

Transformers in 3D Point Clouds: A Survey

Dening Lu, Qian Xie, Mingqiang Wei, *Senior Member, IEEE*, Kyle (Yilin) Gao, *Student Member, IEEE*, Linlin Xu, *Member, IEEE*, and Jonathan Li, *Senior Member, IEEE*

Abstract—Transformers have been at the heart of the Natural Language Processing (NLP) and Computer Vision (CV) revolutions. The significant success in NLP and CV inspired exploring the use of Transformers in point cloud processing. However, how do Transformers cope with the irregularity and unordered nature of point clouds? How suitable are Transformers for different 3D representations (e.g., point- or voxel-based)? How competent are Transformers for various 3D processing tasks? As of now, there is still no systematic survey of the research on these issues. For the first time, we provided a comprehensive overview of increasingly popular Transformers for 3D point cloud analysis. We start by introducing the theory of the Transformer architecture and reviewing its applications in 2D/3D fields. Then, we present three different taxonomies (i.e., implementation-, data representation-, and task-based), which can classify current Transformer-based methods from multiple perspectives. Furthermore, we present the results of an investigation of the variants and improvements of the self-attention mechanism in 3D. To demonstrate the superiority of Transformers in point cloud analysis, we present comprehensive comparisons of various Transformer-based methods for classification, segmentation, and object detection. Finally, we suggest three potential research directions, providing benefit references for the development of 3D Transformers.

Index Terms—Transformer, point cloud analysis, self-attention mechanism, deep neural networks, 3D vision.

1 INTRODUCTION

TRANSFORMERS, in encoder and/or decoder configurations, are now the dominant neural architecture in NLP. In view of the impressive ability to model long-range dependencies, they have been successfully adapted to the field of CV [1]–[3] for autonomous driving, visual computing, intelligent monitoring, and industrial inspection. A standard Transformer encoder generally consists of six main components (Fig. 1): 1) input (word) embedding; 2) positional encoding; 3) self-attention mechanism; 4) normalization; 5) feed-forward operation; and 6) skip connection. As for the Transformer decoder, it is typically designed to mirror the Transformer encoder, except it additionally takes as input latent features from the Transformer-encoder. However, for 3D point cloud applications, decoders can be specifically designed (i.e. not be a pure Transformer) for dense prediction tasks such as part segmentation and semantic segmentation in 3D point cloud analysis. Researchers in 3D computer vision often adopt PointNet++ [4] or convolutional backbones with Transformer blocks incorporated therein.

To describe in more detail, let $P = \{p_1, p_2, p_3, \dots, p_N\} \in R^{N \times D}$ be an input point cloud. D is the feature dimension of the input point. Typically in literature, “ D equals to three” means only the 3D coordinate of each point is taken as

input, while “ D equals to six” means both the 3D coordinate and normal vector are taken as input. The details of the aforementioned encoder components are as follows.

Firstly, for the input embedding, P is projected to a high-dimension feature space which can facilitate subsequent learning. This can be achieved by using a Multi-Layer Perception (MLP) or other feature extraction backbone networks like PointNet [5]. We denote the embedded feature map as $X \in R^{N \times C}$. Secondly, the positional encoding is used to either capture the geometrical information, or the relative ordering of input tokens/points if relevant. Note that the Transformer is order-agnostic without this step, which is not an issue for point clouds, since they are naturally unordered. Nonetheless, the frequency-based positional encoding can be used by mapping spatial coordinates with sine and cosine functions [6]. Moreover, there also exist learned position encoding schemes with a trainable parameter matrix B [7], [8], which are more adaptive to different input data. These positional encodings of spatial coordinates have shown to benefit the learning of features at finer scales [9]. Thirdly, the core component of the Transformer encoder is the self-attention mechanism. If a sine/cosine-based positional encoding is used, it is typically added to the embedded feature map X . This feature map is then projected to three different feature spaces using three learnable weight matrices $W_Q \in R^{C \times C_Q}$, $W_K \in R^{C \times C_K}$, $W_V \in R^{C \times C}$, where typically C_K equals to C_Q . In this way, *Query*, *Key*, and *Value* matrices can be formulated as:

$$\begin{cases} \text{Query} &= XW_Q, \\ \text{Key} &= XW_K, \\ \text{Value} &= XW_V. \end{cases} \quad (1)$$

Given the *Query*, *Key*, and *Value* matrices, an attention map is formulated as:

$$\text{Attentionmap} = \text{Softmax}\left(\frac{QK^T}{\sqrt{C_K}}\right), \quad (2)$$

- Dening Lu and Qian Xie contribute equally to this work and should be considered co-first authors.
- Corresponding authors: Linlin Xu; Jonathan Li.
- Dening Lu, Kyle Gao, Linlin Xu, and Jonathan Li are with the Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (e-mail: d62lu, y56gao, l44xu, junli@uwaterloo.ca).
- Mingqiang Wei is with the Shenzhen Research Institute, Nanjing University of Aeronautics and Astronautics, Shenzhen 518038, China (e-mail: mingqiang.wei@gmail.com).
- Jonathan Li is also with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.
- Qian Xie is with the Department of Computer Science, University of Oxford, Oxford OX1 3QD, U.K. (e-mail: qian.xie@cs.ox.ac.uk).

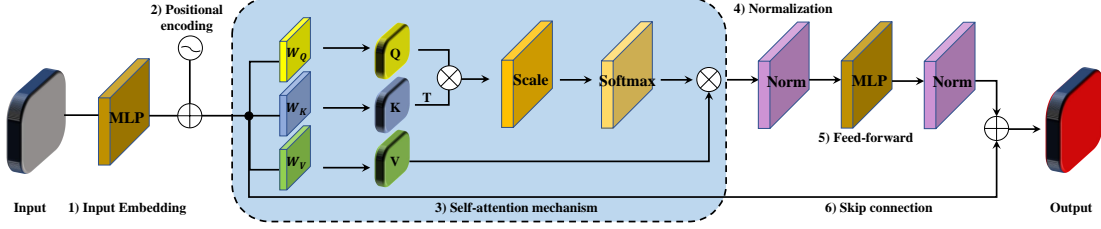


Fig. 1. Illustration of the Transformer encoder architecture.

where Q, K, V denote the *Query*, *Key*, and *Value* matrices respectively. The attention map of size of $N \times N$ measures the similarity of any two input points. It is also called the similarity matrix. Then the attention map and the matrix *Value* are multiplied to generate the new feature map F , of the same size as X . Each feature vector in F is obtained by computing a weighted sum of all input features. It is therefore able to establish connections with all input features. When inputs are global, this process allows for the Transformer to easily learn global features. Therefore, compared with convolutional neural networks (CNNs), Transformers are better at long-range dependency modeling. Additionally, compared with MLPs, Transformers also have two significant advantages. One is that the attention map in the Transformer is dynamic depending on the input during the inference, which is more adaptive than MLPs with fixed weight matrices. Another is that the self-attention mechanism is permutation-equivariant, while for MLPs, the order of input and output is encoded in the weight matrix. Fourthly, a normalization layer is placed before and after the feed-forward layer, performing standardization and normalization on feature maps. There are two kinds of normalization methods used in this layer: LayerNormalization and BatchNormalization. The former is commonly used in NLP, while the latter is commonly used in CV like 2D or 3D data processing. Fifthly, a feed-forward layer is added to enhance the representation of attention features. Generally, it consists of two fully-connection layers with a RELU function. Finally, a skip connection is used between the input and output of the self-attention module. There have been many self-attention variants using various skip connection forms [10]–[12], which we present in more details in Sec. 5.

Note that there are also some 3D Transformers that are not exactly comprised of these six components. For example, early 3D Transformer networks like Point Attention (P-A) [10] and Attentional ShapeContextNet [11] did not have the positional encoding module. They focused on applying the self-attention mechanism to 3D point clouds. Point Cloud Transformer (PCT) [12] proposed a neighbor embedding mechanism achieved by EdgeConv [13]. This mechanism incorporates the positional encoding into the input embedding module. Since the self-attention mechanism is the core component of Transformers, we also classify methods mainly utilizing the self-attention mechanism for point cloud processing into the 3D Transformer family for this survey.

Recently, Transformer models have been introduced to image processing widely, and achieved impressive results

for various tasks, such as image segmentation [14], object detection [15] and tracking [16]. Vision Transformer (ViT) [17] first proposed a pure Transformer network for image classification. It achieved excellent performance compared with the state-of-the-art convolutional networks. Based on ViT, there were numerous Transformer variants proposed for image classification [18]–[21], segmentation [22]–[24], object detection [15], [19], [25], [26], and other vision tasks. Moreover, various innovations on Transformer-based architectures were proposed, such as convolutions+Transformers [20], multi-scale Transformers [21], and self-supervised Transformers [27]. There also have been several surveys [1], [28]–[30] proposed to categorize all involved 2D Transformers into multiple groups. The taxonomies they used were algorithm architecture-based taxonomy and task-based taxonomy.

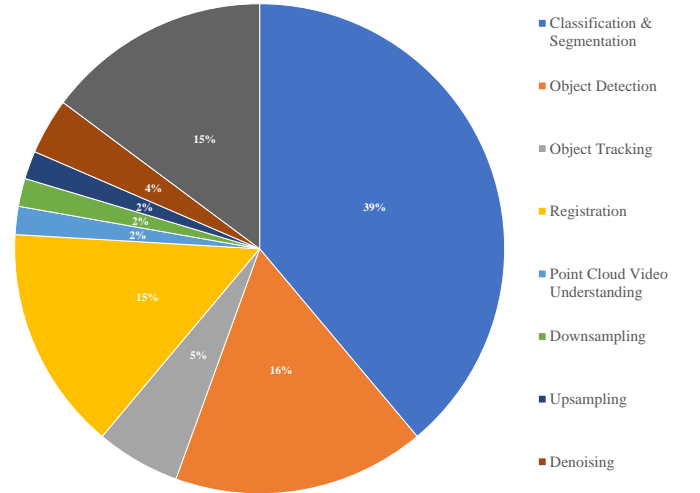


Fig. 2. Applications of Transformers in 3D point clouds.

Due to the remarkable global feature learning ability and permutation-equivariant operations, Transformer architectures are intrinsically suited for point cloud processing and analysis. A number of 3D Transformer backbones were proposed (see Fig. 2) for point cloud classification & segmentation [7], [12], [34], [37], [69], [70], detection [35], [54], tracking [56]–[58], registration [59]–[63], [71], [72], completion [50], [66]–[68], [73], [74], to name a few. Moreover, 3D Transformer networks have also been used for various practical applications, such as structure monitoring [75], medical data analysis [37], and autonomous driving [76], [77]. There-

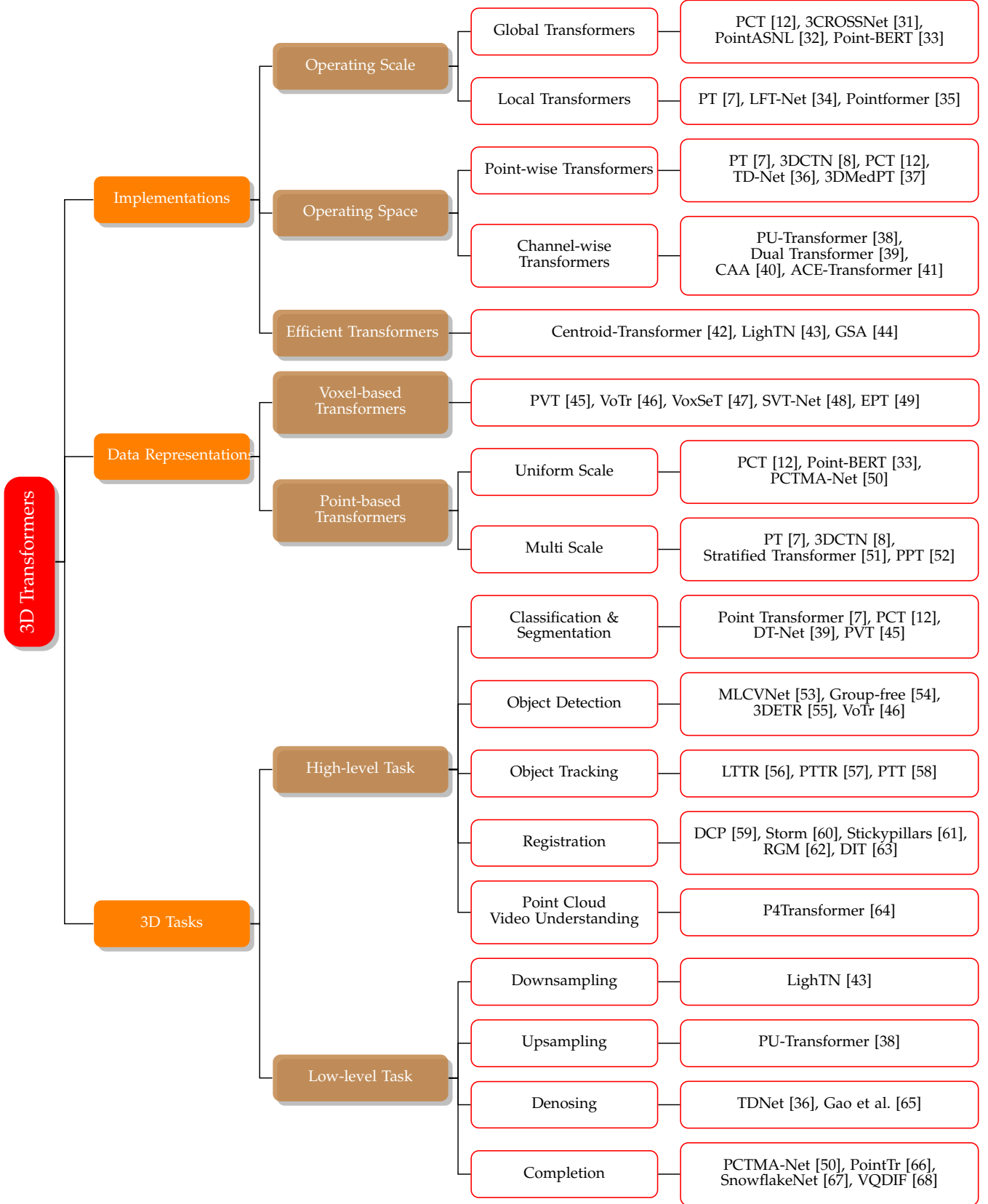


Fig. 3. Taxonomies of 3D Transformers.

fore, it is necessary to conduct a systematic survey for 3D Transformers. Recently, several 3D Transformer/Attention-related reviews have been published. For instance, Khan et al. [29] reviewed the vision Transformers according to the architecture- and task-based taxonomies. However, it mainly focused on Transformers on 2D image analysis, and only provided a brief introduction to 3D Transformer networks. Qiu et al. [78] introduced several variants of the 3D self-attention mechanism, and conducted a detailed comparison and analysis for them on SUN RGBD [79] and ScanNetV2 datasets [80]. However, a comprehensive survey of Transformer models in 3D point clouds has not been conducted so far, which we hope to provide with this paper.

We designed three different taxonomies which are shown in Fig. 3: 1) implementation-based taxonomy; 2) data representation-based taxonomy; 3) task-based taxonomy. In this way, we were able to classify and analyze Transformer networks from multiple perspectives. We note that these taxonomies are not mutually exclusive. Taking Point Transformer (PT) [7] as an example: 1) in terms of the Transformer implementation, it belongs to the local Transformer category, operated in the local neighborhood of the target point cloud; 2) in terms of the data representation, it belongs to the multi-scale point-based Transformer category, extracting the geometrical and semantic features hierarchically; 3) in terms of the 3D task, it is designed for point cloud classification and segmentation. Additionally, we also conducted an investigation of different self-attention variants in 3D point cloud processing. We expect this classification to provide helpful references for the development of Transformer-based networks.

The major contributions of this survey can be summarised as follows:

- This is the first effort, to the best of our knowledge, that focused on comprehensively covering Transformers in point clouds for 3D vision tasks.
- This work investigates a series of self-attention variants in point cloud analysis. It introduced novel self-attention mechanisms aiming to improve the performance and efficiency of 3D Transformers.
- This work provides comparisons and analyses of Transformer-based methods on several 3D vision tasks, including 3D shape classification and 3D shape/semantic segmentation, and 3D object detection on several public benchmarks.
- This work introduces the readers to the SOTA methods as well as to the recent progress of Transformers-based methods for point cloud processing.

The core of this paper is organized into six sections not counting the Introduction. Sec. 2, 3, and 4 introduce the three different taxonomies for 3D Transformer classification. Sec. 5 reviews different self-attention variants proposed in literature to improve the performance of Transformers. Sec. 6 provides a comparison and analysis of surveyed 3D Transformer networks. Lastly, Sec. 7 summarizes our survey work, and points out three potential future directions for 3D Transformers.

2 TRANSFORMER IMPLEMENTATION

In this section, we broadly categorize Transformers in 3D point clouds from multiple perspectives. Firstly, in terms of the operating scale, 3D Transformers can be divided into two parts: Global Transformers and Local Transformers (Sec. 2.1). The operating scale represents the scope of the algorithm with respect to the point cloud, such as the global domain or the local domain. Secondly, in terms of the operating space, 3D Transformers can be divided into Point-wise Transformers and Channel-wise Transformers (Sec. 2.2). The operating scale represents the dimension in which the algorithm is operated, such as the spatial dimension or the channel dimension. Lastly, we provide a review of efficient Transformer networks designed for computational footprint reduction (Sec.2.3).

2.1 Operating Scale

According to the operating scale, 3D Transformers can be divided into two parts: Global Transformers and Local Transformers. The former denotes that Transformer blocks are applied to all the input points for global feature extraction, while the latter denotes that Transformer blocks are applied in a local patch for local feature extraction.

2.1.1 Global Transformers

There are many existing works [8], [10]–[12], [31], [33], [37], [38], [52], [81] focusing on global Transformer. For a global Transformer block, each new output feature in F can establish connections with all input features X . It is both equivariant with respect to permutations of the input and capable of learning the global context features [12].

Following PointNet [5], PCT, as a pure global Transformer network, was proposed in [12]. Taking the 3D coordinates as input P , PCT first proposed a neighbor-embedding architecture to map the point cloud into a high-dimensional feature space. This operation can also incorporate local information into the embedded features. Then these features were fed into four stacked global Transformer blocks to learn semantic information. The global features were finally extracted by a global Max and Average (MA) pooling for classification and segmentation. Moreover, PCT’s improved self-attention module, named Offset-Attention, was inspired by the Laplacian matrix in Graph convolution networks [82]. We detailed the structure of the Offset-Attention module in Sec. 5.1. It is able to sharpen the attention weights and reduce the influence of noise. The state-of-the-art performance of PCT on various tasks proved that Transformers are suitable for 3D point cloud processing.

In contrast to the single scale of PCT, a Cross-Level Cross-Scale Cross-Attention Transformer network was proposed in [31], named 3CROSSNet. Firstly, it performed Farthest Point Sampling (FPS) algorithm [4] on the raw input point cloud to obtain three point subsets with different resolutions. Secondly, it utilized stacked multiple shared Multi-Layer Perception (MLP) modules to extract local features for each sampling point. Thirdly, it applied Transformer blocks to each point subset for global feature extraction. Finally, the Cross-Level Cross-Attention (CLCA) module and Cross-Scale Cross-Attention (CSCA) module were proposed to

build connections between different-resolution point subsets and different-level features for long-range inter- and intra-level dependencies modeling.

A BERT-style pre-training strategy for 3D global Transformers was proposed in [33], which adapted BERT [83] to 3D point cloud processing. Taking the local patches as input, it first utilized the mini-PointNet [5] for the input embedding, following ViT [17]. Then it used a point cloud Tokenizer with a discrete Variational AutoEncoder (dVAE) [84], to convert the embedded points into discrete point tokens for pre-training. The Tokenizer network was adapted from DGCNN [13] which produced meaningful local information aggregating, and was learned through dVAE-based point cloud reconstruction. During pre-training, the point embeddings with some masked tokens were fed into the Transformer encoder. Supervised by the point tokens generated by the Tokenizer, the encoder can be trained to recover the corresponding tokens of the masked locations. The authors have conducted comprehensive experiments to show that the BERT-style pre-training strategy is able to improve the performance of the pure Transformer in point cloud classification and segmentation.

2.1.2 Local Transformers

In contrast to global Transformers, local Transformers [7], [34], [35], [42], [85], [86] aim to achieve feature aggregation in the local patch instead of the entire point cloud.

PT [7] adopted the PointNet++ [4] hierarchical architecture for point cloud classification and segmentation. It focused on local patch processing, and replaced the shared MLP modules in PointNet++ with local Transformer blocks. PT had five local Transformer blocks operated on progressively downsampled point sets. Each block was applied on K-Nearest Neighbor (KNN) neighborhoods of sample points. Specifically, the self-attention operator that PT used was the vector attention [87] instead of the scalar attention. The former has been proven to be more effective for point cloud processing, since it supports channel-wise attention weight assignment, as opposed to assigning one single weight to a whole feature vector. Please refer to Sec. 7 an overview of vector attention.

Pointformer was proposed in [35] to combine the local and global features both extracted by Transformer blocks for 3D object detection. It had three kinds of main blocks: a Local Transformer (LT) block, a Global Transformer (GT) block and a Local-Global Transformer (LGT) block. Firstly, the LT block applied the dense self-attention operation in the neighborhood of each centroid point generated by FPS [4]. Secondly, taking the whole point cloud as input, the GT block aimed to learn global context-aware features via the self-attention mechanism. Lastly, the LGT block adopted a multi-scale cross-attention module, to build connections between local features from the LT and global features from the GT. Specifically, the LGT block took the output of the LT as *query*, and the output of the GT as *key* and *value* to conduct the self-attention operation. As such, all centroid points could be utilized to integrate global information, which led to effective global feature learning.

Inspired by Swin Transformer [21], Stratified Transformer [51] was proposed for 3D point cloud segmentation. It split the point cloud into a group of non-overlapping cubic

windows via 3D voxelization and performed the local Transformer operation in each window. Stratified Transformer was an encoder-decoder architecture. Its encoder was a hierarchical structure consisting of multiple stages, where each stage had two successive Transformer blocks. The former block utilized a Stratified Self-Attention (SSA) to capture the long- and short-range dependencies. The latter block utilized a Shifted SSA to further strengthen the connections between different independent windows, following Swin Transformer [21]. Specifically, to solve the issue that the local Transformer is weak in capturing global information, SSA generated the dense local *key* points and sparse distant *key* points for each *query* point. The former was generated in the window that the *query* point belonged to, while the latter was generated in a larger window by downsampling the entire input point cloud. With this, the receptive field of the *query* point was not limited in the local window, allowing SSA to capture global information. Additionally, Stratified Transformer performed a KPConv [88] embedding in the first stage to extract the local geometric information of the input point cloud. This operation was proven to be effective by their ablation experiments.

2.2 Operating Space

According to the operating space, 3D Transformers can be divided into two categories: Point-wise Transformers and Channel-wise Transformers. The former measures the similarity among input points, while the latter distributes attention weights along channels [40]. Generally, according to Eq. 2, the attention maps of these two kinds of Transformers can be expressed as:

$$\begin{aligned} \text{Point-wise Attn} &= \text{Softmax}\left(\frac{QK^T}{\sqrt{C_K}}\right), \\ \text{Channel-wise Attn} &= \text{Softmax}\left(\frac{Q^T K}{\sqrt{C_K}}\right), \end{aligned} \quad (3)$$

where the size of *Point-wise Attn* is $N \times N$, while the size of *Channel-wise Attn* is $C_K \times C_K$.

2.2.1 Point-wise Transformers

Point-wise Transformers aim to investigate the spatial correlation among points and formulate the output feature map as a weighted sum of all input features.

Since global Transformers and local Transformers in Sec. 2.1 are distinguished by the spatial operating scale, i.e., the whole point cloud or a local patch, all aforementioned methods [7], [8], [10]–[12], [31], [33]–[35], [37], [38], [42], [51], [52], [81] in Sec. 2.1 can be considered as point-wise Transformers.

Point-wise Transformers are also widely applied to other tasks. Xu et al. [36] proposed an encoder-decoder Transformer network (TD-Net) for point cloud denoising. The encoder consisted of a coordinate-based input embedding module, an adaptive sampling module and four stacked point-wise self-attention modules. The outputs of the four self-attention modules were concatenated together as input of the decoder. Additionally, TD-Net used the adaptive sampling approach which can automatically learn the offset of each sampling point generated by FPS [4]. This operation

allows the sampling points closer to the underlying surface. The decoder was applied to construct the underlying manifold according to the extracted high-level features. And finally, a clean point cloud can be reconstructed by manifold sampling.

3D Medical Point Transformer (3DMedPT) [37] was proposed for medical point cloud analysis. Specifically, it included a hierarchical point-wise Transformer for classification and a uniform-scale point-wise Transformer for segmentation. 3DMedPT introduced the convolution operation to the point-wise Transformer block. It added a local feature extraction module achieved using DGCNN [13] before each Transformer block. Point Attention Network (P-A) [10] and Pyramid Point Cloud Transformer (PPT) [52] also had similar structures. Considering insufficient training sample processing in the medical domain, 3DMedPT proposed a special module named Multi-Graph Reasoning (MGR), to enrich the feature representations.

2.2.2 Channel-wise Transformers

In contrast to point-wise Transformers, the channel-wise Transformers [38]–[41], [69] focus on measuring the similarity of different feature channels. They are able to improve the context information modeling by highlighting the role of interaction across different channels [39].

Qiu et al. [40] proposed a back-projection module for local feature capturing, leveraging an idea of error-correcting feedback structure. They designed a Channel-wise Affinity Attention (CAA) module for better feature representations. Specifically, the CAA module consisted of two blocks: a Compact Channel-wise Comparator (CCC) block and a Channel Affinity Estimator (CAE) block. The CCC block could generate the similarity matrix in the channel space. The CAE block further calculated an affinity matrix, in which an element with a higher attention value represented a lower similarity of the corresponding two channels. This operation can sharpen the attention weights and avoid aggregating similar/redundant information. As such, each channel of the output feature had sufficient intersection with other distinct ones, which has been proven to be beneficial to the final results. We also detail the CAA structure in Sec. 5.

Instead of only using the feature channels, Transformer-Conv proposed in [41] combined both coordinate and feature channels to design a novel channel-wise Transformer. Specifically, the *Query* matrix was generated directly by the coordinate information without any linear transformation, while the *Key* matrix was generated by feature channels with MLP. Then the attention matrix was calculated by element-wise multiplication rather than dot product. As such, the attention matrix was able to represent the relationship between the coordinate channels and feature channels of each point. Since all features came from the coordinate space, an element with a higher value in the attention matrix tended to indicate that the corresponding feature channel was more faithful to the coordinate space. After that, the *Value* matrix is created by projecting the *Key* matrix into a latent space using an MLP. Then the new feature map, called the response matrix, can be obtained by element-wise multiplication between the *Value* matrix and attention matrix. The response matrix consisted of the weighted feature

channels of all input points. Finally, the output features were generated by applying a channel max-pooling operation to the response matrix. The max-pooling played a screening role and it could select the most important channels, i.e., the channels which suited the coordinate space best. This process was proven effective for point cloud analysis by the authors’ ablation experiments.

2.3 Efficient Transformers

Despite achieving great success in point cloud processing, standard Transformers tend to incur high computational footprint and memory consumption because of massive linear operations. Given N input points, the computation and memory complexities of the standard self-attention module are quadratic on N , i.e., $O(N^2)$. This is the key drawback when applying Transformers on large-scale point cloud datasets.

Recently, there were several 3D Transformers researching on improving the self-attention module for higher computational efficiency. For instance, Centroid Transformer [42] took N point features as input while outputting a smaller number M of point features. As such, the key information in the input point cloud can be summarized by a smaller number of outputs (called centroids). Specifically, it first constructed M centroids from N input points, by optimizing a general “soft K-means” objective function. Then it used the M centroids and N input points to generate the *Query* and *Key* matrices respectively. The size of the attention map was reduced from $N \times N$ to $M \times N$, so the computational cost of the self-attention was reduced from $O(N^2)$ to $O(NM)$. To further save the computational cost, the authors applied a KNN approximation. This operation essentially converted the global Transformer to a local Transformer. In this case, the similarity matrix was generated by measuring the relationships among each *query* feature vector and its K neighbor *key* vectors, instead of N vectors. So the computational cost can be further reduced to $O(NK)$. Similarly, PatchFormer [89] was also proposed to reduce the size of the attention map. It first split the raw point cloud into M patches, followed by aggregating the local feature in each patch. The significant difference between the two aforementioned models is that PatchFormer used M aggregated local features to generate *Key* matrix, while Centroid Transformer used M centroids to generate *Query* matrix. As such, the computational cost of the self-attention in PatchFormer can also be reduced to $O(NM)$.

Light-weight Transformer Network (LighTN) [43] was proposed to reduce the computational cost in a different way. LighTN aimed to simplify the main components in the standard Transformer, maintaining the superior performance of Transformers while increasing efficiency. Firstly, it removed the positional encoding block because the input 3D coordinates already contain positional information and can be considered as a substitute for the positional encoding. This eliminated the overhead of positional encoding itself. Secondly, it utilized a small-size shared linear layer as the input embedding layer. The dimensions of embedded features were reduced by half compared to the computationally-saving neighbor embedding setting in [12]. With this, the computational costs of the input embedding can be reduced. Thirdly, it presented a single head self-correlation

layer as the self-attention module. The projection matrices of W_Q , W_K , and W_V were removed, to reduce learnable parameters for high efficiency. Since the attention map was generated only by the input self-correlation parameters, the self-attention module was also named the self-correlation module, which can be formulated as:

$$\begin{aligned} SA(X) &= FC_{out}(C(X)), \\ C(X) &= softmax(\frac{XX^T}{\sqrt{C}})X, \end{aligned} \quad (4)$$

where $SA(*)$ represents the self-attention block, FC_{out} represents the linear transformation, $softmax(*)$ is the activation function, and C is the input feature dimension declared in Eq. 2. Lastly, the authors built three linear layers (a standard FFN block generally has two linear layers) in the Feed-Forward Network (FFN) and used the expand-reduce strategy [90] in the middle layer. So the negative impact caused by the decreasing learnable parameters in the self-correlation layer can be mitigated. Similarly, Group Shuffle Attention (GSA) proposed in [44] also simplified the self-attention mechanism in its Transformer network. It integrated the shared projection weight matrix and non-linearity activation function into the self-attention mechanism (please see Sec. 5.1 for a detailed description of GSA).

3 DATA REPRESENTATION

There are several forms of 3D data representation, such as points and voxels, both of which can be used as the input of 3D Transformers. Since points can be represented by or transformed into voxels, several voxel-based approaches can also be performed on point clouds, so as to 3D Transformers. According to different input formats, we divided 3D Transformers into Voxel-based Transformers and Point-based Transformers.

3.1 Voxel-based Transformers

Unlike images, 3D point clouds are generally unstructured, and cannot be directly processed by traditional convolution operators. However, 3D point clouds can be easily converted into 3D voxels, which are structured like images. Thus, some Transformer-based works [45]–[47], [51] explored transforming 3D point clouds into voxel-based representation. The most general voxelization approach can be described as follows [91]: The bounding box of a point cloud is first regularly divided into 3D cuboids via rasterization. Voxels containing points are retained, generating the voxel representation of point clouds.

Inspired by the efficiency of sparse convolution on voxel data [92], [93], Mao et al. [46] first proposed the Voxel Transformer (VoTr) backbone for 3D object detection. They presented the Submanifold Voxel module and the Sparse Voxel module to extract features from non-empty and empty voxels respectively. In both two modules, the Local Attention and Dilated Attention operations were implemented, on the basis of the Multi-head Self-Attention mechanism (MSA), to maintain low computational consumption for numerous voxels. The proposed VoTr can be integrated into most voxel-based 3D detectors. To tackle the computational issue of Transformers as voxel-based outdoor 3D detectors,

Voxel Set Transformer (VoxSeT) [47] was proposed to detect outdoor objects in a set-to-set fashion. Based on the low-rank characteristic of the self-attention matrix, a Voxel-based Self-Attention (VSA) module was designed by assigning a set of trainable “latent codes” to each voxel, which was inspired by the induced set attention blocks in Set Transformer [94].

Inspired by the effectiveness of voxel-based representation on large-scale point clouds, voxel-based Transformers can also be applied to large-scale point cloud processing. For instance, Fan et al. [48] presented Super light-weight Sparse Voxel Transformer (SVT-Net) for large scale place recognition. They designed an Atom-based Sparse Voxel Transformer (ASVT) and a Cluster-based Sparse Voxel Transformer (CSVT). The former was used to encode short-range local relations, while the latter was used to learn long-range contextual relations. Park et al. [49] proposed Efficient Point Transformer (EPT) for large-scale 3D scene understanding from point clouds. To relieve the problem of geometric information loss during voxelization, they introduced the center-aware voxelization and devoxelization operations. On this basis, Efficient Self-Attention (ESA) layers were employed to extract voxel features. Their center-aware voxelization preserved positional information of points in voxels.

3.2 Point-based Transformers

Since voxels are of regular format and points are not, the transformation to voxels would lead to geometric information loss to some extent [4], [5]. On the other hand, since the point cloud is the original representation, it contains the complete geometric information of the data. Thus, most Transformer-based point cloud processing frameworks fall into the category of point-based Transformer. Their architectures are usually classified in two main groups: uniform-scale architecture [12], [33], [50], [63], [81] and multi-scale architecture [7], [8], [37], [39], [51], [52].

3.2.1 Uniform Scale

Uniform-scale architectures usually keep the scale of the point features constant during data processing. The number of output features of each module is consistent with the number of input features. The most representative work is PCT [12], which was discussed in Sec. 2.1. After the input embedding stage, four global Transformer blocks of PCT were directly stacked together to refine point features. There was no hierarchical feature aggregation operation, which facilitated the decoder design for dense prediction tasks like point cloud segmentation. Feeding all input points into the Transformer block is beneficial to global feature learning. However, uniform-scale Transformers tend to be weak in extracting the local features due to the lack of local neighborhoods. Additionally, processing the whole point cloud directly would lead to a high computation footprint and memory consumption.

3.2.2 Multi Scale

Multi-scale Transformers refer to those with progressive point sampling strategies during feature extraction, also called hierarchical Transformers. PT [7] was the pioneering design that introduced the multi-scale structure to a pure Transformer network. The Transformer layers in PT were

applied to progressively (sub)sampled point sets. On one hand, sampling operations could accelerate the computation of the whole network by reducing the parameters of the Transformer. On the other hand, these hierarchical structures usually came with KNN-based local feature aggregation operations. This local feature aggregation was beneficial to the tasks that need fine semantic perception, such as segmentation and completion. And the highly aggregated local features at the last layer of the network could be taken as the global features, which could be used for point cloud classification. Additionally, there also exist many multi-scale Transformer networks [8], [37], [51], [52] that utilized EdgeConv [13] or KPconv [88] for local feature extraction and utilized Transformers for global feature extraction. With this, they are able to combine the strong local modeling ability of convolutions and the remarkable global feature learning ability of Transformers for better semantic feature representation.

4 3D TASKS

Similar to image processing [29], 3D point cloud-related tasks can also be divided into two main groups: high-level and low-level tasks. High-level tasks involve semantic analysis, which focuses on translating 3D point clouds to information that people can understand. Low-level tasks, such as denoising and completion, focus on exploring fundamental geometric information. They are not directly related to human semantic understanding but can indirectly contribute to high-level tasks.

4.1 High-level Task

In the field of 3D point cloud processing, high-level tasks usually include: classification & segmentation [7], [11], [12], [32]–[34], [37], [39], [40], [42], [44], [45], [51], [85], [86], [89], [95]–[99], object detection [35], [46], [47], [53]–[55], [69], [77], [100]–[102], tracking [56]–[58], registration [59]–[63], [71], [72], [103] and so on. Here, we started by introducing classification & segmentation tasks, which are very common and fundamental research topics in the field of 3D computer vision.

4.1.1 Classification & Segmentation

Similar to image classification [104]–[107], 3D point cloud classification methods aim at classifying the given 3D shapes into specific categories, such as chair, bed and sofa for indoor scenes, and pedestrian, cyclist and car for outdoor scenes. In the field of 3D point cloud processing, since the encoders of segmentation networks are usually developed from classification networks, we introduce these tasks together.

Xie et al. [11], for the first time, introduced the self-attention mechanism into the task of point cloud recognition. Inspired by the success of shape context [108] in shape matching and object recognition, the authors first transformed the input point cloud into a form of shape context representation. This representation was comprised of a set of concentric shell bins. Based on the proposed novel representation, they then introduced the ShapeContextNet

(SCN) to perform point feature extraction. To automatically capture the rich local and global information, a dot-product self-attention module was further applied to the shape context representation, resulting in the Attentional ShapeContextNet (A-SCN).

Inspired by the self-attention networks in image analysis [87], [109] and NLP [83], Zhao et al. [7] designed a vector attention-based Point Transformer layer. A Point Transformer block was constructed on the basis of the Point Transformer layer in a residual fashion. The encoder of PT was constructed with only Point Transformer blocks, pointwise transformations and pooling operation for point cloud classification. Moreover, PT also used a U-Net structure for point cloud segmentation, where the decoder was designed to be symmetrical with the encoder. It presented a Transition Up module to recover the original point cloud with semantic features from the downsampled point set. Such module consisted of a linear layer, batch normalization, ReLU, and trilinear interpolation for feature mapping. Additionally, a skip connection between the encoder block and the corresponding decoder block was introduced to facilitate backpropagation. With these carefully designed modules, PT became the first model that reached over 70% mIoU (70.4%) for semantic segmentation on Area 5 of the S3DIS dataset [110]. As for the task of shape classification on the ModelNet40 dataset, Point Transformer also achieved 93.7% overall accuracy.

As illustrated in Sec. 2.1, Point-BERT [33] was proposed to pre-train pure Transformer-based models with a Mask Point Modeling (MPM) task for point cloud classification. It was inspired by the concept of BERT [83] and masked auto-encoder [111]. Specifically, a point cloud was first divided into several local point patches. Then a mini-PointNet was utilized to get the embedded feature (which can be regarded as tokens) for each patch. Like [33], some tokens were randomly discarded (masked) and the rest were fed to the Transformer network, to recover the masked point tokens. This training procedure was entirely self-supervised. With 8192 points as input, Point-BERT achieved 93.8% overall accuracy on ModelNet40 [112].

Zhang et al. [45] proposed a pure Transformer-based point cloud learning backbone, taking 3D voxels as the input, termed Point-Voxel Transformer (PVT). Inspired by the recent Swin Transformer [21], a Sparse Window Attention (SWA) operation was designed to perform the self-attention within non-overlapping 3D voxel windows in a shifting-window configuration. A relative-attention (RA) operation was also introduced to compute fine-grained features of points. With the two aforementioned modules, PVT could take advantage of both point-based and voxel-based structures with one pure Transformer architecture. Similarly, Lai et al. [51] proposed Stratified Transformer to explicitly encode global contexts. It also extended Swin Transformer [21] to point cloud processing by 3D voxelization. The main difference from PVT is that Stratified Transformer took both dense local points and sparse distant points as the *key* vectors for each *query* vector. This operation was beneficial to message passing among cubic windows and as well as to global information capturing. Both PVT and Stratified Transformer achieved 86.6% pIoU for part segmentation on ShapeNet dataset. However, Stratified Transformer per-

formed better for semantic segmentation, surpassing PVT by 4.7% mIoU on the S3DIS dataset.

4.1.2 Object Detection

Thanks to the popularization of 3D point cloud scanners, 3D object detection is becoming a more and more popular research topic. Similar to the 2D object detection task, 3D object detectors aim to output 3D bounding boxes with point clouds as input data. Recently, Carion et al. [15] introduced the first Transformer-based 2D object detector, DETR. It proposed to combine Transformers and CNNs to eliminate non-maximum suppression (NMS). Since then, Transformer-related works have also shown a flourishing growth in the field of point cloud-based 3D object detection.

On the basis of VoteNet [113], Xie et al. [53], [114], for the first time, introduced the self-attention mechanism of Transformers into the task of 3D object detection in indoor scenes. They proposed the Multi-Level Context VoteNet (MLCVNet) to improve detection performance by encoding contextual information. In their papers, each point patch and vote cluster were regarded as tokens in Transformers. Then the self-attention mechanism was utilized to strengthen the corresponding feature representations via capturing relations within point patches and vote clusters, respectively. Due to the integration of the self-attention modules, MLCVNet achieved better detection results than its baseline model on both ScanNet [80] and SUN RGB-D datasets [79]. PQ-Transformer [100] was proposed to detect 3D objects and predict room layouts simultaneously, which was also based on VoteNet. It utilized a Transformer decoder to enhance proposal features. With the assistance of room layout estimation and refined features by the Transformer decoder, PQ-Transformer attained a mAP@0.25 of 67.2% on ScanNet.

Aforementioned methods employed the hand-crafted grouping scheme, obtaining features for object candidates by learning from points within the corresponding local regions. However, Liu et al. [54] argued that the point grouping operation within limited regions tended to hinder the performance of 3D object detection. Thus, they presented a group-free framework with the aid of the attention mechanism in Transformers. The core idea was that the features of an object candidate should come from all the points in the given scene, instead of a subset of the point cloud. After obtaining object candidates, their method first leveraged a self-attention module to capture contextual information between the object candidates. They then designed a cross-attention module to refine the object features with the information of all the points. With the improved attention stacking scheme, their detector achieved the mAP@0.25 of 69.1% on the ScanNet dataset.

Inspired by DETR [15] in 2D object detection, an end-to-end 3D Detection Transformer network, termed 3DETR [55], was first proposed to formulate 3D object detection as a set-to-set problem. Borrowing ideas from both DETR [15] and VoteNet [113], 3DETR was designed in the general encoder-decoder fashion. In the encoder part, sampled points and the corresponding features extracted by MLP were directly fed into a Transformer block for feature refinement. In the decoder part, these features went through a parallel Transformer-fashion decoder and were turned into a set of

object candidate features. These object candidate features were finally used to predict 3D bounding boxes. 3DETR improved on VoteNet by 9.5% AP_{50} and 4.6% AP_{25} on ScanNetV2 and SUN RGB-D respectively.

Apart from the aforementioned methods focusing on indoor scenes, Sheng et al. [69] proposed a Channel-wise Transformer based two-stage framework (CT3D) to improve 3D object detection performance in outdoor LiDAR point clouds. The input of the channel-wise Transformer came from a Region Proposal Network (RPN). Moreover, the Transformer network consisted of two sub-modules: the proposal-to-point encoding module and the channel-wise decoding module. The encoding module first took the proposals and their corresponding 3D points as input. Then it extracted the refined point features through a self-attention-based block. The channel-wise decoding module transformed the extracted features from the encoder module into a global representation through a channel-wise re-weighting scheme. Finally, Feed-Forward Networks (FFNs) were performed for detection predictions. As such, CT3D achieved 81.77% AP in the moderate car category on the KITTI test set.

In a similar paradigm to DETR [15], a LiDAR and Camera fusion based 3D object detector based on Transformers was proposed in [115], called TransFusion. In TransFusion, the attention mechanism was employed to adaptively fuse features from images. It aimed to relieve the problem of bad association between LiDAR points and image points established by calibration matrices. CAT-Det [101] was also proposed to fuse LiDAR point clouds and RGB images more efficiently for 3D object detection performance boosting. A Pointformer and an Imageformer were first introduced in the branches of the point cloud and image respectively to extract multi-modal features. A Cross-Modal Transformer (CMT) module was then designed to combine the features from the aforementioned two streams. With the performance of 67.05% mAP on the KITTI test split, CAT-Det became the first multi-modal solution that significantly surpassed LiDAR-only ones.

4.1.3 Object Tracking

3D object tracking takes two point clouds (i.e., a template point cloud and a search point cloud) as input. It outputs 3D bounding boxes of the target (template) in the search point cloud. It involves feature extraction of point clouds and feature fusion between template and search point clouds.

Cui et al. [56] argued that most existing tracking approaches did not consider the attention changes of object regions during tracking. According to them, different regions in the search point cloud should contribute different importance to the feature fusion process. Based on this observation, they presented a LiDAR-based 3D Object Tracking with a TRansformer network (LTTR). This method was able to improve the feature fusion of template and search point clouds by capturing attention changes over tracking time. Specifically, they first built a Transformer encoder to improve the feature representation of template and search point clouds separately. Then the cross-attention mechanism was employed to build a Transformer decoder. It could fuse features from the template and search point clouds

by capturing relations between the two point clouds. Benefiting from the Transformer-based feature fusion between the template and search point clouds, LTTR reached 65.8% mean Precision on KITTI tracking dataset. Zhou et al. [57] also proposed a Point Relation Transformer (PRT) module to improve feature fusion in their coarse-to-fine Point Tracking Transformer (PTTR) framework. Similar to LTTR, PRT employed self-attention and cross-attention to encode relations within and between point clouds respectively. The difference is that PRT utilized the Offset-Attention [12] to relieve the impact of noise data. Finally, PTTR surpassed LTTR by 8.4% and 10.4% in terms of average Success and Precision, and became a new SOTA on the KITTI tracking benchmark.

Unlike the two aforementioned approaches which focused on the feature fusion step, Shan et al. [58] introduced a Point-Track-Transformer (PTT) module to enhance the feature representation after the feature fusion step. Features from the fusion step and the corresponding point coordinates were both mapped into the embedding space. A position encoding block was also designed to capture positional features using the KNN algorithm and an MLP layer. With the aforementioned two embedded semantic and positional features as input, a self-attention block was finally applied to obtain more representative features. To verify the effectiveness of the proposed PTT, the authors integrated it into the seeds voting and proposal generation stages of the P2B [116] model resulting in the PTT-Net. PTT-Net improved P2B by 9.0% in terms of Precision on KITTI for the car category.

4.1.4 Registration

Given two point clouds as input, the aim of point cloud registration is to find a transformation matrix to align them.

Deep Closest Point (DCP) model proposed in [59] introduced the Transformer encoder into the task of point cloud registration. As usual, the input unaligned point clouds were first sent to a feature embedding module, such as PointNet [5] and DGCNN [13], to transfer 3D coordinates into a feature space. A standard Transformer encoder was then applied to perform context aggregation between two embedded features. Finally, DCP utilized a differentiable Singular Value Decomposition (SVD) layer to compute the rigid transformation matrix. DCP was the first work that employed the Transformer model to improve the feature extraction of point clouds in registration. With the same paradigm, STORM [60] also deployed Transformer layers to refine the point-wise features extracted by EdgeConv [13] layers, capturing the long-term relationship between point clouds. It achieved better performance than DCP for partial registration on ModelNet40 dataset. Similarly, Fischer et al. [61] leveraged multi-head self- and cross-attention mechanisms to learn contextual information between target and source point clouds. Their method focused on processing outdoor scenes, e.g., the KITTI dataset [117].

To find more robust correspondences between two point clouds, Fu et al. [62] presented the first deep graph matching-based framework (RGM) to perform robust point cloud registration, which was less sensitive to outliers. During the graph establishment, they employed Transformer encoders to obtain the soft edges of two nodes within a

graph. With the generated soft graph edges, better correspondences could be obtained for the overlapping parts when registering partial-to-partial point clouds. The effectiveness of the proposed Transformer-based edge generator was demonstrated by the ablation study where the performance dropped on ModelNet40 when replacing the edge generator with either full connection edges or sparse connection edges.

Recently, Yew et al. [72] argued that explicit feature matching and outlier filtering via RANSAC in point cloud registration can be replaced with attention mechanisms. They designed an end-to-end Transformer framework, termed REGTR, to directly find point cloud correspondences. In REGTR, point features from a KPconv [88] backbone were fed into several multi-head self- and cross-attention layers for comparing source and target point clouds. With the aforementioned simple design, REGTR became the current state-of-the-art point cloud registration method on the ModelNet40 [112] and 3DMatch [118] datasets. Similarly, GeoTransformer [71] also utilized self- and cross-attention to find robust superpoint correspondences. In terms of Registration Recall, both REGTR and GeoTransformer achieved 92.0% on the 3DMatch dataset. However, GeoTransformer surpassed REGTR by 10.2% on the 3DLoMatch [119] dataset.

4.1.5 Point Cloud Video Understanding

The 3D world around us is dynamic and consistent in time, which cannot be fully represented by traditional single-frame and fixed point clouds. In contrast, point cloud videos, a set of point clouds captured in a fixed frame rate, could be a promising data representation of dynamic scenes in the real world. Understanding dynamic scenes and dynamic objects is important for the application of point cloud models to many real-world scenarios. Point cloud video understanding involves processing a time sequence of 3D point clouds. Thus, the Transformer architecture could be a promising choice to process point cloud videos, since they are good at dealing with global long-range interactions.

Based on such observation, P4Transformer [64] was proposed to process point cloud videos for action recognition. To extract the local spatial-temporal features of a point cloud video, the input data were first represented by a set of spatial-temporal local areas. Then a point 4D convolution was used to encode features for each local area. After that, the P4Transformer authors introduced a Transformer encoder to receive and integrate the features of local areas via capturing long-range relationships across the entire video. P4Transformer has been successfully applied to the task of 3D action recognition and 4D semantic segmentation from point clouds. It achieved higher results than PointNet++-based methods on many benchmarks (e.g., the MSR-Action3D [120], the NTU RGB+D 60 [121] and 120 [122] datasets for 3D action recognition, and the Synthia 4D [93] dataset for 4D semantic segmentation). It demonstrated the effectiveness of Transformers on point cloud video understanding.

4.2 Low-level Task

The input data of low-level tasks is usually the raw scanned point cloud with occlusion, noise, and uneven densities.

Thus, the ultimate goal of low-level tasks is to get a high-quality point cloud, which could benefit high-level tasks. Some typical low-level tasks include point cloud downsampling [43], upsampling [38], denoising [36], [65], completion [50], [66]–[68], [73], [74], [123], [124].

4.2.1 Downsampling

Given a point cloud with N points, downsampling methods aim at outputting a smaller size of point cloud with M points, while retaining the geometric information of the input point cloud. Leveraging the powerful learning ability of Transformers, LighTN [43] was proposed to downsample point clouds in a task-oriented manner. As mentioned in Sec. 2.3, it first removed the position encoding, then used a small-size shared linear layer as the embedding layer. Moreover, the MSA module was replaced with a single head self-correlation layer. Experimental results demonstrated the aforementioned strategies significantly reduced the computational cost. 86.18% classification accuracy could still be attained while only 32 points were sampled. Moreover, the lightweight Transformer network was designed as a detachable module, which can be easily inserted into other neural networks.

4.2.2 Upsampling

Contrary to downsampling, upsampling methods aim to restore missing fine-scale geometric information by outputting a point cloud of bigger size than the input point cloud [125]. The upsampled points are expected to reflect realistic geometry and lie on the surfaces of the objects represented by the given sparse point clouds. PU-Transformer [38] was the first work to apply the Transformer-based model to point cloud upsampling. The authors designed two novel blocks for the PU-Transformer. The first block was the Positional Fusion block (PosFus), which aimed at capturing local position-related information. The second one was the Shifted Channel Multi-head Self-Attention (SC-MSA) block. It was designed to address the lack of connection between the outputs of different heads in conventional MSA. See the SC-MSA in Sec. 5 for more details. PU-Transformer showed the promising potential of Transformer-based models in point cloud upsampling.

4.2.3 Denoising

Denoising takes point clouds corrupted by noise as input, and outputs clean point clouds by utilizing the local geometry information. TDNet [36] was first proposed for point cloud denoising. Taking each point as a word token, it improved the NLP Transformer [6] making it suitable for point cloud feature extraction. The Transformer-based encoder mapped the input point cloud into a high-dimensional feature space and learned the semantic relationship among points. With the extracted feature from the encoder, the latent manifold of the noisy input point cloud can be obtained. Finally, a clean point cloud can be generated by sampling each patch manifold.

Another category of point cloud denoising method is to filter out noise points directly from the input point clouds. For instance, some Lidar point clouds could contain a huge number of virtual (noise) points. These points are produced

by the specular reflections of glass or other kinds of reflective materials. To detect these reflective noise points, Gao et al. [65] first projected the input 3D LiDAR point cloud into a 2D range image. Then a Transformer-based auto-encoder network was employed to predict a noise mask to indicate the points coming from reflection.

4.2.4 Completion

In most 3D practical applications, it is usually difficult to obtain complete point clouds of objects or scenes due to occlusion from other objects or self-occlusion. This issue makes point cloud completion an important low-level task in the field of 3D vision.

PoinTr proposed in [66], for the first time, converted point cloud completion to a set-to-set translation task. Specifically, the authors claimed that the input point cloud can be represented by a set of groups of local points, termed “point proxies”. Taking a sequence of point proxies as input, a geometry-aware Transformer block was carefully designed to generate the point proxies of the missing parts. In a coarse-to-fine fashion, FoldingNet [126] was finally employed to produce points based on the predicted point proxies. The geometry-aware Transformer block was a self-contained module, which can capture both the semantic and geometric relationship among points. PoinTr attained 8.38 Average L_1 Chamfer Distance (CD) on the PCN dataset [127].

In contrast with PointTr, Xiang et al. [67] proposed to formulate the task of point cloud completion as the growth of 3D points in a snowflake-like fashion. Based on this insight, SnowflakeNet was presented to focus on recovering fine geometric details, such as corners, sharp edges and smooth regions, of the complete point cloud. The core idea was to combine Snowflake Point Deconvolution (SPD) layer with the skip-Transformer to better guide the point splitting process. SPD could generate multiple points from any single one. Skip-Transformer was capable of capturing both contexts and spatial information from the given point and the generated points. With the skip-Transformer integrated, the SPD layers were capable of modeling structure characteristics, thus producing more compact and structured point clouds. Benefiting from SPD and the skip-Transformer, SnowflakeNet surpassed PoinTr by 1.17 Average L_1 Chamfer Distance (CD) on the PCN dataset [127].

Instead of working directly on the point cloud, ShapeFormer proposed in [68] introduced a novel 3D sparse representation named the Vector Quantized Deep Implicit Functions (VQDIF). It converted the 3D point cloud to a set of discrete 2-tuples consisting of the coordinate and the quantized feature index. On this basis, a VQDIF encoder and decoder were designed to perform transformation between the 3D point cloud and the proposed 2-tuples. The sequences of 2-tuples features from partial observations were fed into a Transformer-based autoregressive model to generate complete feature sequences. Then these sequences were projected to a feature grid via the VQDIF decoder. Finally, a 3D-Unet [128] was employed to generate local deep implicit functions of objects’ whole shapes.

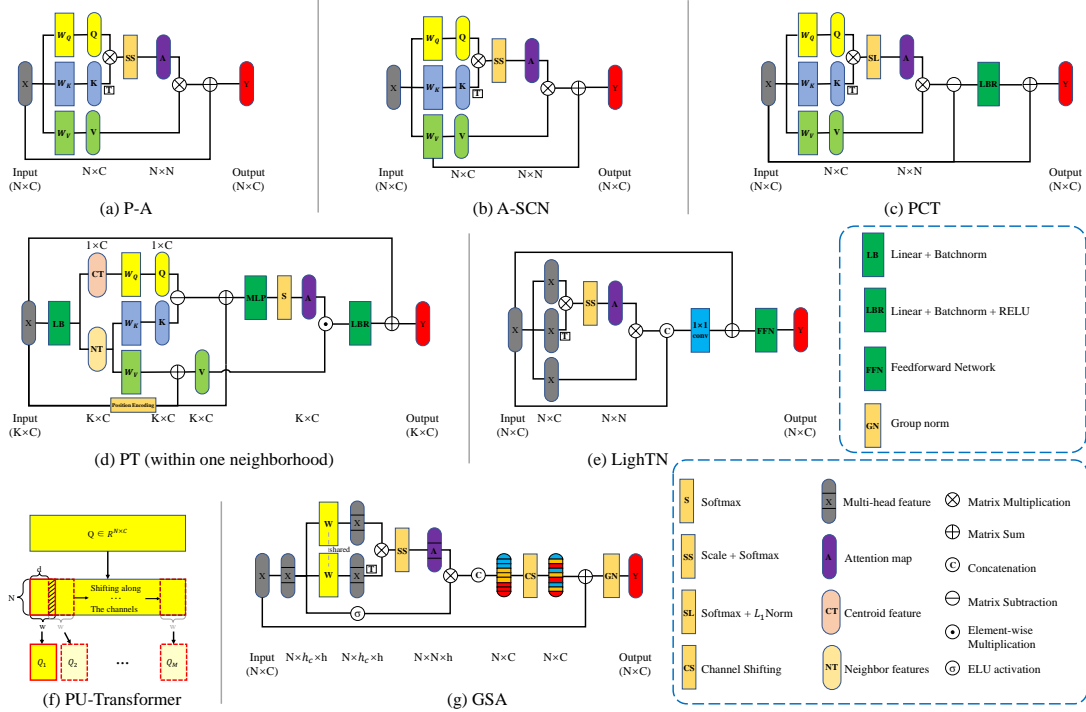


Fig. 4. Architectures of Point-wise Variants.

5 3D SELF-ATTENTION VARIANTS

Based on the standard self-attention module, there are many variants which were designed to improve the performance of Transformers in 3D point cloud processing, as shown in Figs. 4 and 5. As stated in Sec. 2.2, we categorize the relevant variants into two classes: *Point-wise Variants* and *Channel-wise Variants*.

5.1 Point-wise Variants

P-A [10] (Fig. 4(a)) and A-SCN [11] (Fig. 4(b)) have different residual structures in their Transformer encoders. The former strengthened the connection between the output and input of the module, while the later established the relationship between the output and the *Value* matrix of the module. Relevant experiments have demonstrated that the residual connection facilitated model convergence [11].

Inspired by the Laplacian matrix $L = D - E$ in Graph convolution networks [82], the PCT paper [12] further proposed an Offset-Attention module (Fig. 4(c)). This module calculated the offset (difference) between the Self-Attention (SA) features and the input features X by matrix subtraction, which was analogous to a discrete Laplacian operation. Additionally, it refined the normalization of the similarity matrix by replacing *Scale + Softmax* (SS) with *Softmax + L_1 Norm* (SL) operation. It was able to sharpen the attention weights and reduce the influence of noise. Based on the Offset-Attention, Zhou et al. [57] proposed a Relation Attention Module (RAM) which had a similar structure as the Offset-Attention module. The difference was that it first projected *Query*, *Key* and *Value* matrices into latent feature spaces by linear layers. Then, instead of generating the *Attentionmap* by multiplying the *Query* and *Key* matrices directly, it applied the L_2 normalization to the *Query* and

Key matrices. This operation prevented the few feature channels with extremely large magnitudes from overpowering the rest. Ablation experiments in [57] demonstrated that the L_2 normalization was able to improve the model performance.

PT [7] (Fig. 4(d)) introduced the vector subtraction attention operator to its Transformer network, replacing the commonly-used scalar dot-product attention. Compared with the scalar attention, vector attention is more expressive since it supports adaptive modulation of individual feature channels, as opposed to whole feature vectors. This kind of expression appears to be very beneficial in 3D data processing [7]. Point Transformer utilized the subtraction-form vector attention to achieve the local feature aggregation. The attention map was generated by simply building the connections between the centroid feature and its neighbor feature, instead of measuring the similarity between any two point features within a neighborhood. Additionally, the 3D Convolution-Transformer Network (3DCTN) paper [8] conducted a detailed investigation on self-attention operators in 3D Transformers, including the scalar attention and different forms of vector attention.

As mentioned in Sec. 2.3, LighTN [43] presented a self-correlation module, to reduce the computational cost. As shown in Fig. 4(e), it eliminated the projection matrices, W_Q , W_K , and W_V simultaneously in the self-attention mechanism. Only the input self-correlation parameters were used to generate attention features. According to Eq. 4, the self-correlation mechanism generates a symmetry attention map $X \cdot X^T$, which satisfies the permutation invariance in point cloud processing [43]. The authors also conducted a series of ablation studies, removing different projection matrices, to demonstrate the effectiveness of the proposed

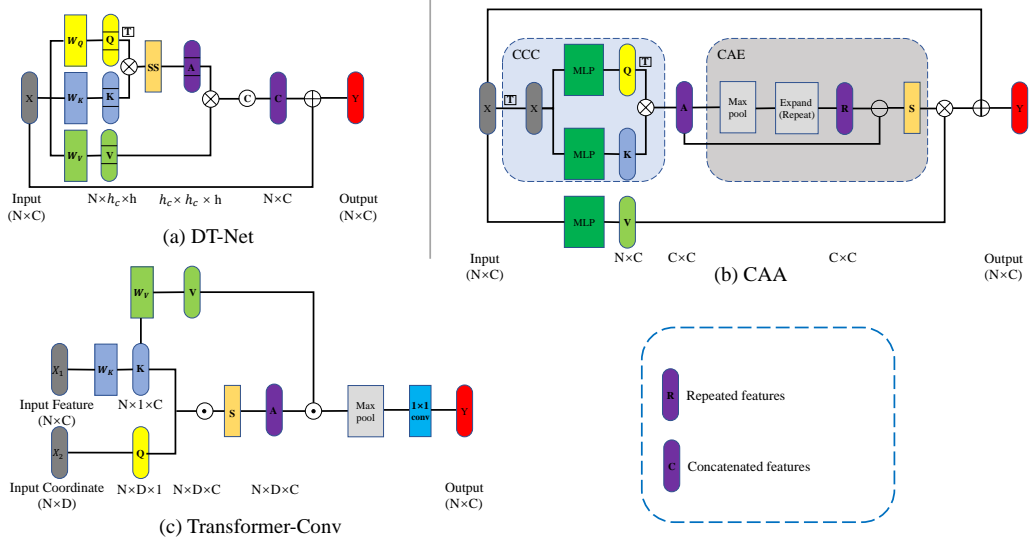


Fig. 5. Architectures of Channel-wise Variants.

self-correlation mechanism.

PU-Transformer [38] proposed the SC-MSA block to improve the MSA mechanism. Specifically, despite the rich information captured by MSA, only feature dependencies within the same head can be estimated. I.e., MSA lacked information propagation between different heads. To address this issue, as shown in Fig. 4(f), PU-Transformer applied a window (dashed square) shift along the channels to ensure that any two consecutive splits had a uniform overlap area. Compared with the independent splits of the standard MSA, SC-MCA is able to enhance the channel-wise relations in the output features.

GSA proposed in [44] had two improvements compared to the standard MSA. The first one was that GSA was a parameter-efficient self-attention mechanism. It used a shared projection matrix W to generate the *Query* and *Key* matrices, and used a non-linearity σ to generate the *Value* matrix:

$$\text{Attn}_\sigma(X) = \text{softmax}\left(\frac{QK^T}{\sqrt{C}}\right)\sigma(X), \quad (5)$$

where $Q = K = XW$ and C is the dimension of X . With this, GSA is able to reduce the computational costs of the self-attention operation. The second one was that GSA introduced channel shuffle to MSA, which enhanced the information flow between heads. As shown in Fig. 4(g), unlike PU-Transformer [38], it re-grouped the channels by rewriting each point feature.

5.2 Channel-wise Variants

Dual Transformer Network (DT-Net) [39] proposed the channel-wise MSA, applying the self-attention mechanism to the channel space. As shown in Fig. 5(a), unlike the standard self-attention mechanism, the channel-wise MSA multiplied the transposed *Query* matrix and *Key* matrix. As such, the attention map could be generated to measure the similarities between different channels, as described in Eq. 3.

As shown in Fig. 5(b), the CAA module [40] utilized a similar approach to generate the similarity matrix between

different channels. Moreover, it designed a CAE block to generate the affinity matrix, strengthening the connection between distinct channels and avoiding aggregating similar/redundant information. The *Value* matrix was generated by an MLP layer, and the final feature map was obtained by multiplying the affinity matrix and *Value* matrix. Additionally, the CAA module used a regular skip connection between the input and output feature map.

The Transformer-Conv module proposed in [41] learned the potential relationship between feature channels and coordinate channels. As shown in Fig. 5(c), The *Query* matrix and *Key* matrix were generated by coordinates and features of the point cloud respectively. Then the similarity matrix was produced by a relation function β (e.g., element-wise multiplication) and channel softmax operation. In contrast with the aforementioned methods, the *Value* matrix in the Transformer-Conv module was generated from the *Key* matrix by linear projection, followed by multiplying the similarity matrix and *Value* matrix in an element-wise manner. Lastly, the final feature was generated by using a channel max-pooling and further 1×1 convolution.

6 COMPARISON AND ANALYSIS

This section gives an overall comparison and analysis of 3D Transformers on several main-stream tasks, including classification, part segmentation, semantic segmentation and object detection.

6.1 Classification & Segmentation

3D point cloud classification and segmentation are two fundamental yet challenging tasks, in which Transformers have played a key role. Classification can best reflect the ability of neural networks to extract salient features. Table. 1 shows the classification accuracy of different methods on the ModelNet40 [112] dataset. For fair comparisons, input data and input size are also shown. We report the Overall Accuracy (OA) as the evaluation metric, which is widely adopted.

TABLE 1

Comparative analysis between involved point cloud classification methods on the ModelNet40 [112] dataset. OA means Overall Accuracy. All results quoted were taken from the cited papers. P = points, N = normals.

Method	input	input size	OA(%)
Non-Transformer			
PointNet [5]	P	1024×3	89.2
PointNet++ [4]	P	1024×3	90.7
PointNet++ [4]	P, N	5120×6	91.9
PointWeb [129]	P	1024×3	92.3
SpiderCNN [130]	P, N	1024×6	92.4
PointCNN [131]	P	1024×3	92.5
PointConv [132]	P, N	1024×6	92.5
FPCNN [133]	P, N	1024×6	92.5
Point2sequence [134]	P	1024×3	92.6
DGCNN [13]	P	1024×3	92.9
KPCNN [88]	P	6800×3	92.9
InterpCNN [135]	P	1024×3	93.0
ShellNet [136]	P	1024×3	93.1
RSMix [137]	P	1024×3	93.5
PAConv [138]	P	1024×3	93.9
RPNNet [139]	P, N	1024×6	94.1
CurveNet [140]	P	1024×3	94.2
PointMLP [141]	P	1024×3	94.5
Attention/Transformer			
ShapeContextNet [11]	P	1024×3	90.0
PATs [44]	P	1024×3	91.7
DT-Net [39]	P	1024×3	92.9
MLMSPT [95]	P	1024×3	92.9
PointASNL [32]	P, N	1024×6	93.2
PCT [12]	P	1024×3	93.2
Centroid Transformers [42]	P	1024×3	93.2
LFT-Net [34]	P, N	2048×6	93.2
3DMedPT [37]	P	1024×3	93.4
3CROSSNet [31]	P	1024×3	93.5
PatchFormer [89]	P, N	1024×6	93.6
Point Transformer [7]	P, N	1024×6	93.7
Point-BERT [33]	P	8192×3	93.8
CAA [40]	P	1024×3	93.8
PVT [45]	P, N	1024×6	94.0

From the table, we can see the recent proliferation of Transformer-based point cloud processing methods from 2020, when the Transformer architecture was first employed in image classification in the ViT paper [17]. Due to the strong ability of global information aggregating, Transformers rapidly achieved leading positions in this task. Most 3D Transformers achieved a classification accuracy of around 93.0%. The newest PVT [45] pushed the limit to 94.0%, which surpassed most non-Transformer algorithms of the same period. As an emerging technology, the success of the Transformer in point cloud classification demonstrates its great potential in the field of 3D point cloud processing. We also presented the results of several state-of-the-art non-Transformer-based methods as reference. As can be seen, the classification accuracy of the recent non-Transformer-based methods has exceeded 94.0%, and the highest one is 94.5%, achieved by PointMLP [141]. The various attention mechanisms used in Transformer methods are versatile and have great future potential for breakthroughs. We believe adapting innovations of general point cloud processing methods to Transformer methods can achieve state-of-the-art results. For example the Geometric Affine Module which resulted in PointMLP’s impressive performance can be easily integrated into a Transformer-based network. I.e. PointMLP’s performance is an indication of the Geometric Affine module’s excellent performance, and not that MLPs are superior

to Transformers as feature extraction backbones.

For part segmentation, ShapeNet part segmentation dataset [142] results were used for comparison. The commonly used part-average Intersection-over-Union was set as the performance metric. As summarised in Table. 2, all the Transformer-based methods achieved a pIoU of around 86%, except for ShapeContextNet [11], which was an early model published before 2019. Note that Stratified Transformer [51] achieved the highest 86.6% pIoU among all the comparative methods. It was also the best model in the task of semantic segmentation on the S3DIS semantic segmentation dataset [110] (Table. 3).

6.2 Object Detection

The application of Transformers to 3D object detection from point clouds remains less explored research area. There are only a few Transformer or Attention-based methods in recent literature. A reason could be that object detection is more complicated than classification. Table. 4 summarises the performance of these Transformer-based networks on two public indoor scene datasets: SUN RGB-D [79] and ScanNetV2 [80]. VoteNet [113] is also reported here as a reference, which is the pioneering work in 3D object detection. In terms of $AP@25$ in the ScanNetV2 dataset, all the Transformer-based methods performed better than VoteNet. Pointformer [35] and MLCVNet [53] were based on VoteNet, and achieved similar performance. Both of them utilized the self-attention mechanism in Transformers to enhance the feature representations. Instead of leveraging the local voting strategy in the aforementioned two approaches, GroupFree3D [54] directly aggregated semantic information from all the points in the scene to extract the features of objects. Its performance of 69.1% demonstrated that aggregating features from all the elements by the self-attention mechanism is a more efficient way than the local voting strategy in VoteNet, MLCVNet, and Pointformer. 3DETR [55], as the first end-to-end Transformer-based 3D object detector, achieved the second best detection performance, 65.0%, in the ScanNetV2 dataset.

7 DISCUSSION AND CONCLUSION

7.1 Discussion

As in 2D computer vision, Transformers also showed its potential in 3D point cloud processing. From the perspective of the 3D tasks, Transformer-based methods mainly focused on high-level tasks, such as classification and segmentation. We argue the reason is that Transformers are better at extracting global contextual information via capturing long-dependency relationships, which corresponds to the semantic information in high-level tasks. On the other hand, low-level tasks, such as denoising and sampling, focus on exploring local geometric features. From the perspective of performance, 3D Transformers improved the accuracy of the aforementioned tasks and surpassed most of the existing methods. However, as shown in Sec. 6, for certain tasks, there is still a gap between them and the start-of-the-art non-Transformer-based methods. This is an indication that simply using Transformers as the backbone is not enough. Other innovative point cloud processing techniques must

TABLE 2

Comparative analysis between different point cloud Transformers in terms of ploU on the ShapeNet part segmentation dataset. ploU means part-average Intersection-over-Union. All results quoted were taken from the cited papers.

Method	ploU	air-plane	bag	cap	car	chair	ear-phone	guitar	knife	lamp	laptop	motor-bike	mug	pistol	rocket	skate-board	table
3DMedPT [37]	84.3	81.2	86.0	91.7	79.6	90.1	81.2	91.9	88.5	84.8	96.0	72.3	95.8	83.2	64.6	78.2	83.8
ShapeContextNet [11]	84.6	83.8	80.8	83.5	79.3	90.5	69.8	91.7	86.5	82.9	96.0	69.2	93.9	82.5	62.9	74.4	80.8
DT-Net [39]	85.6	83.0	81.4	84.3	78.4	90.9	74.3	91.0	87.3	84.7	95.6	69.0	94.4	82.5	59.0	76.4	83.5
3CROSSNet [31]	85.9	83.8	84.9	86.1	79.8	91.2	70.3	91.1	87.0	85.0	95.9	73.2	94.9	83.2	56.2	76.7	83.0
CAA [40]	85.9	84.5	82.2	86.8	78.9	91.1	74.5	91.4	89.0	84.5	95.5	69.6	94.2	83.4	57.8	75.5	83.5
PointASNL [32]	86.1	84.1	84.7	87.9	79.7	92.2	73.7	91.0	87.2	84.2	95.8	74.4	95.2	81.0	63.0	76.3	83.2
LFT-Net [34]	86.2	83.0	83.9	90.9	79.4	93.1	71.4	92.5	88.6	85.7	95.9	69.3	94.2	85.0	65.6	74.6	85.5
PCT [12]	86.4	85.0	82.4	89.0	81.2	91.9	71.5	91.3	88.1	86.3	95.8	64.6	95.8	83.6	62.2	77.6	83.7
MLMSPT [95]	86.4	84.4	84.7	89.2	80.2	89.4	77.1	92.3	87.5	85.3	96.7	71.6	95.2	84.2	61.3	76.0	83.6
PatchFormer [89]	86.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Point Transformer [7]	86.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PVT [45]	86.6	85.3	82.1	88.7	82.1	92.4	75.5	91.0	88.9	85.6	95.4	76.2	94.7	84.2	65.0	75.3	81.7
Stratified Transformer [51]	86.6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

TABLE 3

Comparative analysis between different point cloud Transformers in terms of mIoU/mAcc/OA on the S3DIS Area 5 semantic segmentation dataset. mIoU means mean classwise Intersection over Union, mAcc means mean of classwise ACCuracy, and OA means Overall pointwise Accuracy. All results quoted were taken from the cited papers.

Method	OA	mIoU	mAcc	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
ShapeContextNet [11]	81.6	52.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PATs [44]	-	60.07	70.83	93.04	98.51	72.28	1.00	41.52	85.05	38.22	57.66	83.64	48.12	67.00	61.28	33.64
PCT [12]	-	61.33	67.65	92.54	98.42	80.62	0.00	19.37	61.64	48.00	76.58	85.20	46.22	67.71	67.93	52.29
PointASNL [32]	87.7	62.6	68.5	94.3	98.4	79.1	0.0	26.7	55.2	66.2	83.3	86.8	47.6	68.3	56.4	52.1
MLMST [95]	-	62.9	-	94.5	98.7	90.6	0.0	21.1	60.0	51.4	83.0	89.6	28.9	70.7	74.2	55.5
LFT-Net [34]	-	65.2	76.2	92.8	96.1	81.9	0.0	37.6	70.3	70.4	73.2	76.0	40.9	78.8	71.0	58.2
PVT [45]	-	67.30	-	91.18	98.76	86.23	0.31	34.21	49.90	61.45	81.62	89.85	48.20	79.96	76.45	54.67
EPT [49]	-	67.5	74.7	91.5	97.4	86.0	0.2	40.4	60.8	66.7	87.7	79.6	73.7	58.6	77.2	57.3
PatchFormer [89]	-	68.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Point Transformer [7]	90.8	70.4	76.5	94.0	98.5	86.3	0.0	38.0	63.4	74.3	89.1	82.4	74.3	80.2	76.0	59.3
Stratified Transformer [51]	91.5	72.0	78.1	-	-	-	-	-	-	-	-	-	-	-	-	-

TABLE 4

Comparative analysis between different point cloud Transformers in terms of AP on the ScanNetV2 and SUN RGB-D object detection datasets. AP means Average Precision. All results quoted were taken from the cited papers.

Method	ScanNetV2		SUN RGB-D	
	AP ₂₅	AP ₅₀	AP ₂₅	AP ₅₀
VoteNet [113]	58.6	33.5	57.7	-
3DETR [55]	62.7	37.5	56.8	30.1
Pointformer [35]	64.1	-	61.1	-
MLCVNet [53]	64.5	41.4	59.8	-
3DETR-m [55]	65.0	47.0	59.0	32.7
GroupFree3D [54]	69.1	52.8	63.0	45.2

be employed. Therefore, despite the rapid development of 3D Transformers, as an emerging technology, they still need further exploration and improvement.

Based on the properties of Transformers and their successful applications in the 2D domain, we pointed out several potential future directions for 3D Transformers, hoping it will ignite the further development of this technology.

7.1.1 Patch-wise Transformers

As mentioned in Sec. 2.2, 3D Transformers can be divided into two groups: Point-wise Transformers and Channel-wise Transformers. Moreover, referring to the exploration of Transformers in 2D image processing [87], we are able to further divide Point-wise Transformers into Pair-wise Transformers and Patch-wise Transformers based on the operating form. The former calculates the attention weight for a feature vector by a corresponding pair of points, while the latter incorporates information from all points in a given

patch. Specifically, the self-attention mechanism of pair-wise Transformers can be described as:

$$y_i = \sum_{j \in \mathcal{R}_i} \alpha(x_i, x_j) \odot \beta(x_j), \quad (6)$$

where y_i is the output feature, \mathcal{R}_i is the operating scope of the self-attention module, \odot is the Hadamard product, β projects the feature x_j to a new feature space by linear layers, and $\alpha(x_i, x_j)$ is utilized to measure the relationship between x_i and x_j , which can be decomposed as:

$$\alpha(x_i, x_j) = \rho(\gamma(\delta(x_i, x_j))), \quad (7)$$

where ρ is normalization function like softmax, γ is a mapping function that ensures $\delta(x_i, x_j)$ has the same size as $\beta(x_j)$, and δ is a relation function, the most common examples of which are:

$$\begin{aligned} \text{Concatenation} : \delta(x_i, x_j) &= [\varphi(x_i), \psi(x_j)], \\ \text{Summation} : \delta(x_i, x_j) &= \varphi(x_i) + \psi(x_j), \\ \text{Subtraction} : \delta(x_i, x_j) &= \varphi(x_i) - \psi(x_j), \\ \text{Hadamard product} : \delta(x_i, x_j) &= \varphi(x_i) \odot \psi(x_j), \\ \text{Dot product} : \delta(x_i, x_j) &= \varphi(x_i)^T \psi(x_j), \end{aligned} \quad (8)$$

where the *Dot product* reduces this to a scalar attention operator, while the other forms reduce this to vector attention operators. The subtraction-form vector attention has been used in PT [7]. From the Eq. 6, we can see that the attention weight $\alpha(x_i, x_j)$ is determined by a corresponding pair of point features x_i and x_j . Pair-wise Transformers achieved compelling performance in the 2D image processing and were also commonly used in 3D point cloud processing. Nearly

all algorithms in Sec. 2.2.1 can be considered as pair-wise Transformers, where most of them used the *Dot product*.

Zhao et al. [87] also explored a family of patch-wise Transformers in image processing, whose self-attention mechanism can be expressed as:

$$y_i = \sum_{j \in \mathbb{R}_i} \alpha(x_{\mathbb{R}_i})_j \odot \beta(x_j), \quad (9)$$

where $x_{\mathbb{R}_i}$ is the patch of feature vectors in \mathbb{R}_i , α transforms the $x_{\mathbb{R}_i}$ to a new tensor with the same spatial dimensionality, and $\alpha(x_{\mathbb{R}_i})_j$ is the j -th feature vector in this tensor. Similar to pair-wise Transformers, $\alpha(x_{\mathbb{R}_i})$ can also be decomposed as:

$$\alpha(x_{\mathbb{R}_i}) = \rho(\gamma(\delta(x_{\mathbb{R}_i}))), \quad (10)$$

and δ can be expressed as three different forms [87]:

$$\begin{aligned} \text{Concatenation} : \delta(x_{\mathbb{R}_i}) &= [\varphi(x_i), [\psi(x_j)]_{\forall j \in \mathbb{R}_i}], \\ \text{Star-product} : \delta(x_{\mathbb{R}_i}) &= [\varphi(x_i)^T \psi(x_j)]_{\forall j \in \mathbb{R}_i}, \\ \text{Dot product} : \delta(x_{\mathbb{R}_i}) &= [\varphi(x_j)^T \psi(x_k)]_{\forall j, k \in \mathbb{R}_i}. \end{aligned} \quad (11)$$

By comparing Eq. 6 and 9, we see that the latter aggregates all feature vectors in \mathbb{R}_i to generate the weight matrix that is applied to $\beta(x_j)$, instead of merely utilizing a pair of features. In this way, patch-wise Transformers are able to enhance the connections among different feature vectors, and extract more robust short- and long-range dependencies. However, since the feature vectors are arranged in a particular order in $x_{\mathbb{R}_i}$, patch-wise Transformers are not permutation-equivariant, which may have some negative effects on point cloud processing.

Currently, there is little patch-wise Transformer research in the field of 3D point cloud processing. Considering the advantages of patch-wise Transformers and their outstanding performance in image processing, we believe that introducing patch-wise Transformers to point cloud processing is beneficial to performance improvement.

7.1.2 Adaptive Set Abstraction

PointNet++ [4] proposed a Set Abstraction (SA) module to extract the semantic features of the point cloud hierarchically. It mainly utilized FPS and query ball grouping algorithms to achieve sampling point searching and local patch construction respectively. However, the sampling points generated by FPS tend to be evenly distributed in the original point cloud, while ignoring the geometric and semantic differences between different parts. For example, the tail of the aircraft is more geometrically complex and distinct than the fuselage. As such, the former needs more sampling points to be described. Moreover, query ball grouping focuses on searching the neighbor points only based on the Euclidean distance. However, it ignores the semantic feature differences among points, which makes it easy to group points with different semantic information into the same local patch. Therefore, developing an adaptive set abstraction is beneficial to improving the performance of 3D Transformers. Recently, there have been several Transformer-based methods in the 3D field exploring adaptive sampling [43]. But few of them made full use of the rich short- and long-range dependencies generated by the self-attention mechanism. In the field of image processing,

Deformable Attention Transformer (DAT) proposed in [143] generated the deformed sampling points by introducing an offset network. It achieved impressive results on comprehensive benchmarks with low computational footprint. It will be meaningful to present an adaptive sampling method based on the self-attention mechanism for the hierarchical Transformer. Additionally, inspired by the superpixel [144] in the 2D field, we argue that it is feasible to utilize the attention map in 3D Transformers to obtain the ‘‘superpoint’’ [145] for point cloud oversegmentation, converting point-level 3D data into neighborhood-level data. As such, this adaptive clustering technique can be used to replace the query ball grouping method.

7.1.3 Self-supervised Transformer Pre-training

Transformers have shown impressive performance on NLP and 2D image processing tasks. However, much of their success stems not only from their excellent scalability but also from large-scale self-supervised pre-training [83]. Vision Transformer [17] performed a series of self-supervision experiments, and demonstrated the potential of the self-supervised Transformer. In the field of point cloud processing, despite the significant progress of supervised point cloud approaches, point cloud annotation is still a labor-intensive task. And the limited labeled dataset hinders the development of supervised approaches, especially in terms of the point cloud segmentation task. Recently, there have been a series of self-supervised approaches proposed to deal with these issues, such as Generative Adversarial Networks (GAN) [146] in the 2D field, Auto-Encoders (AE) [147], [148], and Gaussian Mixture Models (GMM) [149]. These methods used auto-encoders and generative models to realize self-supervised point cloud representation learning [96]. Their satisfactory performances have demonstrated the effectiveness of the self-supervised point cloud approaches. However, few self-supervised Transformers have been currently applied to 3D point cloud processing. With the increasing availability of large-scale 3D point clouds, it is worthwhile to explore the self-supervised 3D Transformers for point cloud representation learning.

Overall, Transformers have only started to be applied to point cloud-related tasks. This research area has much space for innovations, especially by integrating breakthroughs from NLP and 2D computer vision.

7.2 Conclusion

Transformer models have attracted widespread attention in the field of 3D point cloud processing, and achieved impressive results in various 3D tasks. In this paper, we have comprehensively reviewed recent Transformer-based networks applied to point cloud-related tasks, such as point cloud classification, segmentation, object detection, registration, sampling, denoising, completion and other practical applications. We first introduced the theory behind the Transformer architecture, and described the development and applications of 2D and 3D Transformers. Then we utilized three different taxonomies to categorize the current methods found in literature into multiple groups, and analyzed them from multiple perspectives. Additionally, we also described a series of self-attention variants that aimed

to improve the performance and reduce the computational cost. In terms of point cloud classification, segmentation and object detection, brief comparisons of the reviewed methods were provided in this paper. Finally, we suggested three potential future research directions for the development of 3D Transformers. We hope this survey gives researchers a comprehensive view of 3D Transformers, and drives their interest to further innovate the research in this field.

REFERENCES

- [1] K. Han *et al.*, “A survey on vision transformer,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022, doi: 10.1109/TPAMI.2022.3152247.
- [2] Y. Li, T. Yao, Y. Pan, and T. Mei, “Contextual transformer networks for visual recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022, doi: 10.1109/TPAMI.2022.3164083.
- [3] J. Xiao, X. Fu, A. Liu, F. Wu, and Z.-J. Zha, “Image de-raining transformer,” *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–18, 2022, doi: 10.1109/TPAMI.2022.3183612.
- [4] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, p. 5105–5114.
- [5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.
- [6] A. Vaswani *et al.*, “Attention is all you need,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
- [7] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, “Point transformer,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 16 259–16 268.
- [8] D. Lu, Q. Xie, L. Xu, and J. Li, “3DCTN: 3D convolution-transformer network for point cloud classification,” *arXiv:2203.00828*, 2022. [Online]. Available: <http://arxiv.org/abs/2203.00828>
- [9] M. Tancik *et al.*, “Fourier features let networks learn high frequency functions in low dimensional domains,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7537–7547.
- [10] M. Feng, L. Zhang, X. Lin, S. Z. Gilani, and A. Mian, “Point attention network for semantic segmentation of 3D point clouds,” *Pattern Recognit.*, vol. 107, p. 107446, 2020, doi: 10.1016/j.patcog.2020.107446.
- [11] S. Xie, S. Liu, Z. Chen, and Z. Tu, “Attentional shapecontextnet for point cloud recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4606–4615.
- [12] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, “PCT: Point cloud transformer,” *Comput. Vis. Media.*, vol. 7, no. 2, pp. 187–199, 2021.
- [13] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph CNN for learning on point clouds,” *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.
- [14] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, “Max-deeplab: End-to-end panoptic segmentation with mask transformers,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5463–5474.
- [15] N. Carion *et al.*, “End-to-end object detection with transformers,” in *Proc. Eur. Conf. Comput. Vis.*, vol. 12346, 2020, pp. 213–229.
- [16] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, “Transformer tracking,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8126–8135.
- [17] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–12.
- [18] B. Wu *et al.*, “Visual transformers: Token-based image representation and processing for computer vision,” *arXiv:2006.03677*, 2020. [Online]. Available: <http://arxiv.org/abs/2006.03677>
- [19] W. Wang *et al.*, “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 548–558.
- [20] H. Wu *et al.*, “CvT: Introducing convolutions to vision transformers,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 22–31.
- [21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 9992–10 002.
- [22] E. Xie and othersg, “Segformer: Simple and efficient design for semantic segmentation with transformers,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 12 077–12 090.
- [23] B. Cheng, A. Schwing, and A. Kirillov, “Per-pixel classification is not all you need for semantic segmentation,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 17 864–17 875.
- [24] Y. Wang *et al.*, “End-to-end video instance segmentation with transformers,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8741–8750.
- [25] Z. Yao, J. Ai, B. Li, and C. Zhang, “Efficient DETR: improving end-to-end object detector with dense prior,” *arXiv:2104.01318*, 2021. [Online]. Available: <http://arxiv.org/abs/2104.01318>
- [26] J. Yang *et al.*, “Focal self-attention for local-global interactions in vision transformers,” *arXiv:2107.00641*, 2021. [Online]. Available: <http://arxiv.org/abs/2107.00641>
- [27] X. Chen, S. Xie, and K. He, “An empirical study of training self-supervised vision transformers,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 9640–9649.
- [28] Y. Liu *et al.*, “A survey of visual transformers,” *arXiv:2111.06091*, 2021. [Online]. Available: <http://arxiv.org/abs/2111.06091>
- [29] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, “Transformers in vision: A survey,” *ACM Computing Surveys (CSUR)*, 2021.
- [30] P. Xu, X. Zhu, and D. A. Clifton, “Multimodal learning with transformers: A survey,” *arXiv:2206.06488*, 2022. [Online]. Available: <http://arxiv.org/abs/2206.06488>
- [31] X.-F. Han, Z.-Y. He, J. Chen, and G.-Q. Xiao, “3CROSSNet: Cross-level cross-scale cross-attention network for point cloud representation,” *IEEE Robotics Autom. Lett.*, vol. 7, no. 2, pp. 3718–3725, 2022.
- [32] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, “PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5589–5598.
- [33] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, “Point-BERT: Pre-training 3D point cloud transformers with masked point modeling,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19 313–19 322.
- [34] Y. Gao, X. Liu, J. Li, Z. Fang, X. Jiang, and K. M. S. Huq, “LFT-Net: Local feature transformer network for point clouds analysis,” *IEEE Trans. Intell. Transport. Syst.*, 2022, doi: 10.1109/TITS.2022.3140355.
- [35] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, “3D object detection with pointformer,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7463–7472.
- [36] X. Xu, G. Geng, X. Cao, K. Li, and M. Zhou, “TDNet: transformer-based network for point cloud denoising,” *Appl. Opt.*, vol. 61, no. 6, pp. C80–C88, 2022.
- [37] J. Yu *et al.*, “3D medical point transformer: Introducing convolution to attention networks for medical point cloud analysis,” *arXiv:2112.04863*, 2021. [Online]. Available: <http://arxiv.org/abs/2112.04863>
- [38] S. Qiu, S. Anwar, and N. Barnes, “PU-Transformer: Point cloud upsampling transformer,” *arXiv:2111.12242*, 2021. [Online]. Available: <http://arxiv.org/abs/2111.12242>
- [39] X.-F. Han, Y.-F. Jin, H.-X. Cheng, and G.-Q. Xiao, “Dual transformer for point cloud analysis,” *arXiv:2104.13044*, 2021. [Online]. Available: <http://arxiv.org/abs/2104.13044>
- [40] S. Qiu, S. Anwar, and N. Barnes, “Geometric back-projection network for point cloud classification,” *IEEE Trans. Multimedia*, vol. 24, pp. 1943–1955, 2022.
- [41] G. Xu, H. Cao, J. Wan, K. Xu, Y. Ma, and C. Zhang, “Adaptive channel encoding transformer for point cloud analysis,” *arXiv:2112.02507*, 2021. [Online]. Available: <http://arxiv.org/abs/2112.02507>
- [42] L. Wu, X. Liu, and Q. Liu, “Centroid transformers: Learning to abstract with attention,” *arXiv:2102.08606*, 2021. [Online]. Available: <http://arxiv.org/abs/2102.08606>
- [43] X. Wang, Y. Jin, Y. Cen, T. Wang, B. Tang, and Y. Li, “LighTN: Light-weight transformer network for performance-overhead tradeoff in point cloud downsampling,” *arXiv:2202.06263*, 2022. [Online]. Available: <http://arxiv.org/abs/2202.06263>
- [44] J. Yang *et al.*, “Modeling point clouds with self-attention and gumbel subset sampling,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3323–3332.

- [45] C. Zhang, H. Wan, S. Liu, X. Shen, and Z. Wu, "PVT: Point-voxel transformer for 3D deep learning," *arXiv:2108.06076*, 2021. [Online]. Available: <http://arxiv.org/abs/2108.06076>
- [46] J. Mao *et al.*, "Voxel transformer for 3D object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 3164–3173.
- [47] S. L. Chenhang He, Ruihuang Li and L. Zhang, "Voxel set transformer: A set-to-set approach to 3D object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8417–8427.
- [48] Z. Fan, Z. Song, H. Liu, Z. Lu, J. He, and X. Du, "SVT-Net: Super light-weight sparse voxel transformer for large scale place recognition," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 551–560.
- [49] C. Park, Y. Jeong, M. Cho, and J. Park, "Efficient point transformer for large-scale 3D scene understanding," 2022. [Online]. Available: <https://openreview.net/forum?id=3SUTolxulT3>
- [50] J. Lin, M. Rickert, A. Perzylo, and A. Knoll, "PCTMA-Net: Point cloud transformer with morphing atlas-based point generation network for dense point cloud completion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5657–5663.
- [51] X. Lai *et al.*, "Stratified transformer for 3D point cloud segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8500–8509.
- [52] L. Hui, H. Yang, M. Cheng, J. Xie, and J. Yang, "Pyramid point cloud transformer for large-scale place recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 6098–6107.
- [53] Q. Xie *et al.*, "MLCVNet: Multi-level context votenet for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10447–10456.
- [54] Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong, "Group-free 3D object detection via transformers," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 2949–2958.
- [55] I. Misra, R. Girdhar, and A. Joulin, "An end-to-end transformer model for 3D object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 2906–2917.
- [56] Y. Cui, Z. Fang, J. Shan, Z. Gu, and S. Zhou, "3D object tracking with transformer," *Proc. Brit. Mach. Vis. Conf.*, p. 317, 2021.
- [57] C. Zhou *et al.*, "PTTR: Relational 3D point cloud object tracking with transformer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8531–8540.
- [58] S. Jiayao, S. Zhou, Y. Cui, and Z. Fang, "Real-time 3D single object tracking with transformer," *IEEE Trans. Multimedia*, 2022, doi: 10.1109/TMM.2022.3146714.
- [59] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3523–3532.
- [60] Y. Wang, C. Yan, Y. Feng, S. Du, Q. Dai, and Y. Gao, "STORM: Structure-based overlap matching for partial point cloud registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022, doi: 10.1109/TPAMI.2022.3148308.
- [61] K. Fischer *et al.*, "StickyPillars: Robust and efficient feature matching on point clouds using graph neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 313–323.
- [62] K. Fu, S. Liu, X. Luo, and M. Wang, "Robust point cloud registration framework based on deep graph matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8893–8902.
- [63] G. Chen, M. Wang, Y. Yue, Q. Zhang, and L. Yuan, "Full transformer framework for robust point cloud registration with deep information interaction," *arXiv:2112.09385*, 2021. [Online]. Available: <http://arxiv.org/abs/2112.09385>
- [64] H. Fan, Y. Yang, and M. Kankanhalli, "Point 4D transformer networks for spatio-temporal modeling in point cloud videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 14204–14213.
- [65] R. Gao, M. Li, S.-J. Yang, and K. Cho, "Reflective noise filtering of large-scale point cloud using transformer," *Remote Sens.*, vol. 14, no. 3, p. 577, 2022.
- [66] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, "PoinTr: Diverse point cloud completion with geometry-aware transformers," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 12498–12507.
- [67] P. Xiang *et al.*, "SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 5499–5509.
- [68] X. Yan, L. Lin, N. J. Mitra, D. Lischinski, D. Cohen-Or, and H. Huang, "ShapeFormer: Transformer-based shape completion via sparse representation," *arXiv:2201.10326*, 2022. [Online]. Available: <http://arxiv.org/abs/2201.10326>
- [69] H. Sheng, S. Cai, Y. Liu, B. Deng, J. Huang, X.-S. Hua, and M.-J. Zhao, "Improving 3D object detection with channel-wise transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 2743–2752.
- [70] Y. Wei, H. Liu, T. Xie, Q. Ke, and Y. Guo, "Spatial-temporal transformer for 3D point cloud sequences," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2022, pp. 1171–1180.
- [71] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, and K. Xu, "Geometric transformer for fast and robust point cloud registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11143–11152.
- [72] Z. J. Yew and G. h. Lee, "REGTR: End-to-end point cloud correspondences with transformers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6677–6686.
- [73] W. Chen, H. Liang, Z. Chen, F. Sun, and J. Zhang, "TransSC: Transformer-based shape completion for grasp evaluation," *arXiv:2107.00511*, 2021. [Online]. Available: <http://arxiv.org/abs/2107.00511>
- [74] X. Liu, G. Xu, K. Xu, J. Wan, and Y. Ma, "Point cloud completion by dynamic transformer with adaptive neighbourhood feature fusion," *IET Comput. Vis.*, 2022, doi: 10.1049/cvi2.12098.
- [75] Y. Zhou, A. Ji, and L. Zhang, "Sewer defect detection from 3D point clouds using a transformer-based deep learning model," *Autom. Constr.*, vol. 136, p. 104163, 2022.
- [76] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7077–7087.
- [77] Z. Yuan, X. Song, L. Bai, Z. Wang, and W. Ouyang, "Temporal-channel transformer for 3D lidar-based video object detection for autonomous driving," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 2068–2078, 2021.
- [78] S. Qiu, Y. Wu, S. Anwar, and C. Li, "Investigating attention mechanism in 3D point cloud object detection," in *3DV*, 2021, pp. 403–412.
- [79] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 567–576.
- [80] A. Dai *et al.*, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5828–5839.
- [81] X.-Y. Gao, Y.-Z. Wang, C.-X. Zhang, and J.-Q. Lu, "Multi-head self-attention for 3D point cloud classification," *IEEE Access*, vol. 9, pp. 18137–18147, 2021.
- [82] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent.*, 2014.
- [83] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.
- [84] J. T. Rolfe, "Discrete variational autoencoders," *arXiv:1609.02200*, 2016. [Online]. Available: <http://arxiv.org/abs/1609.02200>
- [85] Z. Wang, Y. Wang, L. An, J. Liu, and H. Liu, "Local transformer network on 3D point cloud semantic segmentation," *Information*, vol. 13, no. 4, p. 198, 2022.
- [86] S. Liu, K. Fu, M. Wang, and Z. Song, "Group-in-group relation-based transformer for 3D point cloud learning," *Remote Sens.*, vol. 14, no. 7, p. 1563, 2022.
- [87] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10076–10085.
- [88] H. Thomas *et al.*, "KPconv: Flexible and deformable convolution for point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6411–6420.
- [89] Z. Cheng, H. Wan, X. Shen, and Z. Wu, "Patchformer: A versatile 3D transformer based on patch attention," *arXiv:2111.00207*, 2021. [Online]. Available: <http://arxiv.org/abs/2111.00207>
- [90] S. Mehta, M. Ghazvininejad, S. Iyer, L. Zettlemoyer, and H. Hajishirzi, "Delight: Deep and light-weight transformer," *arXiv:2008.00623*, 2020. [Online]. Available: <http://arxiv.org/abs/2008.00623>
- [91] Y. Xu, X. Tong, and U. Stilla, "Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry," *Autom. Constr.*, vol. 126, p. 103675, 2021.
- [92] B. Graham, M. Engelcke, and L. Van Der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9224–9232.

- [93] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3075–3084.
- [94] J. Lee *et al.*, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3744–3753.
- [95] X.-F. Han, Y.-J. Kuang, and G.-Q. Xiao, "Point cloud learning with transformer," *arXiv:2104.13636*, 2021. [Online]. Available: <http://arxiv.org/abs/2104.13636>
- [96] K. Fu, P. Gao, R. Zhang, H. Li, Y. Qiao, and M. Wang, "Distillation with contrast is all you need for self-supervised point cloud representation learning," *arXiv:2202.04241*, 2022. [Online]. Available: <http://arxiv.org/abs/2202.04241>
- [97] C.-K. Yang, J.-J. Wu, K.-S. Chen, Y.-Y. Chuang, and Y.-Y. Lin, "An MIL-Derived transformer for weakly supervised point cloud segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11 830–11 839.
- [98] C. Park, Y. Jeong, M. Cho, and J. Park, "Fast point transformer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16 949–16 958.
- [99] J. Zhang, X. Li, X. Zhao, Y. Ge, and Z. Zhang, "U-shaped network based on transformer for 3D point clouds semantic segmentation," in *ICVIP*, 2021, pp. 170–176.
- [100] X. Chen, H. Zhao, G. Zhou, and Y.-Q. Zhang, "PQ-transformer: Jointly parsing 3D objects and layouts from point clouds," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2519–2526, 2022.
- [101] Y. Zhang, J. Chen, and D. Huang, "CAT-Det: Contrastively augmented transformer for multi-modal 3D object detection," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022.
- [102] Y. Wang *et al.*, "Bridged transformer for vision and point cloud 3D object detection," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 12 114–12 123, 2022.
- [103] G. Trappolini, L. Cosmo, L. Moschella, R. Marin, S. Melzi, and E. Rodolà, "Shape registration in the time of transformers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 5731–5744.
- [104] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1106–1114, 2012.
- [105] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [106] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [107] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
- [108] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, 2002.
- [109] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 68–80.
- [110] I. Armeni *et al.*, "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1534–1543.
- [111] K. He *et al.*, "Masked autoencoders are scalable vision learners," *arXiv:2111.06377*, 2021. [Online]. Available: <http://arxiv.org/abs/2111.06377>
- [112] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.
- [113] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9277–9286.
- [114] Q. Xie, Y.-K. Lai, J. Wu, Z. Wang, Y. Zhang, K. Xu, and J. Wang, "Vote-based 3D object detection with context modeling and SOB-3DNMS," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1857–1874, 2021.
- [115] X. Bai *et al.*, "TransFusion: robust lidar-camera fusion for 3D object detection with transformers," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1090–1099, 2022.
- [116] H. Qi, C. Feng, Z. Cao, F. Zhao, and Y. Xiao, "P2B: Point-to-box network for 3D object tracking in point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6329–6338.
- [117] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
- [118] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1802–1811.
- [119] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "PREDATOR: Registration of 3D point clouds with low overlap," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4267–4276.
- [120] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3D points," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2010, pp. 9–14.
- [121] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1010–1019.
- [122] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2684–2701, 2019.
- [123] S. Huang, Z. Yang, Y. Shi, J. Tan, H. Li, and Y. Cheng, "3DPCTN: Two 3D local-object point-cloud-completion transformer networks based on self-attention and multi-resolution," *Electronics*, vol. 11, no. 9, p. 1351, 2022.
- [124] X. Wen *et al.*, "PMP-Net++: Point cloud completion by transformer-enhanced multi-step point moving paths," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.
- [125] M. Wei *et al.*, "AGConv: Adaptive graph convolution on 3D point clouds," *arXiv:2206.04665*, 2022. [Online]. Available: <http://arxiv.org/abs/2206.04665>
- [126] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Interpretable unsupervised learning on 3D point clouds," *arXiv:1712.07262*, 2017. [Online]. Available: <http://arxiv.org/abs/1712.07262>
- [127] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *3DV*, 2018, pp. 728–737.
- [128] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: learning dense volumetric segmentation from sparse annotation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 2016, pp. 424–432.
- [129] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5565–5573.
- [130] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Eur. Conf. Comput. Vis.*, 2018, pp. 87–102.
- [131] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 828–838.
- [132] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9621–9630.
- [133] Y. Lin *et al.*, "FPConv: Learning local flattening for point convolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4293–4302.
- [134] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, 2019, pp. 8778–8785.
- [135] J. Mao, X. Wang, and H. Li, "Interpolated convolutional networks for 3D point cloud understanding," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1578–1587.
- [136] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1607–1616.
- [137] D. Lee *et al.*, "Regularization strategy for point cloud via rigidly mixed sample," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15 900–15 909.
- [138] M. Xu, R. Ding, H. Zhao, and X. Qi, "PAConv: Position adaptive convolution with dynamic kernel assembling on point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3173–3182.
- [139] H. Ran, W. Zhuo, J. Liu, and L. Lu, "Learning inner-group relations on point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15 477–15 487.

- [140] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, "Walk in the cloud: Learning curves for point clouds shape analysis," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 915–924.
- [141] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual MLP framework," *arXiv:2202.07123*, 2022. [Online]. Available: <http://arxiv.org/abs/2202.07123>
- [142] L. Yi *et al.*, "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.
- [143] Z. Xia, X. Pan, S. Song, L. E. Li, and G. Huang, "Vision transformer with deformable attention," *arXiv:2201.00520*, 2022. [Online]. Available: <http://arxiv.org/abs/2201.00520>
- [144] L. Zhu *et al.*, "Learning the superpixel in a non-iterative and life-long manner," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1225–1234.
- [145] L. Hui, J. Yuan, M. Cheng, J. Xie, X. Zhang, and J. Yang, "Super-point network for point cloud oversegmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 5510–5519.
- [146] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
- [147] M. Gadelha, R. Wang, and S. Maji, "Multiresolution tree networks for 3D point cloud processing," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 103–118.
- [148] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3861–3870.
- [149] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 40–49.