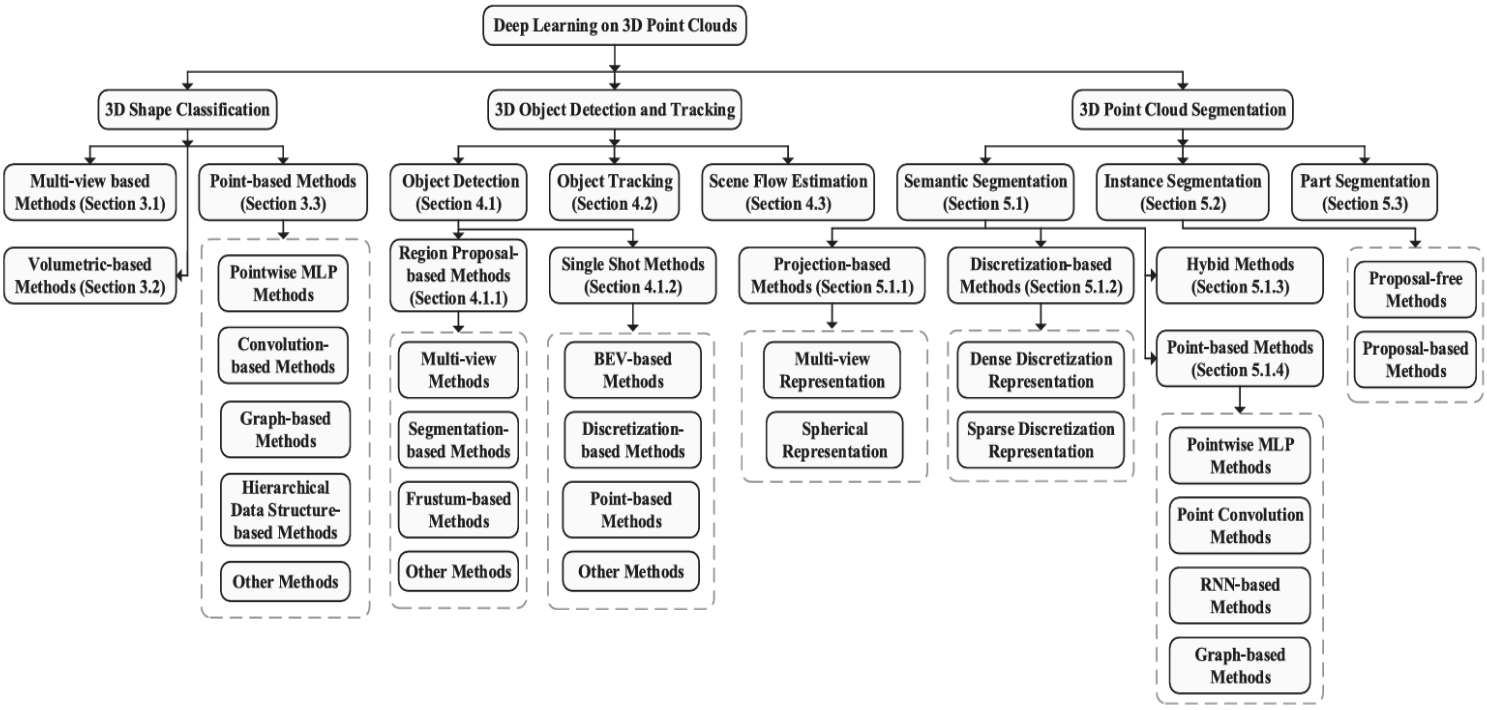


Fig. 1. A taxonomy of deep learning methods for 3D point clouds.

TABLE 1
A Summary of Existing Datasets for 3D Shape Classification, 3D Object Detection and Tracking, and 3D Point Cloud Segmentation

Datasets for 3D Shape Classification							
Name and Reference	Year	#Samples	#Classes	#Training	#Test	Type	Representation
McGill Benchmark [23]	2008	456	19	304	152	Synthetic	Mesh
Sydney Urban Objects [24]	2013	588	14	-	-	Real-World	Point Clouds
ModelNet10 [6]	2015	4899	10	3991	605	Synthetic	Mesh
ModelNet40 [6]	2015	12311	40	9843	2468	Synthetic	Mesh
ShapeNet [8]	2015	51190	55	-	-	Synthetic	Mesh
ScanNet [11]	2017	12283	17	9677	2606	Real-World	RGB-D
ScanObjectNN [7]	2019	2902	15	2321	581	Real-World	Point Clouds
Datasets for 3D Object Detection and Tracking							
Name and Reference	Year	#Scenes	#Classes	#Annotated Frames	#3D Boxes	Secne Type	Sensors
KITTI [14]	2012	22	8	15K	200K	Urban (Driving)	RGB & LiDAR
SUN RGB-D [25]	2015	47	37	5K	65K	Indoor	RGB-D
ScanNetV2 [11]	2018	1.5K	18	-	-	Indoor	RGB-D & Mesh
H3D [26]	2019	160	8	27K	1.1M	Urban (Driving)	RGB & LiDAR
Argoverse [27]	2019	113	15	44K	993K	Urban (Driving)	RGB & LiDAR
Lyft L5 [28]	2019	366	9	46K	1.3M	Urban (Driving)	RGB & LiDAR
A*3D [29]	2019	-	7	39K	230K	Urban (Driving)	RGB & LiDAR
Waymo Open [30]	2020	1K	4	200K	12M	Urban (Driving)	RGB & LiDAR
nuScenes [31]	2020	1K	23	40K	1.4M	Urban (Driving)	RGB & LiDAR
Datasets for 3D Point Cloud Segmentation							
Name and Reference	Year	#Points	#Classes[1]	#Scans	Spatial Size	RGB	Sensors
Oakland [32]	2009	1.6M	5(44)	17	-	N/A	MLS
ISPRS [33]	2012	1.2M	9	-	-	N/A	ALS
Paris-rue-Madame [34]	2014	20M	17	2	-	N/A	MLS
IQmulus [35]	2015	300M	8(22)	10	-	N/A	MLS
ScanNet [11]	2017	-	20(20)	1513	8×4×4	Yes	RGB-D
S3DIS [10]	2017	273M	13(13)	272	10×5×5	Yes	Matterport
Semantic3D [12]	2017	4000M	8(9)	15/15	250×260×80	Yes	TLS
Paris-Lille-3D [36]	2018	143M	9(50)	3	200×280× 30	N/A	MLS
SemanticKITTI [15]	2019	4549M	25(28)	23201/20351	150×100×10	N/A	MLS
Toronto-3D [37]	2020	78.3M	8(9)	4	260×350× 40	Yes	MLS
DALES [38]	2020	505M	8(9)	40	500×500×65	N/A	ALS

- Multi-view based methods project an unstructured point cloud into 2D images, while volumetric-based methods convert a point cloud into a 3D volumetric representation. Then, well-established 2D or 3D convolutional networks are leveraged to achieve shape classification. In contrast, point based methods directly work on raw point clouds without any voxelization or projection. Point-based methods do not introduce explicit information loss and become increasingly popular. Note that, this paper mainly focuses on point based methods, but also includes few multi-view based and volumetric-based methods for completeness.

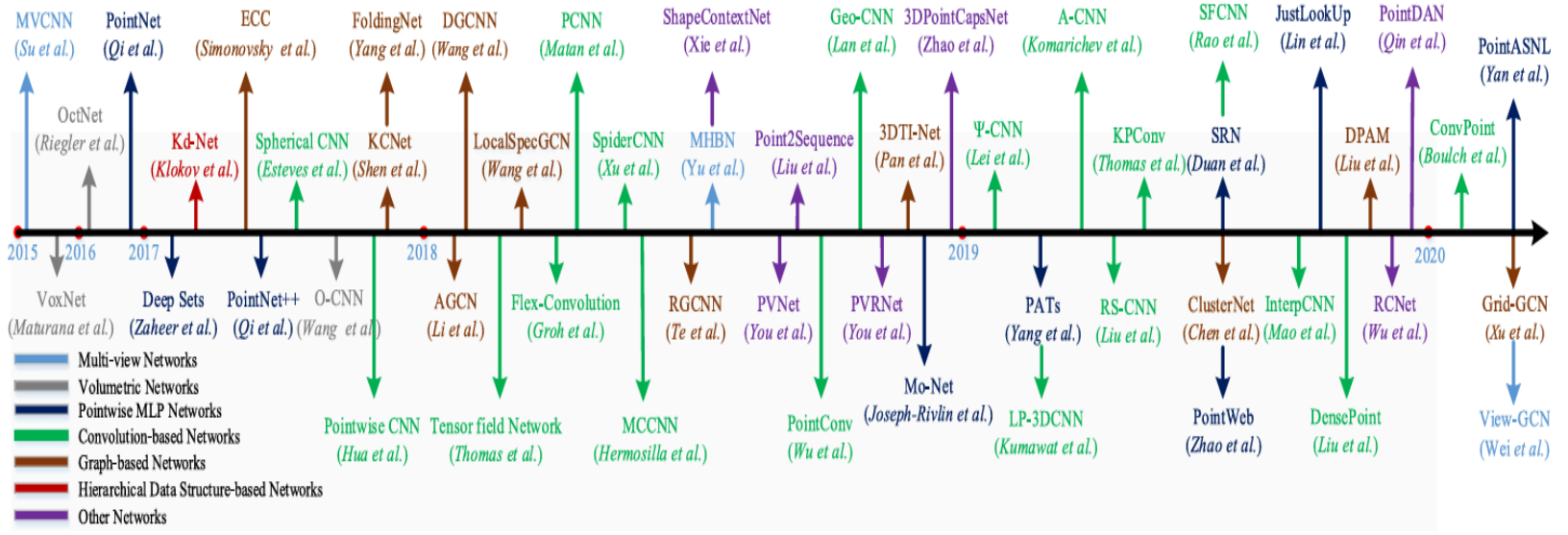


Fig. 2. Chronological overview of the most relevant deep learning-based 3D shape classification methods.

***Multi-View Based Methods :**

These methods first project a 3D shape into multiple views and extract view-wise features, and then fuse these features for accurate shape classification. How to aggregate multiple view-wise features into a discriminative global representation is a key challenge for these methods.

MVCNN [40] is a pioneering work, which simply max pools multi-view features into a global descriptor. However, max-pooling only retains the maximum elements from a specific view, resulting in information loss. MHBN [41] integrates local convolutional features by harmonized bilinear pooling to produce a compact global descriptor. Yang et al. [42] first leveraged a relation network to exploit the inter relationships (e.g., region-region relationship and view view relationship) over a group of views, and then aggregated these views to obtain a discriminative 3D object representation. In addition, several other methods [43], [44], [45], [46] have also been proposed to improve the recognition accuracy. Unlike previous methods, Wei et al. [47] used a directed graph in View-GCN by considering multiple views as graph nodes. The core layer consisting of local graph convolution, non-local message passing and selective view sampling is then applied to the constructed graph. The concatenation of max-pooled node

features at all levels is finally used to form the global shape descriptor

- **Volumetric-Based Methods:**

These methods usually voxelize a point cloud into 3D grids, and then apply a 3D Convolution Neural Network (CNN) on the volumetric representation for shape classification. Maturana et al. [48] introduced a volumetric occupancy network called VoxNet to achieve robust 3D object recognition. Wuetal. [6] proposedaconvolutional deepbelief-based 3DShapeNetstolearnthedistributionofpointsfromvarious 3D shapes (which are represented by a probability distribution of binary variables on voxel grids). Although encouraging performance has been achieved, these methods are unable to scale well to dense 3D data since the computation andmemoryfootprintgrowcubicallywiththeresolution. To this end, a hierarchical and compact structure (such as octree) is introduced to reduce the computational and memory costs of these methods. OctNet [49] first hierarchically partitions a point cloud using a hybrid grid-octree structure, which represents the scene with several shallow octrees along a regular grid. The structure of octree is encoded efficiently using a bit string representation, and the feature vector of each voxel is indexed by simple arithmetic. Wang et al. [50] proposed an Octree-based CNN for 3D shape classification. The average

normal vectors of a 3D model sampled in the finest leaf octants are fed into the network, and 3D-CNNisappliedontheoctants occupied by the 3D shape surface. Compared to a baseline network based on dense input grids, OctNet requires much less memory and runtime for high-resolution point clouds. Le et al. [51] proposed a hybrid network called PointGrid, which integrates the point and grid representation for efficient point cloud processing. A constant number of points is sampled within each embedding volumetric grid cell, which allows the network to extract geometric details by using 3D convolutions. Ben-Shabat et al. [52] transformed the input point cloud into 3D grids whicharefurtherrepresentedby3DmodifiedFisher tor (3DmFV) method, and then learned the global representation through conventional CNNarchitecture.

<u>* Point-Based Methods:</u>

According to the network architecture used for the feature learning of each point, methods in this category can be divided into pointwise MLP, convolution-based, graph based, hierarchical data structure-based methods and other typical methods.

Pointwise MLP Methods These methods model each point independently with several shared Multi-Layer Perceptrons (MLPs) and then aggregate a global feature using a symmetric aggregation function, as showninFig.3. Typical deep learning

methods for 2D images cannot be directly applied to 3D point clouds due to their inherent data irregularities. As a pioneering work,PointNet[5]directly takes point clouds as its input and achieves permutation invariance with a symmetric function. Specifically, PointNet learns pointwise features independently with several MLP layers and extracts global features with a max-pooling layer. Deep sets [53] achieves permutation invariance by summing up all representations and applying nonlinear transformations. Since features are learned independently for each point in PointNet [5], the local structural information between points cannot be captured. Therefore, Qi et al. [54] proposed a hierarchical net work PointNet++ to capture fine geometric structures from the neighborhood of each point. As the core of PointNet++ hierarchy, its set abstraction level is composed of three layers: the sampling layer, the grouping layer and the PointNet based learning layer. By stacking several set abstraction levels, Point Net++ learns features from a local geometric structure and abstracts the local features layer by layer. Because of its simplicity and strong representation ability, many networks have been developed based on PointNet [5]. The architecture of Mo-Net [55] is similar to PointNet [5] but it takes a finite set of moments as its input. Point Attention Trans formers (PATs) [56] represents each point by its own absolute position and relative positions with respect to its neighbors and learns high

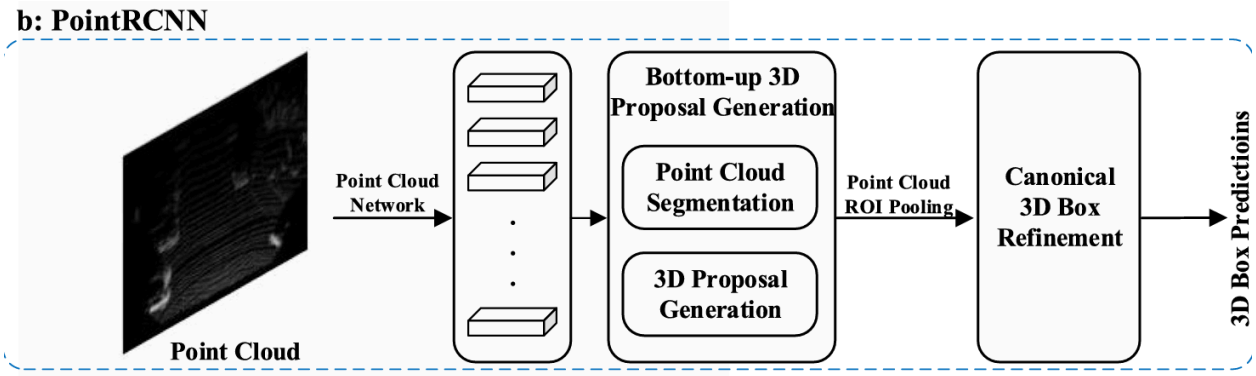
dimensional features through MLPs. Then, Group Shuffle Attention (GSA) is used to capture relations between points, and a permutation invariant, differentiable and train-able end-to-end Gumbel Subset Sampling (GSS) layer is developed to learn hierarchical features. Based on PointNet++ [54], PointWeb [57] utilizes the context of the local neighborhood to improve point features using Adaptive Feature Adjustment (AFA). Duan et al. [58] proposed a Structural Relational Network (SRN) to learn structural relational features between different local structures using MLP. Lin et al. [59] accelerated the inference process by constructing a lookup table for both input and function spaces learned by PointNet.Theinference time on the ModelNet and ShapeNet datasets is sped up by 1.5 ms and 32 times over PointNet on a moderate machine. SRINet [60] first projects a point cloud to obtain rotation invariant representations, and then utilizes PointNet-based backbone to extract a global feature and graph-based aggregation to extract local features. In PointASNL, Yan et al. [61] utilized an Adaptive Sampling (AS) module to adaptively adjust the coordinates and features of points sampled by the Furthest Point Sampling (FPS) algorithm, and proposed local-non-local(L-NL)module to capture the local and long range dependencies of these sampled points

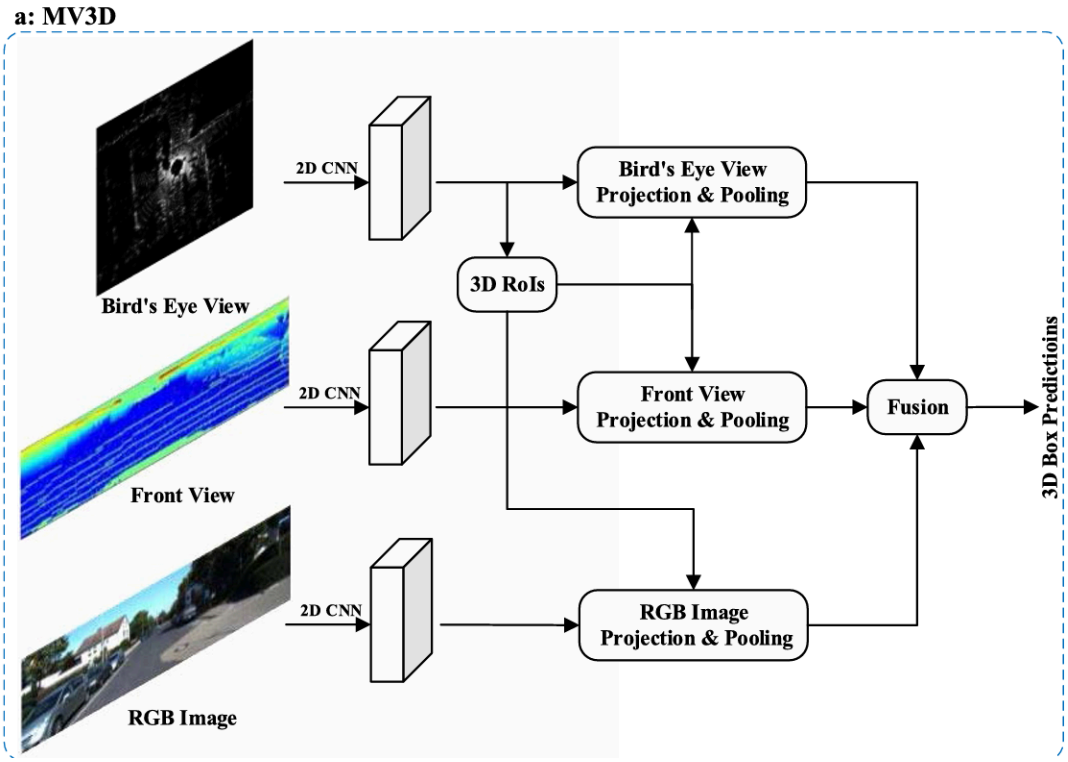
- **Convolution-Based Methods :**

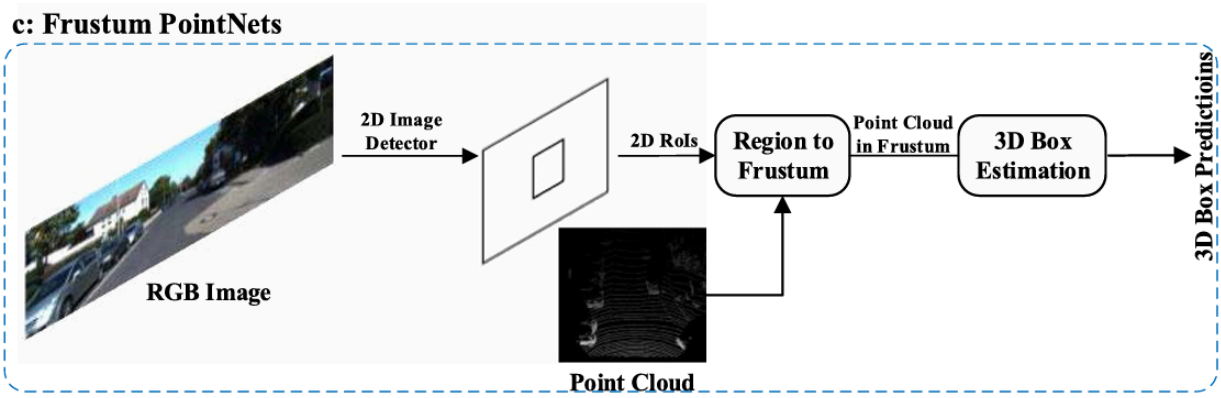
Compared with kernels defined on 2D grid structures (e.g., images), convolutional kernels for 3D point clouds are hard to design due to the irregularity of point clouds. According to the type of convolutional kernels, current 3D convolution methods can be divided into continuous and discrete con volution methods, as shown in Fig. 4. 3D Continuous Convolution Methods. These methods define convolutional kernels on a continuous space, where the weights for neighboring points are related to the spatial distribution with respect to the center point. 3Dconvolution can be interpreted as a weighted sum over a given subset. As the core layer of RS-CNN [62], RS-Conv takes a local subset of points around a certain point as its input, and the convolution is implemented using an MLP by learning themappingfromlow-levelrelations(suchaseuclid ean distance and relative position) to high-level relations between points inthe local subset. In [63], kernel elements are selected randomly in a unit sphere. An MLP-based continu ous function is then used to establish relation between the locations of the kernelelementsandthepointcloud.InDense Point[64], convolutionisdefinedasaSingle-LayerPerceptron (SLP) with a nonlinear activator. Features are learned by concatenating features from all previous layers to sufficiently exploit the contextual information. Thomas et al. [65] pro posed both rigid

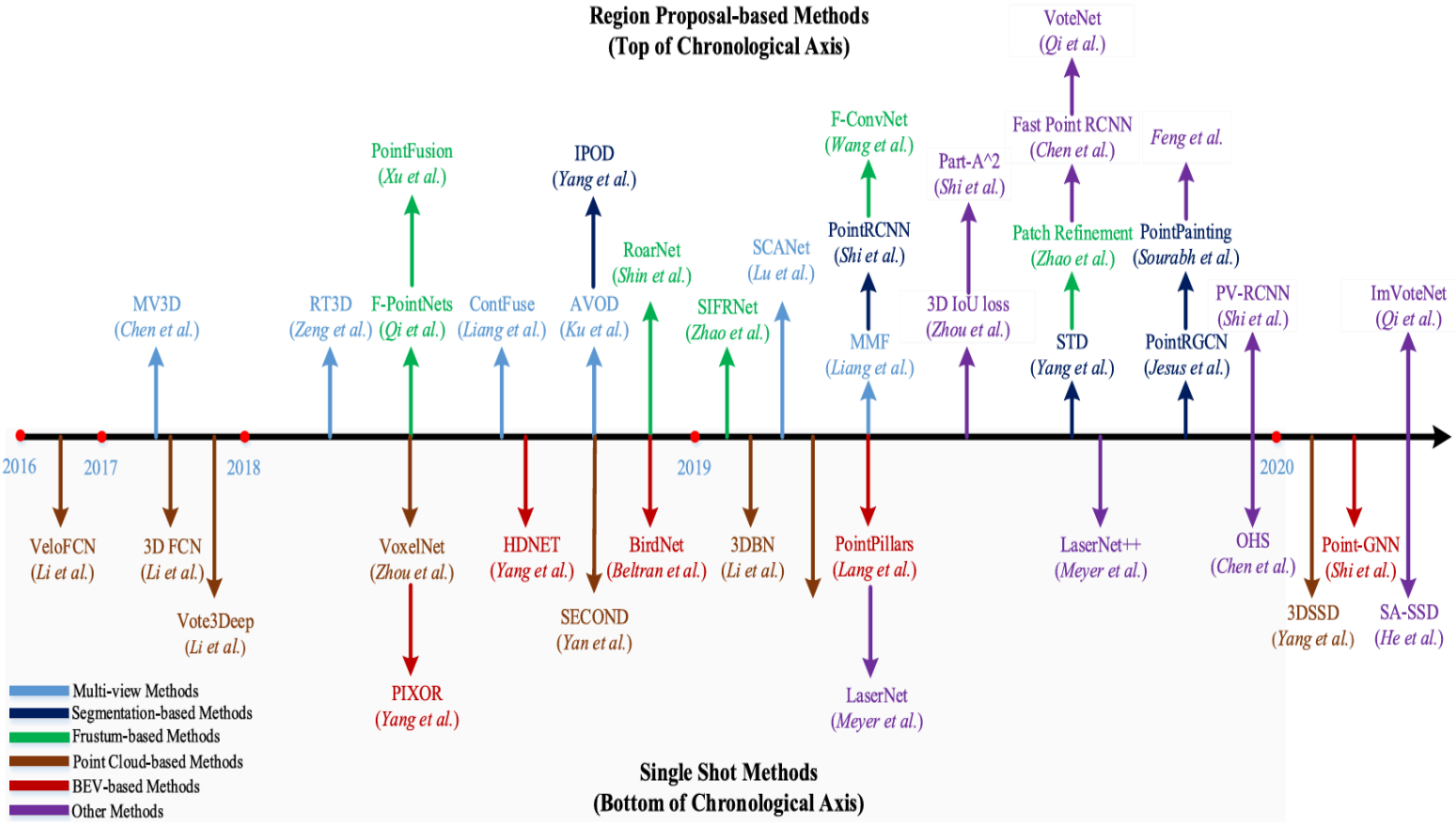
and deformable Kernel Point Convolution (KPConv) operators for 3D point clouds using a set of learn able kernel points. ConvPoint [66] separates the convolution kernel into spatial and feature parts. The locations of the spa tial part are randomly selected from a unit sphere and the weightingfunctionislearnedthroughasimpleMLP. Some methods also use existing algorithms to perform convolution. In PointConv [67], convolution is defined as a Monte Carlo estimation of the continuous 3D convolution with respect to an important sampling. The convolutional kernels consist of a weighting function (which is learned with MLP layers) and a density function (which is learned by a kernelized density estimation and an MLP layer). To improve memory and computational efficiency, the 3D convolution is further reduced into two operations: matrix multiplication and 2D convolution. With the same parameter setting, its memory consumption can be reduced by about 64 times. In MCCNN [68], convolution is considered as a Monte Carlo estimation process relying on a sample's density function (which is implemented with MLP). Poisson disk sampling is then used to construct a point cloud hierarchy. This convolution operator can be used to perform convolution between two or multiple sampling methods and can handle varying sampling densities. In SpiderCNN [69], SpiderConv is proposed to define convolution as the product of a step function and a Taylor expansion defined on the k nearest

neighbors. The step function captures the coarse geometry by encoding the local geodesic distance, and the Taylor expansion captures the intrinsic local geometric variations by interpolating arbitrary values at the vertices of a cube. Besides, a convolution network PCNN [70] is also proposed for 3D point clouds based on the radial basis function.






c: Frustum PointNets
RGB Image
2D Image Detector
2D RoIs
Region to Frustum
Point Cloud in Frustum
3D Box Estimation
3D Box Predictioins
Point Cloud


Region Proposal-based Methods
(Top of Chronological Axis)
VoteNet (Qi et al.)
F-ConvNet (Wang et al.)
Fast Point RCNN (Chen et al.)
Feng et al.
PointFusion (Xu et al.)
IPOD (Yang et al.)
Part-A^2 (Shi et al.)
PointRCNN (Shi et al.)
Patch Refinement (Zhao et al.)
PointPainting (Sourabh et al.)
RoarNet (Shin et al.)
SCANet (Lu et al.)
PV-RCNN (Shi et al.)
ImVoteNet (Qi et al.)
MV3D (Chen et al.)
RT3D (Zeng et al.)
F-PointNets (Qi et al.)
ContFuse (Liang et al.)
AVOD (Ku et al.)
SIFRNet (Zhao et al.)
3D IoU loss (Zhou et al.)
MMF (Liang et al.)
STD (Yang et al.)
PointRGCN (Jesus et al.)
2016
2017
2018
2019
2020
VeloFCN (Li et al.)
3D FCN (Li et al.)
Vote3Deep (Li et al.)
VoxelNet (Zhou et al.)
HDNET (Yang et al.)
BirdNet (Beltran et al.)
3DBN (Li et al.)
PointPillars (Lang et al.)
LaserNet++ (Meyer et al.)
OHS (Chen et al.)
Point-GNN (Shi et al.)
3DSSD (Yang et al.)
SA-SSD (He et al.)
SECOND (Yan et al.)
PIXOR (Yang et al.)
LaserNet (Meyer et al.)
Multi-view Methods
Segmentation-based Methods
Frustum-based Methods
Point Cloud-based Methods
BEV-based Methods
Other Methods
Single Shot Methods
(Bottom of Chronological Axis)

***3D POINT CLOUD SEGMENTATION:**

3D point cloud segmentation requires the understanding of both the global geometric structure and the fine-grained details of each point. According to the segmentation granularity, 3D point cloud segmentation methods can be classified into three categories: semantic segmentation (scene level), instance segmentation (object level) and part segmentation (part level).

*** 3DSemanticSegmentation:**

Given a point cloud, the goal of semantic segmentation is to separate it into several subsets according to the semantic meanings of points. Similar to the taxonomy for 3D shape classification (Section 3), there are four paradigms for semantic segmentation: projection-based, discretization based, point-based, and hybrid methods. The first step of both the projection and discretization based methods is to transform a point cloud to an intermediate regular representation, such as multi-view [181], [182], spherical [183], [184], [185], volumetric [166], [186], [187], permutohedral lattice [188], [189], and hybrid representations [190], [191], as shown in Fig. 11. The intermediate segmentation results are then projected back to the raw point cloud. In contrast, point-based methods directly work on irregular point clouds. Several representative methods are shown in Fig. 10.

TABLE 3
Comparative 3D Object Detection Results on the KITTI Test 3D Detection Benchmark

Method			Modality	Speed (fps)	Cars			Pedestrians			Cyclists		
					E	M	H	E	M	H	E	M	H
Region Proposal -based Methods	Multi-view Methods	MV3D [4]	L & I	2.8	74.97	63.63	54.00	-	-	-	-	-	-
		AVOD [126]	L & I	12.5	76.39	66.47	60.23	36.10	27.86	25.76	57.19	42.08	38.29
		ContFuse [127]	L & I	16.7	83.68	68.78	61.67	-	-	-	-	-	-
		MMF [128]	L & I	12.5	88.40	77.43	70.22	-	-	-	-	-	-
		SCANet [39]	L & I	11.1	79.22	67.13	60.65	-	-	-	-	-	-
		RT3D [131]	L & I	11.1	23.74	19.14	18.86	-	-	-	-	-	-
	Segmentation -based Methods	IPOD [132]	L & I	5.0	80.30	73.04	68.73	55.07	44.37	40.05	71.99	52.23	46.50
		PointRCNN [133]	L	10.0	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
		PointRGCN [134]	L	3.8	85.97	75.73	70.60	-	-	-	-	-	-
		PointPainting [135]	L & I	2.5	82.11	71.70	67.08	50.32	40.97	37.87	77.63	63.78	55.89
		STD [138]	L	12.5	87.95	79.71	75.09	53.29	42.47	38.35	78.69	61.59	55.30
	Frustum -based Methods	F-PointNets [139]	L & I	5.9	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01
		SIFRNet [140]	L & I	-	-	-	-	-	-	-	-	-	-
		PointFusion [142]	L & I	-	77.92	63.00	53.27	33.36	28.04	23.38	49.34	29.42	26.98
		RoarNet [143]	L & I	10.0	83.71	73.04	59.16	-	-	-	-	-	-
		F-ConvNet [144]	L & I	2.1	87.36	76.39	66.69	52.16	43.38	38.80	81.98	65.07	56.54
		Patch Refinement [145]	L	6.7	88.67	77.20	71.82	-	-	-	-	-	-
	Other Methods	3D IoU loss [146]	L	12.5	86.16	76.50	71.39	-	-	-	-	-	-
		Fast Point R-CNN [147]	L	16.7	84.80	74.59	67.27	-	-	-	-	-	-
		PV-RCNN [148]	L	12.5	90.25	81.43	76.82	-	-	-	-	-	-
		VoteNet [124]	L	-	-	-	-	-	-	-	-	-	-
		Feng et al. [149]	L	-	-	-	-	-	-	-	-	-	-
		ImVoteNet [150]	L & I	-	-	-	-	-	-	-	-	-	-
		Part-A^2 [151]	L	12.5	87.81	78.49	73.51	-	-	-	-	-	-
Single Shot Methods	BEV-based Methods	PIXOR [129]	L	28.6	-	-	-	-	-	-	-	-	-
		HDNET [152]	L	20.0	-	-	-	-	-	-	-	-	-
		BirdNet [153]	L	9.1	13.53	9.47	8.49	12.25	8.99	8.06	16.63	10.46	9.53
	Discretization -based Methods	VeloFCN [154]	L	1.0	-	-	-	-	-	-	-	-	-
		3D FCN [155]	L	<0.2	-	-	-	-	-	-	-	-	-
		Vote3Deep [156]	L	-	-	-	-	-	-	-	-	-	-
		3DBN [157]	L	7.7	83.77	73.53	66.23	-	-	-	-	-	-
		VoxelNet [136]	L	2.0	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
		SECOND [158]	L	26.3	83.34	72.55	65.82	48.96	38.78	34.91	71.33	52.08	45.83
		MVX-Net [159]	L & I	16.7	84.99	71.95	64.88	-	-	-	-	-	-
		PointPillars [137]	L	62.0	82.58	74.31	68.99	51.45	41.92	38.89	77.10	58.65	51.92
		SA-SSD [160]	L	25.0	88.75	79.79	74.16	-	-	-	-	-	-
	Point-based Methods	3DSSD [161]	L	25.0	88.36	79.57	74.55	54.64	44.27	40.23	82.48	64.10	56.99
	Other Methods	LaserNet [162]	L	83.3	-	-	-	-	-	-	-	-	-
		LaserNet++ [163]	L & I	26.3	-	-	-	-	-	-	-	-	-
		OHS-Dense [164]	L	33.3	88.12	78.34	73.49	47.14	39.72	37.25	79.09	62.72	56.72
		OHS-Direct [164]	L	33.3	86.40	77.74	72.97	51.29	44.81	41.13	77.70	63.16	57.12
		Point-GNN [125]	L	1.7	88.33	79.47	72.29	51.92	43.77	40.14	78.60	63.48	57.08

- **Quality Control :**

To address the problem of quality control between a Building Information Model (BIM) and point cloud data, the workflow generally involves automating the comparison of the model (BIM) parameters with the reference point cloud to identify discrepancies, or "errors," where points in the point cloud are not represented in the model. The goal is to minimize these errors by improving the model.

- **Automated Point Cloud Comparison: Use point cloud analysis tools or custom scripts to compare the reference point cloud data against the BIM geometry.**

- **Calculate Deviations: Determine the distance between the point cloud and corresponding BIM objects. If a point in the cloud doesn't match a modeled object, it's considered an "error" or unmodeled point.**

- **Bounding Box Method: For each BIM object (walls, floors, etc.), define bounding boxes and check how many point cloud points lie within the boxes.**
- **Mesh Comparison: Convert BIM geometry to a mesh and compare the point cloud points to the mesh surface using tolerance levels.**

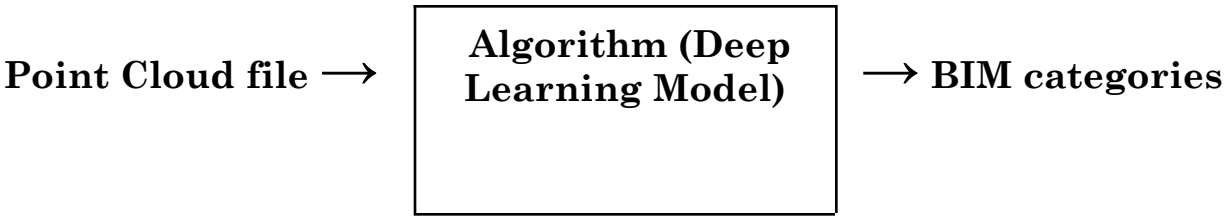
- **Clash Detection Tools: Use existing Revit clash detection tools for rough comparisons or specialized tools like Navisworks to find missing model elements.**
- **Automation minimal human input**
- **Point cloud=>3D reconstruction geometry =>BIM model**
- **Automated Quality Control System:**

  -Visual Control

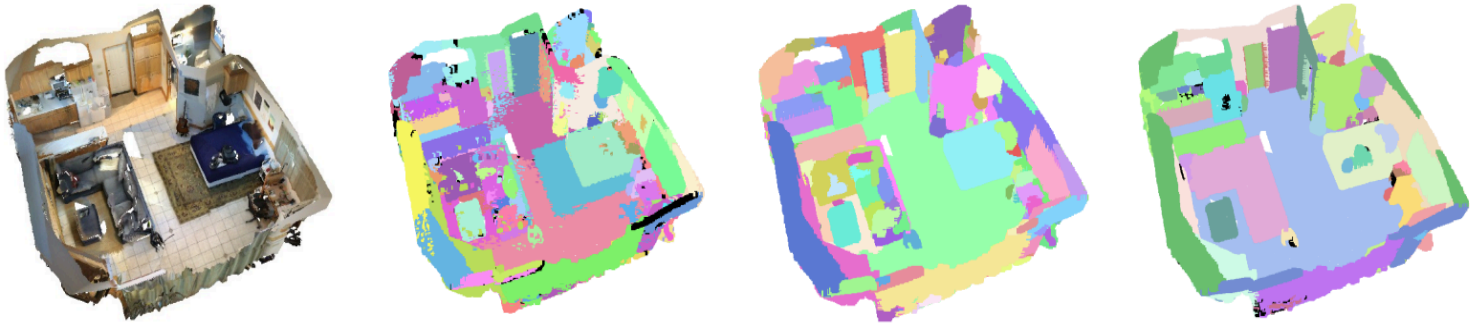
  -Autodesk model checker(BIM interoperability tools)

  -Clash detection

- **Task 1 : Automate the quality control(check) of BIM generated by the user SCAN VS BIM**
- **Task 2 : Automate the construction of BIM : SCAN TO BIM**
- **Task 1.0: Identification of categories in Point Cloud (.rcp file)**



• Segment Anything in 3D scene : SAM 3D



3D Scene SAM3D SAM3D (ensembled) Ground-truth

- **SAM3D, a novel framework that is able to predict masks in 3D point clouds by leveraging the Segment-Anything Model (SAM) in RGB images without further training or finetuning. For a point cloud of a 3D scene with posed RGB images, we first predict segmentation masks of RGB images with SAM, and then project the 2D masks into the 3D points. Later, we merge the 3D masks iteratively with a bottom-up merging approach. At each step, we merge the point cloud masks of two adjacent frames with the bidirectional merging approach. In this way, the 3D masks predicted from different frames are gradually merged into the 3D masks of the whole 3D scene. Finally, we can optionally ensemble the result from our SAM3D with the over-segmentation results based on the geometric**

information of the 3D scenes. Our approach is experimented with ScanNet dataset and qualitative results demonstrate that our SAM3D achieves reasonable and fine-grained 3D segmentation results without any training or finetuning of SAM .

Input:

- **3D Point Cloud: This is the primary input to SAM3D. It represents the geometry of the scene in 3D space.**
- **RGB Images (posed): SAM3D requires RGB images that correspond to the 3D scene. These images can either be captured directly along with the point cloud (if available) or rendered from the point cloud using a virtual camera.**

Process:

- **Segmenting in 2D with SAM: SAM3D uses the Segment Anything Model (SAM) to predict segmentation masks on RGB images. These RGB images are aligned with the 3D point cloud (i.e., the camera poses are known), which allows for projecting the 2D masks onto the corresponding 3D points.**

- **Projecting 2D Masks to 3D: The 2D masks predicted by SAM from the RGB images are then projected onto the 3D points in the point cloud, effectively labeling the 3D points.**

Output:

- **3D Segmentation: SAM3D's final output is a segmentation of the 3D point cloud, where each point in the cloud is labeled based on the 2D masks generated by SAM from the RGB images.**

How SAM3D Works:

1. **Predicting Masks in 2D (RGB Images):**
    - **SAM3D uses the Segment Anything Model (SAM) to predict segmentation masks on RGB images of a 3D scene. SAM operates only on 2D data, meaning it can segment objects or regions within the 2D images effectively without further training or finetuning.**
2. **Projecting 2D Masks to 3D:**
    - **Once SAM predicts segmentation masks in the 2D images, SAM3D projects these 2D masks onto the corresponding 3D points in the point cloud. This is possible because the RGB images are aligned with the 3D scene (i.e., the camera poses are known), so each 2D pixel can be mapped back to its corresponding 3D point.**

3. **Merging 3D Masks:**
    - **SAM3D doesn't stop with just projecting the 2D masks onto the point cloud. Since the scene may be captured from multiple RGB images at different angles, SAM3D iteratively merges the 3D masks from different views using a bidirectional merging approach. This means that the 3D segmentation masks from different frames are merged step by step to create a consistent 3D segmentation of the entire scene.**
4. **Bottom-Up Merging:**
    - **SAM3D uses a bottom-up merging strategy, where adjacent frames are merged progressively. This iterative process refines the 3D masks and builds up a comprehensive mask for the entire 3D point cloud.**

- **3D segmentation aims to predict point-level semantic labels of the 3D point cloud of a scene. Previous approaches either develop semantic segmentation approaches on point cloud [10,12,13,16], or leverage features extracted from RGB images by cross-modal fusion between 2D and 3D [3, 5, 8]. While 3D point clouds provide geometric information that cannot be obtained from images, the 2D visual perception models have demonstrated great success because of the large-scale RGB images and corresponding annotations as training data. Recently, the Segment Anything Model (SAM) [7] has been**

introduced for promptable, fine grained segmentation of RGB images. SAM has achieved astonishing results in fine-grained segmentation of stuff, objects, and object parts in RGB images of complex scenes. Follow-up works SAD [1] leverages SAM to segment the depth maps. With the cues with enhanced geometry information, it addresses the over-segmentation problem by SAM. Another work Anything-3D [15] combines SAM with BLIP [9], Stable Diffusion [14], and Point-E [11] for single-view conditioned 3D reconstruction of objects in an image. In this project, we investigate how to leverage the segmentation results of SAM to generate fine-grained 3D masks for 3D scenes. We propose SAM3D, which projects the SAM segmentation masks from RGB frames into the 3D scene, and merge the results iteratively to get the segmentation masks of the whole 3D scene. Specifically, given the point cloud of a 3D scene with posed RGB images, we first predict 2D segmentation masks by applying SAM on the RGB images, and then project the 2D masks into 3D. Then the 3D masks of partial scenes are merged iteratively in a bottom-up way to generate the 3D masks of the whole scene. We propose a bidirectional merging approach to merge the two masks from adjacent frames at each merging step. Finally, we ensemble the 3D masks with the over segmentation masks of the scene. Our proposed approach

takes advantage of SAM and does not require training or fine tuning the model. We demonstrate the qualitative results on 3D scenes of ScanNet [2] dataset.

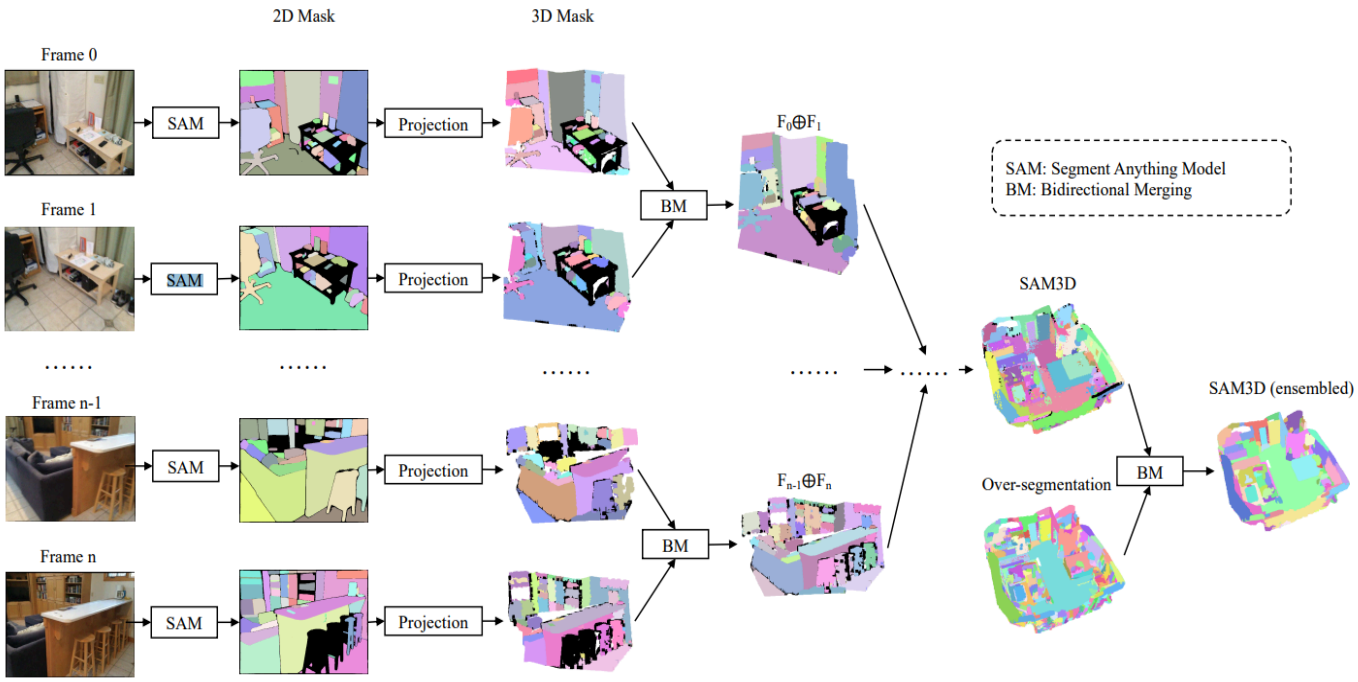


Figure 2. **Overview.** For input images, we first use SAM to generate 2D masks, and then map the 2D masks to 3D. Then we iteratively merge adjacent point clouds with the Bidirectional Merging (BM) approach until we obtain the 3D masks of the whole scene. We finally merge the SAM3D result with the over-segmentation masks to obtain an ensembled result.

Github : https://github.com/Pointcept/SegmentAnything3D

2.1. Single-frame 3D masks from SAM

We first apply SAM [7] automatic mask generation approach on the single-frame RGB images to obtain the pixel-level masks of an image. SAM returns nested masks with different granularity: whole, part, and subpart. In order to obtain non-overlapping masks, if a pixel is covered by multiple masks, we assign the mask ID with the highest predicted IoU to the pixel. Then we map the 2D masks to the 3D space according to the depth of each pixel provided by the RGB-D image. Specifically, the mapping is formulated as,

$$[x_i, y_i, z_i]^T = \mathcal{R}^{-1}\mathcal{M}^{-1}s[u_i, v_i, 1]^T - \mathcal{R}^{-1}T, \quad (1)$$

where $[u_i, v_i, 1]$ and $[x_i, y_i, z_i]$ are 2D homogeneous coordinates and projected 3D homogeneous coordinates, respectively. $\mathcal{M}$ is the intrinsic camera calibration matrix. $\mathcal{R}$ and $\mathcal{T}$ are the rotation matrix and translation matrix obtained from the extrinsic camera pose matrix, and s is a scaling factor. Finally, we apply grid pooling [17] to downsample the point cloud masks.

2.2. Bidirectional Merging between Two Point Clouds

Given the 3D masks derived from the previous subsection, we introduce a bidirectional merging approach to merge the mask predictions from different frames, as shown in Figure 4.

Specifically, we denote the point cloud from two adjacent frames as $X^1 = \{x_1^1, x_2^1, \cdots, x_m^1\}$ and $X^2 = \{x_1^2, x_2^2, \cdots, x_n^2\}$, where m and n are the numbers of points in X^1 and X^2, respectively. We compute the correspondence mapping M between X^1 and X^2. The points x_i^1 and x_j^2 are a pair of matched points between X^1 and X^2 if $(i, j) \in M$. For each mask id m in X^1, we first find the points in X^1 within the same 3D mask, denoted as Q^1. The number of points with mask id m in X^1 is denoted as σ_m^1. We then leverage the correspondence mapping M to find their corresponding points in X^2, denoted as Q^2. For each mask id n of the points in Q^2, we denote σ_{mn}^{1-2} as the number of points in Q^2 whose mask id is n. Similarly, we denote σ_n^2 as the number of points in X^2 whose mask id is n. If $\sigma_{mn}^{1-2} > \delta \times min(\sigma_m^1, \sigma_n^2)$ (where δ is the threshold and it is set to 0.5 in our experiments), it means that the mask with id m from X^1 and the mask with id n from X^2 are highly overlapping masks. So we merge the masks by assigning n as the new mask id for points in X^2 whose previous mask ids are m. This process is followed by a symmetric process where we iterate the mask ids in X^2 and query the points and mask ids in X^1.

• Self Attention :

Self-attention is a mechanism used in machine learning, particularly in natural language processing (NLP) and computer vision tasks, to capture dependencies and relationships within input sequences. It allows the model to identify and weigh the importance of different parts of the input sequence by attending to itself.

- **Self-attention operates by transforming the input sequence into three vectors: query, key, and value. These vectors are obtained through linear transformations of the input. The attention mechanism calculates a weighted sum of the values based on the similarity between the query and key vectors. The resulting weighted sum, along with the original input, is then passed through a feed-forward neural network to produce the final output. This process allows the model to focus on relevant information and capture long-range dependencies.**

Self-attention is closely related to other concepts in machine learning and artificial intelligence:

- **Transformer: Self-attention is a key component of the Transformer model, a powerful architecture that has achieved state-of-the-art results in various NLP and computer vision tasks.**
- **Attention Mechanism: Self-attention is a specific type of attention mechanism that allows the model to selectively focus on relevant information.**
- **BERT (Bidirectional Encoder Representations from Transformers): BERT is a pre-trained Transformer model that utilizes self-attention to capture contextual information in natural language.**

- **Transformers:**

-A transformer model is a neural network that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence

- Transformer models apply an evolving set of mathematical techniques, called attention or

self-attention, to detect subtle ways even distant data elements in a series influence and depend on each other.

-A transformer is a deep learning architecture developed by researchers at Google and based on the multi-head attention mechanism, proposed in a 2017 paper "Attention Is All You Need".[1] Text is converted to numerical representations called tokens, and each token is converted into a vector via looking up from a word embedding table.[1] At each layer, each token is then contextualized within the scope of the context window with other (unmasked) tokens via a parallel multi-head attention mechanism allowing the signal for key tokens to be amplified and less important tokens to be diminished.

Transformers have the advantage of having no recurrent units, and therefore require less training time than earlier recurrent neural architectures (RNNs) such as long short-term memory (LSTM).[2] Later variations have been widely adopted for training large language models (LLM) on large (language) datasets, such as the Wikipedia corpus and Common Crawl.[3]

Transformers were first developed as an improvement over previous architectures for machine translation,[4][5] but have found many applications since then. They are used in large-scale natural language processing, computer vision (vision transformers), reinforcement learning,[6][7] audio,[8] multi-modal processing, robotics,[9] and even playing chess.[10] It has also led to the development of pre-trained systems, such as generative pre-trained transformers (GPTs)[11] and BERT[12] (Bidirectional Encoder Representations from Transformers)

- **Vision Transformer :**

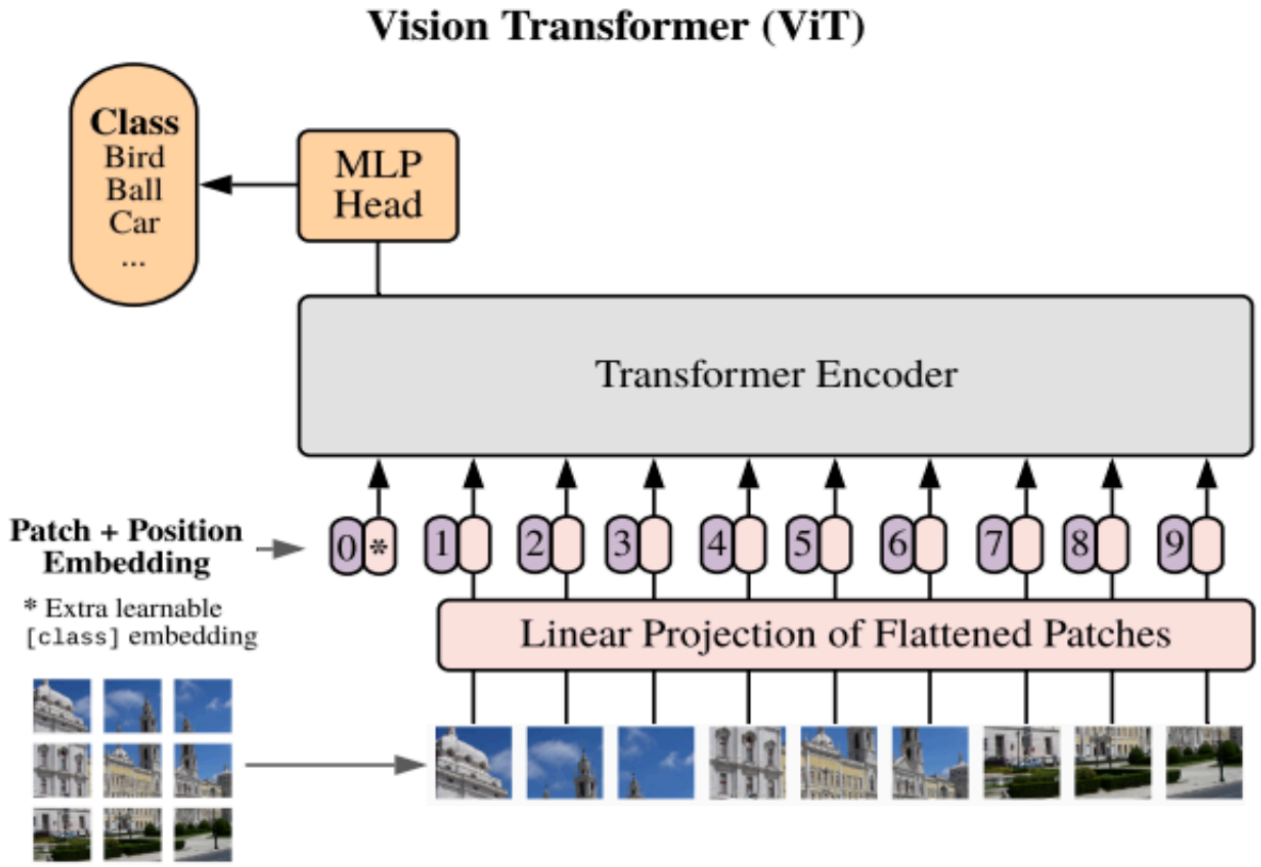
A vision transformer (ViT) is a transformer designed for computer vision.[1] A ViT breaks down an input image into a series of patches (rather than breaking up text into tokens), serializes each patch into a vector, and maps it to a smaller dimension with a single matrix multiplication. These vector embeddings are then processed by a transformer encoder as if they were token embeddings.

ViT were designed as alternatives to convolutional neural networks (CNN) in computer vision applications. They have different inductive biases, training stability, and data efficiency.[2] Compared to CNN, ViT is less data efficient, but has higher capacity. Some of the largest modern computer vision models are ViT, such as one with 22B parameters.[3][4]

- **how Vision Transformers work:**

1. **Patch Embedding: An image is divided into fixed-size patches. Each patch is then flattened and projected into a higher-dimensional space using a linear embedding layer. This converts the 2D image data into a sequence of patch embeddings.**
2. **Position Encoding: Since transformers do not inherently capture the positional information of patches, positional embeddings are added to the patch embeddings to retain spatial information.**

3. **Transformer Encoder: The sequence of patch embeddings (with positional encodings) is fed into a standard transformer encoder, which consists of layers of self-attention and feed-forward networks. The self-attention mechanism allows the model to learn relationships between different patches regardless of their spatial distance.**
4. **Classification or Regression Head: The output from the transformer encoder is then used for specific tasks, such as classification, by adding a classification head or other task-specific layers.**



- **Patch Embeddings: Each patch is flattened and projected into a higher-dimensional embedding space.**
- **Add [CLS] Token: A special [CLS] token is prepended to the sequence of patch embeddings.**

- **Positional Encoding: Positional encodings are added to retain spatial information about the patches.**
- **Transformer Encoder: The sequence (including the [CLS] token) is passed through multiple transformer layers.**
- **[CLS] Output: After passing through the transformer, the output of the [CLS] token is extracted. This output is assumed to capture a global representation of the entire image.**
- **Classification Head: The [CLS] token output is passed to a fully connected (FC) layer or classification head to produce the final classification (e.g., the predicted label for the image).**

Structure of the Transformer Encoder

- **A transformer encoder consists of several identical layers stacked on top of each other. Each of these layers has two main subcomponents:**

1. Multi-Head Self-Attention Mechanism

2. Feed-Forward Neural Network (FFN)

2.1.2. Multi-head attention

Multi-head attention mechanism (Vaswani et al., 2017) has been proposed to model the complex relationships of token entities from different aspects. To be specific, the input sequence X is linear transformed into h groups of $\{K_i, Q_i, V_i\}_{i=0}^{h-1}$, each group repeats the self-attention process. The final output is produced by projecting the concatenation of the outputs from the h groups with a weight matrix $W \in \mathbb{R}^{hd^V \times d}$. The overall process can be described as:

$$\mathrm{MultiHeadAtt}\,(X) = [\mathrm{Att}_0\,(X), \mathrm{Att}_1\,(X), \ldots, \mathrm{Att}_{h-1}\,(X)]W$$

$$\mathrm{Att}_i\,(X) = \mathrm{softmax}\left(\frac{Q_i \cdot K_i^T}{\sqrt{d_i^Q}}\right) \times V_i$$

2.1.1. Self-attention

Self-attention is one of the core mechanisms of transformer, which exists in both encoder and decoder blocks. Taking a sequence of entity tokens $X=\{x_0, x_1, \ldots, x_n\}$ as input (the entity tokens can be word sequence in NLP or video clips in the vision area), self-attention layer first linearly transforms the input tokens into three different vectors: key vector $K \in \mathbb{R}^{n \times d^K}$, query vector $Q \in \mathbb{R}^{n \times d^Q}$ and value vector $V \in \mathbb{R}^{n \times d^V}$ (e.g. $d^K=d^Q=d^V=512$ in practice). The output is produced via $\mathrm{Att}\,(X) = \mathrm{softmax}\,(\frac{Q \cdot K^T}{\sqrt{d^Q}} \times V)$ where $Q \cdot K^T$ is to capture the relevance score between different entities, $\sqrt{d^Q}$ is to reduce the score for gradient stability, softmax operation is to normalize the result for probability distribution and finally, multiplying with V is to obtain the weighted value matrix.

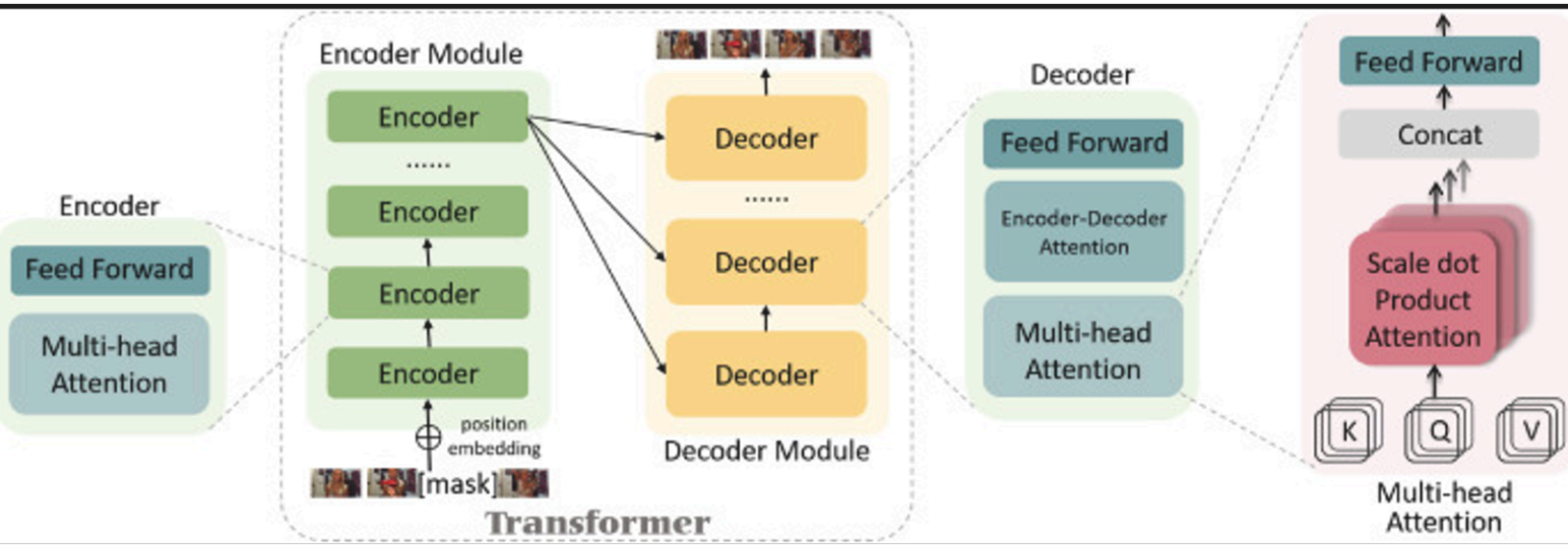


Fig. 1. An overview of the standard transformer architecture. The whole transformer is composed of encoder module and decoder module, with several encoders and decoders stacked in each module respectively. Each encoder consists of a multi-head attention layer and a feed forward layer, while each decoder additionally contains a encoder-decoder attention layer. The multi-head attention mechanism is shown in the right most column, which transfers the input sequence into *h* groups of {K, Q, V} and concatenates the self-attention outputs of each group as the final output.

- **The standard transformer consists of several encoder blocks and decoder blocks. Each encoder block contains a self-attention layer and a feed forward layer, while each decoder block contains**

an encoder-decoder attention layer in addition to the self-attention and feed forward layers.

- **Volumetric CNN :**

3D Convolutions: Instead of applying 2D convolutions on images, Volumetric CNNs use 3D convolutional filters. This means that the convolutional filter slides in three dimensions (height, width, depth) over the input data, allowing the network to capture spatial and structural features within a 3D space.

3D Pooling: Similar to max or average pooling in 2D CNNs, Volumetric CNNs employ 3D pooling layers that reduce the size of the 3D data by downsampling across all three dimensions. This is essential to reduce the computational complexity and extract hierarchical features.

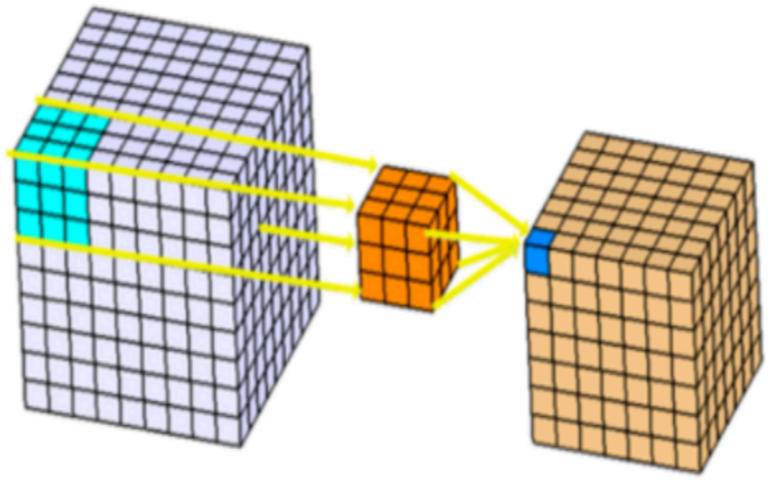
Input Data: Volumetric CNNs process volumetric data, where each input has three spatial dimensions, typically represented as cubes or 3D grids. Examples of such input data include:

- **Medical images: MRI or CT scans produce 3D volumetric data of the human body.**
- **Point clouds: Data used in applications like autonomous driving and 3D modeling.**
- **Voxel data: A 3D analog to pixels, often used in 3D computer graphics or geometric modeling.**

• 3D convolution :

3D Convolutions

3D convolutions applies a 3 dimentional filter to the dataset and the filter moves 3-direction (x, y, z) to calcuate the low level feature representations. Their output shape is a 3 dimentional volume space such as cube or cuboid. They are helpful in event detection in videos, 3D medical images etc. They are not limited to 3d space but can also be applied to 2d space inputs such as images.



2D Convolutions

On image datasets, mostly 2D Convolutional filters are used in CNN architectures. The main idea of 2D convolutions is that the the convolutional filter moves in 2-directions (x,y) to calculate low dimentional features from the image data. The output shape i also a 2 dimentional matrix.

- **Multiview CNN :**

A Multiview CNN (MVCNN) is a convolutional neural network architecture that is designed to handle data from multiple perspectives or views, particularly for tasks involving 3D object recognition, classification, or reconstruction. Instead of processing 3D data directly (like Volumetric CNNs), Multiview CNNs rely on multiple 2D projections of a 3D object from different angles and fuse the information from these views to perform the task.

The key idea behind Multiview CNNs is that 3D objects can be represented by a collection of 2D views (e.g., images of the object from different angles), and a CNN can process each of these views to learn features that represent the underlying 3D structure.

1. **The input to a Multiview CNN consists of several 2D images taken from different angles (viewpoints) around the 3D object. These views might be captured by rendering a 3D model into 2D or using multiple cameras in real-world settings.**
2. **Processing Each View: Each 2D view is processed individually by a standard 2D CNN to extract feature maps. The same CNN is typically applied to all the views, so they**

share weights and extract consistent features from each view.

3. **Feature Aggregation: After processing each view separately, the Multiview CNN aggregates the feature maps or representations from the different views. This is often done using a pooling operation (e.g., max pooling, average pooling) over the views, or more complex fusion techniques like attention mechanisms, to combine the features into a single, unified representation of the object.**
4. **Classification or Other Tasks: Once the views have been aggregated, the resulting combined feature vector is passed through fully connected layers for tasks like classification, detection, or segmentation.**

• MLP :

A multilayer perceptron (MLP) is a name for a modern feedforward artificial neural network, consisting of fully connected neurons with a nonlinear activation function, organized in at least three layers, notable for being able to distinguish data that is not linearly separable

-An artificial neural network (ANN) is a machine learning model inspired by the structure and function of the human brain's interconnected network of neurons. It consists of interconnected nodes called artificial neurons, organized into layers. Information flows through the network, with each neuron processing input signals and producing an output signal that influences other neurons in the network.

A multi-layer perceptron (MLP) is a type of artificial neural network consisting of multiple layers of neurons. The neurons in the MLP typically use nonlinear activation functions, allowing the network to learn complex patterns in data. MLPs are significant in machine learning because they can learn nonlinear relationships in data, making

them powerful models for tasks such as classification, regression, and pattern recognition.

-A neural network consists of interconnected nodes, called neurons, organized into layers. Each neuron receives input signals, performs a computation on them using an activation function, and produces an output signal that may be passed to other neurons in the network. An activation function determines the output of a neuron given its input. These functions introduce nonlinearity into the network, enabling it to learn complex patterns in data.

The network is typically organized into layers, starting with the input layer, where data is introduced. Followed by hidden layers where computations are performed and finally, the output layer where predictions or decisions are made.

Neurons in adjacent layers are connected by weighted connections, which transmit signals from one layer to the next. The strength of these connections, represented by weights, determines how much influence one neuron's

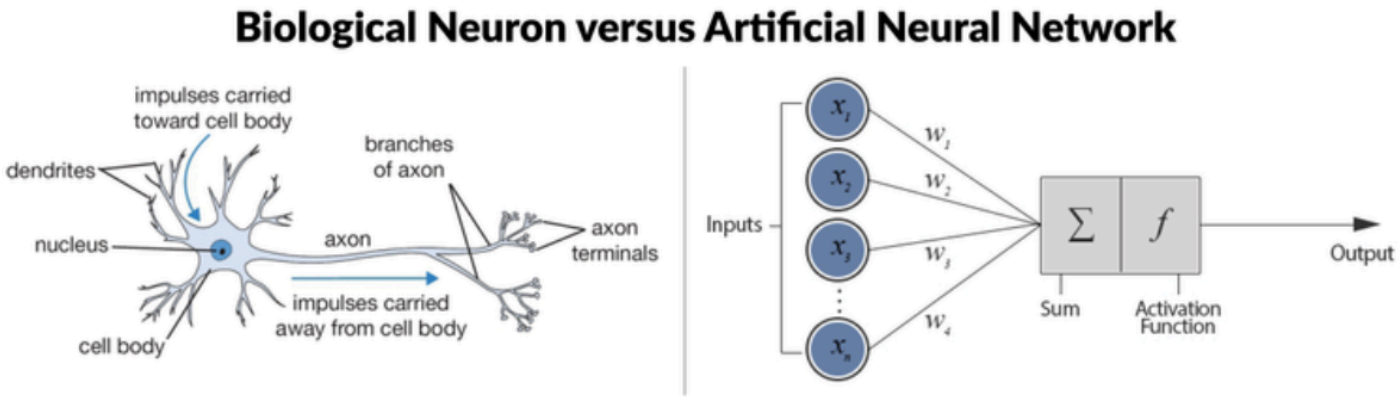
output has on another neuron's input. During the training process, the network learns to adjust its weights based on examples provided in a training dataset. Additionally, each neuron typically has an associated bias, which allows the neuron to adjust its output threshold.

Neural networks are trained using techniques called feedforward propagation and backpropagation. During feedforward propagation, input data is passed through the network layer by layer, with each layer performing a computation based on the inputs it receives and passing the result to the next layer.

Backpropagation is an algorithm used to train neural networks by iteratively adjusting the network's weights and biases in order to minimize the loss function. A loss function (also known as a cost function or objective function) is a measure of how well the model's predictions match the true target values in the training data. The loss function quantifies the difference between the predicted output of the model and the actual output,

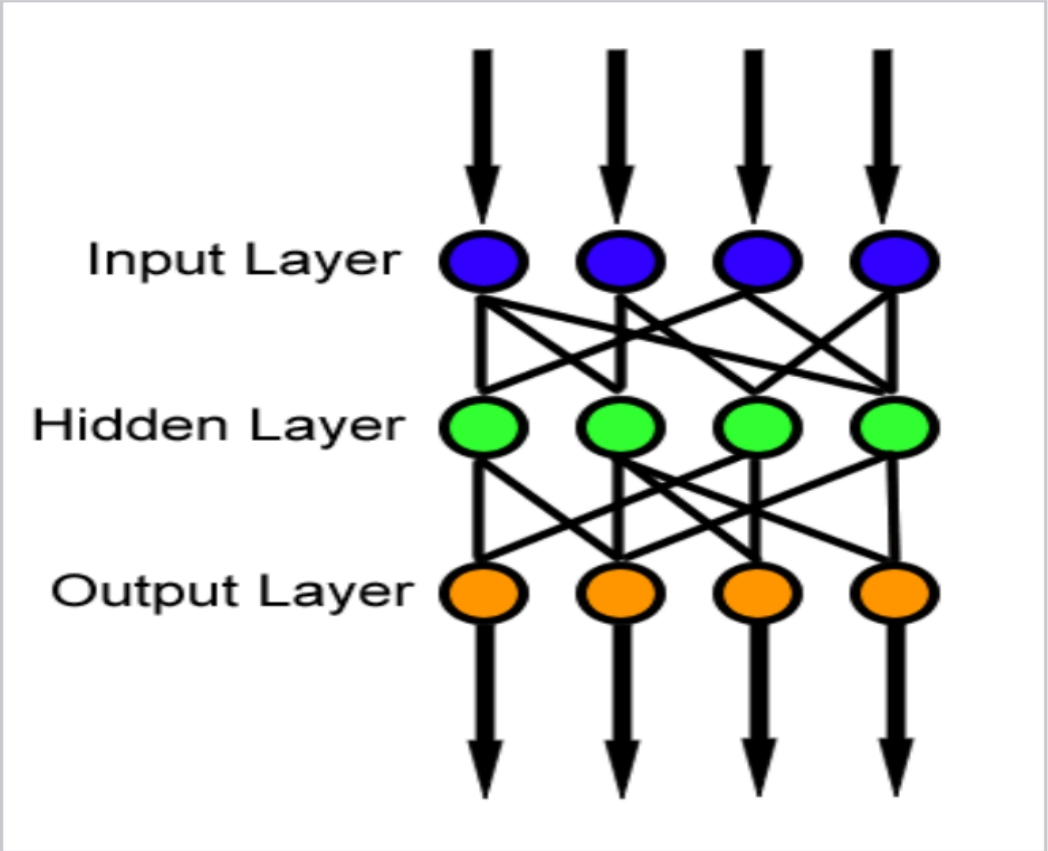
providing a signal that guides the optimization process during training.

The goal of training a neural network is to minimize this loss function by adjusting the weights and biases. The adjustments are guided by an optimization algorithm, such as gradient descent.



- **Types of Neural Network**
  - **Feedforward Neural Networks (FNN)** : These are the simplest form of ANNs, where information flows in one direction, from input to output. There are no cycles or loops in the

network architecture. Multilayer perceptrons (MLP) are a type of feedforward neural network.



In a feedforward network, information always moves one direction; it never goes backwards.

- **Learning occurs by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result.**

This is an example of supervised learning, and is carried out through backpropagation.

• Backpropagation vs Feedback Connections :

- **Backpropagation is a training algorithm that adjusts the weights in the network based on the error (difference between the predicted and actual output). It helps the network learn from mistakes.**
- **Feedback Connections, on the other hand, refer to a network structure where the output of one layer or time step can feed back into earlier layers (as seen in Recurrent Neural Networks).**

-In FNNs, there are no feedback connections between layers or neurons because:

- **The input flows in one direction, from input to output.**
- **The output of a layer doesn't loop back to earlier layers**

-Architecture: FNNs are the simplest type of artificial neural network. They consist of an input layer, one or more hidden layers, and an output layer. Connections between

nodes only move in one direction—forward—from input to output.

Data Processing: FNNs process input data in a fixed way. Each input is processed independently of any other input, and there is no memory of previous inputs.

Use Cases: FNNs are commonly used for tasks where each input is independent, such as image classification or simple regression tasks.

Limitation: They are not well-suited for tasks involving time sequences or dependencies between inputs over time (e.g., language modeling or time series prediction).

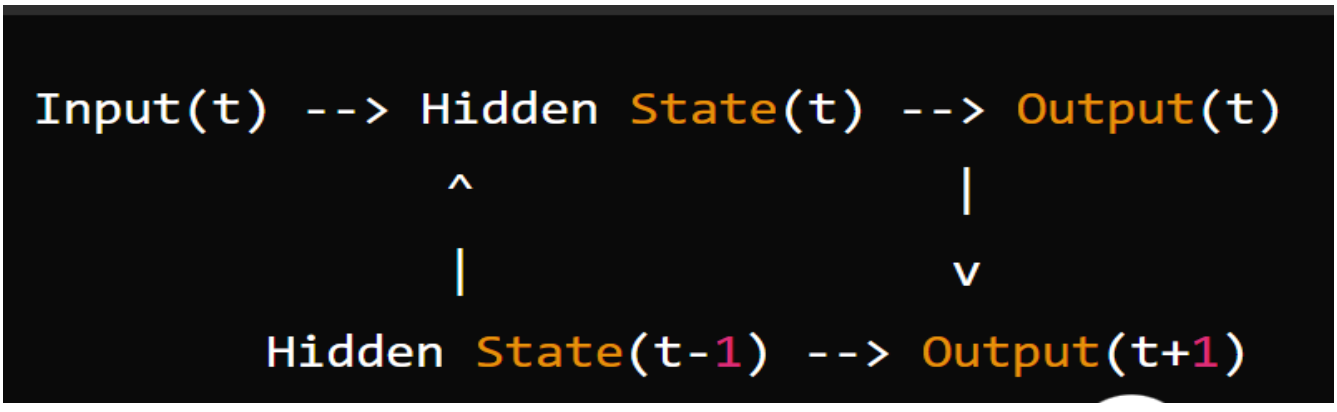
- Recurrent Neural Networks RNN :

Architecture: RNNs have a feedback loop within the hidden layers. This means that information from previous steps is fed back into the network, allowing it to maintain a form of memory.

Data Processing: RNNs process input sequences one element at a time while retaining information about the previous elements through internal states. This ability to remember previous inputs makes RNNs well-suited for sequential data.

Use Cases: RNNs are used for tasks involving sequences, such as language processing (e.g., text generation, translation), time series forecasting, and speech recognition.

Limitation: Standard RNNs suffer from issues like vanishing and exploding gradients, which can make it hard to learn long-range dependencies. Variants like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) are introduced to address these limitations.



• Autoencoder :

It is designed for unsupervised learning and consists of an encoder network that compresses the input data into a lower-dimensional latent space, and a decoder network that reconstructs the original input from the latent representation. **Autoencoders** are often used for dimensionality reduction, data denoising, and generative modeling.

• Multilayer Perceptrons :

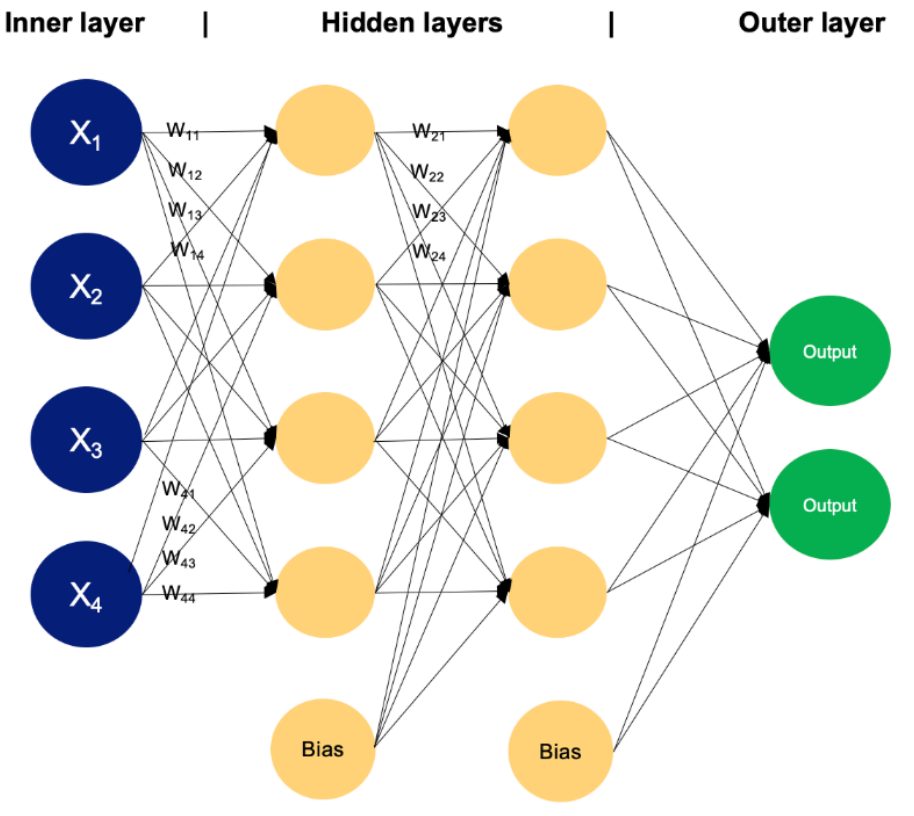
A multilayer perceptron is a type of feedforward neural network consisting of fully connected neurons with a nonlinear kind of activation function. It is widely used to distinguish data that is not linearly separable.

MLPs have been widely used in various fields, including image recognition, natural language processing, and speech recognition, among others. Their flexibility in architecture and ability to approximate any function under certain conditions make them a fundamental building block in deep learning and neural network research.

- **The input layer consists of nodes or neurons that receive the initial input data. Each neuron represents a feature or dimension of the input data. The number of neurons in the input layer is determined by the dimensionality of the input data.**
- **Between the input and output layers, there can be one or more layers of neurons. Each neuron in a hidden layer receives inputs from all neurons in the previous layer (either the input layer or another hidden layer) and produces an output that is passed to the next layer. The number of hidden layers and the number of neurons in each hidden layer are hyperparameters that need to be determined during the model design phase.**
- **This layer consists of neurons that produce the final output of the network. The number of neurons in the output layer depends on the nature of the task. In binary classification, there**

may be either one or two neurons depending on the activation function and representing the probability of belonging to one class; while in multi-class classification tasks, there can be multiple neurons in the output layer.

- Neurons in adjacent layers are fully connected to each other. Each connection has an associated weight, which determines the strength of the connection. These weights are learned during the training process



In a multilayer perceptron, neurons process information in a step-by-step manner, performing computations that involve weighted sums and nonlinear transformations. Let's walk layer by layer to see the magic that goes within.

Input layer

- **The input layer of an MLP receives input data, which could be features extracted from the input samples in a dataset. Each neuron in the input layer represents one feature.**
- **Neurons in the input layer do not perform any computations; they simply pass the input values to the neurons in the first hidden layer.**

Hidden layers

- **The hidden layers of an MLP consist of interconnected neurons that perform computations on the input data.**
- **Each neuron in a hidden layer receives input from all neurons in the previous layer. The inputs are multiplied by corresponding weights, denoted as w. The weights determine how much influence the input from one neuron has on the output of another.**

- **In addition to weights, each neuron in the hidden layer has an associated bias, denoted as b. The bias provides an additional input to the neuron, allowing it to adjust its output threshold. Like weights, biases are learned during training.**
- **For each neuron in a hidden layer or the output layer, the weighted sum of its inputs is computed. This involves multiplying each input by its corresponding weight, summing up these products, and adding the bias:**

Where n is the total number of input connections, w_i is the weight for the i-th input, and x_i is the i-th input value.

- **The weighted sum is then passed through an activation function, denoted as f. The activation function introduces nonlinearity into the network, allowing it to learn and represent complex relationships in the data. The activation function determines the output range of the neuron and its behavior in response to different input values. The choice of activation function depends on the nature of the task and the desired properties of the network.**

Output layer

- **The output layer of an MLP produces the final predictions or outputs of the network. The number of neurons in the output layer depends on the task being performed (e.g., binary classification, multi-class classification, regression).**
- **Each neuron in the output layer receives input from the neurons in the last hidden layer and applies an activation function. This activation function is usually different from those used in the hidden layers and produces the final output value or prediction.**

During the training process, the network learns to adjust the weights associated with each neuron's inputs to minimize the discrepancy between the predicted outputs and the true target values in the training data. By adjusting the weights and learning the appropriate activation functions, the network learns to approximate complex patterns and relationships in the data, enabling it to make accurate predictions on new, unseen samples.

This adjustment is guided by an optimization algorithm, such as stochastic gradient descent

(SGD), which computes the gradients of a loss function with respect to the weights and updates the weights iteratively.

$$\text{Weighted Sum} = \sum_{i=1}^{n} (w_i * x_i) + b$$

- **3 D deep learning :**

3D data -> Regular grids :Rasterized form

->IRregular grids :Geometric form

- Datasets :

- **Shapenet :**

ShapeNet is an ongoing effort to establish a richly-annotated, large-scale dataset of 3D shapes. We provide researchers around the world with this data to enable research in computer graphics, computer vision, robotics, and other

related disciplines. ShapeNet is a collaborative effort between researchers at Princeton, Stanford and TTIC.

–ShapeNetCore

ShapeNetCore is a subset of the full ShapeNet dataset with single clean 3D models and manually verified category and alignment annotations. It covers 55 common object categories with about 51,300 unique 3D models. The 12 object categories of PASCAL 3D+, a popular computer vision 3D benchmark dataset, are all covered by ShapeNetCore.

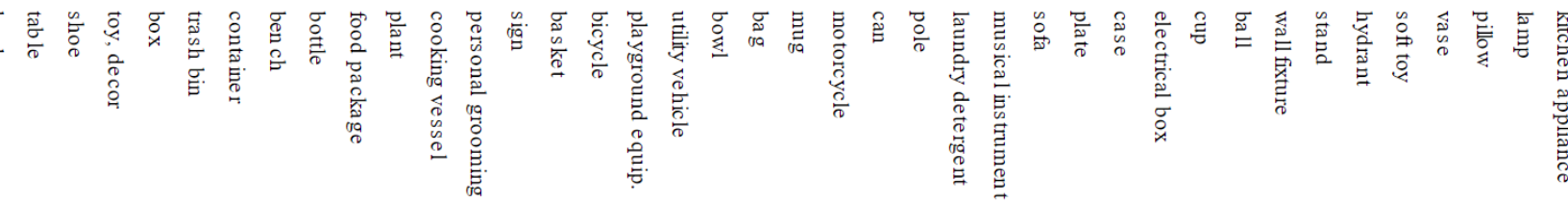
ShapeNetSem

ShapeNetSem is a smaller, more densely annotated subset consisting of 12,000 models spread over a broader set of 270 categories. In addition to manually verified category labels and consistent alignments, these models are annotated with real-world dimensions, estimates of their material composition at the category level, and estimates of their total volume and weight.

- **3 D Scan :**

We have created a dataset of more than ten thousand 3D scans of real objects. To create the dataset, we recruited 70 operators, equipped them with consumer-grade mobile 3D scanning setups, and paid them to scan objects in their environments. The operators scanned objects of their choosing, outside the laboratory and without direct supervision by computer vision professionals. The result is a large and diverse collection of object scans: from shoes, mugs, and toys to grand pianos, construction vehicles, and large outdoor

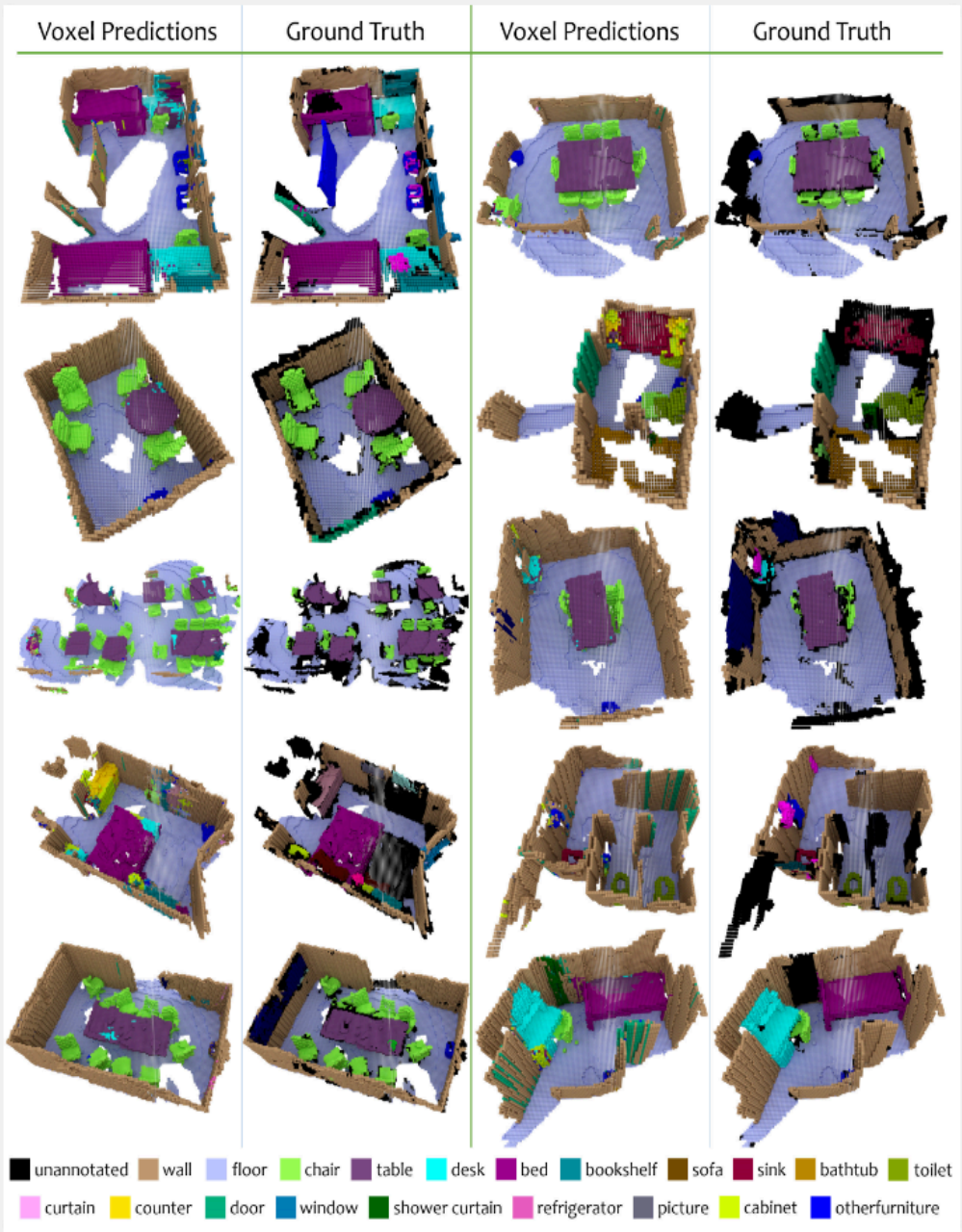
sculptures. We worked with an attorney to ensure that data acquisition did not violate privacy constraints. The acquired data was irrevocably placed in the public domain and is available freely.



- **Scan Net :** Richly-annotated 3D Reconstructions of Indoor Scenes

-ScanNet is an RGB-D video dataset containing 2.5 million views in more than 1500 scans, annotated with 3D camera poses, surface reconstructions, and instance-level semantic segmentations. To collect this data, we designed an easy-to-use and scalable RGB-D capture system that includes automated surface reconstruction and crowdsourced semantic annotation. We show that using this data helps achieve state-of-the-art

performance on several 3D scene understanding tasks, including 3D object classification, semantic voxel labeling, and CAD model retrieval.



Semantic voxel labeling of 3D scans in ScanNet using our 3D CNN architecture. Voxel colors indicate predicted or ground truth category.

- ArchShapesNet:

  - **a novel dataset for benchmarking architectural BIM element classification algorithms**

- **IFC :**

  - Industry Foundation Classes

  - IFC files are platform neutral and can be read and edited by any BIM software.

- **MVCNN :**

  -Multi-view Convolutional Neural Networks for 3D Shape Recognition

-A longstanding question in computer vision concerns the representation of 3D shapes for recognition: should 3D shapes be represented with descriptors operating on their native 3D formats, such as voxel grid or polygon mesh, or can they be effectively

represented with view-based descriptors? We address this question in the context of learning to recognize 3D shapes from a collection of their rendered views on 2D images. We first present a standard CNN architecture trained to recognize the shapes' rendered views independently of each other, and show that a 3D shape can be recognized even from a single view at an accuracy far higher than using state-of-the-art 3D shape descriptors. Recognition rates further increase when multiple views of the shapes are provided. In addition, we present a novel CNN architecture that combines information from multiple views of a 3D shape into a single and compact shape descriptor offering even better recognition performance. The same architecture can be applied to accurately recognize human hand-drawn sketches of shapes. We conclude that a collection of 2D views can be highly informative for 3D shape recognition and is amenable to emerging CNN architectures and their derivatives.

• Cloud Servers

- **Google Colab :**

  Google Colab was initially utilized to perform model training and analysis. However, due to the extensive size of the dataset, memory limitations and automatic session terminations posed significant challenges. Specifically, the available RAM proved insufficient for handling the entire dataset, requiring alternative cloud-based solutions.

- **Google Cloud Compute Engine:**

  To address the limitations encountered with Google Colab, a Virtual Machine (VM) instance was created using Google Cloud Compute Engine. This approach allowed for greater flexibility and scalability. Key configuration steps included:

- **Region and Zone Selection: Choose an optimal region and zone to minimize latency and**

optimize performance based on geographic proximity.

- **Hardware Configuration: Selected the required RAM and GPU specifications to efficiently manage the large dataset and support intensive computational tasks, including semantic segmentation and classification.**

***Point Cloud File Properties :**

- **File Size: The point cloud dataset is approximately 17GB (18,880,303,104 bytes).**

***Chunking Process :**

Due to the large file size, the dataset is processed in manageable chunks. The following steps outline the approach:

1. **Chunk Loading: The point cloud data is divided into smaller chunks to efficiently load into memory.**
2. **Model Feeding: Each chunk is sequentially fed into the PointNet model for processing.**

3. **Segmentation: The model performs semantic segmentation on each chunk independently.**

- HDF5 FILE :

HDF5 (Hierarchical Data Format version 5) is a versatile file format designed to store and manage large amounts of data. It's widely used in fields like scientific research, machine learning, and engineering, where datasets can be too large or complex to fit in memory all at once.

Key Features of HDF5:

1. **Hierarchical Structure:**
   HDF5 files are organized like a file system, with groups and datasets. Groups are similar to directories, while datasets store the actual data, like files.
   - **Groups: Containers for other groups or datasets.**
   - **Datasets: Multidimensional arrays that store raw data.**
2. **Efficient Storage for Large Data:**
   HDF5 is designed for high-performance I/O, making it efficient for reading and writing large datasets. It allows data to be compressed without losing precision, which reduces the file size and storage space requirements.

3. **Scalability:**

   HDF5 supports extremely large datasets, even those larger than your system's memory, by only loading portions of the data as needed (chunking). This makes it ideal for big data applications like handling point clouds, 3D data, and high-resolution images.
4. **Cross-Platform Compatibility:**

   HDF5 files can be read and written across different operating systems (Windows, Linux, macOS), programming languages (Python, C++, MATLAB), and environments without needing to change the file format.
5. **Metadata Support:**

   In addition to storing raw data, HDF5 files can include metadata. This allows for efficient tagging of data with additional information, like dimensions, units, or descriptions, which is helpful for organization and analysis.

<u>Benefits for Point Cloud Data:</u>

HDF5 is especially useful in research projects involving large datasets like point clouds, as it allows for efficient storage and manipulation. You can store chunks of point clouds, feed them into models like PointNet, and perform

operations on each chunk without loading the entire dataset into memory.

- The **h5py** library, is a common way to interact with HDF5 files in Python

```
import numpy as np
import h5py

# Example: Simulate a large point cloud data array (each point has
x, y, z coordinates)
point_cloud_data = np.random.rand(10000000, 3)  # 10 million points,
each with x, y, z coordinates

# Define HDF5 file and chunk size
hdf5_file = "point_cloud_data.h5"
chunk_size = 1000000  # Processing 1 million points at a time

# Writing the point cloud data into HDF5 with chunking
with h5py.File(hdf5_file, 'w') as f:
    # Create a dataset for point cloud data with a chunked layout
    dataset = f.create_dataset('points', data=point_cloud_data,
chunks=(chunk_size, 3), compression='gzip')

    print(f"Dataset 'points' written with chunk size: {chunk_size}
and compression: gzip")

# Reading the point cloud data back in chunks
with h5py.File(hdf5_file, 'r') as f:
    dataset = f['points']
    total_points = dataset.shape[0]
```

```
    print(f"Total points in the dataset: {total_points}")

    # Read and process data in chunks
    for i in range(0, total_points, chunk_size):
        chunk_data = dataset[i:i+chunk_size]

        # Here, you would feed the chunk into your PointNet model
for segmentation
        # For demonstration, we'll just print the chunk size
        print(f"Processing chunk {i // chunk_size + 1}: shape
{chunk_data.shape}")
```

- **CRASLAB _annotated dataset :**

- indoor point cloud dataset for BIM related applications

- **IM (building information modeling) has gained wider acceptance in the AEC (architecture, engineering, and construction) industry. Conversion from 3D point cloud data to vector BIM data remains a challenging and labor-intensive process, but particularly relevant during various stages of a project lifecycle. While the challenges associated with processing very large 3D point cloud datasets are widely known, there is a pressing need for intelligent geometric feature extraction and reconstruction algorithms for automated point cloud processing.**

Compared to outdoor scene reconstruction, indoor scenes are challenging since they usually contain high amounts of clutter. This dataset comprises the indoor point cloud obtained by scanning four different rooms (including a hallway): two office workspaces, a workshop, and a laboratory including a water tank. The scanned space is located at the Electrical and Computer Engineering department of the Faculty of Engineering of the University of Porto. The dataset is fully labeled, containing major structural elements like walls, floor, ceiling, windows, and doors, as well as furniture, movable objects, clutter, and scanning noise. The dataset also contains an as-built BIM that can be used as a reference, making it suitable for being used in Scan-to-BIM and Scan-vs-BIM applications.

-SCAN VS BIM :

- **Automatic identifying 3D model objects by aligning a point Cloud of a construction site with BIM =>Compare the two according to some evaluation criteria or metrics**

- DURAARK :

- Research Area: Complex Modeling

DURAARK is a three year research project, which tackles the challenge of developing secure and efficient long-term archival processes for 3D architectural data. The project has a focus on the combination of unconstructed data from Building-Information-Modelling (BIM) environments and will develop strategies to search and detect architecturally meaningful information in the automatically created rich data in order to overcome current problems that inhibit a successful long-term preservation of 3D architectural content.

Problems to tackle:

Non-existent semantic relations between architectural representations in Building Information Models, point clouds and geo-referenced data from web and images.

No automated systems to detect and search for architectural structures in point clouds.

Heterogeneous and inconsistent metadata schemes and ontologies for the description of building elements and their properties in highly enriched BIM models.

Fast changing development of architectural software, files and referenced web-based data as metadata schemes prohibits successful and easy long-term archival for companies and institutions.

- **Aim and approach**

The DURAARK project will develop query and preservation tools that are tailored to the domain of architectural 3D content. The objective is to ensure future-proof reuse of 3D objects, using open file standards and providing techniques to gain new architectural information through enrichment and automated combination of heterogeneous data.

The research will be developed, implemented and evaluated with partners from the professional communities in architecture, engineering and building administration. Cases and short-term hands-on projects will help to guide and evaluate the overall project.

- **DURAARK will develop:**

Methods for a metadata rich long-term archival of architectural content with its vast domain-specific spectrum of semantic

detail levels and fragmented nature of naming schemes and dynamic references to other web-based content.

Tools to automatically link point cloud scans and BIM data and the enrichment of architectural data with other georeferenced content such as images and semantic information from the web.

Queries on point clouds and BIM data in order to allow a future long-term archive to answer questions as they appear in a practice that needs to work seamlessly across heterogeneous sets of data.

Tools for the automatic detection of architectural meaningful structures as walls, openings and floors in point clouds.

Means to allow queries about the changing state to buildings by accessing data from different periods of time.

Collaborators

Leibniz Universität Hannover, LUH, Germany University Bonn - Institute of Computer Science, UBO, Germany Fraunhofer Austria Research GmbH, FhA, Austria Eindhoven University of Technology, TUE, Netherlands Centre for Information Technology and Architecture

Github Link : https://github.com/DURAARK

• ISPRS benchmark on Semantic Segmentation of High-Resolution 3D Point Clouds and Meshes :

Automated extraction of geographic objects from airborne data has been an important research topic in photogrammetry and remote sensing for decades. In addition to images, 3D point clouds from airborne LiDAR and Multi-View-Stereo-Image-Matching became more and more important as basic data sources. The aim of H3D is to provide state-of-the-art data sets to the community, which can be used by interested researchers to test their own methods and algorithms on semantic segmentation for geospatial applications. We propose a benchmark consisting of highly dense LiDAR point clouds captured at three different epochs. The respective point clouds are manually labeled into at least 11 classes and are used to derive labeled textured 3D meshes as an alternative representation. Core features of H3D are:

- **UAV-based simultaneous data collection of both LiDAR data and imagery from the same platform**
- **High density LiDAR data of 800 points/m² enriched by RGB colors of on board cameras incorporating a GSD of 2-3 cm → H3D(PC)**
- **High resolution 3D textured mesh data generated from both LiDAR data and imagery in an hybrid manner → H3D(Mesh)**
- **Manually set labels for the LiDAR point cloud, which are automatically transferred to the 3D mesh**

-The ISPRS Benchmark on Indoor Modeling provides a public benchmark dataset to enable performance evaluation and benchmarking of indoor modeling methods. The dataset consists of six

point clouds captured by different sensors in indoor environments of various complexities.

Link : https://www.gaofen-challenge.com/benchmark

- **Clutter removal - extract structural features for modeling purposes -Automated Reconstruction techniques**
- **Labeled high_resolution point cloud =>Creation of a semantically rich digital twin**

-Digital Twin :

A digital twin is a virtual representation of an object or system designed to reflect a physical object accurately. It spans the object's lifecycle, is updated from real-time data and uses simulation, machine learning and reasoning to help make decisions.

A digital twin is a digital model of an intended or actual real-world physical product, system, or process (a *physical twin*) that serves as the effectively indistinguishable digital counterpart of it for practical purposes, such as simulation, integration, testing, monitoring, and maintenance , A digital twin is set of adaptive models that emulate the behavior of a physical system in a virtual system getting real time data to update itself along its life cycle. The digital twin replicates the physical system to predict failures and opportunities for

changing, to prescribe real time actions for optimizing and/or mitigating unexpected events observing and evaluating the operating profile system.[2] Though the concept originated earlier (as a natural aspect of computer simulation generally), the first practical definition of a digital twin originated from NASA in an attempt to improve the physical-model simulation of spacecraft in 2010.[4] Digital twins are the result of continual improvement in modeling and engineering.

In the 2010s and 2020s, manufacturing industries began moving beyond digital product definition to extending the digital twin concept to the entire manufacturing process. Doing so allows the benefits of virtualization to be extended to domains such as inventory management including lean manufacturing, machinery crash avoidance, tooling design, troubleshooting, and preventive maintenance. Digital twinning therefore allows extended reality and spatial computing to be applied not just to the product itself but also to all of the business processes that contribute toward its production.

What is the difference between digital twins and BIM models?

Digital twins and BIM models are digital representations of physical spaces. The main difference between the two is that 3D BIM models are used to visualize the design and construction of

an asset, while a digital twin allows for virtual interaction with that asset. To clarify, here are the key differences between digital twins and BIM models: BIM models created in software like Revit represent the design intent of a building, allowing for visualization of desired physical features and dependencies. By their nature, 3D models linked to a BIM process do not automatically track or represent changes in the state of a building over time. Digital twins provide a photorealistic virtual rendering of a physical space, and the space can be scanned multiple times over time to allow teams to track its evolution within its digital twin. This helps stakeholders track milestones and better understand the lifecycle of a project. Digital twins integrate with IoT sensors and other digital solutions. When you use these technologies to access real-time data about your building system, digital twins act as a visual replica that accurately captures the physical characteristics of that system . In other words, BIM modeling software can help you design and construct a building, while digital twins can help you maintain and operate that building. For example, you can use BIM modeling software to determine where a building's HVAC system will be installed. Once it's in place, you can use digital twins to monitor its operation throughout its lifecycle.

Aspect	BIM (Building Information Modeling)	Digital Twin
Definition	A digital representation of the physical and functional characteristics of a building. Used for design, construction, and management.	A real-time virtual representation of a physical object, process, or system that continuously syncs with data from the physical world.
Purpose	Facilitates design, collaboration, construction planning, and project management.	Enables real-time monitoring, performance optimization, and predictive maintenance during operation.
Life Cycle Stage	Primarily used in the **design and construction** phases, and sometimes for early facility management.	Used mainly in the **operational phase** and beyond, for continuous monitoring and analysis.
Data Source	Relies on **static data** generated during design and construction. Updates occur manually when changes are made.	Continuously updated with **real-time data** from IoT sensors, devices, and systems in the physical world.
Usage Context	Architects, engineers, and construction teams use BIM for **design validation, clash detection, and construction planning**.	Facility managers, operators, and maintenance teams use Digital Twins for **monitoring, diagnostics, and predictive analysis**.
Real-Time Interaction	No real-time interaction. It is typically a static model.	Full real-time interaction, reflecting the live state of the asset.
Main Technologies	3D modeling, CAD, and information-rich models (e.g., Revit, ArchiCAD).	IoT, sensors, data analytics, AI, and machine learning for real-time data processing and prediction.
Examples of Use	Creating a detailed 3D model of a building for construction planning and coordination between different systems (e.g., HVAC, electrical, structural).	Monitoring the performance of a smart building's HVAC system, tracking energy consumption, or simulating future performance scenarios based on real-world conditions.
Simulation and Prediction	Limited to design simulations before construction begins.	Enables **real-time simulation** and predictive analytics for operational optimization and maintenance planning.

Integration	Primarily focuses on the design and construction phase, though some BIM models may be updated post-construction.	Continuously evolves during the operational life of the building or system, integrating real-time performance data.

- **Understanding Point Net code :**

- num_votes: how many times each point cloud in the dataset is evaluated

-pointclouds_pl : input data :point clouds (tensorflow placeholder)

shape :[Batch size,num_points, 3]

where Batch size:number of point clouds processed simultaneously in one forward pass

where num_points : number of points in each point cloud

-Labels -pl : holds the ground truth labels for each point cloud in a batch

Type: tf.placeholder

example: batch_size= 4 , labels_pl =[3,7,12,1]

⇒Point cloud n^1 belongs to class 3

⇒Point cloud n^2 belongs to class 7

-Each label corresponds to one point cloud in the batch

-Labels are integer values representing the class of each cloud

- pred : it contains 40 scores (logits) for each point cloud in the batch (small group of point clouds that are processed together in one forward and backward pass through the model)

- `pred[i]`: A vector of length `num_classes` for the i-th point cloud in the batch, containing 40 values. Each value represents the model's confidence score for how likely the point cloud belongs to one of the 40 possible classes. The score `pred[i][j]` is a raw logit corresponding to class `j`, which can be converted into a probability using the softmax function.

=> Highest score(logit) = Predicted class

- the model processes each point independently to extract features (passing it through MLPs)

-Each point is transformed from (x,y,x) coordinates into a feature vector that captures the more complex information about the point (Per_Point features)

- After computing features for each point, the model aggregates the Per_Point features into a global feature that represents the entire point cloud ,it summarizes the information from all N points in the point cloud

-During Training Phase : point cloud data + labels(ground truth)

-During Evaluation Phase : point cloud data+true labels , Loss = difference between true labels and predicted labels (the output of the model)

-During Prediction phase : point cloud is used to generate predictions , labels are not provided because they are unknown

+ Num_batches = file size / batch size

-Top K : It's a function considers the top k predictions for each point cloud when computing accuracy

example: Tiger = 0.4,dog = 0.3, cat=0.1, lion = 0.09 .../True value=Cat

- **Top 1 accuracy : predicted output (highest prediction) is Tiger**

=>Wrong prediction because its different to the real value

- **Top 5 accuracy : Cat is among the top 5 guesses**

=>Prediction is correct.

-Total_seen : number of point clouds that have been evaluated

-Total_correct : track the total number of correct predictions

-Loss_sum : accumulates total loss overall batches for reporting the average loss .

-Total seen _class : how many point clouds from each class have been seen during evaluation

-Total correct _class: how many point clouds from each class have been correctly classified

-Feed_dict : dict that feeds the rotated point cloud data / labels into the model for evaluation

-Correct : number of correct predictions in the current batch

- **CheckPoint :**
  - **A checkpoint is a snapshot of a model's state, saved at specific points during the training process. Checkpoints capture the model's architecture, weights, and other relevant parameters, allowing for several important functionalities.**

- **Resume Training: If training is interrupted (e.g., due to time constraints or system issues), checkpoints enable you to resume from the saved state instead of starting over from the beginning.**
- **Model Evaluation at Different Stages: By saving checkpoints at regular intervals, you can evaluate how the model performs at various stages of the training process. This allows for detailed comparisons of model performance and behavior over time.**
- **Inference Without Retraining: Once the model is trained and saved in a checkpoint, you can load it at any time to perform inference (e.g., making predictions or processing new data) without needing to retrain the model from scratch.**

Checkpoints are essential for managing long training processes, experimenting with model variations, and ensuring that valuable training progress is preserved

- **Github Link :https://github.com/nicolas-chaulet**
    - **Processing 3D Point Cloud data :**

-3D point cloud is Three Dimensional (3D) data representing a physical object in a 3D coordinate system. Each point represents a single measurement, capturing the precise location of a specific point in space with an x, y, and z coordinate. It also stores some information such as geometric coordinates, RGB color, intensity, reflectance value, and normal vectors.

Although a point cloud has some data embedded in each point, that data will not give us any information if we do not extract semantic information from it. There are several steps needed to extract semantic information

from a 3D point cloud. The process can include manual processing or automatic processing using specific programming methods. Below are general steps to process 3D point cloud data.

1. Pre-Processing: Preparing the Data

Before diving into analysis, we ensure the point cloud is ready for interpretation:

- **Filtering: Removing noise and outliers that might distort results, like stray points from reflections or sensor errors.**
- **Downsampling: Reducing the number of points to make processing more efficient, especially for large datasets, while preserving essential details.**
- **Registration: Aligning multiple point clouds from different scans into a unified coordinate system, creating a seamless representation of the entire scene.**

2. Segmentation: Grouping the Dots into Meaningful Clusters

- **Region Growing: Connecting neighboring points based on similarity in properties like color, intensity, or geometric features.**
- **Model Fitting: Applying geometric models (planes, spheres, cylinders) to segment structures like walls, pipes, or tree trunks.**
- **Machine Learning: Employing algorithms to learn patterns and classify points into categories like trees, buildings, or vehicles.**

3. Feature Extraction: Drawing Out the Fingerprints

We now extract key characteristics to describe each segmented object or region:

- **Geometric Features: Calculating surface normals, curvature, or bounding boxes to reveal shape characteristics.**
- **Spatial Features: Analyzing spatial relationships between objects, like distance, proximity, or alignment.**

4. Classification and Analysis: Deriving Meaning from the Features

Finally, we assign semantic meaning to the extracted features:

- **Object Classification: Labeling objects based on their features (e.g., “car,” “tree,” “building”).**

- **Change Detection: Identifying differences between point clouds of the same area over time, useful for monitoring construction progress or environmental changes.**
- **Measurement and Modeling: Extracting dimensional information for tasks like volume estimation, model creation, or structural analysis**

Software for Process 3D Point Cloud

General-Purpose Point Cloud Processing:

- **CloudCompare: A free, open-source software with extensive tools for filtering, registration, segmentation, analysis, and visualization. This is a very popular software used by point cloud user to read and modify their data.**
- **MeshLab: Another free, open-source software focused on mesh processing, with point cloud editing and cleaning capabilities.**
- **Geomagic Studio: A comprehensive commercial software with advanced tools for point cloud editing, mesh creation, analysis, and CAD integration.**
- **PointCab: Commercial software with a focus on architectural and construction applications, offering point cloud cleaning, modeling, and BIM integration.**

Industry-Specific Point Cloud Software:

- **Autodesk ReCap: Designed for architecture, engineering, and construction, specializing in point cloud cleaning, registration, and integration with Autodesk design suites.**
- **Trimble RealWorks: Suite of software for surveying, mapping, and construction, offering point cloud processing, analysis, and integration with Trimble hardware.**
- **Leica Cyclone: Comprehensive software for surveying and mapping, including point cloud processing, 3D modeling, and geospatial analysis.**
- **Terrasolid: Suite of software for photogrammetry and LiDAR processing, with advanced tools for point cloud classification, feature extraction, and terrain modeling.**
- **Pix4D: Popular photogrammetry software that also handles LiDAR point cloud processing, offering 3D modeling, mapping, and orthophoto generation.**

Cloud-Based Point Cloud Processing:

- **Vercator: Automatic cloud-based processing for large point cloud datasets, providing cleaning, compression, and visualization capabilities.**

- **PointFuse: Cloud-based software for converting point clouds into lightweight 3D meshes for easier sharing and visualization.**

Automatic 3D Point Cloud Processing

General-Purpose Machine Learning Library with Point Cloud Capabilities:

- **TensorFlow: A versatile open-source framework from Google with extensive support for deep learning, 3D data processing, and visualization tools like TensorFlow Graphics and PointNet++.**
- **PyTorch: Another popular open-source framework known for its flexibility and ease of use, with libraries like Kaolin and PyTorch3D specifically designed for 3D deep learning tasks.**
- **Scikit-learn: A comprehensive library for traditional machine learning algorithms in Python, including modules for point cloud classification and clustering.**

Specialized Point Cloud Processing Libraries with ML Integration:

- **PCL (Point Cloud Library): A comprehensive open-source library with a vast array of point cloud processing algorithms, including machine learning modules for tasks like segmentation and classification.**
- **Open3D: An open-source library offering a versatile set of tools for 3D data processing, including machine learning capabilities for point cloud analysis and feature extraction. We can use open3D in python to automate the process.**

- **Cloud Based Services :**

  -Cloud Storage Services

  ***Amazon S3 *Google Cloud Storage * Azure VMs**

  -Big Data Platform

***Google BigQuery (run complex SQL queries) on large datasets**

-Managed Jupyter Environments

***Google Ai platform *Azure ML**

-Point Cloud classifier : classify /vectorize data in large 3D point clouds

example :pointly :cloud based B2B software

Link : https://pointly.ai/

=> Use APIS : integrate classification models into external workflows /platforms : when the user clicks on the button in ribbon panel , it classifies the point cloud .

- **Model Fine Tuning :**

  Unlike initial training which requires massive datasets like ImageNet, this refinement focuses on more restricted and specialized data. It is an iterative process that aims to improve the performance of the model on a particular task, without losing the prior knowledge acquired during initial training.The central idea lies in the model's ability to generalize to new domains while retaining its ability to specialize. Fine-Tuning makes it possible to adjust the

weights of the connections between neurons so that they are more adapted to the new task without significantly disrupting pre-existing knowledge. This approach finds varied applications across many fields. In Computer Vision, for example, a model pre-trained on a large collection of images can be fine-tuned for the detection of specific objects in a particular context such as autonomous vehicles or surveillance cameras.

Fine-tuning requires a methodical and precise approach. The process begins with collecting and preparing data. It must be of high quality, specific to the target task, and representative of the real-world scenarios the model will face.Data cleaning is also essential to eliminate errors, duplicates, and inconsistencies. This formatting then facilitates fine-tuning.The second step is choosing the pre-trained model based on the specific task. For example, a model pre-trained for image recognition may be more suitable for object detection than another pre-trained for NLP.Before starting fine-tuning, the initial performance of the selected model on the target task is evaluated. This provides a baseline against which improvement can be measured later.Next comes the time to fine-tune hyperparameters such as the learning rate, number of iterations, and batch size. This is what makes the difference between model convergence and overfitting. To find the best performing hyperparameter combinations, we can use techniques such as random search, grid search, or Bayesian

optimization. Beyond simple parameter tuning, advanced strategies can further optimize model performance while avoiding potential pitfalls. Transfer learning involves using the prior knowledge acquired by a model on a task to improve its performance on a similar task. The lower layers responsible for detecting general features are often retained, while the upper layers can be tuned for the new task.However, this transfer can lead to overfitting if the training data is too specific. Using regularization techniques such as dropout can mitigate this risk by introducing randomness during training.Another technique is progressive fine-tuning, which involves fine-tuning the model in several steps. We start with higher and more specific layers, before moving to lower layers.The goal? To allow a smoother adaptation and reduce the risk of losing crucial knowledge. An evaluation of the model's performance at each step is essential to understand its evolution.It provides the ability to detect signs of overfitting or underfitting, and adjust the strategy accordingly. All these advanced strategies help optimize model specialization while minimizing risks.

- **WSL-Based Fine-Tuning of PointNet for Semantic Segmentation of Complex Industrial Point Clouds**

Installation

Install TensorFlow. You may also need to install h5py. The code has been tested with Python 2.7, TensorFlow 1.0.1, CUDA 8.0 and cuDNN 5.1 on Ubuntu 14.04.
If you are using PyTorch, you can find a third-party pytorch implementation here.
To install h5py for Python:

```
sudo apt-get install libhdf5-dev
sudo pip install h5py
```

Part Segmentation

To train a model for object part segmentation, firstly download the data:

```
cd part_seg
sh download_data.sh
```

The downloading script will download the ShapeNetPart dataset (around 1.08GB) and our prepared HDF5 files (around 346MB).

```
malek@Administrator:~$ source /home/malek/pointnet-env/bin/activate
(pointnet-env) malek@Administrator:~$ cd pointnet.pytorch
(pointnet-env) malek@Administrator:~/pointnet.pytorch$ cd utils
```

```
(pointnet-env) malek@Administrator:~/pointnet.pytorch/utils$ python 3
mainevaluate.py  --model  "checkpoints/seg_model_Chair_19.pth"
--dataset "/mnt/c/Users/Pc/Downloads/trial.npy"  --output
"/mnt/c/Users/Pc/Desktop/Project\ STAGE/"
```

```
Model loaded successfully.
Loading point cloud from: /mnt/c/Users/Pc/Downloads/pointcloud.npy
Point cloud shape: (22268413, 3)
Point cloud shape after adding batch dimension: torch.Size([1,
22268413, 3])
Point cloud shape after transposing: torch.Size([1, 3, 22268413])
Performing inference in 45 chunks...
Processing chunk 1/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 2/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 3/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 4/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 5/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 6/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 7/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 8/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 9/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 10/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 11/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 12/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 13/45 with shape: torch.Size([1, 3, 500000])
```

```
Processing chunk 14/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 15/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 16/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 17/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 18/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 19/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 20/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 21/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 22/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 23/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 24/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 25/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 26/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 27/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 28/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 29/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 30/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 31/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 32/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 33/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 34/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 35/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 36/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 37/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 38/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 39/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 40/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 41/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 42/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 43/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 44/45 with shape: torch.Size([1, 3, 500000])
Processing chunk 45/45 with shape: torch.Size([1, 3, 268413])
Final prediction shape: torch.Size([1, 22268413])
Prediction shape after squeezing: (22268413,)
Visualizing the point cloud with Open3D...
```

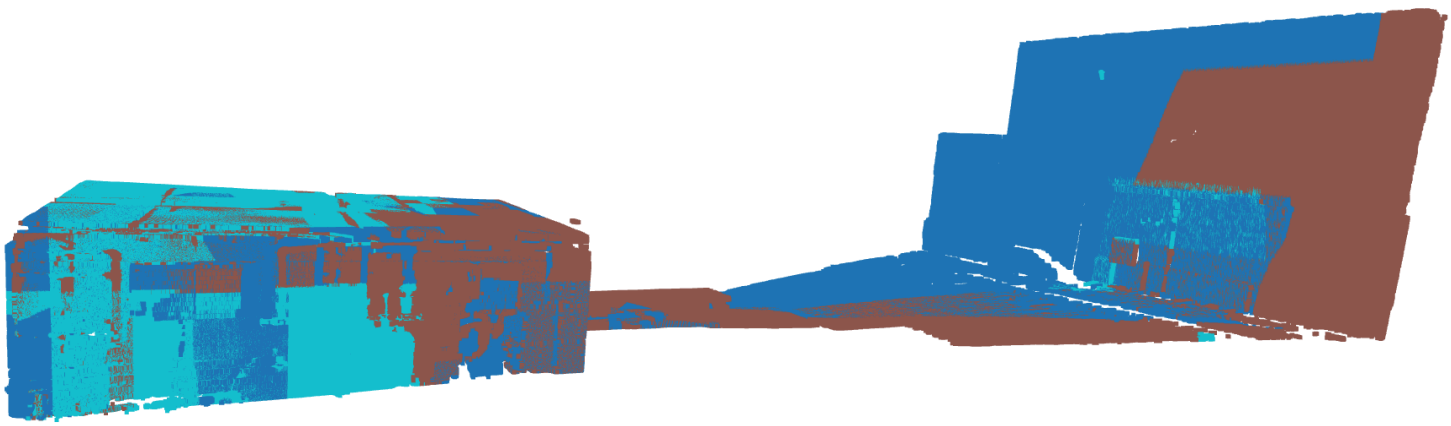


Figure: Advanced 3D Visualization of Segmented Point Clouds via Open3D

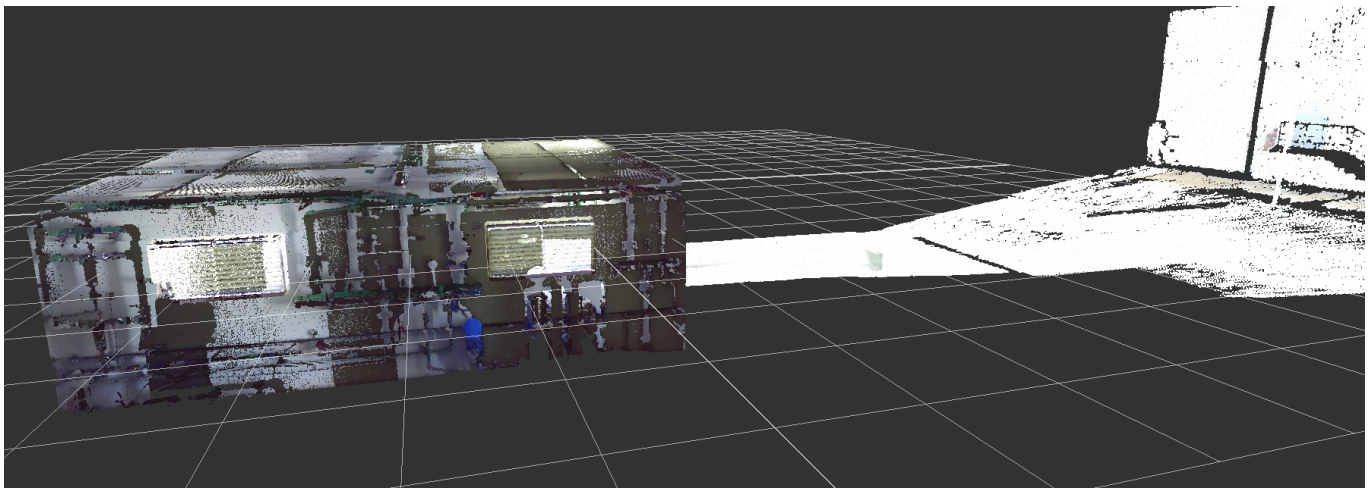


Figure : 3D view of real point cloud via AutoDesk Recap pro

In this project, we utilized PointNet within the Windows Subsystem for Linux (WSL) environment to perform semantic segmentation on point cloud data. We set up a virtual machine (VM) named pointcloud_env in WSL to manage the PointNet model and applied it to segment point clouds. Despite following the prescribed methodology, challenges related to dataset compatibility impacted the segmentation results.

WSL Setup and Implementation

1. **WSL Environment:**
    - **What is WSL? Windows Subsystem for Linux (WSL) allows you to run a Linux distribution directly on Windows without the need for a virtual machine or dual-boot setup. This provides a convenient way to develop and run Linux-based applications and scripts directly from a Windows environment.**
    - **VM Setup: We configured a VM named pointcloud_env within WSL to create a controlled environment for running PointNet. This setup facilitated the management of dependencies and execution of the model.**
2. **Model Setup and Training:**
    - **Repository Cloning: We cloned the PointNet GitHub repository into the WSL environment. This repository includes the necessary code and pre-trained models for training and evaluating PointNet.**
    - **Fine-Tuning and Inference: Using the WSL environment, we fine-tuned the PointNet model on the provided datasets and applied it for inference to segment point clouds.**
3. **Datasets Utilized:**
    - **ShapeNetPart Dataset: Approximately 1.08 GB in size, this dataset was used for training and testing the model. It includes**

various object categories but lacks BIM objects and industrial equipment.

- **Custom HDF5 Files: We created additional HDF5 files (around 346 MB) containing point cloud data tailored for segmentation.**

Challenges and Observations

Despite setting up the WSL environment and fine-tuning the model, we encountered challenges with segmentation accuracy. The main issue was that both the ShapeNetPart dataset and our custom HDF5 files did not include BIM objects or industrial equipment. As a result, the model produced inaccurate segmentation results for these specific categories.

Repository and Methodology

Our approach is based on the PointNet architecture, as described in the arXiv tech report (arXiv:1612.00593). PointNet is notable for:

- **Direct Point Cloud Processing: It handles raw point clouds directly, avoiding the need for conversion to voxel grids or images.**
- **Permutation Invariance: It respects the order of input points, ensuring robustness.**
- **Unified Architecture: Suitable for various tasks including object classification, part segmentation, and scene parsing.**

Repository Features:

- **Installation Instructions: Includes guidance on setting up TensorFlow, h5py, and other dependencies within the WSL environment.**

- **Training and Evaluation: Commands for training classification models, performing part segmentation, and visualizing results using TensorBoard.**
- **Data Preparation: Tools and scripts for managing and preparing HDF5 files.**

To address the issues with segmentation accuracy for BIM objects and industrial equipment, future work could involve:

- **Specialized Datasets: Incorporating datasets that specifically include BIM and industrial equipment.**
- **Enhanced Model Fine-Tuning: Adapting the model with more relevant data to improve segmentation performance.**

For additional resources and details, please visit the PointNet GitHub repository.