



Science Arts & Métiers (SAM)

is an open access repository that collects the work of Arts et Métiers Institute of Technology researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: <https://sam.ensam.eu>
Handle ID: <http://hdl.handle.net/10985/23047>

To cite this version :


Sijie HU, Arnaud POLETTE, Jean-Philippe PERNOT - SMA-Net: Deep learning-based identification and fitting of CAD models from point clouds - Engineering with Computers - Vol. 38, n°6, p.5467-5488 - 2022

Any correspondence concerning this service should be sent to the repository

Administrator : scienceouverte@ensam.eu



SMA-Net: Deep learning-based identification and fitting of CAD models from point clouds

Sijie Hu¹ · Arnaud Polette¹ · Jean-Philippe Pernot¹ 

Received: 30 November 2021 / Accepted: 18 March 2022

© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract

Identification and fitting is an important task in reverse engineering and virtual/augmented reality. Compared to the traditional approaches, carrying out such tasks with a deep learning-based method have much room to exploit. This paper presents SMA-Net (Spatial Merge Attention Network), a novel deep learning-based end-to-end bottom-up architecture, specifically focused on fast identification and fitting of CAD models from point clouds. The network is composed of three parts whose strengths are clearly highlighted: voxel-based multi-resolution feature extractor, spatial merge attention mechanism and multi-task head. It is trained with both virtually-generated point clouds and as-scanned ones created from multiple instances of CAD models, themselves obtained with randomly generated parameter values. Using this data generation pipeline, the proposed approach is validated on two different data sets that have been made publicly available: robot data set for Industry 4.0 applications, and furniture data set for virtual/augmented reality. Experiments show that this reconstruction strategy achieves compelling and accurate results in a very high speed, and that it is very robust on real data obtained for instance by laser scanner and Kinect.

Keywords Identification · Fitting · Deep learning · Transformer · Data generation · Virtual reality · Reverse engineering

1 Introduction

At a time when companies and individuals are increasingly immersed in the digital world, even the full 3D world, solving the question of moving from the real world to the virtual one is becoming central. Thus, there is a clear need to develop fast and accurate reconstruction techniques also able to keep track of the possible evolutions and changes. This is particularly true in the context of the Industry 4.0 for which there is a large demand to model digital twins of products and systems, and to maintain the coherence between the real and virtual worlds in time [39]. With the development and spreading of 3D acquisition technologies, reverse engineering (RE) has been extensively exploited in recent years to

support various applications, for instance in engineering design, robotics, metrology or cultural heritage [6]. For most engineering applications, the main objective is to extract information from the acquired raw data to reconstruct parametric CAD models that best fit to the current state of the object. Today, most existing RE techniques follow a time-consuming patch-by-patch reconstruction strategy, wherein users have to face cumbersome trimming and stitching issues. At the end, dead CAD models are obtained and cannot be modified later on. Moreover, in the absence of suitable approaches, very few RE techniques are able to deal directly and efficiently with the case of assemblies. Thus, the method generally consists in disassembling the various parts that will then serve as inputs of the RE process, and then most probably reassembled in the virtual world. This does not allow to study assemblies which cannot be disassembled easily. As a consequence, these approaches do not fully address the requirements of the Industry 4.0 in terms of fast reconstruction or update of virtual environments and complete systems, which is the focus of this paper.

With the rapid development of computer technologies and hardware, deep learning has received increasing attention due to its numerous applications in different areas, such as

✉ Jean-Philippe Pernot
jean-philippe.pernot@ensam.eu

Sijie Hu
sijie.hu@ensam.eu

Arnaud Polette
arnaud.polette@ensam.eu

¹ LISPEN, Arts et Métiers Institute of Technology,
13617 Aix-en-Provence, France

computer vision, simulation, natural language processing, and robotics. As a dominating branch of artificial intelligence, deep learning has been successfully used to solve complex problems that are sometimes difficult to solve with more traditional approaches. Specifically, being able to identify parts and assemblies, segment point clouds and retrieve CAD model parameter values from a point cloud are essential steps, which could benefit a lot from a deep learning-based approach. However, the use of deep learning to reverse engineer products or systems is a relatively poorly explored strategy, and notably when it comes to the reconstruction of parametric CAD assemblies [6]. Although some 3D reconstruction methods based on deep learning have emerged in recent years [2, 3, 26, 27], these methods are all based on existing open datasets and are not specifically designed for reverse engineering, so their accuracy is far from meeting the requirements of industrial applications. The lack of proper datasets is certainly partially responsible for such a low attention, as it is a bottleneck that makes very challenging this type of research. Indeed, most existing datasets are only designed for tackling part-level or instance-level 3D scene understanding, like 3D semantic segmentation and 3D object detection or parameter-independent 3D reconstruction. Moreover, these datasets usually originate from the Internet and require a huge amount of manual annotations. The work presented in this paper not only aims to identify parts and assemblies from point clouds, but also to retrieve parameter values from the corresponding CAD models.

To solve those issues and allow a more global approach to reverse engineering, this paper introduces a deep learning-based reconstruction technique. Figure 1 shows a reconstruction example following the proposed approach. It is based on a novel end-to-end bottom-up architecture, specially focused on fast identification and fitting of CAD models from 3D point clouds. If the point cloud embeds several semantic parts (e.g., chairs or tables), they are treated separately after a segmentation step (Fig. 1b). The core of the approach relies on the so-called SMA-Net (Spatial Merge Attention Network) composed of three main parts: voxel-based multi-resolution feature extractor, spatial merge attention mechanism and multi-task head. It is trained with

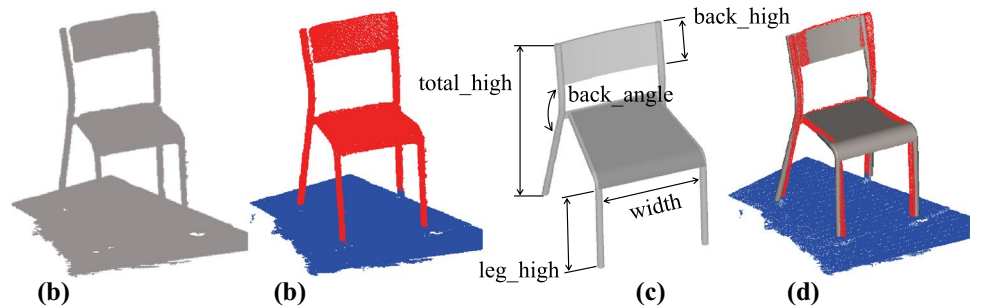
both virtually-generated point clouds and as-scanned ones created from multiple instances of several CAD templates, themselves obtained with randomly generated parameter values. Using this data generation pipeline, the proposed approach has been validated on two different datasets: robot dataset for Industry 4.0 applications, and furniture dataset for virtual/augmented reality. Once trained, SMA-Net can be used on a new point cloud, to identify the corresponding CAD template (Fig. 1c) and estimate its parameter values (Fig. 1d). The CAD templates gather together prior knowledge on the parts to be reconstructed, including all the constraints to define the related B-Rep models. The geometric reconstructions are performed within a CAD software in charge of updating the CAD models using the outputs of SMA-Net. The adopted modeler also keeps track of the consistency of the generated models, including assembly constraints between parts if needed.

Experiments show that this reconstruction strategy achieves compelling and accurate results in a very high speed, and that it is very robust on real datasets obtained for instance by laser scanner and Kinect. Thus, these results are of particular interest in the context of the Industry 4.0, to maintain the coherence between real products/systems/environments and their digital twins whose geometric models are a prior known, but whose configurations are evolving.

The contribution is threefold: (i) a novel reverse engineering framework able to fit CAD models on point clouds segmented using a deep segmentation network; (ii) a virtual data generation pipeline able to generate datasets of CAD model instances together with their as-scanned point clouds and related parameter values; (iii) a light and fast bottom-up 3D classification and fitting framework, named SMA-Net, able to identify objects from point clouds and estimate the parameter values of the corresponding CAD templates in an end-to-end manner. For the first time, such datasets used to validate the proposed identification and fitting technique have been made publicly available.

The paper is organized as follows. Section 2 reviews the works related to reverse engineering and deep learning. The overall framework is presented in Sect. 3 with the details of the three parts of SMA-Net. The experimental set up is detailed in Sect. 4, and the results are presented and

Fig. 1 Fast reconstruction of a chair CAD model: **a** acquired chair point cloud and its environment, **b** segmented point cloud, **c** CAD template and its parameters, **d** fitted CAD model generated by a CAD modeler with parameter values resulting from SMA-Net



discussed in Sect. 5. Section 6 ends this paper with conclusion and discussion.

2 Prior work

This section reviews previous reverse engineering and deep learning approaches for point cloud analysis. In reverse engineering, a particular attention is paid on the template-based methods. Fayolle and Pasko proposed a system to reconstruct solid models from digital point clouds by means of a genetic evolutionary algorithm [18]. Robertson et al. [45] have exploited the way to extract parametric models of features from poor quality 3D data using RANSAC [19], but it can only estimate parameters of simple primitives (e.g. cylinder, plan). As an extension of RANSAC [31, 37, 48], try to recover multiple primitives from a point cloud. However, the performance of these RANSAC-based methods is highly dependent on the tuning of parameters based on prior knowledge of different primitive shapes and it is not able to reverse a full CAD model. These algorithms focus on the resolution of general reverse engineering problems. Therefore, it is difficult to achieve accurate reconstructions of 3D targets like template-based methods do. Bey et al. [5] have introduced a template-based method to reconstruct CAD models from 3D point clouds, which is applied for civil engineering purposes. Erdös et al. have presented a CAD model matching method and they have extended the type of features to torus and cuboids [16]. In these works, the authors also focus on basic shapes (e.g. cylinder and cuboids) and do not work on complete CAD models. Inspired by template-based reverse engineering approaches, Buonamici et al. [7] proposed a technique to fit a single CAD template on a point cloud, while using particle swarm optimization. Another fitting approach is proposed by Shah et al. [50], with a template-based CAD reconstruction process optimized part-by-part using simulated annealing. These template-based methods show good results, but suffer from several limitations. For instance, manual intervention is required when pre-aligning CAD templates onto the point clouds, and a lot of time is spent in the iterative resolution process. In recent years, some template-based point cloud fitting methods based on deep learning have emerged. Armen et al. [2] perform scan-to-CAD alignment by predicting 9-DoF (degree of freedom) of the CAD model. [3, 26, 27] use similar way to align the CAD template with the 3D input. Then, to improve surface reconstruction accuracy, Vladislav et al. [30] proposed to deform retrieved CAD models after their alignment to scans. However, this mesh deformation based method can only achieve rough approximations, and it results in changes in the geometric properties of the CAD model that is transformed in a mesh. Thus, it is still difficult to meet the

industrial requirements in terms of parameter-driven CAD template reconstruction.

Many different ways have been proposed to process point cloud data using deep learning, such as point-wise CNNs [8, 29, 44, 64, 67], volume-wise CNNs [10–12, 24, 25, 53], multi-view CNNs [9, 35, 43, 52] and special CNNs, namely kernel-based parametric CNN [54, 59, 62] and graph reasoning [36, 51, 57, 58]. Considering point-wise CNNs, PointNet [8] is a pioneer in this direction. However, PointNet does not capture local structures limiting both its ability to recognize fine-grained patterns and its generalizability to complex scenes. There have been some attempts to strengthen the ability to explain local features to learn deep point set features more efficiently and robustly [44, 64, 67]. Instead of directly processing the irregular points, voxel-based methods project point cloud on regular 3D grids in Euclidean space. Graham et al. [24, 25] try to handle voxel-based data by sparse tensor and sparse convolution and achieved good results for the segmentation of indoor scenes. Christopher et al. [11] introduced 4D Spatio-Temporal ConvNets that further improve the efficiency of sparse convolution through a more complex network structure. Multi-view CNNs represent the 3D point cloud with images rendered from different views and analyze the information based on a 2D CNN. Recently, Chen et al. [9] proposed to encode the sparse 3D point cloud with a compact multi-view representation for the task of object detection. Abhijit et al. [35] chooses multiple different virtual views from a 3D input to generate representative 2D channels for training a 2D model for the task of indoor 3D semantic segmentation. Moreover, some work designed convolutional kernels in special ways to improve the parsing ability of convolutional operations. Xu et al. [65] introduced SpiderCNN whose kernel is defined by a Taylor polynomial function with neighbors weighted so as to extend convolutional operations from regular grids to irregular point sets. Thomas et al. [54] introduced KPConv a new point convolution kernel which optimize the position of each weights dynamically. Besides, Li et al. [36] explored the improvement of 3D scene understanding performance in a non-Euclidean space.

Deep learning approaches often rely on huge datasets. ScanNet [14] and S3DIS [1] are richly annotated datasets of 3D reconstructed meshes of indoor scenes specifically designed for scene understanding. For the task of part-level understanding, ShapeNetPart [66] and PartNet [40] introduce a large-scale dataset with instance-level 3D objects annotated. As an outdoor scene dataset, KITTI [20] is widely used for optical flow estimation, 3D object detection and 3D tracking. However, such datasets are not yet fully available when considering point clouds for parameter values estimation. Moreover, aforementioned deep learning methods as a backbone are often used in 3D classification, 3D object detection and 3D scene segmentation tasks. There

exist few attempts to study their application in reverse engineering, especially to reconstruct parametric CAD models [34, 60]. To circumvent those limitations, this paper introduces SMA-Net, a multi-task convolutional neural network designed to perform both the classification and parameter estimation in one forward propagation during fitting process. Furthermore, a virtual data generation pipeline is proposed to allow the creation of large datasets ready for training, and to validate the approach.

3 RE framework and SMA-Net architecture

This section introduces the novel RE framework together with the architecture and technicalities of SMA-Net, the Spatial Merge Attention Network at the core of the proposed approach.

3.1 Reverse engineering framework

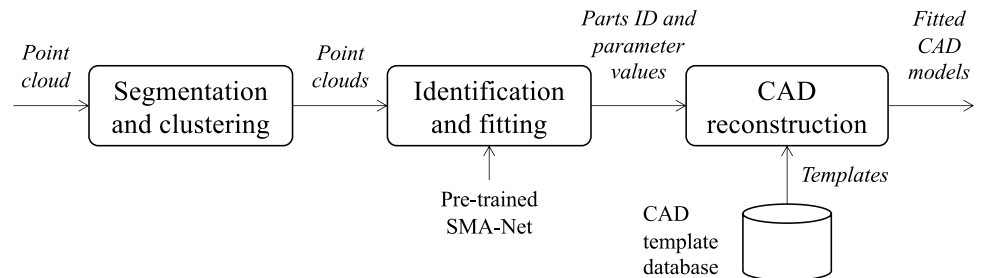
The RE framework is presented on Fig. 2. It starts with the acquisition of a raw point cloud, which then undergoes a segmentation and clustering step, which can be done by the existing well-designed semantic segmentation networks [63] and clustering algorithms [47]. In our implementation, SSCNs [23] pretrained on ScanNet dataset and DBSCAN [17] are used for segmentation and clustering respectively. The output point clouds are then analyzed by SMA-Net, which has been pre-trained upstream. Thanks to an elegant deep neural network and an end-to-end learning, SMA-Net allows object identification and parameter estimation into one module. It outputs the classes and estimated parameter values of the parts and assemblies identified in the point cloud. This information is then used as inputs of the reconstruction module in charge of generating all the CAD model instances using pre-defined templates. CAD templates include built-in constraints (e.g. symmetries, joints between parts) to be satisfied by the modeler during the reconstruction step. The whole framework is an end-to-end architecture without any loop or iteration, which makes it very fast and easy to control. Here, CAD models do not need to be updated at multiple iteration steps, and they are generated only once after the fitting.

3.2 Overview of the identification and fitting framework

To obtain the right class and precise parameter values, three problems have been tackled. The first is to extract features from the 3D object space in an efficient way, the second is to efficiently integrate information from a large amount of extracted features, and the third is to build a robust loss function to guide the learning process. SMA-Net has an elegant network architecture for solving these three problems. It consists of three main components depicted on Fig. 3:

- *Voxel-based multi-resolution feature extractor* to efficiently extract features from different spatial resolutions (Fig. 3a). The input is a set of N_p points of 3D coordinates $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, with $i \in [1 \dots N_p]$. Firstly, the complete point cloud is voxelized at a voxel size of V_s (s refers to the size of voxel) and then fed into a 3D CNN for feature extraction. The output is a set of feature maps generated at different resolutions and denoted as F^j , with $j \in [1 \dots N_l]$ and j the index of the extracted feature maps with different spatial resolutions.
- *Spatial merge attention mechanism* for robustly integrating features of different resolutions (Fig. 3b). Each extracted feature map F^j is forwarded to the spatial attention (SA) mechanism, which emphasizes spatial meaningful features of F^j along spatial axis. It generates a SA score M_s^j for each spatial resolution layer. Then, spatial recalibrated features F_s^j are computed by voxel-wise multiplications between M_s^j and F^j . A merge function is followed to squeeze and extract the important features from F_s^j , which are then fused to form the global feature vector V . During this step, the information related to the size of the raw point cloud are also incorporated. Specifically, the merge function is composed of a set of transformer blocks and a channel attention block combined to produce the final feature vector.
- *Multi-task head* for multi-parameters learning (Fig. 3c). After obtaining the final feature vector, highly characterized information is input to multi-heads part. The multi-heads is composed of a classifier and several parameter regressors. The output of the classifier is used to retrieve the CAD template model from data-

Fig. 2 Proposed RE framework exploiting the pre-trained SMA-Net for fast identification and fitting of CAD models from point clouds



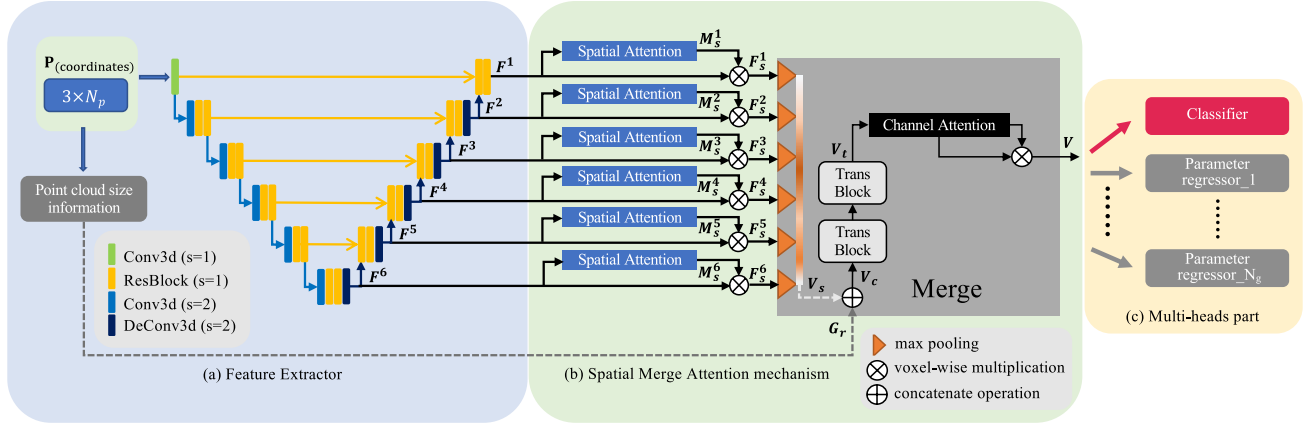


Fig. 3 Spatial Merge Attention Network (SMA-Net) highlighting three main components: **a** feature extractor extracting per-volume features F^j from different resolutions, **b** spatial merge attention mechanism

aggregating multi-resolution features to a vector, **c** multi-task head predicting the class and its parameters

base, and the parameter regressors are used for estimating parameter values precisely.

The outputs of SMA-Net then serve as inputs of the CAD reconstruction step, which extracts the identified CAD template and instantiates it with the estimated parameter values to get the final CAD model fitting the point cloud (Fig. 2).

3.3 Voxel-based multi-resolution feature extractor

Generally, any point feature extraction network can serve as the feature extractor. On the other hand, the grid-based methods have proven to be computationally efficient and are particularly suitable for image feature extraction due to their controllable receptive fields. Thanks to the emergence of sparse representation and sparse convolution technology [11, 25], these attributes have been well extended to the 3D field. Hence, the first component of SMA-Net exploits a voxel-based CNN for efficient feature encoding (Fig. 3a). In particular, the U-Net [46] architecture has been widely used in 2D and 3D domains especially for the semantic segmentation because of its efficient encoder-decoder structure and hierarchical expression. Thus, SMA-Net follows the U-Net like architecture with sparse representation and sparse convolution technology. Unlike previous methods [11, 23] which only use the information of the last layer of U-Net to derive the final output, SMA-Net exploits different feature extraction layers containing different structure size information. Indeed, even if their final output benefits from previous layers by skip-connections, these methods unavoidably miss useful information present at lower resolutions, which undoubtedly reduces the quality of the results. Actually, in a deep neural network, the receptive field increases as the depth of the network increases, and the deeper of the network, the more abstract of the extracted features. In other

words, the shallow layer contains rich local structure size information, while the deep layer contains more overall structure size information. For parameter estimation task, extracted features can represent the detail information of the target, but they are also able to express more abstract overall structural information.

To this end, the input point set are voxelized into volumes, then a series of sparse convolution operations are performed with skip connections to extract features from different spatial resolutions (Fig. 3a). Thus, the geometric and contextual features of different spatial resolutions are extracted. The output is a set of N_l feature maps generated at different spatial resolutions and denoted as $F^j \in \mathbb{R}^{C_j \times L_j \times W_j \times H_j}$, with $j \in [1 \dots N_l]$, j the index of features at different spatial resolutions, and C_j the feature channels with spatial resolution at $L_j \times W_j \times H_j$.

3.4 Spatial merge attention mechanism

The attention mechanism has made great achievements in many visual perception tasks such as image caption generators, multi-object detection and semantic segmentation. More recently, the attention mechanism represented by transformer has been widely used in the image field with very impressive results [15, 33, 38]. Thus, this part of SMA-Net consists of two submodules detailed in the next subsections (Fig. 3b): (1) voxel-based multi-resolution spatial attention (SA), (2) data merge (DM) consisting of two transformer blocks and a channel attention block. They have been specifically designed for enhancing the representational power of the network. The output of the Spatial Merge Attention mechanism is the feature vector V used as input of the multi-heads part (Fig. 3c).

3.4.1 Multi-resolution spatial attention

To further recalibrate spatial information, the SA submodule learns a set of spatial probability maps to weight the conv-layer feature maps of the CNN. Specifically, given a feature map F^j encoded by the hierarchical U-Net structure, SA sequentially infers a 3D spatial attention map $M_s^j \in \mathbb{R}^{1 \times L_j \times W_j \times H_j}$ (Fig. 4a). Next, the recalibrated feature map $F_s^j \in \mathbb{R}^{C_j \times L_j \times W_j \times H_j}$ can be computed as follows:

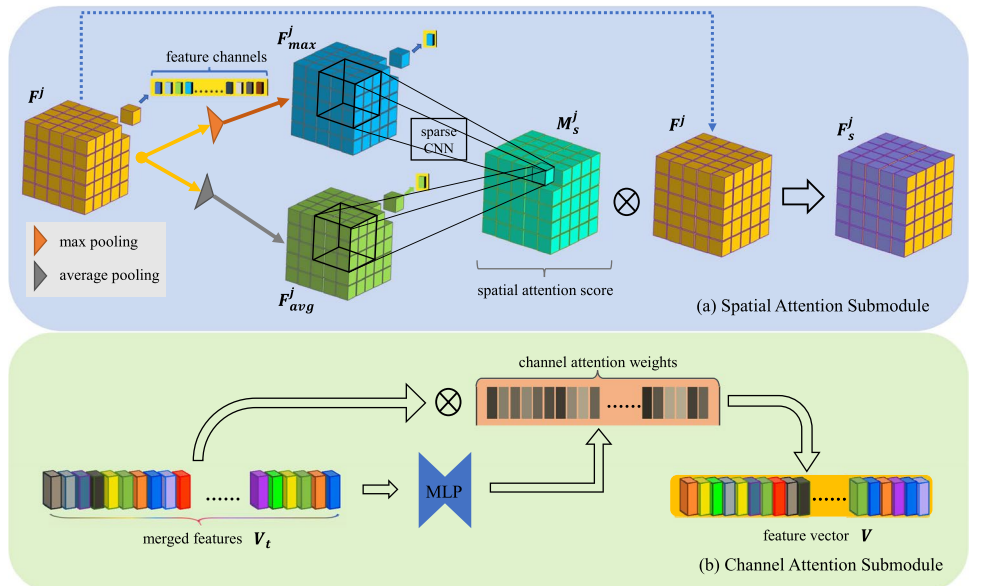
$$\begin{aligned} M_s^j &= \Phi(F^j), \text{ with } j \in [1 \dots N_l], \\ F_s^j &= M_s^j \otimes F^j, \end{aligned} \quad (1)$$

where $\Phi(\cdot)$ is the spatial attention function and \otimes the element-wise multiplication. In SMA-Net, this function first aggregates spatial information of different feature maps by means of max-pooling and average-pooling operations. The outputs of those operations are then forwarded to a light sparse convolutional neural network. The process can be summarized as:

$$\begin{aligned} \Phi(F^j) &= \sigma[\text{SConvs}(A^j)], \\ A^j &= \text{ReLU}\left(\text{IN}\left(\text{SConvs}\left(\text{Cat}\left(F_{\max}^j, F_{\text{avg}}^j\right)\right)\right)\right), \end{aligned} \quad (2)$$

where σ denotes the sigmoid function, **SConvs** denotes sparse convolution, **ReLU** and **IN** refer to Rectified Linear Unit activation function [22] and instance normalization [55] respectively, and **Cat** is the concatenate operation.

Fig. 4 Architecture of the attention mechanisms: **a** spatial attention submodule taking advantage of max-pooling and average-pooling operations forwarded to a 3D sparse CNN, **b** channel attention submodule combining outputs of max-pooling and average-pooling using a shared MLP network



3.4.2 Data merge

The data merge consists of two transformer blocks and a channel attention block. Figure 5.a illustrates the structure of a transformer block which consist of layer normalization (**LN**) [4], multi-head self-attention (**MSA**) [15, 56, 61] and a feed forward network (**FFN**) that can be replaced by a MLP (Sect. 4.3.1). Each transformer block inputs a sequence, and outputs another sequence after a forward propagation [56] in which each sequence is composed of several tokens, where each token is represented by a vector of dimension \mathbb{R}^{C_i} . Formally, the calculation of the output Z^l of a transformer block can be formulated as follows:

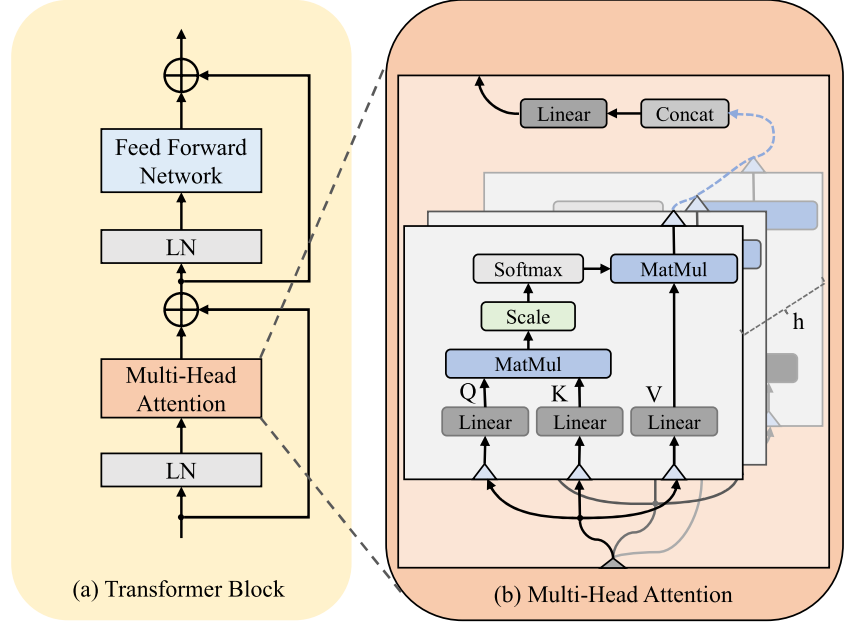
$$\begin{aligned} Z^l &= \text{FFN}\left(\text{LN}\left(\hat{Z}^l\right)\right) + \hat{Z}^l, \\ \hat{Z}^l &= \text{MSA}\left(\text{LN}\left(Z^{l-1}\right)\right) + Z^{l-1}, \end{aligned} \quad (3)$$

where Z^{l-1} is the output of the last block. Furthermore, **MSA** stands for multi-head self-attention, which is the core of transformer block and which can be formulated as follows (Fig. 5b):

$$\begin{aligned} \text{MSA}(Z) &= \text{Cat}(A_0, A_1, \dots, A_{h-1})M^c, \\ A_i &= \text{Softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_i^k}}\right)V_i, \\ Q_i &= ZM_i^q, K_i = ZM_i^k \text{ and } V_i = ZM_i^v, \end{aligned} \quad (4)$$

where $i \in [0, 1, \dots, h-1]$ is the head index and h is the number of heads, d_i^k is the dimension of K_i , and $M_i^q \in \mathbb{R}^{\frac{C_i}{h} \times \frac{C_i}{h}}$, $M_i^k \in \mathbb{R}^{\frac{C_i}{h} \times \frac{C_i}{h}}$, $M_i^v \in \mathbb{R}^{\frac{C_i}{h} \times \frac{C_i}{h}}$ and $M^c \in \mathbb{R}^{C_i \times C_i}$ are weight

Fig. 5 Overview of trans-former block. **a** architecture of transformer block, **b** multi-head attention in transformer block



matrices of linear projection functions. The self-attention operation in transformer block makes all tokens mutual interact, which in theory can yield global receptive field. Thus, in the field of 2D images, the input image is usually divided into a number of fixed-size patches, and each patch is considered as a token, thereby converting the 2D image into a sequence [13, 15, 61].

Considering the identification and fitting tasks, a simple way to generate a token is transforming a feature map F^j into a token T^j . However, the number of tokens depends on the depth of the U-Net pyramid structure, which is very limited. This inevitably causes a design contradiction: obtaining more tokens by increasing the depth of U-Net. Moreover, although this way can fuse features from different resolutions, considering the whole feature map in a resolution as a token will lose valuable details. To overcome this problem, a simple channel reshape strategy is used. Unlike [15, 61], tokens are not generated in spatial domain. Instead, tokens are produced along channel axis. L tokens are generated, and each token is represented by a C_t -dimensional vector.

In practice, data merge module takes the output of multi-resolution spatial attention F_s^j as input. It is first squeezed into a vector along spatial axis, so the length of the vector is the number of feature channels $V_s^j \in \mathbb{R}^{C_t}$. Then, all V_s^j , with $j \in [1 \dots N_l]$ are concatenated into one vector V_s . This process can be summarized as follows (Fig. 3):

$$\begin{aligned} V_s &= \text{Cat}(V_s^1, V_s^2, \dots, V_s^{N_l}), \\ V_s^j &= \text{MaxPool}(F_s^j), \text{ with } j \in [1 \dots N_l]. \end{aligned} \quad (5)$$

Moreover, the original size information, such as size d of the bounding box diagonal, maximum $(x_{\max}, y_{\max}, z_{\max})$ and

minimum $(x_{\min}, y_{\min}, z_{\min})$ of raw coordinate values P of point cloud provide a vital cues for 3D scene perception and understanding. Thus, the size information G_r of the original point cloud is also provided (Fig. 3):

$$\begin{aligned} G_r &= \text{Cat}(X_{\max}, X_{\min}, d) \in \mathbb{R}^7, \\ \text{with } \begin{cases} X_{\max} = (x_{\max}, y_{\max}, z_{\max}) = \mathbf{Max}(P, \text{axis} = 1) \\ X_{\min} = (x_{\min}, y_{\min}, z_{\min}) = \mathbf{Min}(P, \text{axis} = 1) \\ d = \|X_{\max} - X_{\min}\|. \end{cases} \end{aligned} \quad (6)$$

Then, V_s and G_r are further projected into a high-dimensional space and then combined through a concatenate operation to obtain V_c :

$$\begin{aligned} V_c &= \text{Cat}(V'_s, G'_r), \\ V'_s &= V_s M^s \text{ and } G'_r = G_r M^r, \end{aligned} \quad (7)$$

where $M^s \in \mathbb{R}^{len(V_s) \times (L-1) \cdot C_t}$ and $M^r \in \mathbb{R}^{7 \times C_t}$ are weight matrices. After, V_c is divided into L groups, resulting in L tokens.

$$\text{Tokens} = \text{Reshape}(V_c) \in \mathbb{R}^{L \cdot C_t}, \quad (8)$$

where $L \cdot C_t$ is equal to the length of V_c . In this way, tokens are defined more efficiently. More explanations are unfolded in the ablation study. Next, transformer blocks take the generated tokens as input to further aggregate the information and learn a global context. Then, all tokens are recompressed into a vector V_r . Finally, given the merged feature vector V_r , a channel attention network is applied to refine the features and produce the final feature vector to be used for the multi-task head (Fig. 4b):

$$V = A_c \otimes V_r, \quad (9)$$

$$A_c = \text{Softmax}(V_r M^1 M^2)$$

where $M^1 \in \mathbb{R}^{(L \cdot C_r) \times C_h}$, $M^2 \in \mathbb{R}^{C_h \times (L \cdot C_r)}$ are weight matrices.

3.5 Multi-task head

Multi-group joint parameter regression can be considered as a multi-task learning problem, which is intrinsically a multi-objective optimization problem because different tasks may conflict each other [49]. It is very intuitive that different classes which have similar shapes or sizes can contribute to each other when trained jointly, because they often share common features. During the training phase, these features can boost each other to achieve higher prediction results [68].

Thus, the last part of SMA-Net is a multi-group head network designed so that groups of similar shapes or sizes benefit from each other, and groups of different shapes or sizes stop interfering with each other. Let's consider N_c classes distributed in N_g groups. For instance, the furniture dataset, as introduced in the results section, is composed of 10 classes (4 chairs and 6 tables) distributed in 2 groups (the group of the chairs and the one of the tables). Each class $c \in [1 \dots N_c]$ belongs to a group $g \in [1 \dots N_g]$ and is characterized by N_p^c parameters p_i^c , with $i \in [1 \dots N_p^c]$. To be able to identify the class of a given object in a point cloud, and also find out its parameter values, the multi-task head module of SMA-Net is composed of (Fig. 3c):

- One *classification head* in which a multilayer perceptron (MLP) is applied to produce N_c classification scores $\{sc_1, sc_2, \dots, sc_{N_c}\} \in \mathbb{R}^{N_c}$ that are further regularized by a cross entropy loss function \mathcal{L}_{cls} . This part of SMA-Net allows identifying the class of an object in a point cloud, i.e. the class with the maximum score.
- Several *parameter regressors*, one for each of the N_g groups of classes. Each regressor is able to estimate all the parameter values of all the classes belonging to the corresponding group. Thus, depending on the class of the object identified by the classification head, only the subset of parameter values corresponding to that class is to be considered, and all the other parameter values are disregarded. The regressions are also realized by a MLP, followed by a parameter residual loss for guiding the parameter estimation learning. For a parameter p_i^c , the absolute deviation δp_i^c and the relative deviation Δp_i^c between the estimated value \tilde{p}_i^c and the ground truth value \bar{p}_i^c can be computed as follows:

$$\Delta p_i^c = \frac{\delta p_i^c}{\bar{p}_i^c} = \frac{\tilde{p}_i^c - \bar{p}_i^c}{\bar{p}_i^c}, \quad \forall i \in [1 \dots N_p^c]. \quad (10)$$

The deviations are collected within the parameter residual loss function of the regressor related to the g -th group for enabling SMA-Net reducing relative errors:

$$\mathcal{L}_{\text{reg}}^g = \sum_{c=1}^{N_c} \left(\delta_{c,g} \cdot \sum_{i=1}^{N_p^c} \mathbf{S}(\Delta p_i^c) \right), \quad (11)$$

$$\text{with } \mathbf{S}(\Delta p_i^c) = \begin{cases} 0.5(\Delta p_i^c)^2, & \text{if } \|\Delta p_i^c\| < 1 \\ \|\Delta p_i^c\| - 0.5, & \text{otherwise} \end{cases}$$

where $\delta_{c,g}$ is equal to 1 if c is a class of group g and 0 otherwise, and $\mathbf{S}(\cdot)$ denotes the standard SmoothL1 function. It makes the gradient of the model smoother, thereby, making the model more stable and easier to converge during the training process.

To improve the overall performance of classification and parameter estimation, SMA-Net is trained to minimize a joint loss function:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \sum_{g=1}^{N_g} \lambda_g \cdot \mathcal{L}_{\text{reg}}^g \quad (12)$$

with λ_g the hyper-parameters controlling the balance between the different loss functions. Finally, even though SMA-Net is a unique identification and fitting framework, Euclidian distances between the point cloud and the CAD model are not directly minimized during the learning, and the fitting is carried out at the parameter level. Those distances are however used at the end to check the quality of the fitting.

4 Experimental setup

This section introduces the virtual data generation pipeline designed to set up the databases and validate SMA-Net on both identification and estimation tasks. It also details the technicalities and settings of the different components of SMA-Net, including the implementation details on how two known network have been modified to allow a comparison with SMA-Net.

4.1 Virtual data generation pipeline

To evaluate and test SMA-Net on both identification and parameter estimation tasks, annotated 3D datasets are to be defined. Montlahuc et al. [41] have introduced an as-scanned point clouds generation strategy able to produce point clouds of CAD assembly models, similar to the ones that would have been obtained with a real scanner, i.e. including artifacts resulting from real acquisitions (e.g. noise, heterogeneous density, incompleteness). However, their

method produces point clouds without associating both the classes of the parts (e.g., tables and chairs) and the numerical parameter values of the CAD models, which are to be used for training a network. Here, the idea is to propose a virtual data generation pipeline, able to generate as-scanned point clouds, labeled for learning on both identification and parameter estimation tasks. The complete pipeline is presented in Fig. 6. For a given CAD template, it allows the creation of multiple CAD model instances as well as multiple as-scanned point clouds. The process starts by randomly generating parameter values used to generate the instances with a CAD modeler, i.e. the open-source parametric modeler FreeCAD in this work. The output of the

CAD modeler is input to the point cloud virtual generation algorithm. If virtual scan flag is set, HPR-based algorithm is used for generating as-scanned point clouds [32]. At the beginning, the origin and view direction of each virtual camera are randomly defined. After that, a post-processing step is performed to insert noise and to sub-sample the point clouds. At the end of each loop, the information is collected and stored: generated CAD model instance and mesh, as-scanned point cloud, label and parameter values.

4.2 Robot and furniture datasets

Two datasets have been designed to validate SMA-Net: robot dataset and furniture dataset. Generally, from the perspective of machine learning, angle-type parameters are more difficult to understand than general distance-type parameters. Thus, the considered robot template only includes angle-type parameters. This is used to verify the angle perception ability of SMA-Net, and to study its application for assembly fitting. The furniture dataset contains some common furnitures, i.e. chairs and tables, and the CAD templates include both angle-type and distance-type parameters. This second database is used to test the general performance of the proposed approach. Moreover, due to the presence of occlusion and noise in the real scene, as-scanned point clouds are generated for both datasets. Thus, it is possible to evaluate SMA-Net on point clouds obtained in real-world conditions, and not only using virtually-generated data. Importantly, SMA-Net has been trained and tested on the two datasets separately, and the datasets (labeled model instances, point clouds and parameter values) are made publicly available at the following URL: <https://doi.org/10.5281/zenodo.5031617>.

Robot dataset This dataset is composed of 5000 point clouds generated from a robot template (Fig. 7a). The pose of the robot is controlled by 4 angles A_1 – A_4 , whose values have been randomly changed in the range of $[0, 90]$. For each pose, an instance of the CAD template is generated, together with its mesh and as-scanned point cloud. Figures 7b₁ and b₂ shows meshes corresponding to two different poses, 7c₁ and c₂ are the virtually-generated point clouds, 7d₁ and d₂

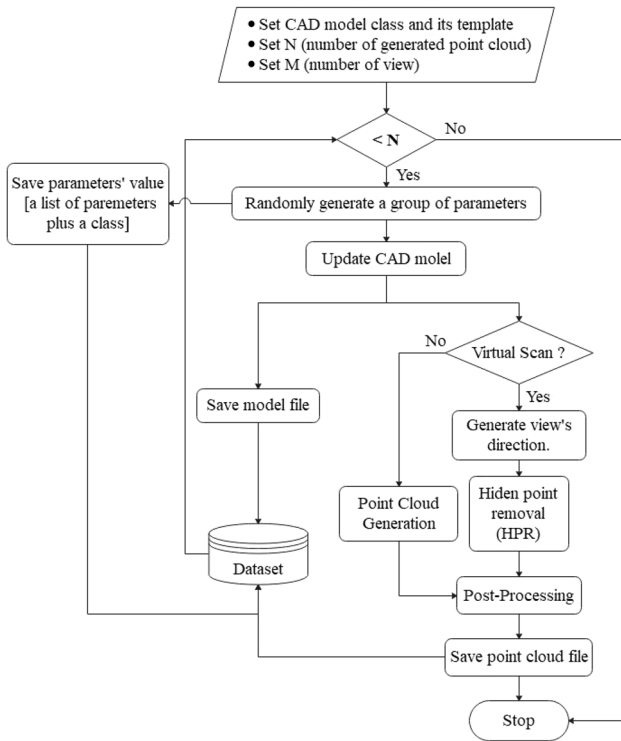


Fig. 6 Virtual data generation pipeline that uses labelled CAD template to generate many point cloud instances with their parameter values

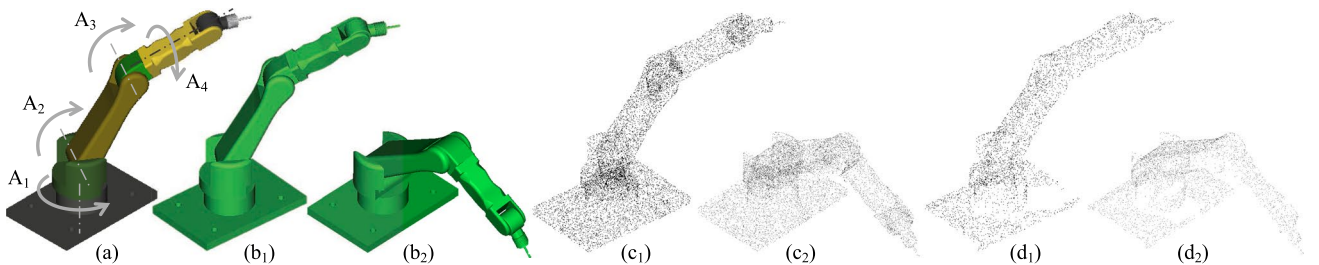


Fig. 7 Generation of as-scanned point clouds for the robot dataset: **a** robot template and its 4 control angles, **b_i** updated meshes of two different poses, **c_i** virtually-generated point clouds for the two poses, **d_i** as-scanned counterparts for the two poses

the as-scanned point clouds of those poses. The data is then split to training sub-dataset including 4000 data and testing sub-dataset including 1000 data. In addition, to evaluate the performance of the model during training, the training data is further divided into 3200 for training and 800 for validating. In this database, there is a single class ($N_c = 1$) belonging to a unique group ($N_g = 1$), and therefore a single regressor is used in the multi-task head. Here, the validation consists in testing the performance of SMA-Net to estimate the values of the 4 angles for an unknown pose represented by its possibly incomplete point cloud.

Furniture dataset This dataset contains two groups ($N_g = 2$) for a total of 10 classes ($N_c = 10$). Thus, SMA-Net is here composed of two regressors in its multi-task head, one for each group. Each regressor is in charge of estimating the parameter values of all the classes of the corresponding group. The first group gathers together 4 types of chairs, and the second 6 types of tables. Figure 8 shows the elements of the database, i.e. all the CAD templates of the considered classes, and examples of point clouds produced with different parameters. The 10 CAD templates are parameterized by more than 50 control parameters. Figure 9 shows the parameterization of a chair and a table for a total of 13 parameters. The final dataset consists of 20,000 point clouds, 2000 for each of the 10 classes. The point clouds are divided into 17,000 training and 3000 testing samples. To ensure the fairness of the test, 300 samples are uniformly randomly selected from each class for testing, and the rest for training. For experimental studies, the original training set is split into 15,000 training samples and 2000 validation samples. In the same way, 200 samples are uniformly and randomly selected from each class for validating and the rest for training. Here, the validation consists in testing the performance of SMA-Net to identify the class of an object and estimate its parameter values from a raw point cloud.

4.3 Technicalities and settings

This section details the way SMA-Net has been implemented and parameterized. It also explains how two classic networks have been tuned for comparison with SMA-Net.

4.3.1 SMA-Net

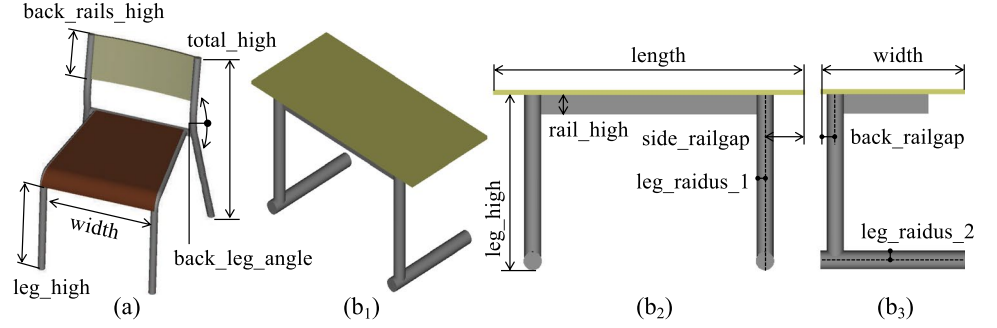
The implementation of SMA-Net is based on PyTorch [42] and MinkowskiEngine [11] for sparse convolution and other essential layers. The voxel size is set to 10 to balance the computational cost and accuracy. Higher performance could be achieved through setting a small voxel size, but it would bring more footprint than expected, especially in low-latency systems. Then, the three components of SMA-Net are customized as follows:



Fig. 8 Generation of as-scanned point clouds for the furniture dataset. First four rows are the chairs (Chair_1 to 4), and next six rows are the tables (Table_1 to 6). First column are CAD templates, second and third are updated meshes, fourth and fifth are virtually-generated point clouds, and last are as-scanned ones

- For the multi-resolution feature extractor, the depth of the U-Net hierarchy is set to 6 ($N_l = 6$). Deeper layers help extracting higher level features, and consequently more information after combination. To balance performance and computational complexity, a slim layer design strategy has been adopted. Thus, the output channels under each level in U-Net are set to m , $2m$, $3m$, $4m$, $5m$, $6m$ respectively, where m is empirically set to 24 to have a good balance between the performance of the model and the memory consumption.
- For each spatial attention module, 2 sparse convolutional layers are used with kernel size equal to 3. Normally,

Fig. 9 Examples of CAD templates and related parameters: **a** Chair_3 template and its parameters, **b₁** Table_1 template and its parameters (**b₂**, **b₃**)



convolution with kernel size equal to 3 is more efficient on GPU than other kernel size.

- For data merge, L and C_i are set to 32 and 64, respectively. The number of head in **MSA** is set to 16 ($h = 16$), while C_h is set to $\frac{L \cdot C_i}{16}$ in the current implementation. Moreover, **FFN** has been implemented by a two-layer MLP, where the dimension of the features in the hidden layer is equal to 4 times the dimension of the input features.
- For multi-task head, one head is used for classification, the others regress the parameters of different groups. MLP is used in both cases. Specifically, MLP in classifier has 2 hidden layers. Regressors have more parameters to predict and more abstract prediction goals, which is more difficult. Thus, 4 hidden layers are deployed in regressors. Empirically, one can observe that batch normalization largely hinders the regressor's learning ability, so the batch normalization of the last two layers has been removed.

During the training, data augmentation is applied on-the-fly by flipping with 50% probability, randomly rotating the objects along up-axis, and jittering data with Gauss noise. This is needed to be able to identify objects in any configuration when considering real-world examples. To strengthen the identification and estimation capacities of SMA-Net in various conditions, virtually-generated point clouds are also merged with as-scanned ones. Specifically, during the training, 40% of virtually-generated data are randomly selected and replaced by the as-scanned data. For testing, only as-scanned point clouds are used to validate the approach on real-life data.

The whole SMA-Net is trained in an end-to-end manner while minimizing the total loss function of Eq. (12) with hyper-parameters $\lambda_g = 1.0$. For the minimization, the Adam optimizer is adopted with an initial learning rate of 1×10^{-3} and decay the learning rate by a factor of 0.7 every 20 epochs and train for 200 epochs. All experiments have been performed on a laptop with NVIDIA RTX2080 GPU and Intel(R) Core(R) i7-9750 CPU.

4.3.2 Comparison setting

Two known models, namely *Pointnet2*[†] and *Resnet*[†] [28, 44], have been tuned for comparison with SMA-Net on our dataset. In particular, for *Pointnet2*[†], the implementation of [44] has been used as a starting point. In order to make it suitable for both classification and estimation tasks, the classification head of the original implementation has been replaced by the multi-heads part proposed in this paper. For training, the original input have been uniformly sampled with 4096 points. For *Resnet*[†], the 3D-Resnet [11] based on MinkowskiEngine sparse convolutional library has been re-implemented. More concretely, sparse tensor and sparse operation replace the dense tensor and related operations in original *Resnet*[†]. Similarly to *Pointnet2*[†], the multi-heads part of SMA-Net replaces the original classification head of *Resnet*[†]. In addition, the structure of Res-34 has been used in our experiment. Finally, the same training strategy of SMA-Net has been adopted to train both *Pointnet2*[†] and *Resnet*[†]. In this paper, [†] thus indicates that multi-head is used instead of the original classification head to make the model compatible with the fitting task.

5 Results and discussion

The proposed approach has been validated on two up-to-date challenging scenarios. The first scenario illustrates how SMA-Net can be used to follow the evolutions of a real robot and update its digital twin for Industry 4.0 applications. The second scenario shows how SMA-Net can identify objects in a real environment, and estimate their parameter values to allow fast reconstruction of 3D environments for BIM applications.

5.1 Evaluation criteria

Several criteria are here adopted to evaluate the performances of SMA-Net on both identification and parameter estimation tasks. For the identification task, the adopted

criterion is the one of [8]. For parameter estimation task, the accuracy of the estimations can be carried out in an absolute but also relative manner. Therefore, for a given parameter p_i^c related to a class c , the average of the absolute and relative errors on a test dataset can be computed as follows:

$$E_r^{\text{avg}}(c, i) = \frac{1}{N_{\text{test}}^c} \sum_{k=1}^{N_{\text{test}}^c} \|\Delta p_i^c(k)\|$$

$$E_a^{\text{avg}}(c, i) = \frac{1}{N_{\text{test}}^c} \sum_{k=1}^{N_{\text{test}}^c} \|\delta p_i^c(k)\|, \quad (13)$$

where N_{test}^c refers to the number of samples of class c in the test dataset, $\Delta p_i^c(k)$ and $\delta p_i^c(k)$ represent respectively the relative and absolute errors of parameter p_i^c on the k -th element of class c in the test dataset. Errors are computed using Eq. (10). If angle-type and distance-type parameters appear at the same time when computing the average value, by default all parameter values are directly added to calculate the average value. Further, the mean average values (E_r^m and E_a^m) are used to describe the overall errors, which are defined as follows:

$$E_r^m = \frac{1}{N_c} \sum_{c=1}^{N_c} \frac{1}{N_p^c} \sum_{i=1}^{N_p^c} E_r^{\text{avg}}(c, i)$$

$$E_a^m = \frac{1}{N_c} \sum_{c=1}^{N_c} \frac{1}{N_p^c} \sum_{i=1}^{N_p^c} E_a^{\text{avg}}(c, i), \quad (14)$$

where N_c and N_p^c refer respectively to the number of classes and the number of parameters in class c .

In addition, to assess the quality of a fitting result, Euclidian distances are computed between the point cloud and the CAD model updated with parameters estimated by SMA-Net. This is performed in CloudCompare, and for a sample

of the test dataset, average distance, max distance, and standard deviation are provided. Finally, the efficiency of SMA-Net is evaluated in terms of time required to perform both the identification and estimation tasks. Classically, the efficiency is expressed in Hertz (Hz) to show how many samples the network treats every second.

5.2 Quantitative analysis

Figure 10 shows the evolution of the relative and absolute errors during the training on the robot dataset, and Table 1 gathers the results of the evaluation of *Pointnet2*[†], *Resnet*[†] and SMA-Net on the test dataset composed of as-scanned point clouds. Here, there is no identification task as the dataset is composed of a single class, i.e. the four-axis robot. Compared with *Pointnet2*[†] and *Resnet*[†], SMA-Net achieves the best results for the estimation of the four angles. From the results, axis-1 (A_1) and axis-2 (A_2) have smaller absolute error around 0.7° and 0.6° , respectively. The absolute error of axis-4 (A_4) is larger which around 3.1. The results are very intuitive, the spatial position of axis-3 and axis-4 strongly dependent on the position of axis-1 and axis-2. SMA-Net thus gets more information from the changes of axis-1 and axis-2 whose behavior is therefore better approximated. In terms of efficiency, the training takes around 5 h, and SMA-Net is able to estimate the angles at a frequency of 30.83 Hz, which is far enough for Industry 4.0 applications, including updating the digital twin of such a system evolving in time. Moreover, the size of the model is 12M parameters, which makes it possible to consider applications on low performance devices.

Table 2 show the results of *Pointnet2*[†], *Resnet*[†] and SMA-Net for both the identification and parameter estimation

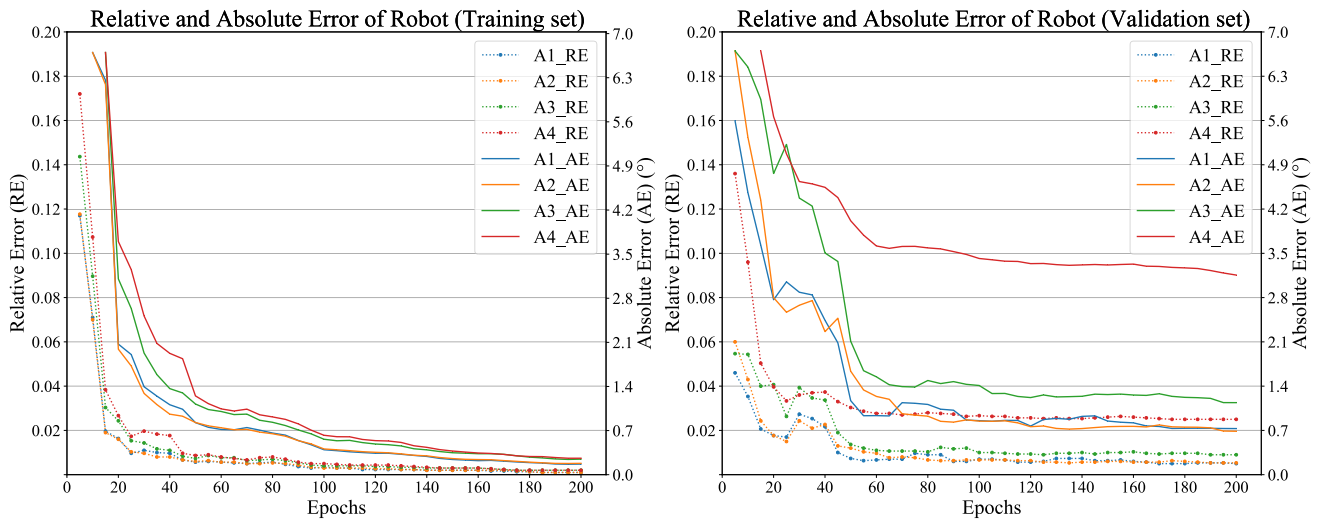


Fig. 10 Training curves with relative errors E_r^{avg} (dotted line) and absolute error E_a^{avg} (solid line) for the robot dataset and its four angle values to estimate: result curve on training set (top), result curve on validation set (down)

Table 1 Evaluation result on the robot test dataset ($N_{\text{test}}^{\text{Robot}} = 1000$)

Model	Parameter A_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	A_1		A_2		A_3		A_4	
<i>Pointnet2</i> [†]	0.007	0.92°	0.007	0.84°	0.010	1.27°	0.028	3.62°
<i>ResNet</i> [†]	0.006	0.82°	0.006	0.69°	0.013	1.62°	0.027	3.46°
SMA-Net	0.005	0.68°	0.004	0.57°	0.007	1.05°	0.025	3.09°
Mean								
Model	E_r^m	E_a^m	Params. (M)					
<i>Pointnet2</i> [†]	0.013	1.66°	4.6					
<i>ResNet</i> [†]	0.013	1.65°	64.1					
SMA-Net	0.011	1.35°	12.1					

Table 2 Results on the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$)

Model	Mean		Acc.	Params. (M)
	E_r^m	E_a^m		
<i>Pointnet2</i> [†]	0.097	20.15	1.0	6.2
<i>ResNet</i> [†]	0.033	8.26	1.0	64.7
SMA-Net	0.027	6.48	1.0	13.9

tasks on the furniture dataset. It proves that the three models can all accurately identify CAD models from the database (Acc.= 1.0 in Table 2), but that SMA-Net far outperforms *Pointnet2*[†] and *Resnet*[†] on the estimation task. Indeed, *Pointnet2*[†] fails to estimate accurately the parameter values, and *Resnet*[†] also generates large estimation errors compared with SMA-Net. In the appendix, Tables 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16 illustrate the result of each parameter. From the results, SMA-Net is able to manage both angle-type and distance-type parameters, to identify object and to perform estimations in a highly efficient way. For example, considering Chair_3 (Fig. 9a, Table 9), 'back_leg_angle' (p_{16}) relative error is 0.013 which is on the same order of magnitude as the relative error of the other distance-type parameters. Table_1 (Fig. 9b_i) has 8 parameters, and compared to parameters of 'length' (p_{23}), 'width' (p_{24}) and 'leg_high' (p_{30}), parameters of 'back_railgap' (p_{25}), 'side_railgap' (p_{26}), 'rail_high' (p_{27}), 'leg_radius_1' (p_{28}) and 'leg_radius_2' (p_{29}) have a very small size, which

make them more difficult to fit. Indeed, the fitting of small-size features usually consumes more iteration cycles than large-size ones. In terms of efficiency, the training takes around 14 h, and SMA-Net is able to identify objects and estimate parameter values in one forward propagation, and at a frequency of 30.03 Hz, which is very promising for the development of AR/VR applications. Here, the model only takes 13.9 M parameters.

Furthermore, the sensitivity of SMA-Net to noise has been evaluated while generating a series of virtual scan data incorporating different levels of noise. Specifically, noise is added using a Gaussian noise distribution controlled by an amplitude factor varying from 0 to 30 mm as shown in Table 3. For all evaluations in Table 3, the same pre-trained model is used. One can see that when the noise increases from 0 to 5 mm, the absolute error of the parameter changes less than 0.6. Then, the absolute error grows slightly with the noise of 10 mm, however, when the noise increases to 30 mm, the absolute error of the model increases significantly. It can be concluded that SMA-Net is not sensitive to the presence of a reasonable noise, i.e. a level of noise that can be encountered with current acquisition devices.

5.3 Qualitative analysis

Figures 11 and 12 illustrate how the proposed RE framework can be used to fit efficiently CAD parts and assemblies on point clouds. Here, the point clouds have been obtained

Table 3 Evaluation result of the absolute errors $E_a^{\text{avg}}(c, i)$ with respect to the amplitude of the inserted noise (0 mm, 1 mm, 5 mm, 10 mm, 30 mm) on the robot test dataset ($N_{\text{test}}^{\text{Robot}} = 1000$)

Class c	Parameter p_i^c	Absolute errors $E_a^{\text{avg}}(c, i)$				
		0 mm	1 mm	5 mm	10 mm	30 mm
Robot	A_0	0.78°	0.68°	1.11°	2.39°	8.68°
	A_1	0.67°	0.57°	0.84°	1.62°	4.94°
	A_2	1.17°	1.05°	1.26°	1.64°	6.98°
	A_3	3.39°	3.09°	3.78°	5.72°	18.95°

following the novel virtual data generation pipeline. Given an input point cloud, SMA-Net identifies the label of the object (e.g. Robot and Chair_3 in those examples) and estimates the parameter values in one forward propagation. Those values are then forwarded to the CAD modeler that updates the corresponding CAD model and creates the mesh. Then, it is possible to compute distances between the point cloud and the mesh to display a color map. With quite low mean distances when compared to the size of the objects, those results clearly show the capacity of SMA-Net to identify and fit an object in an incomplete point cloud.

The proposed identification and fitting technique has also been validated on real scanned data. Figure 13 shows the result of a fitting on a point cloud acquired with a laser scanner ROMER Absolute Arm 7520 SI and sub-sampled to 10,000 points. Here, an ICP [21] algorithm aligns the input point cloud and the mesh resulting from the update of the CAD template with the parameter values obtained from

SMA-Net. Results are good, even though a bit below the ones obtained on as-scanned point clouds. This is because the adopted CAD template does not fully match the scanned chair. Moreover, chairs are slightly deformable objects, which may also explain possible deviations between a real-world point cloud and a perfect CAD template.

Figure 14a, b illustrate how the proposed RE framework can be used for real-time reconstruction of CAD models within a virtual scene. From left to right, the raw point cloud is first acquired using Kinect V1, and then transferred to a pre-trained semantic segmentation model [23] to label points and extract objects of interest by clustering [17]. Then, for each segmented point cloud, SMA-Net is used to identify the underlying object and estimate its parameters in one forward propagation. Finally, using the outputs of SMA-Net, CAD models can be reconstructed and located in the virtual scene using ICP. Here, measured deviations can be ascribed to the fact that the underlying CAD templates are simplified ones,

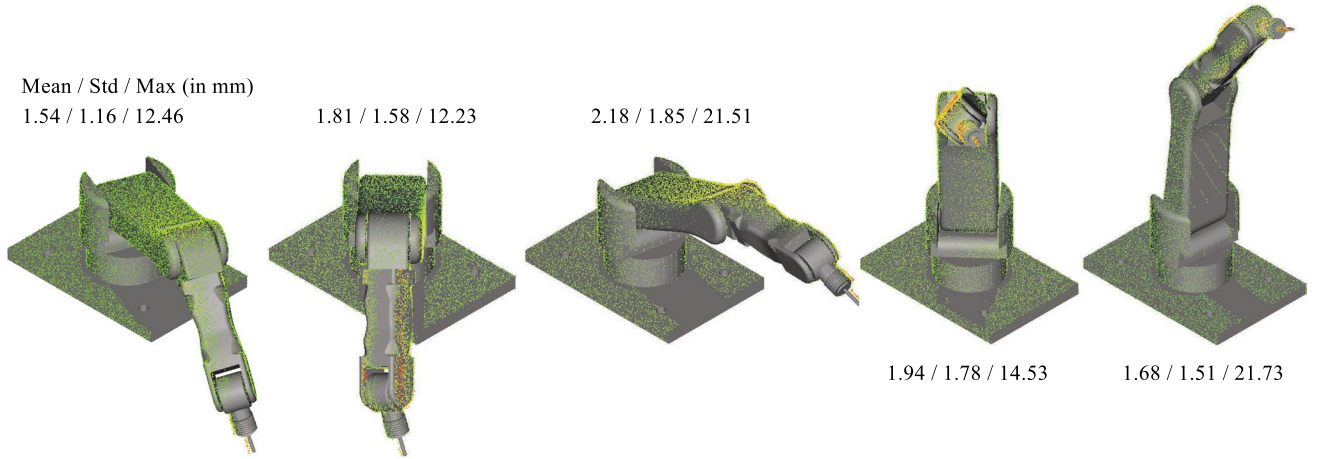


Fig. 11 Robot fitting results on five different poses of the test dataset: as-scanned point clouds (colored points where red indicates larger deviations) used as inputs of SMA-Net, and meshes (gray) generated from the updated CAD models using outputs of SMA-Net

Fig. 12 Chair_3 fitting results on samples from the test dataset: as-scanned point clouds used as inputs of SMA-Net, and meshes generated from the updated CAD models using estimated outputs of SMA-Net

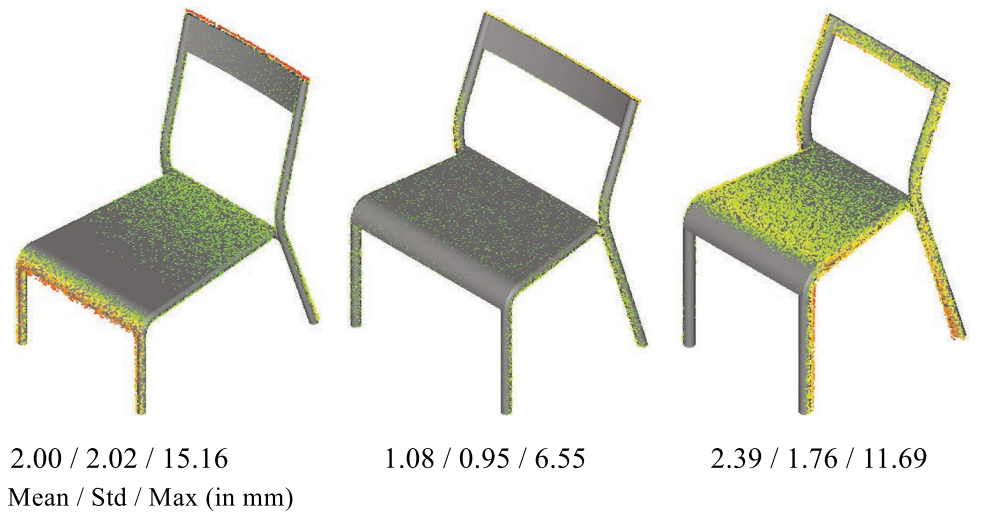


Fig. 13 Multiple views (b_1 – b_3) of the fitting result of Chair_3 on the point cloud acquired with a laser scanner (a)

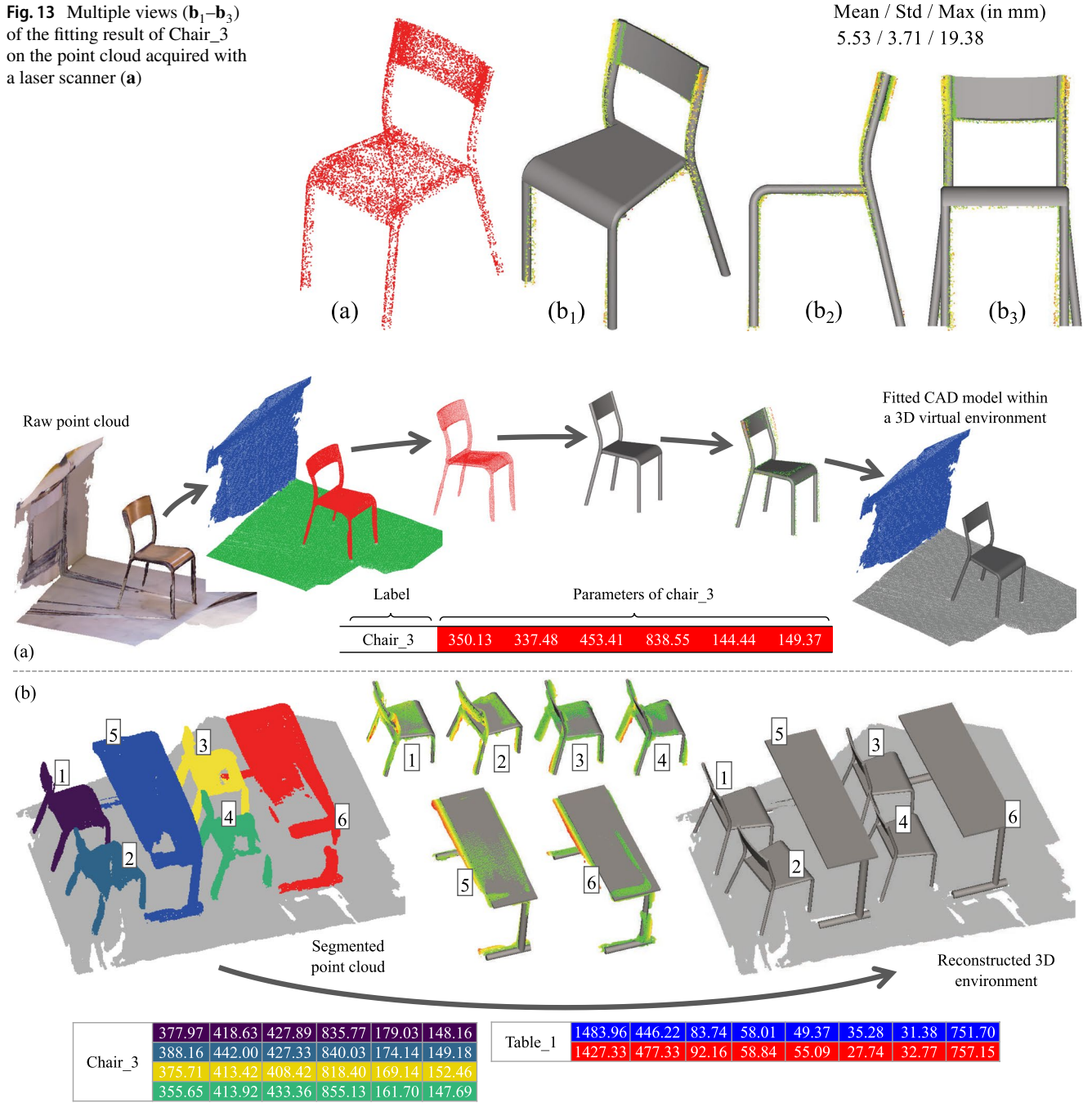


Fig. 14 Complete reconstruction of virtual environments from point clouds captured with Kinect V1. The whole scene is first segmented by a deep segmentation network, and a clustering algorithm is used to group points. Then, for each point cloud, SMA-Net identifies the

label of the underlying object and estimate its parameters in one forward propagation. Finally, CAD models are reconstructed using the output values of SMA-Net, and they are located in the 3D scene using ICP

which cannot capture the full complexity of the acquired objects, and more parameters would be necessary. This need is also true when considering possible object deformations due to aging, which would require additional modeling parameters. Finally, outliers and artifacts due to the acquisition with Kinect also have an impact on the results and create a larger variance in some places.

5.4 Ablation study

The architecture of SMA-Net has been designed around three components, namely multi-resolution feature extractor, spatial merge attention module and multi-head module, all of which have a vital impact on the performances. This ablation study evaluates how each component affects the

average of the relative errors. For fast evaluation, a sub-dataset of furniture dataset is exploited, which consists of six classes namely Chair_1, Chair_2, Chair_3, Table_4, Table_5, and Table_6. The configuration of the hyperparameters is the same as for the above experiments. Table 4 shows the experimental results. The five leftmost columns are used to configure the experiments. Indeed, the second column represents the possibility to combine or not the information from multiple resolutions (MR) during the feature extraction stage. The third one indicates the adoption or not of the spatial attention (SA) mechanism. The fourth column tells the data merge module (DM) is applied or not. The fifth column characterizes whether multiple-heads (MH) are used or not to regress the N_g groups. To highlight the contribution of each component, seven experiments were conducted.

The first experiment can be considered as a baseline. Specifically, as each component is set up to 'No', only the output of the last layer of the feature extractor is used as input of the next part, the spatial attention and merge modules are removed, and for the final parameter prediction, only one regressor is integrated. The second experiment clearly shows that multi-heads can help to decouple features between different groups to a certain extent. Then, for the third experiment, the MR feature extractor is activated to integrate features from different scales. The result shows that rich multi-scale features help improve the performance of SMA-Net. Instead of getting better results, deploying the SA sub-module slightly hurts the performance of the model. This is because the features after calibration of SA are too abstract to parse it through a simple network. This reflects the effectiveness and importance of the transformer-based DM sub-module from the other side. Finally, in the sixth experiment, all four modules are activated, and the best results are obtained.

In addition, for the last experiment, tokens are generated by transforming a feature map F^j into a token T^j that is detailed in Sect. 3.4. From the results, it is clear that the proposed token generation process is more efficient than just simply transforming a feature map F^j into a token T^j .

As a conclusion, the extraction of features from multiple resolutions can get more useful information than just

collecting features from only one layer. Moreover, introducing spatial merge attention module can help SMA-Net integrating more meaningful information and have a larger receptive field during the feature merge stage. In addition, multi-head makes it possible to deal with different target groups. Moreover, it can also reduce the coupling between different groups, and at the same time it helps promoting mutual boost between the classes which belong to the same group.

6 Conclusion and future work

This paper introduces a novel reverse engineering framework able to identify and fit CAD models from incomplete point clouds. The core of the approach is SMA-Net, a fast bottom-up 3D classification and fitting framework able to identify objects and estimate the parameter values in an end-to-end manner. The network is trained using point clouds generated following a virtual data generation pipeline able to create many as-scanned point clouds.

Differently, from the traditional reverse engineering techniques working at the level of the patches and facing numerous trimming and stitching issues, the proposed approach carries out an end-to-end identification and parameter estimation in one forward propagation, and it achieves very good results on both virtually-generated datasets and real-life acquisitions. The geometric modeling operations are left to the CAD modeler in charge of generating CAD models from CAD templates using the output values of SMA-Net. Considering possibly incomplete point clouds, resulting from the occlusion phenomenon, the results have shown that SMA-Net is able to rely on other features to infer reasonable parameter values for the partially visible parts. In addition, future works should focus on integrating all modules shown in Fig. 2 into a unified reverse engineering framework to realize end-to-end reasoning from the entire scene to the CAD model. Also, needs concern the possibility to automatically adjust CAD templates to take into account local deformations and thus be able to model structural ageing.

Table 4 Ablation study evaluating the interest of each component of SMA-Net

Model	MR	SA	DM	MH	E_r^m	E_a^m (mm)
SMA-Net	No	No	No	No	0.035	13.19
SMA-Net	No	No	No	Yes	0.031	10.94
SMA-Net	Yes	No	No	Yes	0.024	8.06
SMA-Net	Yes	Yes	No	Yes	0.025	8.66
SMA-Net	Yes	No	Yes	Yes	0.023	7.23
SMA-Net	Yes	Yes	Yes	Yes	0.022	6.5
SMA-Net [‡]	Yes	Yes	Yes	Yes	0.025	8.29

The values in bold indicate the optimal results

The "‡" refers to tokens generated by transforming a feature map F^j into a token T^j , following details in 3.4

Moreover, two application scenarios have been presented with promising results to serve in the context of the Industry 4.0. The one on the robot is particularly adapted to maintain the coherence of the digital twin to track the evolution of the physical twin in time. The second one involves the complete reconstruction of a virtual environment from a Kinect acquisition. In both cases, SMA-Net demonstrated good accuracy and very high speed in identification and estimation tasks, and its performances are compatible with Industry 4.0 real-time simulation requirements. Both datasets are made publicly available at the following URL: <https://doi.org/10.5281/zenodo.5031617>. Last but not least, the results obtained from our method can also be considered as a group of initial parameters for the next step of fine-tuning based on more traditional methods. At the same time, a good initialization will allow the algorithm to converge quickly to a local minimum and thus gain higher accuracy, which is of great research interest and practical value.

Appendix 1

See Tables 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16.

Table 5 Parameter name and index relationship of the chairs in the furniture dataset

Class c	Name	Parameter p_i
Chair_1	p_seat	p_1
	u_shape_dia	p_2
	u_loc	p_3
	leg_loc	p_4
	leg_high	p_5
Chair_2	sideRails_length	p_6
	seat_length	p_7
	width	p_8
	backlegs_high	p_9
Chair_3	frontLegs_high	p_{10}
	length	p_{11}
	width	p_{12}
	leg_high	p_{13}
	total_high	p_{14}
	back_rails_high	p_{15}
Chair_4	back_leg_angle	p_{16}
	width_seat	p_{17}
	fillet_seat	p_{18}
	rail_seat	p_{19}
	high	p_{20}
	u_len	p_{21}
	u_dis	p_{22}

Table 6 Parameter name and index relationship of the tables in the furniture dataset

Table_1	length	p_{23}
	width	p_{24}
	back_railgap	p_{25}
	side_railgap	p_{26}
	rail_high	p_{27}
	leg_radius_1	p_{28}
Table_2	leg_radius_2	p_{29}
	leg_high	p_{30}
	upLength	p_{31}
	upWidth	p_{32}
	downLength	p_{33}
	downWidth	p_{34}
	legHigh	p_{35}
Table_3	legDis	p_{36}
	legDia	p_{37}
	upLength	p_{38}
	downLength	p_{39}
Table_4	legHigh	p_{40}
	legDia	p_{41}
	high	p_{42}
	dia	p_{43}
	leg_dis	p_{44}
	support_dis	p_{45}
	u_len	p_{46}
	u_dia	p_{47}
Table_5	u_loc	p_{48}
	width	p_{49}
	length	p_{50}
	high	p_{51}
Table_6	radius	p_{52}
	high	p_{53}
	dia	p_{54}
	leg_angle	p_{55}
	support_dis	p_{56}

Table 7 Result of Chair_1 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 5

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_1		p_2		p_3		p_4	
<i>Pointnet2</i> [†]	0.022	6.02	0.270	11.14	0.059	8.88	0.023	5.92
<i>ResNet</i> [†]	0.011	2.96	0.035	1.37	0.026	4.07	0.011	2.55
SMA-Net	0.008	1.89	0.028	1.12	0.014	2.16	0.007	1.56
Model	p_5							
<i>Pointnet2</i> [†]	0.012	6.51						
<i>ResNet</i> [†]	0.010	5.25						
SMA-Net	0.004	2.01						

Table 8 Result of Chair_2 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 5

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_6		p_7		p_8		p_9	
<i>Pointnet2</i> [†]	0.060	27.03	0.053	26.88	0.070	33.28	0.008	9.38
<i>ResNet</i> [†]	0.012	5.66	0.010	5.78	0.014	7.72	0.006	6.90
SMA-Net	0.010	5.21	0.009	5.26	0.011	5.88	0.003	3.51
Model	p_{10}							
<i>Pointnet2</i> [†]	0.018	14.91						
<i>ResNet</i> [†]	0.008	5.98						
SMA-Net	0.006	4.07						

Table 9 Result of Chair_3 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 5

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{11}		p_{12}		p_{13}		p_{14}	
<i>Pointnet2</i> [†]	0.094	48.48	0.084	46.71	0.039	18.61	0.026	24.69
<i>ResNet</i> [†]	0.019	9.06	0.016	7.94	0.011	5.50	0.012	11.09
SMA-Net	0.016	7.83	0.012	6.25	0.014	6.72	0.005	4.67
Model	p_{15}							
<i>Pointnet2</i> [†]	0.571	101.89	0.029	4.37				
<i>ResNet</i> [†]	0.031	4.38	0.014	2.21				
SMA-Net	0.022	3.10	0.013	1.94				

Table 10 Result of Chair_4 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 5

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{17}		p_{18}		p_{19}		p_{20}	
<i>Pointnet2</i> [†]	0.064	34.90	0.173	10.76	0.123	24.15	0.025	11.12
<i>ResNet</i> [†]	0.050	26.42	0.117	7.77	0.083	15.03	0.033	14.86
SMA-Net	0.043	23.29	0.090	5.97	0.078	13.48	0.026	11.56
Model	p_{21}		p_{22}					
<i>Pointnet2</i> [†]	0.394	21.20	0.663	45.81				
<i>ResNet</i> [†]	0.099	5.01	0.250	10.99				
SMA-Net	0.090	4.49	0.229	9.50				

Table 11 Result of Table_1 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 6

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{23}		p_{24}		p_{25}		p_{26}	
<i>Pointnet2</i> [†]	0.021	28.42	0.025	14.80	0.189	11.67	0.101	7.71
<i>ResNet</i> [†]	0.015	18.79	0.021	12.17	0.088	4.83	0.053	3.73
SMA-Net	0.010	12.92	0.015	8.16	0.063	3.39	0.042	3.09
Model	p_{27}		p_{28}		p_{29}		p_{30}	
<i>Pointnet2</i> [†]	0.332	16.57	0.176	4.49	0.123	3.19	0.020	13.36
<i>ResNet</i> [†]	0.059	2.62	0.030	0.72	0.027	0.72	0.014	8.83
SMA-Net	0.058	2.56	0.022	0.54	0.027	0.71	0.007	4.54

Table 12 Result of Table_2 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 6

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{31}		p_{32}		p_{33}		p_{34}	
<i>Pointnet2</i> [†]	0.019	19.94	0.029	16.54	0.058	34.93	0.061	27.07
<i>ResNet</i> [†]	0.011	11.68	0.017	9.90	0.018	10.69	0.022	9.59
SMA-Net	0.009	9.71	0.011	6.54	0.013	7.72	0.013	5.86
Model	p_{35}		p_{36}		p_{37}			
<i>Pointnet2</i> [†]	0.011	8.51	0.118	27.68	0.118	12.49		
<i>ResNet</i> [†]	0.010	7.07	0.022	4.97	0.015	1.82		
SMA-Net	0.003	2.39	0.019	4.34	0.014	1.61		

Table 13 Result of Table_3 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 6

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{38}	p_{39}	p_{40}	p_{41}	p_{42}	p_{43}	p_{44}	p_{45}
<i>Pointnet2</i> [†]	0.014	12.90	0.019	10.16	0.012	6.15	0.047	6.03
<i>ResNet</i> [†]	0.008	6.88	0.010	4.96	0.009	4.72	0.016	2.25
SMA-Net	0.005	4.56	0.009	4.57	0.004	1.79	0.017	2.34

Table 14 Result of Table_4 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 6

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{42}	p_{43}	p_{44}	p_{45}	p_{46}	p_{47}	p_{48}	p_{49}
<i>Pointnet2</i> [†]	0.016	13.13	0.022	26.59	0.035	30.10	0.107	63.67
<i>ResNet</i> [†]	0.024	19.11	0.024	2.87	0.037	31.94	0.053	35.47
SMA-Net	0.014	11.01	0.020	23.37	0.033	29.37	0.043	29.65
Model	p_{46}	p_{47}	p_{48}	p_{49}	p_{50}	p_{51}	p_{52}	p_{53}
	p_{46}	p_{47}	p_{48}	p_{49}	p_{50}	p_{51}	p_{52}	p_{53}
<i>Pointnet2</i> [†]	0.280	12.48	0.421	14.38	0.092	7.80	0.323	71.72
<i>ResNet</i> [†]	0.176	7.67	0.230	6.37	0.048	4.51	0.035	6.30
SMA-Net	0.129	5.91	0.202	5.89	0.038	3.73	0.029	5.29

Table 15 Result of Table_5 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 6

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{49}	p_{50}	p_{51}	p_{52}	p_{53}	p_{54}	p_{55}	p_{56}
<i>Pointnet2</i> [†]	0.024	19.63	0.014	15.69	0.010	7.87	0.323	71.72
<i>ResNet</i> [†]	0.011	8.65	0.010	12.63	0.007	5.94	0.035	6.30
SMA-Net	0.009	7.14	0.008	9.28	0.003	2.12	0.029	5.29

Table 16 Result of Table_6 in the furniture test dataset ($N_{\text{test}}^{\text{Furniture}} = 3000$), with the name of each parameter p_i provided in Table 6

Model	Parameter p_i and its $E_r^{\text{avg}}(c, i)$ and $E_a^{\text{avg}}(c, i)$ values							
	p_{53}	p_{54}	p_{55}	p_{56}	p_{57}	p_{58}	p_{59}	p_{60}
<i>Pointnet2</i> [†]	0.011	8.50	0.017	14.50	0.028	2.59	0.083	18.06
<i>ResNet</i> [†]	0.010	8.19	0.011	9.69	0.016	1.50	0.026	5.83
SMA-Net	0.003	2.68	0.011	9.08	0.014	1.32	0.025	5.35

References

- Armeni I, Sener O, Zamir AR, Jiang H, Brilakis I, Fischer M, Savarese S (2016) 3D semantic parsing of large-scale indoor spaces. In IEEE Conf. on comput. vision and pattern recognition, pp 1534–1543
- Avetisyan A, Dahnert M, Dai A, Savva M, Chang AX, Nießner M (2019) Scan2cad: learning cad model alignment in rgb-d scans. In Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 2614–2623
- Avetisyan A, Dai A, Nießner M (2019) End-to-end cad model retrieval and 9dof alignment in 3d scans. In: Proceedings of the IEEE/CVF International Conference on computer vision, pp 2551–2560
- Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. arXiv preprint [arXiv:1607.06450](https://arxiv.org/abs/1607.06450)
- Bey A, Chaine R, Marc R, Thibault G, Akkouche S (2011) Reconstruction of consistent 3d CAD models from point cloud data using a priori CAD models. ISPRS 3812:289–294
- Buonamici F, Carfagni M, Furferi R, Governi L, Lapini A, Volpe Y (2018) Reverse engineering modeling methods and tools: a survey. Comput-Aided Des Appl 15(3):443–464

7. Buonamici F, Carfagni M, Furferi R, Governi L, Lapini A, Volpe Y (2018) Reverse engineering of mechanical parts: a template-based approach. *J Comput Des Eng* 5(2):145–159
8. Charles RQ, Su H, Kaichun M, Guibas LJ (2017) PointNet: deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conf. on comput. vision and pattern recognition (CVPR), pp 77–85
9. Chen X, Ma H, Wan J, Li B, Xia T (2017) Multi-view 3D object detection network for autonomous driving. In: 2017 IEEE Conference on computer vision and pattern recognition (CVPR), pp 6526–6534
10. Choy C, Dong W, Koltun V (2020) Deep global registration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2514–2523
11. Choy C, Gwak J, Savarese S (2019) 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), pp 3070–3079
12. Choy C, Park J, Koltun V (2019) Fully convolutional geometric features. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp 8958–8966
13. Chu X, Tian Z, Wang Y, Zhang B, Ren H, Wei X, Xia H, Shen C (2021) Twins: revisiting the design of spatial attention in vision transformers. 1(2):3 arXiv preprint [arXiv:2104.13840](https://arxiv.org/abs/2104.13840)
14. Dai A, Chang AX, Savva M, Halber M, Funkhouser T, Nießner M (2017) ScanNet: richly-annotated 3D reconstructions of indoor scenes. In: IEEE Conf. on comput. vision and pattern recognition, pp 2432–2443
15. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al (2020) An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint [arXiv:2010.11929](https://arxiv.org/abs/2010.11929)
16. Erdős G, Nakano T, Váncza J (2014) Adapting CAD models of complex engineering objects to measured point cloud data. *CIRP Ann* 63(1):157–160
17. Ester M, Kriegl H-P, Sander J, Xu X (1996) Density-based spatial clustering of applications with noise. In: *Int. Conf. knowledge discovery and data mining*, vol 240, p 6
18. Fayolle P-A, Pasko A (2015) User-assisted reverse modeling with evolutionary algorithms. In: IEEE Congress on evolutionary computation, pp 2176–2183. <https://doi.org/10.1109/CEC.2015.7257153>
19. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 24(6):381–395
20. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? The kitti vision benchmark suite. In: 2012 IEEE Conference on computer vision and pattern recognition, pp 3354–3361
21. Gelfand N, Mitra NJ, Guibas LJ, Pottmann H (2005) Robust global registration. In: *Symposium on geometry processing*, vol 2, no 3, p 5
22. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, pp 315–323
23. Graham B, Engelcke M, Maaten LVD (2018) 3D semantic segmentation with submanifold sparse convolutional networks. In: 2018 IEEE/CVF Conference on computer vision and pattern recognition, pp 9224–9232
24. Graham B, Engelcke M, Van Der Maaten L (2018) 3d semantic segmentation with submanifold sparse convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 9224–9232
25. Graham B, van der Maaten L (2017) Submanifold sparse convolutional networks. arXiv preprint [arxiv:1706.01307](https://arxiv.org/abs/1706.01307)
26. Guo R, Zou C, Hoiem D (2015) Predicting complete 3d models of indoor scenes. [arxiv:1504.02437](https://arxiv.org/abs/1504.02437)
27. Gupta S, Arbeláez P, Girshick R, Malik J (2015) Aligning 3d models to rgb-d images of cluttered scenes. In: 2015 IEEE Conference on computer vision and pattern recognition (CVPR), pp 4731–4740
28. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 770–778
29. Hu Q et al (2021) Learning semantic segmentation of large-scale point clouds with random sampling. In: *IEEE transactions on pattern analysis and machine intelligence*. <https://doi.org/10.1109/TPAMI.2021.3083288>
30. Ishimtsev V, Bokhovkin A, Artemov A, Ignatyev S, Niessner M, Zorin D, Burnaev E (2020) Cad-deform: deformable fitting of cad models to 3d scans. arXiv preprint [arXiv:2007.11965](https://arxiv.org/abs/2007.11965)
31. Kang Z, Li Z (2015) Primitive fitting based on the efficient multi-baysac algorithm. *PLoS One* 10(3):e0117341
32. Katz S, Tal A (2015) On the visibility of point clouds. In: 2015 IEEE International Conference on computer vision (ICCV), pp 1350–1358
33. Khan S, Naseer M, Hayat M, Zamir SW, Khan FS, Shah M (2021) Transformers in vision: A survey. arXiv preprint [arXiv:2101.01169](https://arxiv.org/abs/2101.01169)
34. Kim H, Yeo C, Lee ID, Mun D (2020) Deep-learning-based retrieval of piping component catalogs for plant 3d cad model reconstruction. *Comput Ind* 123:103320
35. Kundu A, Yin X, Fathi A, Ross D, Brewington B, Funkhouser T, Pantofaru C (2020) Virtual multi-view fusion for 3d semantic segmentation. In: *European Conference on computer vision*. Springer, pp 518–535
36. Li D, Shen X, Yu Y, Guan H, Wang H, Li D (2020) Ggm-net: graph geometric moments convolution neural network for point cloud shape classification. *IEEE Access* 8:124989–124998
37. Li Y, Wu X, Chrysanthou Y, Sharf A, Cohen-Or D, Mitra NJ (2011) Globfit: consistently fitting primitives by discovering global relations. *ACM Trans Graph* 30(4):52:1–52:12
38. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021) Swin transformer: hierarchical vision transformer using shifted windows. arXiv preprint [arXiv:2103.14030](https://arxiv.org/abs/2103.14030)
39. Lu Y (2017) Industry 4.0: a survey on technologies, applications and open research issues. *J Ind Inf Integr* 6:1–10
40. Mo K, Zhu S, Chang AX, Yi L, Tripathi S, Guibas LJ, Su H (2019) PartNet: a large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: 2019 IEEE/CVF Conference on computer vision and pattern recognition (CVPR), pp 909–918
41. Montlahuc J, Shah GA, Polette A, Pernot J-P (2019) As-scanned point clouds generation for virtual reverse engineering of CAD assembly models. *Comput-Aided Des Appl* 16(6):1171–1182
42. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: an imperative style, high-performance deep learning library. *Adv Neural Inf Process Syst* 32:8026–8037
43. Qi CR, Su H, Nießner M, Dai A, Yan M, Guibas LJ (2016) Volumetric and multi-view cnns for object classification on 3d data. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 5648–5656
44. Qi CR, Yi L, Su H, Guibas LJ (2017) PointNet++: deep hierarchical feature learning on point sets in a metric space. In: *Proc. of the 31st Int. Conf. on neural information processing systems, NIPS'17*, pp 5105–5114
45. Robertson C, Fisher RB, Werghi N, Ashbrook AP (2000) Fitting of constrained feature models to poor 3D data. In: Parmee IC (ed) *Evolutionary design and manufacture*. Springer, London, pp 149–160

46. Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Cham, 2015. Springer International Publishing, pp 234–241
47. Saxena A, Prasad M, Gupta A, Bharill N, Patel OP, Tiwari A, Er MJ, Ding W, Lin C-T (2017) A review of clustering techniques and developments. *Neurocomputing* 267:664–681
48. Schnabel R, Wahl R, Klein R (2007) Efficient ransac for point-cloud shape detection. *Comput Graphics Forum* 26(2):214–226
49. Sener O, Koltun V (2018) Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, vol 31, pp 525–536
50. Shah GA, Polette A, Pernot JP, Giannini F, Monti M (2021) Simulated annealing-based fitting of CAD models to point clouds of mechanical parts' assemblies. *Eng Comput* 37(4):2891–2909
51. Shi W, Rajkumar R (2020) Point-gnn: graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 1711–1719
52. Su H, Maji S, Kalogerakis E, Learned-Miller E (2015) Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE International Conference on computer vision, pp 945–953
53. Tang H, Liu Z, Zhao S, Lin Y, Lin J, Wang H, Han S (2020) Searching efficient 3D architectures with sparse point-voxel convolution. In: European Conference on computer vision (ECCV), pp 685–702
54. Thomas H, Qi CR, Deschaud J, Marcotegui B, Goulette F, Guibas L (2019) KPConv: flexible and deformable convolution for point clouds. In: IEEE Int. Conf. on computer vision (ICCV), pp 6410–6419
55. Ulyanov D, Vedaldi A, Lempitsky V (2016) Instance normalization: the missing ingredient for fast stylization. *arXiv preprint [arXiv:1607.08022](https://arxiv.org/abs/1607.08022)*
56. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: *Advances in neural information processing systems*, vol 30, pp 5998–6008
57. Wang C, Samari B, Siddiqi K (2018) Local spectral graph convolution for point set feature learning. In: Proceedings of the European Conference on computer vision (ECCV), pp 52–66
58. Wang L, Huang Y, Hou Y, Zhang S, Shan J (2019) Graph attention convolution for point cloud semantic segmentation. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 10296–10305
59. Wang S, Suo S, Ma W-C, Pokrovsky A, Urtasun R (2018) Deep parametric continuous convolutional neural networks. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 2589–2597
60. Willis KD, Pu Y, Luo J, Chu H, Du T, Lambourne JG, Solar-Lezama A, Matusik W (2021) Fusion 360 gallery: a dataset and environment for programmatic cad construction from human design sequences. *ACM Trans Graph (TOG)* 40(4):1–24
61. Wu S, Wu T, Lin F, Tian S, Guo G (2021) Fully transformer networks for semantic image segmentation. *arXiv preprint [arXiv:2106.04108](https://arxiv.org/abs/2106.04108)*
62. Wu W, Qi Z, Fuxin L (2019) Pointconv: deep convolutional networks on 3d point clouds. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 9621–9630
63. Xie Y, Tian J, Zhu XX (2020) Linking points with labels in 3d: a review of point cloud semantic segmentation. *IEEE Geosci Remote Sens Mag* 8(4):38–59
64. Xu Y, Fan T, Xu M, Zeng L, Qiao Y (2018) Spidernn: deep learning on point sets with parameterized convolutional filters. In: Proceedings of the European conference on computer vision (ECCV), pp 87–102
65. Xu Y, Fan T, Xu M, Zeng L, Qiao Y (2018) Spidernn: deep learning on point sets with parameterized convolutional filters. In: Proceedings of the European conference on computer vision (ECCV), pp 87–102
66. Yi L, Kim VG, Ceylan D, Shen IC, Yan M, Su H et al (2016) A scalable active framework for region annotation in 3d shape collections. *ACM Trans Graph (ToG)* 35(6):1–12
67. Zhao H, Jiang L, Fu C-W, Jia J (2019) Pointweb: enhancing local neighborhood features for point cloud processing. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 5565–5573
68. Zhu B, Jiang Z, Zhou X, Li Z, Yu G (2019) Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint [arXiv:1908.09492](https://arxiv.org/abs/1908.09492)*

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.