Computational Visual Media
https://doi.org/10.26599/CVM.2025.9450383 Vol. 11, No. 1, February 2025, 83–101

Research Article

Swin3D: A pretrained transformer backbone for 3D indoor scene understanding

Yu-Qi Yang[1], Yu-Xiao Guo[3], Jian-Yu Xiong[2], Yang Liu[3] (✉), Hao Pan[3], Peng-Shuai Wang[4], Xin Tong[3], and Baining Guo[3]

Abstract The use of pretrained backbones with fine-tuning has shown success for 2D vision and natural language processing tasks, with advantages over task-specific networks. In this paper, we introduce a pretrained 3D backbone, called Swin3D, for 3D indoor scene understanding. We designed a 3D Swin Transformer as our backbone network, which enables efficient self-attention on sparse voxels with linear memory complexity, making the backbone scalable to large models and datasets. We also introduce a generalized contextual relative positional embedding scheme to capture various irregularities of point signals for improved network performance. We pretrained a large Swin3D model on a synthetic Structured3D dataset, which is an order of magnitude larger than the ScanNet dataset. Our model pretrained on the synthetic dataset not only generalizes well to downstream segmentation and detection on real 3D point datasets but also outperforms state-of-the-art methods on downstream tasks with +2.3 mIoU and +2.2 mIoU on S3DIS Area5 and 6-fold semantic segmentation, respectively, +1.8 mIoU on ScanNet segmentation (val), +1.9 mAP@0.5 on ScanNet detection, and +8.1 mAP@0.5 on S3DIS detection. A series of extensive ablation studies further validated the scalability, generality, and superior performance enabled by our approach.

1 Institute for Advanced Study, Tsinghua University, Beijing 100084, China. E-mail: yangyq18@mails.tsinghua.edu.cn.

2 Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China. E-mail: xjy21@mails.tsinghua.edu.cn.

3 Internet Graphics Group, Microsoft Research Asia, Beijing 100080, China. E-mail: Y.-X. Guo, yuxgu@microsoft.com; Y. Liu, yangliu@microsoft.com (✉); H. Pan, haopan@microsoft.com; X. Tong, xtong@microsoft.com; B. Guo, bainguo@microsoft.com.

4 Wangxuan Institute of Computer Technology, Peking University, Beijing 100080, China. E-mail: wangps@hotmail.com.

Manuscript received: 2023-08-06; accepted: 2023-10-01

Keywords 3D pretraining; ponitcloud analysis; transformer backbone; Swin Transformer; 3D semantic segmentation; 3D object detection

1 Introduction

A paradigm shift has been seen in the fields of Natural Language Processing (NLP) and 2D vision, where the use of large pre-trained backbones has been highly successful [1–6]. This approach has the advantage of being able to generalize to various tasks, while also reducing the amount of network design and training needed as well as the amount of labeled data required for various vision tasks. This involves pre-training a backbone network with a simple design on a large dataset and then fine-tuning it for different downstream tasks. However, the development of generic and scalable pretrained 3D backbones for point cloud understanding is still in its early stages, and existing pretrained 3D backbones are not as effective as state-of-the-art non-pretrained approaches for many 3D vision tasks.

This study aimed to investigate the scalability and generality of a 3D pretrained model, without the need for a complex network design. We introduce a pretrained 3D backbone, Swin3D, for 3D indoor scene understanding tasks. Our method uses sparse voxels to represent the 3D point cloud of an input 3D scene and adapts the network design of Swin Transformer [5], which was originally designed for regular 2D images, to unorganized 3D points as the 3D backbone. We identify two key issues that prevent the naive 3D extension of Swin Transformer from exploring large models and achieving high performance: *high memory complexity* and *ignorance of signal irregularity*. To address these issues, we

developed a novel memory-efficient self-attention operator to compute the self-attentions of sparse voxels within each local window. This reduces the memory cost of self-attention from quadratic to linear with respect to the number of sparse voxels within a window and computes efficiently without sacrificing self-attention accuracy. We extend the contextual relative positional embedding [7, 8] to contextual relative signal embedding to capture point signal irregularities that have not been taken into account by prior research, leading to a significant enhancement in network performance and scalability.

Our novel Swin3D backbone design allows us to scale up the backbone model and utilize large amount of data for pretraining. To demonstrate this, we pretrained a large Swin3D model with 60 M parameters on a 3D semantic segmentation task using a large synthetic 3D dataset: Structured3D [9]. This dataset is ten times larger than the ScanNet dataset [10], which contains 21k rooms. After pretraining, we combined the pretrained Swin3D backbone with task-specific back-end decoders and fine-tuned the models for various 3D indoor scene understanding tasks.

We evaluated the performance of our method on both 3D detection and semantic segmentation tasks on real data, including the ScanNet and S3DIS datasets. Experimental results show that our Swin3D pretrained on the synthetic dataset exhibits good generality and outperforms all existing state-of-the-art methods with +8.1 mAP@0.5 on S3DIS detection, +2.2 mIoU on 6-fold S3DIS segmentation [11], +1.9 mAP@0.5 on ScanNet detection, and +1.8 mIoU on ScanNet segmentation (validation set). We carefully analyzed the contributions of different factors (e.g., model size, data size, pretraining, memory-efficient self-attention, and contextual relative signal embedding) to the performance of our model. Our results show that our pretrained backbones with fine-tuning are superior to the same models trained from scratch and significantly outperform other existing models pretrained with the same large data.

The large backbone model and large amount of 3D data used for pretraining enabled by our backbone design are critical to the superior performance of our method in downstream tasks. We believe that our work illustrates the great potential of a unified pretrained backbone for various 3D understanding tasks. To facilitate and inspire future research along this path, our code and trained models are available at `https://github.com/microsoft/Swin3D`.

2 Related work

Vision transformers. Transformers based on the attention mechanism have been used successfully in computer vision and have achieved great results in many 2D vision tasks, such as image classification, semantic segmentation, and object detection (cf. the comprehensive surveys [12–14]). The plain vision transformer [1] computes global self-attention over the entire image, thus providing long-range attention between image patches; however, this leads to high memory and computational costs due to the quadratic complexity of self-attention. To address this issue, local-window self-attention over small non-overlapping patches [5, 15] was introduced. Various techniques have been proposed to improve the long-range attention of window-based self-attention, such as using a window hierarchy [5], constructing non-local self-attention patterns [16–23], and expanding the receptive field via convolution [24, 25]. Most vision transformers are pretrained with large-scale image datasets and serve as generic vision backbones for multiple purposes.

3D transformers for point cloud understanding. Transformers have been quickly adapted to 3D [26]. Guo et al. [27] used global attention on points and achieved good results in object classification and shape segmentation. Zhao et al. [28] introduced local attention on point clouds, which reduced memory and computational complexity and enabled the point transformer to be used for point clouds at the scene level. Wu et al. [29] further improved the point transformer by using grouped vector attention and partition-based pooling. Fast Point Transformer [30] employed voxel hashing and a lightweight self-attention layer to enhance network efficiency. Stratified Transformer [8] adapted the Swin Transformer design for 3D point clouds and proposed a stratified strategy to expand the receptive field and used contextual relative positional encoding [7] to strengthen self-attention with position information. Nevertheless, its computational and memory requirements remain high because of the inefficient self-attention implementation, which is not able to scale up to accommodate larger model designs. Furthermore, most 3D transformers that have been

created have been tailored to particular tasks, and simply (pre)training them on a large amount of data does not always result in better performance, as we evaluated (Section 6).

Pretrained 3D backbones. Self-supervised learning strategies have been applied to 3D backbone pretraining. PointContrast [31] used point-level losses to pretrain a sparse convolution-based 3D U-Net. MID-Net [32] pretrained an Octree-based HRNet with multiresolution contrastive losses. Hou et al. [33] improved the efficacy of PointContrast by taking advantage of spatial information. DepthContrast [34] employed depth maps to enhance contrastive learning. Masked-signal-modeling-based transformer models, such as BEiT [2] and the masked autoencoder [35], have been used for 3D pretraining [36–39]. Wu et al. [40] integrated the masked point modeling strategy with contrastive learning to boost backbone pretraining. Recently, pretrained image or CLIP models have been utilized for 3D learning [41–44], forming a new type of pretrained 3D backbone. ShapeNet [45] and ScanNet [10] are the main 3D data sources for the above pre-training work. Despite the rapid development of 3D pretraining, its performance on 3D indoor scene understanding is still inferior to that of state-of-the-art non-pretrained approaches.

3 Architecture overview

3.1 Naive 3D extension of Swin Transformer

Hierarchical window-based transformers, such as Swin Transformer [5], are widely used in generic vision due to their high efficiency, multiscale feature learning, scalability, and improved performance compared to 2D CNN backbones. Thus, it is a logical step to extend Swin Transformer-like architectures for 3D point cloud learning. The implementation of the window attention mechanism in 3D appears to be straightforward—partition the input 3D point cloud into non-overlapping 3D windows and compute self-attention on nonempty-voxel features within regular and shifted windows. However, as Lai et al. [8] observed in their ablation study, this naive extension does not lead to superior performance. We have identified two key issues that explain this unimproved performance: *memory complexity* and *signal irregularity*.

Memory complexity. The quadratic complexity of self-attention leads to a high memory cost in 3D. For a 3D window of size $M \times M \times M$, the average number of non-empty voxels within the window is approximately $\mathcal{O}(M^2)$; thus, the memory cost of executing vanilla self-attention within a window is approximately $\mathcal{O}(M^4)$. Depending on the size of a 3D scene, the number of windows N_{w} could be considerable. Therefore, the total memory cost $\mathcal{O}(N_{\mathrm{w}}M^4)$ in 3D could be much higher than its 2D counterpart, where the image size is usually low. This memory issue prevents the use of large windows and more Swin layers, making it difficult to design large models that can benefit from large data.

Signal irregularity. The locations of 3D points can be highly irregular; points can be found anywhere within their occupied voxel, while for 2D visual transformers, image pixels are regularly distributed at grid cell centers. Additionally, because points are usually equipped with other raw signals, such as RGB colors, if we consider both point positions and other pointwise signals as voxel signals, the signal irregularity, i.e., relative signal variation between any two voxels in a window, can be quite varied. Previous works [8] have addressed point irregularity only by using positional encoding in self-attention, but they are unaware of the variations of other signals. In Fig. 1, we illustrate sparse voxels in a window and signal irregularity on a 2D example.

3.2 Swin3D architecture

We designed our Swin3D backbone to address the issues mentioned above. It has a hierarchical structure similar to that of Swin Transformer and is composed of the following modules: *voxelization*, *initial feature embedding*, Swin3D *block*, and *downsample*. The *voxelization* module discretizes an input point cloud into a multiscale sparse voxel grid,

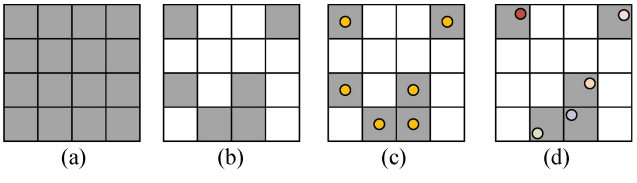


Fig. 1 A 2D illustration of sparse points in a 4×4 window. (a) A fully-occupied window. (b) A sparsely-occupied window, with white cells being empty. (c) Regularly-distributed sparse points in a window. (d) Sparse points irregularly distributed in the window, where different circle colors indicate the varying point-wise signal, such as the RGB color. For simplicity, only one point is drawn on non-empty cells.

the *initial feature embedding* module generates sparse voxel features at the finest voxel level for feature attention, the SWIN3D *block* performs memory-efficient self-attention on sparse voxel features within regular and shifted windows and addresses signal irregularity by *contextual relative signal encoding*, and the *downsample* module aggregates the sparse voxel features at the l-th to $(l+1)$-th levels. SWIN3D contains 5-stage SWIN3D blocks, each of which operates at different voxel resolution. It serves as a multiscale feature encoder for any input point cloud. By default, the voxel resolutions are 2, 4, 8, 16, and 32 cm, which is in line with the multi-resolution selection of most 3D CNN approaches and the Stratified Transformer [8]. It can be easily combined with task-specific decoders for a variety of 3D tasks. Figure 2 illustrates the architecture of our backbone. The details of each module are presented in Section 4.

4 Module design of Swin3D Transformer

4.1 Voxelization

Point cloud input. An input point cloud is typically associated with point-wise signals, such as point position, color, and normal. For a point $\boldsymbol{p}$, the concatenation of these signals is represented by $\boldsymbol{s_p} \in \mathbb{R}^m$. For any color or normal component, it is mapped to the range $[-1, 1]$. A common setting is $m = 6$, representing 3D point coordinates and RGB color.

Voxelization. We employ sparse voxels as point proxies in our backbone. A 5-level hierarchical sparse voxel grid is constructed from the input point cloud, as shown in Fig. 2 with a 2D example. By default, the voxel size at the finest level is set to 2 cm for indoor scenes. The voxel size is doubled when the voxel level is increased by one. Point information is stored in the voxels from the finest to coarsest level in the following manner:

- For the voxel $\boldsymbol{v}$ at the finest level, we randomly select one point within $\boldsymbol{v}$ and designate it as the *representative point* of $\boldsymbol{v}$, which is denoted by $\boldsymbol{r_v}$.
- For the voxel $\boldsymbol{v}$ at the $(l+1)$-th level, we first select all representative points from its child voxels. We then choose the representative point closest to the center of these points and make it the representative point of $\boldsymbol{v}$.

The voxelization step assigns unstructured points to structured sparse voxel grids, and the representative point is used to supply raw features to the initial feature embedding and provide contextual information for computing SWIN3D self-attention (see Section 4.3). For simplicity, we denote the signal at the representative point of voxel $\boldsymbol{v}$ as $\boldsymbol{s_v}$.

4.2 Initial feature embedding

Motivated by the observation that using a linear layer or MLP to project raw features to a high dimension does not yield good performance for Swin-like transformer architectures [8], we propose lifting the raw feature via sparse convolution. At the finest voxel level, we apply one layer of sparse convolution with a $3\times3\times3$ kernel, followed by batch normalization (BN) and a ReLU layer, to transform the input voxel

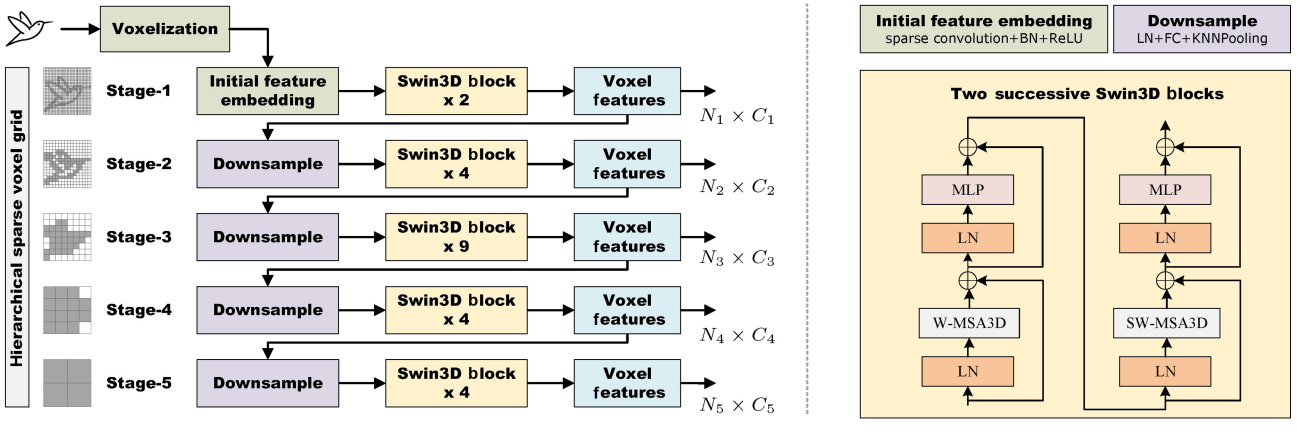


Fig. 2 Left: architecture of SWIN3D, which consists of five-stage transformer blocks that apply self-attention to sparse voxels within regular and shifted windows at various levels of a hierarchical sparse grid. The grids on the left illustrate sparse grids in 2D, with gray cells representing non-empty voxels. N_i denotes the number of sparse voxels at the i-th level, and C_i is the feature channel dimension. Right: detailed operations of each module.

feature to $\mathbb{R}^{C_1}$. The input feature on voxel $\boldsymbol{v}$ is set as the concatenation of the positional offset: $\boldsymbol{r_v} - \boldsymbol{c_v}$ and other point signals stored at $\boldsymbol{r_v}$, where $\boldsymbol{c_v}$ is the center of $\boldsymbol{v}$. We do not use the absolute point position because our goal is to learn local priors via convolution. Compared to KPConv [46] utilized by Ref. [8], our initial feature embedding is much lighter and five times faster.

4.3 Swin3D block

Our Swin3D block is based on the Swin Transformer block design; it operates on both regular and shifted windows in 3D. The voxel grid is split into non-overlapping windows at level l, with a window size of $M \times M \times M$. The shifted window is created by shifting the regular window with an offset of $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$. We modified the standard multi-head self-attention to improve memory efficiency and address signal irregularity, as described below. The number of heads is denoted by N_H.

4.3.1 Memory-efficient self-attention scheme

Vanilla self-attention. For a given input window with N non-empty voxels, denoted by $\{\boldsymbol{v}_i\}_{i=1}^N$, the network features of these voxels are represented by $\{\boldsymbol{f}_i\}_{i=1}^N$. A vanilla multi-head self-attention is then applied to the input voxel features, which is a weighted sum of the projected voxel features:

$$\boldsymbol{f}^{\star}_{i,h} = \sum_{j=1}^{N} \alpha_{ij,h} \cdot \boldsymbol{f}_j \boldsymbol{W}_{V,h}, \qquad i = 1, \cdots, N \tag{1}$$

where $\{\boldsymbol{f}^{\star}_{i,h}\}_{i=1}^N$ are the output feature vectors at the h-th head. The weight coefficient $\alpha_{ij,h}$ is the SoftMax version of $e_{ij,h}$, i.e., $\exp(e_{ij,h}) / \sum_{k=1}^{N} \exp(e_{ik,h})$, and $e_{ij,h}$ is in a scaled dot-product attention form:

$$e_{ij,h} = \frac{(\boldsymbol{f}_i \boldsymbol{W}_{Q,h})(\boldsymbol{f}_j \boldsymbol{W}_{K,h})^{\mathrm{T}}}{\sqrt{d}} \tag{2}$$

Here, $\boldsymbol{W}_{Q,h}, \boldsymbol{W}_{K,h}, \boldsymbol{W}_{V,h}$ are linear projection matrices for *Query*, *Key*, and *Value* computation, respectively, and d is the channel number of the h-th head.

Memory-efficient self-attention. The common multi-head self-attention implementation requires two passes to calculate $\{\alpha_{ij,h}\}$. The first pass computes all $\exp(e_{ij})$, and the second pass accumulates their sums, i.e., $\sum_{k=1}^{N} \exp(e_{ik})$. This necessitates storing all α_{ij} for Eq. (1), resulting in $\mathcal{O}(N^2 \times N_H)$ memory complexity. For Swin-like Transformer architectures, this complexity is $\mathcal{O}(N^2 \times N_H \times N_{\mathrm{w}})$, where N_{w} is the number of windows. In the case of a 3D Swin Transformer with window size $M \times M \times M$, $N = \mathcal{O}(M^2)$ and N_{w} can be very large if the scale of the input point cloud varies greatly. This high memory cost in 3D limits the use of large windows and deeper networks; however, this is not a significant issue in the 2D Swin Transformer because N_{w} is at least one order smaller than its 3D counterpart, and the input image usually has a fixed size.

We observe that Eq. (1) can be rewritten in Eq. (3):

$$\boldsymbol{f}^{\star}_{i,h} = \frac{\sum_{j=1}^{N} (\exp(e_{ij,h}) \boldsymbol{f}_j \boldsymbol{W}_{V,h})}{\sum_{j=1}^{N} \exp(e_{ij,h})}, \qquad i = 1, \cdots, N \tag{3}$$

which allows us to postpone the SoftMax normalization and avoid constructing and storing $\{\alpha_{ij,h}\}$ explicitly. Therefore, we modify the second pass by calculating the denominator and numerator of Eq. (3) simultaneously. During computation, $\{\exp(e_{ij,h})\}_{j=1}^N$ for $\boldsymbol{f}^{\star}_{i,h}$ are calculated on the fly, without storage. This eliminates the quadratic complexity of the memory cost of $\{\alpha_{ij,h}\}$. For gradient propagation, each $\exp(e_{ij,h})$ is computed twice during the training stage. However, this additional computation cost is negligible as the self-attention computation is a memory-intensive operation, not a computation-intensive operation. Thus, our memory reduction does not slow down the computation and could reduce the execution latency as well (see evaluation in Section 6).

4.3.2 Contextual relative signal encoding

Swin Transformer utilizes *relative position bias* [47] to enhance the performance of its backbone. Wu et al. [7] proposed a novel contextual mode for relative positional encoding, referred to as *contextual relative position encoding* (cRPE), which adds relative position encoding to both queries and keys. Lai et al. [8] used cRPE to capture fine-grained position information in 3D self-attention computation. In our work, we extended cRPE to all kinds of signals, not just point positions, as other signals such as RGB color also display high variation within a window, and these variations should be captured by self-attention. We refer to this generalized version as *contextual relative signal encoding* or cRSE. The multi-head self-attention with cRSE is formulated as follows.

We first modify e_{ij} to include the difference between the voxel signals $\Delta \mathbf{s}_{ij} := \boldsymbol{s}_{\boldsymbol{v}_i} - \boldsymbol{s}_{\boldsymbol{v}_j}$:

$$e_{ij,h} = \frac{(\boldsymbol{f}_i \boldsymbol{W}_{Q,h})(\boldsymbol{f}_j \boldsymbol{W}_{K,h})^{\mathrm{T}} + b_{ij,h}}{\sqrt{d}} \tag{4}$$

Here, $b_{ij,h}$ is the contextual signal encoding:

$$b_{ij,h} = (\boldsymbol{f}_i \boldsymbol{W}_{Q,h})(\boldsymbol{t}_{Q,h}(\Delta \boldsymbol{s}_{ij}))^{\mathrm{T}} + (\boldsymbol{f}_j \boldsymbol{W}_{K,h})(\boldsymbol{t}_{K,h}(\Delta \boldsymbol{s}_{ij}))^{\mathrm{T}} \quad (5)$$

The output of self-attention is revised to

$$\boldsymbol{f}^{\star}_{i,h} = \frac{\sum_{j=1}^{N} \exp(e_{ij,h})(\boldsymbol{f}_j \boldsymbol{W}_{V,h} + \boldsymbol{t}_{V,h}(\Delta \boldsymbol{s}_{ij}))}{\sum_{j=1}^{N} \exp(e_{ij,h})} \quad (6)$$

where $\boldsymbol{t}_{K,h}$, $\boldsymbol{t}_{Q,h}$, and $\boldsymbol{t}_{V,h}$ are trainable functions that map the signal differences to $\mathbb{R}^d$.

To make these trainable functions lightweight, we follow Refs. [8, 47] to quantify signal differences by a set of learnable look-up tables: $\{t_1^{Q,h}, \cdots, t_m^{Q,h}\}$, $\{t_1^{K,h}, \cdots, t_m^{K,h}\}$, and $\{t_1^{V,h}, \cdots, t_m^{V,h}\}$, each with a fixed length L_i. For an input vector Δ, its table indices are determined by

$$I_l(\Delta) = \lfloor \frac{(\Delta[l] - \texttt{minquat}[l])L_l}{\texttt{quat}[l]} \rfloor \quad (7)$$

where $\Delta[l]$ is the l-th component of Δ, and $\texttt{quat}[l]$ and $\texttt{minquat}[l]$ are the quantification range and lower bound of signal difference for the l-th signal, respectively. For common signal types, $\texttt{quat}[l]$ and $\texttt{minquat}[l]$ are defined as follows:

- If the l-th signal corresponds to point position, $\texttt{quat}[l] = 2h$ and $\texttt{minquat}[l] = -h$, where h is the physical height of the cubic window.
- If the l-th signal corresponds to one of the RGB components, $\texttt{quat}[l] = 2$ and $\texttt{minquat}[l] = -1$.
- If the l-th signal corresponds to one of the point normal components, $\texttt{quat}[l] = 2$ and $\texttt{minquat}[l] = -1$.

With the look-up tables and index functions, we have $\boldsymbol{t}_{Q,h}(\Delta) = \sum_{l=1}^{m} t_{l,h}^{Q}[I_l(\Delta)]$, $\boldsymbol{t}_{K,h}(\Delta) = \sum_{l=1}^{m} t_{l,h}^{K}[I_l(\Delta)]$, $\boldsymbol{t}_{V,h}(\Delta) = \sum_{l=1}^{m} t_{l,h}^{V}[I_l(\Delta)]$. The use of look-up tables for cRSE introduces additional $3\sum_{i=1}^{m} L_i \times N_H$ parameters. By default, we set $L_l = 4$ if the l-th signal corresponds to point position and $L_l = 16$ if the l-th signal corresponds to color or normal components. The effectiveness of cRSE is evaluated extensively in Section 6.4.

4.3.3 Transformer block

Our Swin3D transformer block, denoted by `W-MSA3D` (on regular windows) and `SW-MSA3D` (on shifted windows), is composed of the revised multi-head self-attention along with other transformer components, such as LayerNorm and MLP layer, as illustrated in Fig. 2(right).

4.3.4 Efficient implementation

We revised the self-attention module of Stratified Transformer [8] to improve its efficiency. The revision includes optimizing kernel scheduling, enabling half-precision, and reducing accesses of atomic operations; we call this revision our vanilla implementation. We further extend cRPE to cRSE and integrate our memory-efficient design. More details on the implementation are presented in the Appendix.

4.4 Downsample

The voxel features at the l-th level are downsampled to the $(l+1)$-th level by first lifting all sparse voxel features to $\mathbb{R}^{C_{l+1}}$ through a LayerNorm and FC layer. Then, for any sparse voxel $\boldsymbol{v}$ at the $(l+1)$-th level, the features of its k-nearest voxels at the l-th level are maxpooled and assigned to $\boldsymbol{v}$. This downsampling strategy is referred to as `KNNPooling`, where k is set to 16 by default.

5 Swin3D backbone pretraining

5.1 Backbone models

We created two versions of Swin3D: Swin3D-S and Swin3D-L. The window size for the first stage is set to $5 \times 5 \times 5$, and for the remaining stages, it is $7 \times 7 \times 7$. The layer numbers are $\{2, 4, 9, 4, 4\}$ with downsample strides $\{3, 2, 2, 2\}$. The feature dimensions (#FD) and head numbers (#HD) at each stage are:

- Swin3D-S: #FD = $\{48, 96, 192, 384, 384\}$, #HD = $\{6, 6, 12, 24, 24\}$;
- Swin3D-L: #FD = $\{80, 160, 320, 640, 640\}$, #HD = $\{10, 10, 20, 40, 40\}$.

By default, the input point signal contains positional and color information only. When the input signal contains point normals and cRSE uses normal signals, we use Swin3D$_n$-S and Swin3D$_n$-L to denote our respective backbone models. The efficiency and support of our backbone design for large models is determined by the following assessments.

Model efficiency. We evaluated the memory and computation efficiency of our backbone model by computing the average memory usage and computational time based on 70 point clouds from ScanNet [10] (see Table 1). Our Swin3D-S model is referred to as *Our-Efficient*. We also compared it with (1) Stratified Transformer [8], which also adapted the Swin Transformer for 3D point clouds; (2) an improved version of Stratified Transformer based on our revision described in Section 4.3.4 and contextual relative signal encoding, excluding our

Table 1 We compare the efficiency of self-attention by reporting the average statistics of point number (as with the number of non-empty voxels), execution time (in ms), and memory footprint (in megabytes) for a single forward-backward iteration. The implementation of Ref. [8] only works with double-precision

		Impl. of Ref. [8]		Our-Vanilla		Our-Efficient	
Block	#Pts	Time	Mem.	Time	Mem.	Time	Mem.
Stage-1	109.48k	487.7	1380.1	25.7	555.4	**20.3**	**268.68**
Stage-2	15.05k	122.9	467.4	16.0	180.4	**14.1**	**95.4**
Stage-3	4.01k	56.5	233.6	9.3	104.8	**7.1**	**47.9**
Stage-4	1.01k	29.0	120.5	6.5	60.0	**4.8**	**24.4**
Stage-5	0.25k	9.6	36.7	4.3	21.2	**2.4**	**8.7**

memory-efficient self-attention scheme, referred to as *Our-Vanilla* implementation. All models had the same number of transformer blocks and the same amount of transformer parameters. Compared with Ref. [8], our vanilla implementation already significantly decreased the computational and memory cost, and our memory-efficient self-attention further halved memory consumption and sped up the computation. The memory and computation efficiency of our design allowed us to explore large Swin3D models to take advantage of large datasets.

Support for large models. We assessed the scalability of our design by examining its GPU memory consumption when applied to large models. We used Swin3D-S as the base model and tested its GPU memory utilization on ScanNet data, increasing the width and depth of the network, number of heads, and window size. Additionally, we compared the results to the vanilla version of our model, which does not employ memory-efficient self-attention. The experimental setup was as follows.

- Wider networks. We created five models based on Swin3D-S by increasing the feature channels with five different ratios: $\frac{2}{3}, 1, \frac{4}{3}, \frac{5}{3}, 2$, The model with ratio $\frac{5}{3}$ is equivalent to Swin3D-L.
- Deeper networks. We created five models based on Swin3D-S by changing the number of Swin block layers at Stage-3 to 6, 9, 12, 15, and 18. Here, the default Swin3D-S uses 9 layers.
- Large head number. We created five models based on Swin3D-S by increasing the number of heads by four different factors: 1, 2, 4, and 8.
- Large window size. We tested different window sizes: $5^3, 7^3, 9^3, 11^3, 13^3, 15^3$.

We present the GPU memory consumption of the test in Fig. 3. We observe that as the channel and block numbers are proportional to memory storage, the rate of memory usage is almost constant, and the models with our memory-efficient self-attention save approximately 25% memory compared to the models with the vanilla-version self-attention. The quadratic memory saving by our memory-efficient self-attention is clearly visible when increasing the head number or window size compared to the models with the vanilla-version self-attention. In conclusion, the reduction of GPU memory enabled by our memory-efficient self-attention makes our backbone architecture suitable for designing large backbones.

From the test, it is evident that our backbone design can accommodate models with greater capacity than Swin3D-L. However, we found experimentally that more large models tend to overfit our pre-training dataset, so we did not explore them further in this study.

5.2 Backbone pretraining

Pretraining data preparation. We opted to pretrain our backbones with the Structured3D dataset [9], which contains 21,835 rooms in 3500 synthetic scenes and is equipped with a variety of high-quality 3D objects and layouts. This dataset is one order of magnitude larger than other real indoor datasets, such as MatterportLayout [48] (2295 rooms) and ScanNet [10] (1613 rooms). We used the provided RGBD and panoramic images to generate the training data as follows.

We acquired all RGBD and panoramic images

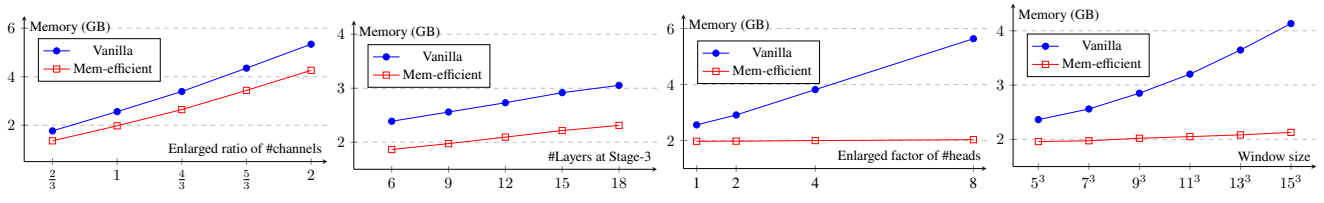


Fig. 3 Evaluation of support for large models. From left to right: GPU memory consumption of wider networks; GPU memory consumption of deeper networks; GPU memory consumption with respect to head numbers; GPU memory consumption with respect to window size. The GPU memory was measured on a forward-and-backward iteration, averaged on all ScanNet data.

associated with the room using the official room description and projected RGB and semantic labels of all images into 3D points based on their intrinsic and extrinsic camera parameters. To reduce the large number of points, we divided the space into cubes of 1 cm^3 and kept only one point in each cube. The remaining points form the point cloud of the room. We also estimated point normals from depth maps for training SWIN3D$_n$ models.

Backbone pretraining. We selected 3D semantic segmentation as our pre-training task, with 25 semantic labels. Four original segmentation labels (counter, box, toilet, and bathtub) were excluded due to their rarity. Table 2 shows the segmentation labels used in pretraining, as well as those in the downstream tasks of ScanNet segmentation and detection and S3DIS segmentation and detection. There is some overlap between our pre-training data and datasets of the downstream tasks, and some labels used in the downstream tasks are not present in our pre-training data.

We followed the original data split: 18,349 rooms for training, 1776 rooms for validation, and 1691 rooms for testing. We used SWIN3D as the encoder and a simple decoder to output semantic labels of input points. The decoder is similar to the UNet decoder. We upsampled the features from the coarsest level using interpolation, followed by a Linear Layer to match the dimensions. We then added fine-level features from the encoder using skip-connection. The purpose of the simple decoder is to make the backbone the main factor in feature learning. The network was trained with 100 epochs, a batch size of 12, and augmented input data through random cropping and rotation. We used the AdamW optimizer with a Cosine learning rate scheduler. Table 3 reports the network parameters, amortized inference latency measured in the Structured3D validation set, and average and peak memory footprint of network training in a subset (600 samples) of the training dataset. Training SWIN3D-S and SWIN3D-L took 488 and 703 GPU hours with NVidia V100 GPUs, respectively. We also pretrained SWIN3D$_n$-S and SWIN3D$_n$-L and used them only for the ScanNet segmentation task. During the network training phase, we employed the data cropping strategy of Ref. [8] to randomly crop a portion from the input scene for training, with a maximum of 120,000 sparse voxels. Additionally, we used the data augmentation technique of Ref. [8] for network training.

Table 3 Model parameters, inference latency, and memory footprint evaluated on Structured3D segmentation

Model	Params (M)	Latency (ms)	Memory (Avg./peak) (GB)
SWIN3D-S	23.57	377.98	2.24/3.69
SWIN3D-L	60.75	554.58	4.11/6.73

Fine-tuning for downstream tasks. We employed our pretrained backbone as a multi-resolution feature encoder and attached it to a task-specific decoder for downstream tasks. The pretrained network weights and look-up tables were loaded for initialization, while the decoder weight was randomly initialized. Our experimental results on downstream tasks are presented in Section 6.

6 Experimental analysis

We conducted experiments to assess the scalability, generality, and effectiveness of our backbone design for typical indoor scene understanding tasks. This section is structured as follows: We first present the experimental setup of downstream tasks in Section 6.1, and then evaluate our model's capability through a series of experiments, comparisons, and ablation studies in Sections 6.2–6.4.

Table 2 We provide a list of semantic labels for our Structured3D pretraining dataset, as well as the datasets of the downstream tasks. The list is denoted by "#C", which represents the number of segmentation labels. Additionally, "Seg." and "Det." indicate the tasks of semantic segmentation and 3D detection, respectively

Dataset	Task	#C	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	bookcase	picture	counter	desk	shelves	curtain	dresser	pillow	mirror	ceiling	refrigerator	television	shower curtain	nightstand	toilet	sink	lamp	bathtub	garbagebin	board	beam	column	clutter	otherstructure	otherfurniture	otherprop.
Structured3D	Seg.	25	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓		✓	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓							✓	✓	✓
ScanNet	Seg.	20	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓					✓		✓		✓	✓		✓						✓		
	Det.	18			✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓					✓		✓		✓	✓		✓	✓							
S3DIS	Seg.	13	✓	✓			✓	✓	✓	✓	✓		✓									✓										✓	✓	✓	✓			
	Det.	5					✓	✓	✓				✓																			✓						

6.1 Experimental settings for downstream tasks

6.1.1 Semantic segmentation

Datasets. ScanNet [10] contains 1613 indoor scans with 20 common semantic labels. We followed the original training/validation/test data split and reported the mean Intersection over Union (IoU) on the validation and test datasets. S3DIS contains 272 rooms from 6 large-scale areas. One area was selected as the validation set, and the other areas were used for training. We report the mean IoU on Area-5 and 6-fold cross-validation results.

Networks. To adapt our pretrained encoders to the segmentation task, we revised the decoder structure used for our pretraining by adding a Swin3D block at each level after the skip-connection to improve the decoder capability. The minimum voxel sizes for ScanNet and S3DIS are 2 and 5 cm, respectively. As ScanNet point clouds possess point normal information and many existing methods have utilized it, we used Swin3D$_n$-S and Swin3D$_n$-L for this task.

Training. During the training stage, we augmented the data through random cropping, scaling, and rotation. We fine-tuned our models (Swin3D-S and Swin3D-L) for 600 epochs for both ScanNet and S3DIS, with a batch size of 12. We used the AdamW optimizer with a Cosine learning rate scheduler with WarmUp, and the maximum learning rate was set to 0.006.

6.1.2 3D detection

Datasets. The ScanNet [10] dataset contains labels for 18 objects, with 1201 scans used for training and 312 for validation. The S3DIS [11] dataset contains instance labels of 7 categories, including floor and ceiling. Following FCAF3D [49], we excluded these two categories and trained and validated our models on the remaining five categories. AP@0.25 and AP@0.5 are the evaluation metrics employed in our experiment.

Networks. We replaced the encoders of two state-of-the-art 3D detection networks (FCAF3D [49] and CAGroup3D [50]) with our pretrained Swin3D to demonstrate the advantages of using our pretrained backbones for 3D detection. Specifically, for FCAF3D, we replaced the Sparse-Convolution-based ResNet in FCAF3D with our pretrained Swin3D and kept the other modules unchanged, which we named Swin3D-S+FCAF3D and Swin3D-L+FCAF3D, respectively; for CAGroup3D, we replaced its feature extractor (BiResNet) with our pretrained encoder with upsample layers and kept the other modules for proposal generation and the detection head unchanged, which we named Swin3D-S+CAGroup3D and Swin3D-L+CAGroup3D, respectively.

Training. For network training, we set the finest voxel size to 2 cm for both ScanNet and S3DIS and used the same data augmentation as for CAGroup3D and FCAF3D. We adjusted our Swin3D-S+CAGroup3D and Swin3D-S+FCAF3D models for ScanNet 3D detection by training them for 200 epochs with a batch size of 12. Similarly, for S3DIS 3D detection, we fine-tuned our Swin3D-L+FCAF3D model for 200 epochs with a batch size of 8. We used the AdamW optimizer with a step-wise learning rate scheduler, and the maximum learning rate was set to 0.001.

6.2 Performance on downstream tasks

6.2.1 Semantic segmentation

We quantitatively evaluated our pretrained models with fine-tuning on the semantic segmentation task and compared them to state-of-the-art methods, as shown in Table 4. Following existing approaches, such as Stratified Transformer [8], PointransformerV2 [29], O-CNN [55], and OctFormer [53], which perform test-time augmentation in evaluating segmentation performance, we voted our segmentation results via 12 rotation augmentations. For reference, we also report the unvoted results of several methods, including ours. The unvoted results are reported in parentheses. In the following, we analyze the results in detail.

Comparison with supervised methods. Among the existing supervised methods, PointTransformerV2 [29], O-CNN [55], and OctFormer [53] utilize point normals on ScanNet segmentation. Our Swin3D-S outperforms the best supervised method by +0.4 mIoU on ScanNet val, +0.8 mIoU on S3DIS Area5, and +0.6 mIoU on S3DIS 6-fold. With greater capacity, our Swin3D-L surpasses all compared supervised methods with +1.0 mIoU on ScanNet val, +2.3 mIoU on S3DIS Area5, and +2.2 mIoU on S3DIS 6-fold. With point normal inputs for pre-training and fine-tuning, our Swin3D$_n$-L further enhances performance on ScanNet val by +0.8 mIoU. We also present the results of Swin3D$_n$-S on

Table 4 Quantitative evaluation on semantic segmentation. The methods in the upper part of the table are supervised methods, while those in the lower part are based on pre-training. We present the highest scores achieved by the compared approaches. We use a star symbol * to signify our SWIN3D without any pre-training and SWIN3D$_n$ to denote our SWIN3D (pre)trained with point normal data. On the ScanNet benchmark (test dataset), we ensembled the results of three trained models by voting the prediction on over-segmented meshes, similar to Mix3D. WindowNorm [54] is a concurrent and unpublished work. As some existing works do not perform all the tests, we use a hyphen symbol (—) to indicate this. For S3DIS segmentation, previous methods did not make use of point normal information, and we also chose not to use it in our experiments. For ScanNet segmentation on the test set, we only report the performance of SWIN3D$_n$-L due to the restrictions on the number of submissions to the ScanNet benchmark website

Method	ScanNet segmentation		S3DIS segmentation	
	Val mIoU (%)	Test mIoU (%)	Area5 mIoU (%)	6-fold mIoU (%)
LargeKernel3D [51]	73.5	73.9	—	—
Mix3D [52]	73.6	**78.1**	—	—
Stratified Transformer [8]	74.3(73.1)	74.7	72.0	—
PointTransformerV2 [29]	75.4(74.4)	75.2	71.6	—
OctFormer [53]	75.7(74.5)	76.6		—
WindowNorm [54]	—	—	72.2	**77.6**
SWIN3D-S*	75.5(74.5)	—	**72.5**	76.9
SWIN3D$_n$-S*	**76.4**(75.2)	—	—	—
SWIN3D-L*	73.9(74.8)	—	69.6	74.9
SWIN3D$_n$-L*	74.2(75.2)	—	—	—
DepthContrast [34]	71.2	—	70.6	—
SceneContext [33]	73.8	—	72.2	—
PointContrast [31]	74.1	—	70.9	—
MaskContrast [40]	75.5(74.4)	—	—	—
SWIN3D-S	76.1(75.0)	—	73.0	78.2
SWIN3D$_n$-S	76.8(75.7)	—	—	—
SWIN3D-L	76.7(75.7)	—	**74.5**	**79.8**
SWIN3D$_n$-L	**77.5**(76.4)	77.9	—	—

ScanNet, which is +0.7 mIoU better than SWIN3D-S.

Comparison with pretraining methods. We compared our approach to existing unsupervised pretraining methods, such as PointContrast [31], SceneContext [33], DepthContrast [34], and MaskContrast [40], which use multi-view data from ScanNet for pretraining and Minkowski U-Net [56] as the encoder architecture. MaskConstrast also combines the ArkitScenes dataset [57] with ScanNet for pretraining. As can be seen in Table 4 (lower section), these unsupervised pretrained methods have lower performance than state-of-the-art supervised methods, with large gaps to our approach. One may wonder if unsupervised pretrained methods can take advantage of large datasets. Our initial tests (see Section 6.3) indicate that the advantages are restricted, probably because the convolutional encoder structure is not able to effectively extract data priors compared to the transformer structure.

Non-pretrained Swin3D. We also evaluated our backbone structure without pretraining, i.e., training SWIN3D from scratch for downstream tasks. We also trained SWIN3D-S from scratch for comparison (600 epochs for ScanNet and 3000 epochs for S3DIS), denoted by SWIN3D-S* and SWIN3D$_n$-S*, respectively. SWIN3D-S* has a comparable performance to existing works and is only inferior to WindowNorm [54] on S3DIS 6-fold. SWIN3D$_n$-S* can further improve SWIN3D-S* due to the use of point normals. Despite their good performance, they are consistently inferior to their pre-trained versions, demonstrating the effectiveness of pretraining.

We noticed that SWIN3D-L trained from scratch does not produce excellent results. SWIN3D-L* and SWIN3D$_n$-L* only achieved 73.9 and 74.2 mIoU on ScanNet segmentation (val), respectively. This is due to the overfitting caused by the large model size in comparison to the size of the training data (ScanNet or S3DIS). This is in agreement with previous findings in the NLP and vision fields (e.g., Bert, SwinTransformerV2) that large transformer models require a considerable amount of data for pretraining to show their advantages.

Additional results. In Table 5 and Table 6, we

Table 5 Category-wise segmentation results evaluated on ScanNet validation set

Method	mIoU	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	refri.	shower cur.	toilet	sink	bathtub	other
Stratified Transformer [8]	74.3	86.2	95.1	66.4	80.9	90.0	82.9	76.6	69.0	71.7	82.2	30.1	66.0	71.1	75.8	66.5	68.6	94.6	67.6	88.3	56.3
PointTransformerV2 [29]	75.4	86.1	95.4	67.4	**81.9**	91.9	**86.5**	77.5	68.8	68.7	84.5	34.1	66.7	71.1	78.7	69.2	71.2	94.5	65.6	89.1	60.5
OctFormer [53]	75.7	86.7	**95.6**	70.4	80.8	92.0	85.1	77.4	66.2	65.8	82.9	30.5	**71.3**	70.4	79.4	62.6	**78.4**	95.1	**71.3**	88.3	**64.4**
Swin3D$_s$-S	76.8	**87.6**	95.1	67.0	79.3	91.6	83.7	78.2	69.6	71.0	86.8	**39.9**	68.1	**73.6**	80.2	72.6	75.8	**95.6**	**71.2**	**89.7**	60.2
Swin3D$_s$-L	**77.5**	87.0	95.0	**73.2**	81.5	**92.1**	84.2	**79.1**	**70.6**	**72.8**	**87.3**	36.9	68.3	73.3	**80.9**	**73.4**	77.8	95.4	70.9	88.6	60.8

Table 6 6-fold S3DIS segmentation results. "An" means that the n-th area is the test data and other 5 areas are used for training. The reported number is mIoU

Method	6-fold mIoU	A1	A2	A3	A4	A5	A6
Swin3D-S	78.2	**82.2**	**64.2**	83.7	75.9	73.0	86.3
Swin3D-L	**79.8**	82.1	66.3	**85.9**	**76.7**	**73.4**	**87.2**

further provide semantic segmentation performance reports on all categories of ScanNet and S3DIS. For 6-fold cross validation of S3DIS, we fine-tuned our models for 600 epochs only. In the test phase, we voted our predictions across 12 rotation augmentations. For Area 5, we could achieve 74.5 mIoU (voted) by fine-tuning our model for 3000 epochs. For visualization, we show three segmentation results in Fig. 4.

6.2.2 3D detection

We quantitatively evaluated our pretrained models with fine-tuning on the 3D detection task and compared them to state-of-the-art methods, as shown in Tables 7 and 8. In the following, we analyze the results in detail.

Comparison with supervised methods. On ScanNet 3D detection (as shown in Table 7),

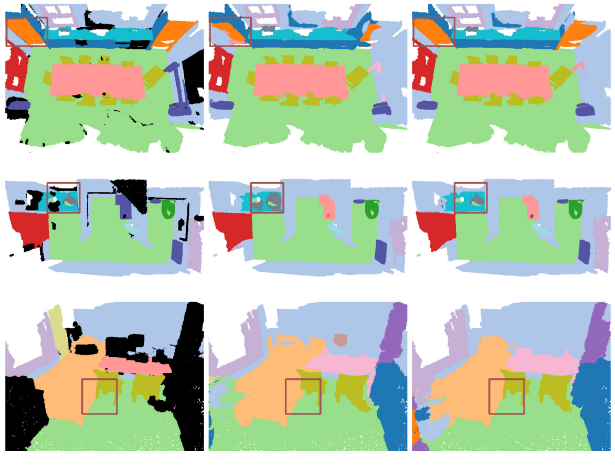


Fig. 4 Visual comparison of ScanNet segmentation. Left: ground truth segmentation labels (points colored in black are not labeled in the original dataset). Middle: Stratified Transformer's results. Right: Swin3D$_s$-L's results.

Table 7 Quantitative evaluation on 3D detection (ScanNet). The methods in the upper part of the table are supervised methods, while those in the lower part are based on pretraining

Method	mAP@0.25	mAP@0.5
RepSurf [58]	71.2	54.8
FCAF3D [49]	71.5	57.3
SoftGroup [59]	71.6	59.4
CAGroup3D [50]	**75.1**	**61.3**
Swin3D-S*+FCAF3D	72.1	56.8
Swin3D-S*+CAGoup3D	73.3	58.6
Point-M2AE [37]	50.1	33.2
PointContrast [31]	59.2	37.3
RandomRooms [60]	68.6	51.5
Swin3D-S+FCAF3D	74.2	59.5
Swin3D-L+FCAF3D	74.2	58.6
Swin3D-S+CAGoup3D	76.4	62.7
Swin3D-L+CAGoup3D	**76.4**	**63.2**

Table 8 Quantitative evaluation of 3D detection (S3DIS)

Method	mAP@0.25	mAP@0.5
GSDN [61]	47.8	25.1
FCAF3D [49]	**66.7**	**45.9**
Swin3D-S*+FCAF3D	64.6	40.7
Swin3D-S+FCAF3D	69.9	50.2
Swin3D-L+FCAF3D	72.1	54.0
Swin3D-S+FCAF3D(2-stage)	**75.4**	**58.6**

Swin3D-S+FCAF3D and Swin3D-L+FCAF3D both outperformed FCAF3D, with an increase of 2.7 points in mAP@0.25. Swin3D-S+CAGroup3D and Swin3D-L+CAGroup3D both surpassed CAGroup3D, with Swin3D-L+CAGroup3D achieving a new record on mAP@0.50. Table 8 shows the performance of our model on S3DIS. Compared to FCAF3D, our pretrained backbones significantly improve the performance, with Swin3D-S+FCAF3D increasing by 4.3 points and Swin3D-L+FCAF3D increasing by 8.1 points in mAP@0.5. The detection experiments lead us to the conclusion that semantic segmentation is an effective pretext task for 3D pretraining, as our pretrained backbone demonstrates remarkable generality to the 3D detection task.

Two-stage fine-tuning. We discovered through experimentation that S3DIS detection can be improved by a two-step fine-tuning process. Initially, we fine-tuned SWIN3D-S on ScanNet detection. Then, we loaded the fine-tuned SWIN3D and continued the fine-tuning on S3DIS training data, taking advantage of the prior knowledge gained from real data in the first step. The improved performance is reported in Table 8.

Comparison with pretrained methods. We compared our approach to three existing unsupervised pretraining methods: PointContrast [31], Point-M2AE [37], and RandomRooms [60]. PointContrast employed ScanNet as its pretraining dataset and Minkowski U-Net as its encoder, Point-M2AE utilized both ShapeNet [45] and ScanNet as its pretraining dataset and a transformer encoder, and RandomRooms adopted random samples of multiple objects from ShapeNet as its pretraining dataset and PointNet++ as its backbone. In line with the observations in semantic segmentation, these unsupervised pretraining methods were unable to compete with supervised methods and our approaches.

Non-pretrained Swin3D. Without pretraining, our model SWIN3D-S* that was trained from scratch showed average results, which were not as good as those of state-of-the-art methods. We hypothesize that a larger amount of training data is necessary to train our transformer from scratch. The higher performance of our pretrained SWIN3D also reflects this fact.

Additional results. In Tables 9 and 10, we present the category-wise performance report on ScanNet and S3DIS detection. Visualizations of some 3D detection results can be seen in Figs. 5 and 6.

6.3 Model scalability

Our backbone design takes advantage of large pretrained datasets, particularly for our large model SWIN3D-L. To evaluate the impact of the amount of pretrained data on the performance of downstream tasks with respect to backbone architectures, we conducted the following experiments.

Scalability of Swin3D. We pretrained our backbones with different amounts of training data: 10%, 33%, and 100%. We then fine-tuned them for downstream tasks. Figure 7 shows the segmentation

Table 10 Quantitative comparison of 3D detection (S3DIS)

Method	Pre.	mAP@0.5	table	chair	sofa	bookcase	board
GSDN [61]	✗	25.1	36.6	75.3	6.1	6.5	1.2
FCAF3D[49]	✗	45.9	45.4	88.3	70.1	19.5	5.6
SWIN3D-S	✓	50.2	52.8	**90.4**	78.8	**20.9**	7.9
SWIN3D-L	✓	**54.0**	**56.2**	90.3	**95.1**	19.6	**8.9**

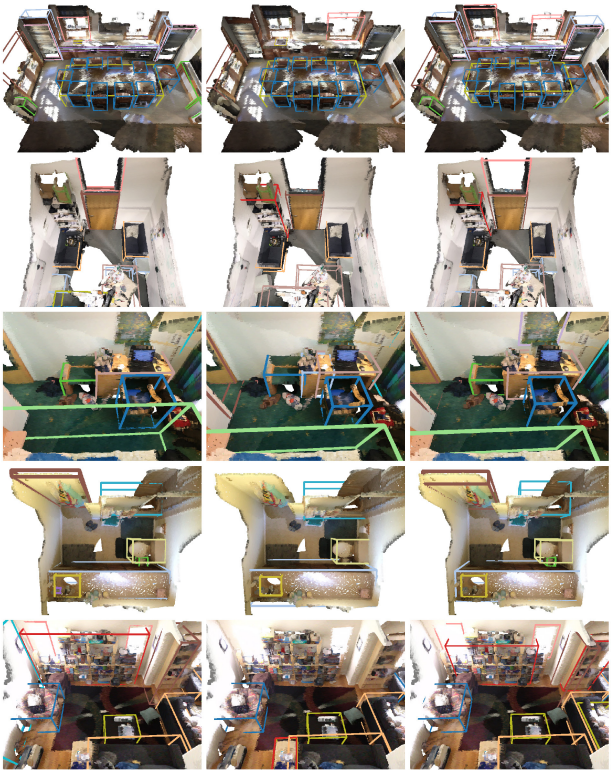


Fig. 5 Visual comparison of ScanNet 3D detection. Left: ground truth. Middle: FCAF3D's result. Right: SWIN3D-L+CAGroup3D's results. Note that the original CAGroup3D work did not release its checkpoint, so no visual results provided in this figure. The five examples (kitchen, storage area, bedroom, bathroom, and apartment) were chosen from the validation set.

Table 9 Quantitative comparison of 3D detection (ScanNet)

Method	Pre.	mAP@0.5	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	refri.	shower cur.	toilet	sink	bathtub	other
FCAF3D [49]	✗	57.3	35.8	81.5	89.8	85.0	62.0	44.1	30.7	58.4	17.9	31.3	53.4	44.2	46.8	**64.2**	91.6	52.6	84.5	57.1
CAGroup3D [50]	✗	61.3	41.4	82.8	90.8	**85.6**	64.9	54.3	37.3	64.1	31.4	41.1	**63.6**	44.4	57.0	49.3	98.2	**55.4**	82.4	58.8
SWIN3D-S+CAGroup3D	✓	62.7	45.7	82.7	**91.0**	79.5	**67.4**	**57.5**	**42.7**	**59.8**	36.8	40.4	62.6	**48.2**	**60.7**	59.8	**99.7**	55.1	77.6	60.7
SWIN3D-L+CAGroup3D	✓	**63.2**	**46.1**	**85.5**	**91.0**	81.1	64.5	52.9	42.4	57.8	**38.2**	**47.2**	63.4	46.0	59.3	61.7	98.3	54.2	**85.4**	**63.6**

Fig. 6 Visual comparison of S3DIS 3D detection. The examples were chosen from the validation set, including conference room, hallway, office, and storage. Left: ground-truth 3D bounding boxes. Middle: FCAF3D's detection results. Right: Swin3D-L+FCAF3D's detection results. Our Swin3D-L+FCAF3D generated more compact and accurate proposals of table, sofa, and board, which is consistent with the results shown in Table 10.

performance of our Swin3D-S and Swin3D-L models on the test dataset of Structured3D (left) and validation set of ScanNet segmentation (right). The plots demonstrate that (1) with more training data, the performance of both models increases significantly, and (2) Swin3D-L has more capacity to benefit from large data and performs better than Swin3D-S.

Scalability with the use of cRSE. We pre-trained our backbones with cRPE instead of our cRSE, referred to as Swin3D-S$^-$ and Swin3D-L$^-$, respectively. We then fine-tuned them on the ScanNet segmentation task, and the results are shown in the right side of Fig. 7. The plots clearly show that pretraining with our cRSE scheme exhibits better scalability than using cRPE. We also found that Swin3D-S outperforms Swin3D-L$^-$, which indicates that the capture of irregular signals is essential for backbone learning.

Scalability of other backbone architectures. We selected a few representative backbone architectures, such as SparseConv Net used by MaskContrast [40] and PointTransformerV2, and pre-trained them on the Structured3D dataset with a segmentation pretext task similar to ours. Additionally, we retrained MaskContrast in its unsupervised manner with the Structured3D dataset. All these pretrained backbones were finetuned on the ScanNet segmentation task. For fair comparison, all these backbones used both color and normal signals for input in the pre-training and fine-tuning stages. Their performance with respect to different amounts of pre-training data (10%, 33%, and 100%) is plotted in Fig. 8. We discovered that these backbones had limited advantages over large datasets and had a large performance gap compared to our Swin3D-S, which has similar or fewer network parameters than theirs. The results of this experiment demonstrate that simply increasing the amount of data used for pre-training does not necessarily lead to significant performance improvement for some existing 3D backbones.

6.4 Ablation study

Dataset for pretraining. We contrasted our backbones that were pre-trained on either ScanNet

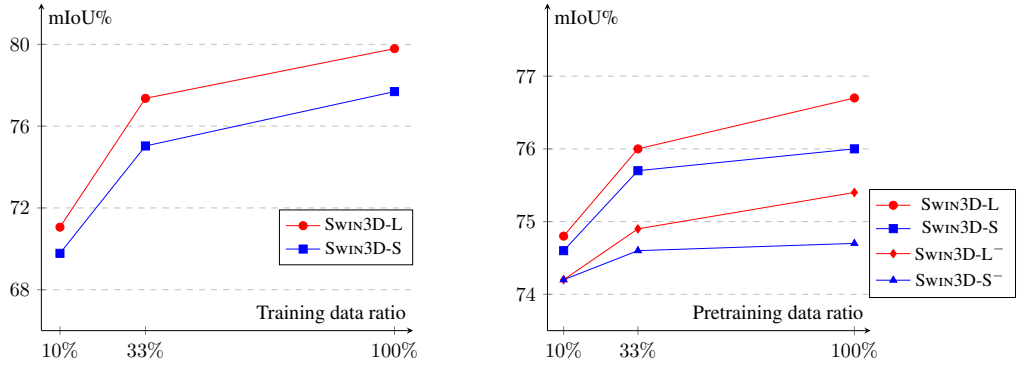


Fig. 7 Model scalability with respect to different ratios of data for pretraining. Left: test on Structured3D segmentation. Right: Swin3D-L on downstream ScanNet segmentation. Swin3D-L$^-$ and Swin3D-S$^-$ are the models pretrained and fine-tuned with cRPE instead of cRSE.

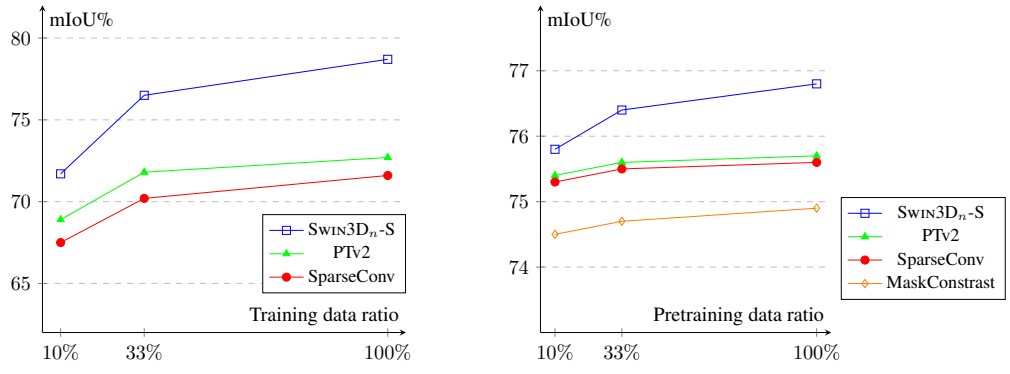


Fig. 8 We tested the scalability of different 3D backbones with respect to different ratios of data for pretraining. "PTv2" refers to PointTranformerV2. Left: results for Structured3D segmentation. Right: results for ScanNet segmentation (val).

or Structured3D for the downstream S3DIS segmentation task. Table 11 clearly demonstrates that our backbones gain more from the extensive synthetic Structured3D data than from the real but limited ScanNet data. This implies that the size of the training data is the most critical factor in 3D backbone training, despite the disparity between real and synthetic data.

Efficacy of cRSE. We conducted two experiments to assess the effectiveness of cRSE. In the first, we trained Swin3D-S from scratch in three different configurations: (1) using cRSE on point position only, similar to Ref. [8]; (2) using cRSE on point position and color; and (3) using cRSE on point position, color, and point normal. We also either excluded or included the normal information from the input point cloud for training and testing. The results in Table 12 demonstrate that taking into account color and normal variation via cRSE can significantly enhance the performance. In the second experiment, we used the pretrained Swin3D$_n$-L for ScanNet segmentation but did not load the pretrained look-up tables for color and/or normal components and trained these unloaded look-up tables from scratch during the network fine-tuning stage. The results in Table 13 show that the use of pretrained tables is essential for improved performance.

Table 11 S3DIS segmentation results using the backbones pretrained on different datasets

Backbone	Pre. dataset	Area5 mIoU (%)	6-fold mIoU (%)
Swin3D-S	ScanNet	71.8	76.7
Swin3D-S	Structured3D	**73.0**	**78.2**
Swin3D-L	ScanNet	68.9	73.6
Swin3D-L	Structured3D	**74.5**	**79.8**

Table 12 Efficacy evaluation of cRSE on ScanNet segmentation

Input point signal	cRSE	Val mIoU (%)
pos+color	pos	73.1
pos+color	pos+color	**74.5**
pos+color+normal	pos	73.1
pos+color+normal	pos+color	74.4
pos+color+normal	pos+color+normal	**75.2**

Table 13 Ablation study using the pretrained look-up tables on ScanNet segmentation

Backbone	Loaded look-up table	Val mIoU (%)
Swin3D$_n$-L	pos	75.4
Swin3D$_n$-L	pos+color	75.9
Swin3D$_n$-L	pos+color+normal	**76.4**

7 Conclusions

In this paper, we introduced a novel 3D backbone for indoor scene understanding—Swin3D, which demonstrated its scalability, generality, and superior performance through extensive experiments. In future research, we would like to strengthen Swin3D by incorporating self-supervised pretraining that combines real and synthetic 3D data at a larger scale. Additionally, it would be beneficial to combine pretrained image features with our backbones to enhance 3D learning, as point clouds are usually accompanied by high-resolution multiview images from 3D capture devices. It is essential to explore how our pre-trained backbone can be adjusted and used for online scene detection and segmentation [62–64] to make our architecture more practical.

Appendix Efficient self-attention implementation

Besides our memory-efficient self-attention approach,

we also enhanced our self-attention computation in the following ways.

Kernel scheduling. Designing an optimal scheduling strategy for CUDA kernels is essential to make the most of the memory bandwidth and computational capabilities of modern GPUs. Stratified Transformer [8] builds CUDA blocks whose number is equal to the number of points in the window and creates GPU threads whose number is the maximum number of points in the window. However, the varying number of points in each window makes it difficult for the CUDA compiler to optimize the execution speed. To tackle this issue, our kernel is designed to calculate the channel-wise contribution to the weight coefficients $c_{ij,h,d}$ (i, j are the indices of queries and keys respectively, h is the index of the head, and d is the index of the channel) per thread and binds blocks to a fixed number of threads (256 in our implementation). We use a local synchronized operation (*__shfl_down_sync*) to perform *reduce-sum* to obtain the coefficient weight $c_{ij,h}$. Thanks to the compacted memory access between neighboring threads and fixed and balanced operations per thread, our implementation results in higher memory bandwidth and efficient computation.

Half-precision support. We enable half-precision for all CUDA kernels that we implemented. Additionally, we reorganized the memory layout of look-up tables from dimension-last to channel-last. This allows a single thread to process two consecutive elements with a single 32-bit memory access for the query, key, and look-up tables.

Computation speedup. We combine the calculation of coefficient weights and cRPE/cRSE into a single kernel, thus decreasing the memory access of queries and keys and speeding up GPU performance.

Reduction of atomic operations. We employ atomic operations to compute the updated features and gradients of the lookup tables. However, half-precision atomic operators can be inefficient due to double writing/reading conflicts. To address this issue, we use shared memory to combine two consecutive 16-bit atomic operations into a single 32-bit atomic operation, thus reducing the number of atomic operations and speeding up GPU execution.

Availability of data and materials

Structured3D, ScanNet, and S3DIS are publicly released datasets. Our code and trained models are available at `https://github.com/microsoft/Swin3D`, under MIT license.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Funding

This research received no specific grant from any funding agency.

Author contributions

Yu-Qi Yang proposed and implemented the key idea, conducted the primary experiments, and contributed to paper writing. Yu-Xiao Guo contributed to network design, efficient implementation, and pretraining. Jian-Yu Xiong explored optimal network structures. Hao Pan and Peng-Shuai Wang assisted with the design of the network and experiments. Xin Tong and Banining Guo provided guidance and feedback on the main idea and results. Yang Liu led the project and contributed to the key concept, experimental design, and paper writing.

References

[1] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint* arXiv:2010.11929, 2020.

[2] Bao, H.; Dong, L.; Piao, S.; Wei, F. BEiT: BERT pre-training of image transformers. *arXiv preprint* arXiv:2106.08254, 2021.

[3] Devlin, J.; Chang, M. W.; Lee, K.; Toutanova, K.; Hulburd, E.; Liu, D.; Wang, M.; Catlin, A. G.; Lei, M.; Zhang, J.; et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the NAACL-HLT, 4171–4186, 2019.

[4] Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems, Article No. 159, 1877–1901, 2020.

[5] Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of

the IEEE/CVF International Conference on Computer Vision, 9992–10002, 2021.

[6] Liu, Z.; Hu, H.; Lin, Y.; Yao, Z.; Xie, Z.; Wei, Y.; Ning, J.; Cao, Y.; Zhang, Z.; Dong, L.; et al. Swin transformer V2: Scaling up capacity and resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11999–12009, 2022.

[7] Wu, K.; Peng, H.; Chen, M.; Fu, J.; Chao, H. Rethinking and improving relative position encoding for vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 10013–10021, 2021.

[8] Lai, X.; Liu, J.; Jiang, L.; Wang, L.; Zhao, H.; Liu, S.; Qi, X.; Jia, J. Stratified transformer for 3D point cloud segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8490–8499, 2022.

[9] Zheng, J.; Zhang, J.; Li, J.; Tang, R.; Gao, S.; Zhou, Z. Structured3D: A large photo-realistic dataset for structured 3D modeling. In: *Computer Vision – ECCV 2020. Lecture Notes in Computer Science, Vol. 12354*. Vedaldi, A.; Bischof, H.; Brox, T.; Frahm, J. M. Eds. Springer Cham, 519–535, 2020.

[10] Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; Nießner, M. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2432–2443, 2017.

[11] Armeni, I.; Sener, O.; Zamir, A. R.; Jiang, H.; Brilakis, I.; Fischer, M.; Savarese, S. 3D semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1534–1543, 2016.

[12] Khan, S.; Naseer, M.; Hayat, M.; Zamir, S. W.; Khan, F. S.; Shah, M. Transformers in vision: A survey. *ACM Computing Surveys* Vol. 54, No. 10, Article No. 200, 2022.

[13] Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 45, No. 1, 87–110, 2023.

[14] Guo, M. H.; Xu, T. X.; Liu, J. J.; Liu, Z. N.; Jiang, P. T.; Mu, T. J.; Zhang, S. H.; Martin, R. R.; Cheng, M. M.; Hu, S. M. Attention mechanisms in computer vision: A survey. *Computational Visual Media* Vol. 8, No. 3, 331–368, 2022.

[15] Han, Q.; Fan, Z.; Dai, Q.; Sun, L.; Cheng, M. M.; Liu, J.; Wang, J. Demystifying local vision transformer: Sparse connectivity, weight sharing, and dynamic weight. *arXiv preprint* arXiv:2106.04263, 2021.

[16] Dong, X.; Bao, J.; Chen, D.; Zhang, W.; Yu, N.; Yuan, L.; Chen, D.; Guo, B. CSWin transformer: A general vision transformer backbone with cross-shaped windows. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 12114–12124, 2022.

[17] Tu, Z.; Talebi, H.; Zhang, H.; Yang, F.; Milanfar, P.; Bovik, A.; Li, Y. MaxViT: multi-axis vision transformer. In: *Computer Vision – ECCV 2022. Lecture Notes in Computer Science, Vol. 13684*. Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; Hassner, T. Eds. Springer Cham, 459–479, 2022.

[18] Zhang, Q.; Xu, Y.; Zhang, J.; Tao, D. VSA: Learning varied-size window attention in vision transformers. In: *Computer Vision – ECCV 2022. Lecture Notes in Computer Science, Vol. 13685*. Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; Hassner, T. Eds. Springer Cham, 466–483, 2022.

[19] Wu, S.; Wu, T.; Tan, H.; Guo, G. Pale transformer: A general vision transformer backbone with pale-shaped attention. *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 36, No. 3, 2731–2739, 2022.

[20] Yang, J.; Li, C.; Zhang, P.; Dai, X.; Xiao, B.; Yuan, L.; Gao, J. Focal attention for long-range interactions in vision transformers. In: Proceeding of the 35th Conference on Neural Information Processing Systems, 30008–30022, 2021.

[21] Li, W.; Wang, X.; Xia, X.; Wu, J.; Li, J.; Xiao, X.; Zheng, M.; Wen, S. SepViT: Separable vision transformer. *arXiv preprint* arXiv:2203.15380, 2022.

[22] Wang, W.; Yao, L.; Chen, L.; Lin, B.; Cai, D.; He, X.; Liu, W. CrossFormer: A versatile vision transformer hinging on cross-scale attention. *arXiv preprint* arXiv:2108.00154, 2021.

[23] Chu, X.; Tian, Z.; Wang, Y.; Zhang, B.; Ren, H.; Wei, X.; Xia, H.; Shen, C. Twins: Revisiting the design of spatial attention in vision transformers. In: Proceedings of the 35th International Conference on Neural Information Processing Systems, Article No. 716, 9355–9366, 2021.

[24] Chen, Q.; Wu, Q.; Wang, J.; Hu, Q.; Hu, T.; Ding, E.; Cheng, J.; Wang, J. MixFormer: Mixing features across windows and dimensions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5239–5249, 2022.

[25] Yuan, Y.; Fu, R.; Huang, L.; Lin, W.; Zhang, C.; Chen, X.; Wang, J. HRFormer: High-resolution vision

transformer for dense predict. In: Proceedings of the 35th Conference on Neural Information Processing Systems, 7281–7293, 2021.

[26] Lahoud, J.; Cao, J.; Khan, F. S.; Cholakkal, H.; Anwer, R. M.; Khan, S.; Yang, M. H. 3D vision with transformers: A survey. *arXiv preprint* arXiv:2208.04309, 2022.

[27] Guo, M. H.; Cai, J. X.; Liu, Z. N.; Mu, T. J.; Martin, R. R.; Hu, S. M. PCT: Point cloud transformer. *Computational Visual Media* Vol. 7, No. 2, 187–199, 2021.

[28] Zhao, H.; Jiang, L.; Jia, J.; Torr, P.; Koltun, V. Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 16239–16248, 2021.

[29] Wu, X.; Lao, Y.; Jiang, L.; Liu, X.; Zhao, H. Point transformer V2: Grouped vector attention and partition-based pooling. In: Proceedings of the 36th International Conference on Neural Information Processing Systems, Article No. 2415, 33330–33342, 2024.

[30] Park, C.; Jeong, Y.; Cho, M.; Park, J. Fast point transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 16928–16937, 2022.

[31] Xie, S.; Gu, J.; Guo, D.; Qi, C. R.; Guibas, L.; Litany, O. PointContrast: unsupervised pre-training for 3D point cloud understanding. In: *Computer Vision – ECCV 2020. Lecture Notes in Computer Science, Vol. 12348*. Vedaldi, A.; Bischof, H.; Brox, T.; Frahm, J. M. Eds. Springer Cham, 574–591, 2020.

[32] Wang, P.-S.; Yang, Y.-Q.; Zou, Q.-F.; Wu, Z.; Liu, Y.; Tong, X. Unsupervised 3D learning for shape analysis via multiresolution instance discrimination. *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 35, No. 4, 2773–2781, 2021.

[33] Hou, J.; Graham, B.; Nießner, M.; Xie, S. Exploring data-efficient 3D scene understanding with contrastive scene contexts. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 15582–15592, 2021.

[34] Zhang, Z.; Girdhar, R.; Joulin, A.; Misra, I. Self-supervised pretraining of 3D features on any point-cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 10232–10243, 2021.

[35] He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; Girshick, R. Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 15979–15988, 2022.

[36] Liu, H.; Cai, M.; Lee, Y. J. Masked Discrimination for Self-supervised Learning on Point Clouds. In: *Computer Vision – ECCV 2022. Lecture Notes in Computer Science, Vol. 13662*. Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; Hassner, T. Eds. Springer Cham, 657–675, 2022.

[37] Zhang, R.; Guo, Z.; Gao, P.; Fang, R.; Zhao, B.; Wang, D.; Qiao, Y.; Li, H. Point-M2AE: Multi-scale masked autoencoders for hierarchical point cloud pre-training. In: Proceedings of the 36th International Conference on Neural Information Processing Systems, Article No. 1962, 27061–27074, 2024.

[38] Yu, X.; Tang, L.; Rao, Y.; Huang, T.; Zhou, J.; Lu, J. Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 19291–19300, 2022.

[39] Pang, Y.; Wang, W.; Tay, F. E. H.; Liu, W.; Tian, Y.; Yuan, L. Masked autoencoders for point cloud self-supervised learning. *arXiv preprint* arXiv:2203.06604, 2022.

[40] Wu, X.; Wen, X.; Liu, X.; Zhao, H. Masked scene contrast: A scalable framework for unsupervised 3D representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9415–9424, 2023.

[41] Wang, Z.; Yu, X.; Rao, Y.; Zhou, J.; Lu, J. P2P: Tuning pre-trained image models for point cloud analysis with point-to-pixel prompting. In: Proceedings of the 36th International Conference on Neural Information Processing Systems, Article No. 1046, 14388–14402, 2024.

[42] Dong, R.; Qi, Z.; Zhang, L.; Zhang, J.; Sun, J.; Ge, Z.; Yi, L.; Ma, K. Autoencoders as cross-modal teachers: Can pretrained 2D image transformers help 3D representation learning? In: Proceedings of the 11th International Conference on Learning Representations, 2023.

[43] Huang, T.; Dong, B.; Yang, Y.; Huang, X.; Lau, R. W. H.; Ouyang, W.; Zuo, W. CLIP2Point: Transfer CLIP to point cloud classification with image-depth pre-training. *arXiv preprint* arXiv:2210.01055, 2022.

[44] Huang, X.; Huang, Z.; Li, S.; Qu, W.; He, T.; Hou, Y.; Zuo, Y.; Ouyang, W. EPCL: Frozen CLIP transformer is an efficient point cloud encoder. *arXiv preprint* arXiv:2212.04098, 2022.

[45] Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint* arXiv:1512.03012, 2015.

[46] Thomas, H.; Qi, C. R.; Deschaud, J. E.; Marcotegui, B.; Goulette, F.; Guibas, L. KPConv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 6410–6419, 2019.

[47] Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-attention with relative position representations. *arXiv preprint* arXiv:1803.02155, 2018.

[48] Zou, C.; Su, J. W.; Peng, C. H.; Colburn, A.; Shan, Q.; Wonka, P.; Chu, H. K.; Hoiem, D. Manhattan room layout reconstruction from a single 360° image: A comparative study of state-of-the-art methods. *International Journal of Computer Vision* Vol. 129, No. 5, 1410–1431, 2021.

[49] Rukhovich, D.; Vorontsova, A.; Konushin, A. FCAF3D: Fully convolutional anchor-free 3D object detection. In: *Computer Vision – ECCV 2022. Lecture Notes in Computer Science, Vol. 13670*. Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; Hassner, T. Eds. Springer Cham, 477–493, 2022.

[50] Wang, H.; Ding, L.; Dong, S.; Shi, S.; Li, A.; Li, J.; Li, Z.; Wang, L. CAGroup3D: Class-aware grouping for 3D object detection on point clouds. In: Proceedings of the 36th International Conference on Neural Information Processing Systems, Article No. 2173, 29975–29988, 2024.

[51] Chen, Y.; Liu, J.; Qi, X.; Zhang, X.; Sun, J.; Jia, J. LargeKernel3D: Scaling up kernels in 3D CNNs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 13488–13498, 2023.

[52] Nekrasov, A.; Schult, J.; Litany, O.; Leibe, B.; Engelmann, F. Mix3D: Out-of-context data augmentation for 3D scenes. In: Proceedings of the International Conference on 3D Vision, 116–125, 2021.

[53] Wang, P. S. OctFormer: Octree-based transformers for 3D point clouds. *ACM Transactions on Graphics* Vol. 42, No. 4, Article No. 155, 2023.

[54] Wang, Q.; Shi, S.; Li, J.; Jiang, W.; Zhang, X. Window normalization: Enhancing point cloud understanding by unifying inconsistent point densities. *arXiv preprint* arXiv:2212.02287, 2022.

[55] Wang, P. S.; Liu, Y.; Guo, Y. X.; Sun, C. Y.; Tong, X. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 72, 2017.

[56] Choy, C.; Gwak, J.; Savarese, S. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3070–3079, 2019.

[57] Baruch, G.; Chen, Z.; Dehghan, A.; Dimry, T.; Feigin, Y.; Fu, P.; Gebauer, T.; Joffe, B.; Kurz, D.; Schwartz, A.; et al. Arkitscenes: A diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data. *arXiv preprint* arXiv:2111.08897, 2021.

[58] Ran, H.; Liu, J.; Wang, C. Surface representation for point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 18920–18930, 2022.

[59] Vu, T.; Kim, K.; Luu, T. M.; Nguyen, T.; Yoo, C. D. SoftGroup for 3D instance segmentation on point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2698–2707, 2022.

[60] Rao, Y.; Liu, B.; Wei, Y.; Lu, J.; Hsieh, C. J.; Zhou, J. RandomRooms: Unsupervised pre-training from synthetic shapes and randomized layouts for 3D object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 3263–3272, 2021.

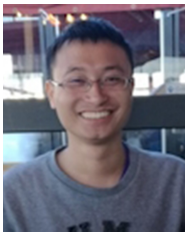
[61] Gwak, J.; Choy, C.; Savarese, S. Generative sparse detection networks for 3D single-shot object detection. In: *Computer Vision – ECCV 2020. Lecture Notes in Computer Science, Vol. 12349*. Vedaldi, A.; Bischof, H.; Brox, T.; Frahm, J. M. Eds. Springer Cham, 297–313, 2020.

[62] Zhang, J.; Zhu, C.; Zheng, L.; Xu, K. Fusion-aware point convolution for online semantic 3D scene segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4533–4542, 2020.

[63] Huang, S. S.; Ma, Z. Y.; Mu, T. J.; Fu, H.; Hu, S. M. Supervoxel convolution for online 3D semantic segmentation. *ACM Transactions on Graphics* Vol. 40, No. 3, Article No. 34, 2021.

[64] Cai, J. X.; Mu, T. J.; Lai, Y. K.; Hu, S. M. LinkNet: 2D–3D linked multi-modal network for online semantic segmentation of RGB-D videos. *Computers & Graphics* Vol. 98, 37–47, 2021.

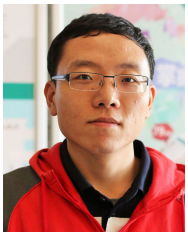
Yu-Qi Yang is a Ph.D. student at the Institute for Advanced Study, Tsinghua University. He received his bachelor degree in computer science from the University of Science and Technology of China in 2018. His research interests include computer graphics and 3D vision.



Yu-Xiao Guo is a senior researcher at Microsoft Research Asia. He received his Ph.D. degree from University of Electronic Science and Technology of China (UESTC) in 2018. His research interests include computer graphics and 3D vision.

Jian-Yu Xiong is a master student at Tsinghua Shenzhen International Graduate School, Tsinghua University. He received his bachelor degree in computer science from Harbin Institute of Technology in 2021. His areas of study are computer vision and image processing.

Yang Liu is a principal researcher at Microsoft Research Asia. He received his Ph.D. degree from the University of Hong Kong in 2008, and his master and bachelor degrees in computational mathematics from the University of Science and Technology of China in 2003 and 2000 respectively. His recent research is centered on geometry processing and 3D learning. His recent research focuses on geometry processing and 3D learning. He is on the editorial boards of *IEEE Transactions on Visualization and Computer Graphics* and *ACM Transactions on Graphics*.



Hao Pan a senior researcher at Microsoft Research Asia. He received his B.Eng. degree in software engineering from Shandong University and his Ph.D. degree in computer science from The University of Hong Kong. His research interests include computer graphics, 3D vision, and geometry modeling and processing.

Peng-Shuai Wang is an assistant professor at Peking University. He received his Ph.D. degree in computer science and his bachelor degree in automation from Tsinghua University in 2018 and 2013, respectively. His research focuses on computer graphics and 3D vision.

Xin Tong is a principal research manager at Microsoft Research Asia, where he is in charge of the Internet Graphics Group. He obtained his Ph.D. degree from Tsinghua University in 1999. His research focuses on computer graphics and computer vision, such as texture synthesis, appearance modeling, light transport simulation and acquisition, 3D facial animation, and data-driven geometric processing. He has been on the editorial boards of *IEEE Transactions on Visualization and Computer Graphics*, *ACM Transactions on Graphics*, and *Computer Graphics Forum*.

Baining Guo (ACM Fellow & IEEE Fellow) obtained his bachelor degree from Peking University, and his master and doctoral degrees from Cornell University. He is currently the assistant managing director of Microsoft Research Asia, and also the head of the Graphics Lab. Before joining Microsoft in 1999, he was a senior staff researcher at Intel Corporation's Microcomputer Research Labs in Santa Clara, California. His research focuses on computer graphics and visualization, such as texture and reflectance modeling, texture mapping, translucent surface appearance, real-time rendering, and geometry modeling.

To submit a manuscript, please go to `https://jcvm.org`.