Learned Suppression for 3D Keypoint Detection with a Graph-Transformer Backbone

Batuhan Arda Bekar, Can Sarı, Hüseyin Can Gülkan, and Barış Özcan

Department of Computer Engineering, Bahçeşehir University, Istanbul, Turkiye
{batuhanarda.bekar, can.sari, huseyin.gulkan}@bahcesehir.edu.tr
baris.ozcan@bau.edu.tr

Abstract. Detecting 3D keypoints is a long-standing challenge in computer vision. Most detectors end with a heuristic post-processing step that is not learned. We propose a 3D keypoint detector that improves on this step with a learned suppression module, paired with a Point Transformer backbone that we extend with a directional graph neural network. The module is a graph network over candidates that learns which to keep, which to suppress, and how to relocate the remaining ones. Paired with three backbones, it improves over DBSCAN and greedy non-maximum suppression, and because it operates on candidate features rather than raw geometry, the same formulation applies to both structural and semantic keypoints. Our model surpasses the per-category trained KeypointDETR on 12 of 16 KeypointNet categories, attains the best Corner F1 on the Building3D Entry-Level benchmark, and remains competitive with BWFormer on the larger Tallinn split. GitHub implementation: https://github.com/cansdev/learned-suppression-3d.

Keywords: 3D keypoint detection · point clouds · learned suppression · graph neural networks

1 Introduction

Detecting 3D keypoints is a long-standing challenge in vision and graphics, underpinning object detection, shape matching, retrieval, and registration [12, 42]. Unlike 2D images, point clouds are unordered, irregularly sampled, and sparse, so each keypoint must be recovered from a small subset of the points in a cloud.

Most learned 3D keypoint detectors predict a dense saliency or vote field over the points; a separate post-processing step, typically non-maximum suppression (NMS) on heatmaps or DBSCAN clustering on votes [4, 24], then refines this field into a small set of discrete keypoints. This step is external to training: the suppression decision receives no supervision, and the final keypoints are shaped by the heuristic as well as by the features that precede it.

Our approach learns this decision, formulating keypoint extraction as candidate elimination on a graph. Hosang *et al.* [6] showed that NMS can be learned, replacing the fixed deletion rule with a network in which each detection exchanges messages with its neighbors over several rounds and is rescored jointly

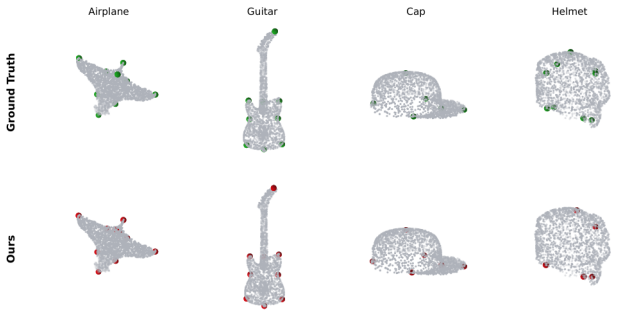

Fig. 1: Qualitative results on KeypointNet (Airplane, Guitar, Cap, Helmet): predictions (red), ground truth (green).

with them. That network only rescores: detections keep their boxes, and a threshold on the new scores produces the final set. In 3D, rescoring alone sets a limit on localization accuracy: a keypoint rarely coincides with a candidate, and the correct location is often a nearby position that no candidate occupies. Our learned suppression module therefore both suppresses and relocates: a graph over candidate keypoints decides, over several rounds of message passing, which candidates to keep, and moves the remaining ones between rounds so that later rounds operate on more accurate positions.

The learned suppression module operates only on the candidates it receives, so their quality bounds the accuracy it can reach. We therefore build the backbone on a hierarchical Point Transformer encoder [45] and extend it in two places: inside each attention block, a spectral positional encoding conditions attention on the local geometry, and after the decoder, a directional graph neural network (GNN) passes signed messages over the feature contrasts between neighboring points to reinforce geometrically distinctive points such as corners.

We evaluate on two datasets with different types of 3D keypoints: the geometric roof corners of Building3D, whose point clouds come from airborne LiDAR [36], and the human-annotated semantic landmarks of KeypointNet, which span 16 object categories [42]. The leading method in each domain builds in domain-specific assumptions, such as parametric roof priors and bird's-eye-view 2D projections for buildings [10, 21], or a fixed per-category keypoint set for objects [12]. Our module operates on candidate features rather than raw geometry, so a single formulation with no domain-specific components applies to both. Our pipeline surpasses the per-category KeypointDETR [12] on 12 of the 16 KeypointNet categories with a single trained model, and attains the best Corner Recall and F1 on Building3D Entry-Level while remaining competitive with BWFormer [21] on the full Tallinn split. Figures 1 and 2 show example detections on both benchmarks.

Our contributions are as follows:

- A learned, graph-based suppression module that outperforms heuristic post-processing such as DBSCAN or greedy NMS for 3D keypoint detection.
- A graph-transformer backbone that extends a Point Transformer encoder with a spectral positional encoding and a directional GNN, which, when paired with the learned suppression module, achieves results that are competitive or better than state-of-the-art detectors on both structural and semantic keypoints.

2 Related Work

2.1 Point Cloud Processing and Graph Neural Networks

Learning directly on point sets started with PointNet [27], which applies shared per-point multi-layer perceptrons (MLPs) and permutation-invariant pooling. To capture global context, PointNet++ [28] added a hierarchical set abstraction (SA) encoder that groups and aggregates local neighborhoods at progressively coarser scales, which is the encoder structure we build on. Point Transformer [45] then replaced pooling with local vector self-attention, which our encoder adopts, and Dynamic Graph CNN [37] builds edge features from the differences between a point and its neighbors, a contrast formulation that our graph layer extends with explicit edge direction and signed messages. Other point and graph networks (KPConv [34], PAConv [40], RandLA-Net [8], Stratified Transformer [14], FG-Net [19], PRA-Net [3]) explore alternative convolution, sampling, and attention schemes; we use them as baselines for comparison. A separate line of work pretrains point cloud representations without labels [25, 43, 44, 46]. These representations transfer well to classification, but they fall behind supervised training when the task requires precise localization, as our Building3D results confirm.

2.2 3D Keypoint Detection

Classic 2D detectors such as SIFT [22], SURF [2], and ORB [29] select interest points by handcrafted local criteria. Harris 3D [32] and learned descriptor-specific detectors [30] carry this notion of geometric saliency to meshes and point clouds, spreading points across the surface without semantic bias.

KeypointNet [42] introduced a large-scale benchmark of human-annotated semantic keypoints over 16 ShapeNet categories, and showed that learned detectors outperformed geometric ones in semantic keypoint detection. KeypointDETR [12] treats keypoint detection as set prediction with a transformer decoder and predicts per-query heatmaps to avoid explicit clustering, but requires a fixed per-category keypoint set. Others forgo landmark supervision: Skeleton Merger [31] merges skeletal hypotheses, UKPGAN [41] distills saliency through reconstruction and adversarial losses, and Wei *et al.* [38] jointly predict saliency and cross-shape correspondence. Many of these methods end with a heuristic post-processing step such as greedy NMS.

Hosang *et al.* [6] showed that this step can be learned by jointly rescoring detections. We bring learned suppression to 3D keypoints, extending it to move

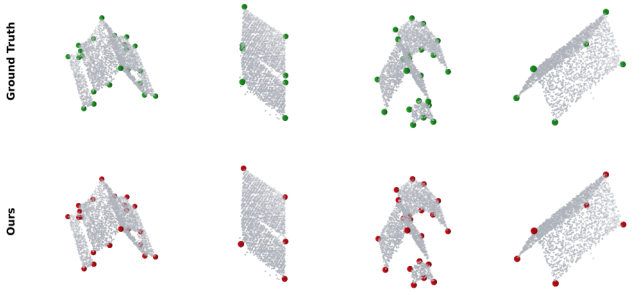

Fig. 2: Qualitative results on Building3D (Tallinn): predictions (red), ground truth (green).

and refine the remaining keypoint locations instead of only rescoring them in place.

2.3 Structural Building Analysis from LiDAR

Learned detection of building structure from airborne LiDAR has progressed from synthetic to real, urban-scale data. PC2WF [20] detects structural corner points directly from raw point clouds, but only on synthetic, rectilinear shapes. Point2Roof [16] was the first to learn roof modeling end to end, regressing roof corner points and consolidating them with DBSCAN clustering. Building3D [36] introduced an urban-scale airborne LiDAR benchmark with annotated roof corners, and we adopt its corner-detection task and metrics. More recent methods raise precision by encoding building-specific priors: PBWR [10] fits parametric roof typologies, and BWFormer [21] detects corners in a 2D top-down projection and lifts them to 3D with a transformer. These priors are specific to roof shapes and do not apply to object-level keypoints. BuildingWorld [11] argues that the limited architectural and geographic diversity of current datasets constrains the generalization of learned building models, which motivates our approach.

3 Method

Our pipeline has two stages (Fig. 3). A hybrid backbone, a hierarchical Point Transformer encoder (Sec. 3.1) and a UNet3+ decoder with a directional GNN (Sec. 3.2), extracts per-point features trained to predict dense classification scores and spatial offsets (Sec. 3.4). High-confidence predictions then vote to form a candidate set, which our learned suppression module (Sec. 3.3) iteratively suppresses and relocates into the final keypoints.

3.1 Hierarchical Encoder: Point Transformer Blocks

The encoder is a stack of four set abstraction stages, SA1 to SA4, each of which downsamples the cloud, grows the feature dimension, and applies two vector

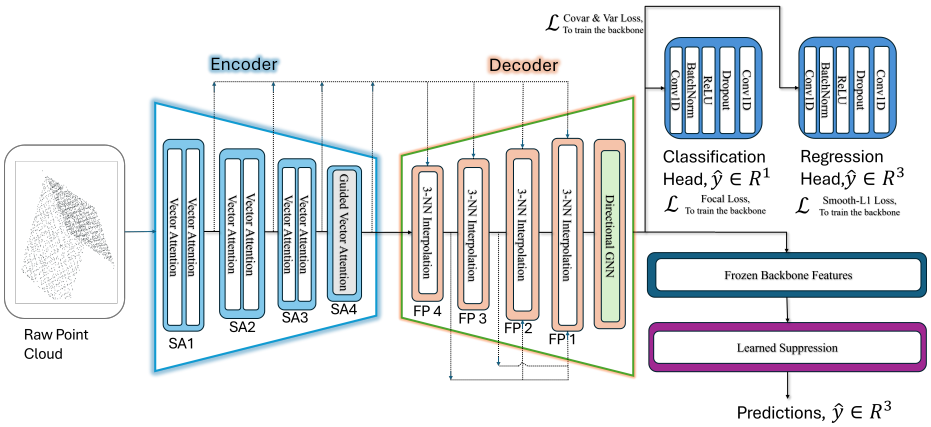

Fig. 3: Model architecture. A hierarchical Point Transformer encoder (SA1–SA4; SA4 injects a ground truth log prior at training time only) feeds a UNet3+ decoder (FP1–FP4) with dense cross-scale skips; a directional GNN refines the full-resolution features before the classification and offset regression heads, whose candidates our learned suppression module consolidates into the final keypoint predictions $\hat{y} \in \mathbb{R}^3$ at inference (Sec. 3.3).

attention Point Transformer blocks in which a center point aggregates its k nearest neighbors (Fig. 4).

Inside a block, a shared linear map with no bias first projects the center feature f_i and each neighbor feature f_j; three further maps with no bias then produce a query q_i, key k_j, and value $v_j \in \mathbb{R}^C$, where C is the feature dimension at that stage (Fig. 4).

A vector attention block relates the center and its neighbor in feature space only, with no information about where the neighbor lies relative to the center. We supply this with a relative-position term built from the offset $\Delta p_{ij} = p_j - p_i$, where p_i and p_j are the coordinates of the center and the neighbor. Since coordinate-based MLPs are biased toward low-frequency functions and represent fine geometric detail poorly [33], we lift the offset into a sinusoidal feature space with the $[\sin, \cos]$ Fourier-feature map of [33], using the octave band schedule $\omega^{\text{base}} = (\pi, 2\pi, \ldots, 2^{L-1}\pi)$ of NeRF-style positional encoding [23], where L is the number of frequency bands. A directional encoding γ_{ij}^{dir} applies this map channel-wise to Δp_{ij}, and a radial encoding γ_{ij}^{rad} applies it to the length $\|\Delta p_{ij}\|$.

Fixed bands cannot adjust to the size and shape of each local neighborhood. Following the learnable-frequency idea of [17], we make the directional bands geometry-adaptive: a 64-dimensional descriptor g_i of the neighborhood (obtained by Fourier-embedding the scale-normalized neighbor offsets and pooling them by mean, standard deviation, and maximum over the k neighbors, shown as the "geo" vector in Fig. 4) is passed through an MLP that predicts per-axis band multipliers $\mu_i = \text{MLP}(g_i)$. These multipliers rescale the octave bands along each

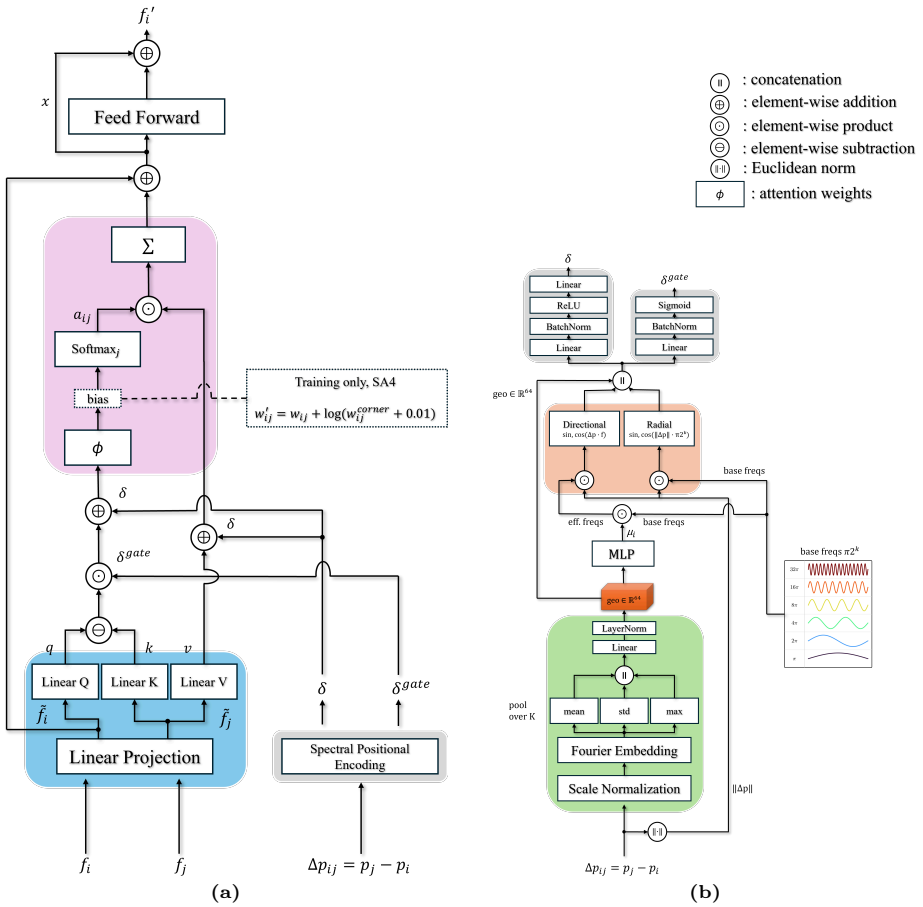

Fig. 4: (a) Point Transformer block (Sec. 3.1); the dashed callout is the SA4 training-only log prior of Eq. (3). **(b)** Spectral positional encoding (Eq. (1)).

spatial axis (with $\odot$ the element-wise product) to give the effective frequencies ω_i (shown as eff. freqs in Fig. 4), while the radial bands stay fixed. A further projection ψ maps the two encodings and the descriptor to an additive position bias δ_{ij} and a multiplicative gate δ_{ij}^{gate},

$$\omega_i = \omega^{\text{base}} \odot \mu_i, \qquad (\delta_{ij},\, \delta_{ij}^{gate}) = \psi\big([\,\gamma_{ij}^{\text{dir}},\ \gamma_{ij}^{\text{rad}},\ g_i\,]\big), \tag{1}$$

where the directional encoding γ_{ij}^{dir} is evaluated at the effective frequencies ω_i. This geometry-adaptive map is the spectral positional encoding of Fig. 4b.

The gate $\delta_{ij}^{gate} \in (0,1)^C$ multiplies the content difference $q_i - k_j$ channel by channel, attenuating the feature contrast between the center and the neighbor, similar to the multiplicative position term in Point Transformer v2 [39]. The bias δ_{ij} is applied as an offset to the gated relation and then added again into the

values. A small per-pair MLP ϕ (two 1×1 layers with BatchNorm and ReLU) maps the gated and biased relation to the attention weights. As in the vector attention of Point Transformer [45], the attention weights modulate the values v per channel rather than through a single shared scalar; to keep this compact, ϕ outputs one weight for every group of g channels, where g=8. A softmax over the neighbors normalizes them into α_{ij}, which aggregate the values, each shifted by the position bias δ_{ij},

$$\begin{aligned} w_{ij} &= \phi\big(\delta_{ij}^{gate} \odot (q_i - k_j) + \delta_{ij}\big), \qquad \alpha_{ij} = \mathrm{softmax}_j(w_{ij}), \\ \mathrm{out}_i &= \sum_j \alpha_{ij} \odot (v_j + \delta_{ij}). \end{aligned} \tag{2}$$

The aggregated output out_i is added to the center feature through a residual connection and refined by a feed-forward layer with a second residual (Fig. 4).

This Point Transformer block is distinct at SA4 in training time, the final and coarsest Set Abstraction stage (Guided Vector Attention at Fig. 3), where a training-time-only prior following the paradigm of learning using privileged information [35] is used. We bias the attention logits with a log term derived from the soft ground truth keypoint labels,

$$\begin{aligned} w'_{ij} &= w_{ij} + \log\big(w_{ij}^{corner} + 0.01\big), \\ w_{ij}^{corner} &= y_j, \qquad y_i = \mathrm{clip}\big(\exp(-d_i^{\min}/r),\, 0,\, 1\big), \end{aligned} \tag{3}$$

where y_i is the soft ground truth label of point i, set from its distance $d_i^{\min}$ to the nearest annotated keypoint with radius r, and w_{ij}^{corner} carries this label down the sampling hierarchy to SA4 and across each center's neighbors. This prior is used during training only: no ground truth label enters the network at inference, where Eq. (3) reduces exactly to the plain vector attention of Eq. (2).

3.2 UNet3+ Skip Connections and Directional GNN

The decoder mirrors the encoder with four feature propagation stages, FP4 to FP1, each of which upsamples the features by three-nearest-neighbor interpolation. Following UNet3+ [9], it fuses dense skip connections from every encoder and decoder stage; Fig. 3 shows the connectivity.

After FP1, a directional GNN refines the per-point features over a k-nearest-neighbor (k-NN) graph. On Building3D the graph uses Euclidean neighborhoods; on KeypointNet, where landmarks lie along the object surface, the neighborhood and the edge length dist_{ij} are instead measured as geodesic distances, which strengthens the contrast features (Tab. 2). We follow the edge-feature formulation of Dynamic Graph CNN [37]: each edge (i, j) carries an edge feature e_{ij} that concatenates the feature contrast $f_i - f_j$ between a point and its neighbor with the unit edge direction dir_{ij} and the edge length dist_{ij},

$$e_{ij} = [\, f_i - f_j,\ \mathrm{dir}_{ij},\ \mathrm{dist}_{ij}\,], \qquad \mathrm{dir}_{ij} = \frac{p_j - p_i}{\|p_j - p_i\|}, \qquad \mathrm{dist}_{ij} = \|p_j - p_i\|. \tag{4}$$

Two MLPs act on each edge feature: MLP_a produces an attention logit that is normalized over the neighbors j by a softmax (scaled by $1/\sqrt{C}$, with C the feature dimension at the FP1 stage) to give the weight a_{ij}, and MLP_m produces a signed message m_{ij}. We aggregate the messages and add them back through a learnable gated residual,

$$\begin{aligned} a_{ij} &= \mathrm{softmax}_j\Big(\tfrac{\mathrm{MLP}_a(e_{ij})}{\sqrt{C}}\Big), \qquad m_{ij} = \mathrm{MLP}_m(e_{ij}), \\ f'_i &= f_i + \sigma(\lambda)\,\mathrm{MLP}_o\Big(\textstyle\sum_j a_{ij}\, m_{ij}\Big), \end{aligned} \tag{5}$$

where MLP_o is an output projection, σ is the logistic sigmoid, and λ is a single learnable scalar that gates the update; MLP_m has no final activation, so a message can either increase or decrease a feature's importance. The messages are signed and carry direction, so that points receive different contrasts depending on their positions with respect to their neighbors. The directional GNN outputs refined per-point features, which are then fed as input to the learned suppression module (Sec. 3.3).

3.3 Learned Suppression

Inspired by three lines of work, our learned suppression module makes the final selection itself learned: a network over a small candidate set, rather than the full cloud, decides which candidates become keypoints and where they lie. Learned NMS replaces a fixed suppression rule with a network that rescores detections from their neighbors [6]; relation networks remove duplicate detections by modeling pairwise relations between objects [7]; and permutation-invariant set networks process an unordered set with attention and impose no ordering [15]. For 3D keypoints, our module is a permutation-invariant graph over candidates that learns pairwise suppression, and it additionally iterates and relocates the remaining candidates between rounds.

Candidate generation. Every point whose predicted corner probability exceeds a threshold casts a Hough-style vote at its offset-displaced position $p_i + r\,\hat{\Delta}_i$ [26]. We denoise the votes with two iterations of mean-shift and initially reduce them to a candidate set by greedy NMS. Each candidate j pools the statistics of its voters together with the refined features from Eq. (5), which yields the candidate feature z_j and its position c_j.

Learned elimination. The learned suppression module is a small graph network over the candidate set (Algorithm 1), built from five MLPs. A shared MLP, $\mathrm{MLP}_{\mathrm{emb}}$, first embeds each candidate feature z_j into a hidden state $h_j \in \mathbb{R}^{128}$. The module then runs T rounds of message passing within a single forward pass; a round is one iteration of the module's computation, different from a training epoch. In each round, for every ordered pair of candidates within a radius ρ, $\mathrm{MLP}_{\mathrm{rel}}$ maps the two hidden states and their current distance d_{ij} to a suppression relation $s_{ij} \in [0, 1]$, the degree to which candidate i suppresses

Algorithm 1 Learned suppression module. $\mathrm{MLP}_{\mathrm{emb}}$ and $\mathrm{MLP}_{\mathrm{keep}}$ are shared across rounds; $\mathrm{MLP}_{\mathrm{rel}}$, $\mathrm{MLP}_{\mathrm{upd}}$, and $\mathrm{MLP}_{\mathrm{mov}}$ have independent weights in each round t. ρ: suppression radius; T: message-passing rounds per forward pass; τ: keep threshold.

Require: candidate features $\{z_j\}$, candidate positions $\{c_j\}$
$h_j \leftarrow \mathrm{MLP}_{\mathrm{emb}}(z_j)$ ▷ shared embedding into $\mathbb{R}^{128}$
for $t = 1, \ldots, T$ **do**
  $d_{ij} \leftarrow \|c_i - c_j\|, \quad \mathcal{A} \leftarrow \{(i,j) : d_{ij} < \rho,\ i \neq j\}$
  $s_{ij} \leftarrow \sigma\big(\mathrm{MLP}_{\mathrm{rel}}([\,h_i, h_j, d_{ij}\,])\big)$ for $(i,j) \in \mathcal{A}$, else 0 ▷ s_{ij}: i suppresses j
  $u_j \leftarrow \sum_i s_{ij}\, h_i$ ▷ incoming suppression
  $h_j \leftarrow h_j + \mathrm{MLP}_{\mathrm{upd}}([\,h_j, u_j\,])$
  $c_j \leftarrow c_j + \mathrm{MLP}_{\mathrm{mov}}(h_j)$ ▷ relocate; c_j accumulates the displacement
end for
$\pi_j \leftarrow \sigma\big(\mathrm{MLP}_{\mathrm{keep}}(h_j)\big) \cdot \big(1 - \max_i s_{ij}\big)$ ▷ s_{ij} from the final round
return $\{c_j : \pi_j > \tau\}$

candidate j. Each candidate then aggregates the suppression it receives into a message $u_j = \sum_i s_{ij}\, h_i$; $\mathrm{MLP}_{\mathrm{upd}}$ updates its hidden state from $[h_j, u_j]$; and $\mathrm{MLP}_{\mathrm{mov}}$ predicts a displacement that relocates the candidate before the next round. After the final round, a shared $\mathrm{MLP}_{\mathrm{keep}}$ maps each hidden state to a keep score, which is multiplied by $1 - \max_i s_{ij}$ (the strongest suppression the candidate received) to give an effective keep probability π_j. Candidates with π_j above a threshold τ are kept and moved by their respective displacement.

3.4 Loss Design

Each point carries the soft keypoint label y_i from Eq. (3), and the positive set P collects all points within radius r of an annotated keypoint; for these the regression target is the displacement to the nearest keypoint, scaled by $1/r$. The classification and regression components of our loss are inspired by Point2Roof [16].

Classification. The per-point corner logits x_i are trained with a focal loss [18] to counter class imbalance, with two additions. The focal exponent γ is annealed from 2.0 to a floor of 1.05 over the first 100 epochs and then held fixed, so the loss concentrates on rare positives early and relaxes once they are learned. Each point is further reweighted by $1 + \beta\, e^{-d_i^{\min}/r}$, where $d_i^{\min}$ is the distance from point i to its nearest annotated keypoint (as in Eq. (3)), r is the label radius, and $\beta = 2.0$; this weight emphasizes points near a keypoint,

$$\begin{aligned} L_{cls} &= \mathbb{E}_i\Big[\big(1 + \beta\, e^{-d_i^{\min}/r}\big)\,(1 - p_{t,i})^{\gamma}\,\mathrm{BCE}(x_i, y_i)\Big], \\ p_{t,i} &= \sigma(x_i)\, y_i + (1 - \sigma(x_i))(1 - y_i). \end{aligned} \tag{6}$$

Here $\mathbb{E}_i$ denotes the mean over all points i, BCE is the binary cross-entropy, σ is the logistic sigmoid, and $p_{t,i}$ is the focal loss probability of the true class at point i.

Table 1: Comparison with prior methods on Building3D (test split). Best per section in **bold**; prior results from [36] (Entry-Level) and [21] (Tallinn). †"Supervised (PT)" uses Point-Transformer [45]; "Self-supervised"/"Linear self-supervised" are Point-MAE-based variants from [36].

Entry-Level					*Tallinn*				
Method	ACO↓	CP	CR	CF1	Method	ACO↓	CP	CR	CF1
Point-BERT [43]	0.25	88.0	69.0	77.0	Point-MAE [25]	0.330	75.0	47.0	58.0
Point-MAE [25]	0.27	85.0	69.0	76.0	Point-M2AE [44]	0.320	79.0	58.0	67.0
Point-M2AE [44]	0.26	88.0	69.0	77.0	Point2Roof [16]	0.390	65.0	30.0	41.0
3D-OAE [46]	0.27	86.0	68.0	76.0	Linear self-supervised† [36]	0.350	70.0	60.0	65.0
Self-supervised† [36]	0.26	87.0	69.0	77.0	Supervised (PT)† [36]	0.290	90.0	53.0	66.0
PointNet [27]	0.36	71.0	50.0	59.0	PC2WF [20]	0.520	18.0	67.0	28.0
PointNet++ [28]	0.34	79.0	52.0	63.0	PBWR [10]	0.222	**98.5**	68.8	81.0
RandLA-Net [8]	0.35	70.0	60.0	65.0	BWFormer [21]	**0.204**	94.9	82.7	**88.4**
DGCNN [37]	0.32	73.0	58.0	65.0	Ours	0.216	92.3	**83.7**	87.8
PAConv [40]	0.33	75.0	57.0	65.0					
Stratified Transformer [14]	0.38	72.0	51.0	62.0					
FG-Net [19]	0.32	77.0	64.0	70.0					
Point2Roof [16]	0.30	66.0	48.0	56.0					
Supervised (PT)† [36]	0.26	89.0	66.0	76.0					
PBWR [10]	**0.22**	**97.0**	85.0	91.0					
Ours	**0.22**	96.5	**90.1**	**93.2**					

Offset regression. An annotated keypoint rarely coincides with a sampled input point [36], so per-point classification alone cannot predict it exactly; for each point in the positive set P, the offset regression head predicts displacements to its target keypoint, recovering locations that lie off the input point set. The predicted offsets of the positive set are trained with a Smooth-L_1 loss [5], summed over the three coordinates and averaged over $|P|$; we denote it L_{off}.

Feature regularization. Because the learned suppression module consumes the post-GNN per-point features, we guard them against representational collapse with the variance and covariance terms of VICReg [1] (L_{var}, L_{cov}); Tab. 2 reports their effect.

Total. The backbone is trained end to end with

$$L = L_{cls} + L_{off} + 0.1\, L_{var} + 0.01\, L_{cov}. \tag{7}$$

Two-stage training. The learned suppression module is trained separately, on top of the frozen backbone, so the candidate features it consumes are exactly those it sees at inference. Candidates are matched to ground truth keypoints by Hungarian assignment [13]; a matched candidate supervises a binary keep loss (cross-entropy with target 1) and a Smooth-L_1 displacement loss toward its matched keypoint, while unmatched candidates are supervised to be suppressed (keep target 0).

Table 2: Component ablations; Building3D Entry-Level numbers are measured on a held-out validation split, a 10% hold-out of the Entry-Level training set. The Op. (Operation) column marks whether each row adds a new component, swaps one for another ($x \to y$), or sets the parameters of a component already in the pipeline to the best configuration on this validation split; the running model carries every change above it. The base of (b) is the full backbone built in (a), the running model after the spectral positional encoding row, with the directional GNN on Euclidean neighborhoods. † marks the same running model shared across the two sections. Best per section in **bold**.

(a) Building3D Entry-Level

Configuration	Op.	ACO↓	CP	CR	CF1	ΔCF1
PointNet++	base	0.311	74.7	80.9	77.66	–
Contrast GNN	add	0.308	76.5	81.9	79.10	+1.44
Point Transformer blocks	add	0.300	84.2	83.9	84.03	+4.93
SA4 attention prior	add	0.291	89.0	80.1	84.32	+0.29
DBSCAN grid search	set	0.301	86.3	83.8	85.00	+0.68
Var/Covar regularizer	add	0.298	87.8	83.8	85.72	+0.72
ReLU → GELU	swap	0.297	88.0	84.0	85.95	+0.23
UNet skip conn. → UNet3+ skip conn.	swap	0.290	88.6	84.3	86.44	+0.49
Edge direction in GNN	add	0.279	88.1	86.0	87.08	+0.64
Spectral positional encoding†	add	0.280	90.3	86.8	88.55	+1.47
DBSCAN → learned supp. (rescore only)	swap	0.280	95.3	**89.9**	92.50	+3.95
Rescore only → rescore & relocation	swap	**0.251**	**96.1**	89.5	**92.70**	+0.20

(b) KeypointNet (all 16 categories)

Configuration	Op.	mIoU	CD↓
Full backbone from (a)†	base	77.37	0.091
Euclidean → geodesic graph	swap	79.67	0.085
DBSCAN → NMS	swap	84.25	0.080
NMS → learned supp. (relocate & rescore)	swap	**92.01**	**0.055**

4 Experiments

4.1 Datasets

Building3D. We use the corner-detection task of Building3D [36] on two splits, the Entry-Level subset and the full Tallinn city split. Predicted corners are matched with the Hungarian assignment [13] to the ground truth and scored by Corner Precision, Recall, and F1 (CP, CR, CF1) and the Average Corner Offset (ACO, lower is better).

KeypointNet. KeypointNet [42] provides human-annotated semantic keypoints on ShapeNet objects across 16 categories. We follow the standard split and evaluate with the KeypointDETR [12] evaluation protocol: Hungarian-matched mean intersection-over-union (mIoU) at a distance threshold ϵ and Chamfer Distance (CD). The mIoU and CD metrics are averaged over all test shapes, so the aggregate values in Tabs. 2 and 4 are shape-weighted rather than the mean of the per-category columns in Tab. 3. The training regimes differ: KeypointDETR trains one model per category with a fixed keypoint set, and its numbers are taken from [12], whereas our model is trained on all 16 categories simultaneously.

4.2 Implementation Details

Common setup. Training follows the same two-stage procedure on both datasets. The feature propagation and graph layer MLPs use GELU, and the classification and offset heads use ReLU. The backbone is trained with AdamW under a OneCycleLR schedule with a peak learning rate of 3×10^{-3}; we then freeze it and train the learned suppression module with Adam. Candidate generation

Table 3: Per-category keypoint localization on all 16 KeypointNet categories: Ours versus KeypointDETR and prior detectors; baseline numbers as reported by KeypointDETR [12]. Best per column in **bold**. *All methods use the identical evaluation protocol, adopted from KeypointDETR.

(a) Hungarian mIoU* (%) at $\epsilon = 0.10$.

Method	Air.	Bat.	Bed	Bot.	Cap	Car	Cha.	Gui.	Hel.	Kni.	Lap.	Mot.	Mug	Ska.	Tab.	Ves.
PRA-Net [3]	82.88	72.72	49.32	40.11	28.20	82.06	70.44	73.71	31.24	38.15	84.70	60.73	76.50	49.47	69.69	48.17
PointNet++ [28]	82.31	71.77	57.85	50.77	45.78	84.06	57.50	72.34	48.51	62.57	87.20	69.93	68.75	67.67	60.01	58.52
DGCNN [37]	78.49	60.33	64.30	56.79	47.56	79.12	59.49	63.88	46.79	70.19	85.22	68.69	63.23	76.78	59.92	54.56
Wei *et al.* [38]	79.86	65.50	52.05	55.58	36.14	86.13	86.28	78.08	47.72	63.66	94.40	69.38	70.79	72.91	86.61	58.38
KeypointDETR [12]	91.37	71.37	70.83	87.22	79.43	**94.29**	91.89	93.96	**85.05**	63.12	99.11	87.52	**96.62**	92.28	**98.05**	59.97
Ours	**96.93**	**90.83**	**87.61**	**90.68**	**91.74**	92.25	**94.58**	**97.84**	52.36	**91.12**	**99.65**	**92.28**	85.55	**94.25**	95.96	**77.35**

(b) Chamfer Distance (CD*, lower is better).

Method	Air.	Bat.	Bed	Bot.	Cap	Car	Cha.	Gui.	Hel.	Kni.	Lap.	Mot.	Mug	Ska.	Tab.	Ves.
PRA-Net [3]	0.049	0.082	0.182	0.163	0.238	0.059	0.060	0.053	0.170	0.188	0.064	0.099	0.070	0.163	0.053	0.146
PointNet++ [28]	0.050	0.075	0.113	0.092	0.134	0.057	0.077	0.060	0.144	0.112	0.046	0.075	0.076	0.080	0.062	0.108
DGCNN [37]	0.058	0.074	0.086	0.083	0.101	0.066	0.076	0.072	0.123	0.083	0.036	0.079	0.083	0.069	0.066	0.107
Wei *et al.* [38]	0.048	0.088	0.198	0.093	0.211	0.048	0.055	0.066	0.172	0.107	0.078	0.085	0.101	0.081	0.039	0.100
KeypointDETR [12]	0.045	0.074	0.081	0.050	0.066	**0.045**	0.055	0.045	**0.087**	0.089	**0.024**	0.053	**0.049**	0.055	**0.027**	0.105
Ours	**0.038**	**0.049**	**0.056**	**0.049**	**0.062**	0.051	**0.051**	**0.029**	0.407	**0.036**	0.033	**0.047**	0.096	**0.037**	0.059	**0.079**

(Sec. 3.3) uses vote threshold 0.10 and NMS radius 0.05; the module runs T=2 suppression rounds with radius ρ=0.08 and keep threshold τ=0.40, and training matches candidates to keypoints by Hungarian assignment at threshold 0.10. These settings are chosen on the validation split; the heuristic baselines in Tab. 4 use the same ones, greedy NMS being exactly this candidate generation stage and DBSCAN the configuration set in Tab. 2a (eps 0.05, threshold 0.19), held fixed across backbones and datasets. The whole pipeline trains and runs on a single NVIDIA RTX 4070 (12 GB VRAM), where the backbone holds roughly 22.7M parameters and the suppression module adds 0.15M.

Building3D. Each scene is subsampled to 2560 points carrying XYZ, RGBA color, LiDAR intensity, and four scale-conditioning channels (the log normalization scale and bounding-box aspect ratios, broadcast to every point to recover the scale and shape lost to centroid-radius normalization), for C=12 input channels. The backbone is trained for 300 epochs at batch size 32 and the suppression module for 60 epochs at batch size 8, independently on the Entry-Level subset and on the full Tallinn city split. Training augments each scene with random flips and rotations, and soft labels (Eq. (3)) use radius r=0.05. At inference we use test-time augmentation: each scene is passed through the network at four 90° rotations, and a predicted corner is kept only if it is recovered in at least two of the four passes.

KeypointNet. A single backbone is trained jointly across all 16 categories for 100 epochs at batch size 14, on shapes subsampled to 2048 points carrying XYZ, color, and surface normals from principal component analysis (PCA) (C=9).

Table 4: Our learned suppression versus heuristic post-processing (greedy NMS, DBSCAN) per backbone, on Building3D (Entry-Level, Tallinn) and KeypointNet (all 16 categories), all on the test splits. Best per backbone in **bold**.

		Building3D Entry-Level				Building3D Tallinn				KeypointNet	
Backbone	Post-proc.	ACO↓	CP	CR	CF1	ACO↓	CP	CR	CF1	mIoU	CD↓
Ours	NMS	0.285	92.3	89.2	90.70	0.310	85.4	**84.8**	85.10	84.25	0.080
	DBSCAN	0.260	95.9	82.9	89.00	0.290	**94.2**	75.2	83.60	79.67	0.085
	Learned supp.	**0.220**	**96.5**	**90.1**	**93.20**	**0.216**	92.3	83.7	**87.80**	**92.01**	**0.055**
PointNet++ [28]	NMS	0.323	44.2	**95.0**	60.40	0.366	54.6	**91.2**	68.30	46.33	0.130
	DBSCAN	0.277	89.5	76.1	82.30	0.312	90.2	72.5	80.40	70.92	**0.111**
	Learned supp.	**0.270**	**89.9**	84.2	**87.00**	**0.295**	**92.4**	78.5	**84.90**	**74.99**	0.115
DGCNN [37]	NMS	0.338	27.8	**94.8**	43.00	0.390	28.9	**94.7**	44.30	42.87	0.129
	DBSCAN	**0.287**	82.1	75.0	78.40	0.337	77.5	72.4	74.90	70.68	0.100
	Learned supp.	0.295	**90.7**	79.8	**84.90**	**0.305**	**89.7**	75.0	**81.70**	**77.87**	**0.099**

Augmentation applies random flips, rotations, and scale jitter; soft labels use radius r=0.10. The suppression module uses the same settings as on Building3D.

4.3 Experimental Results

Building3D. On Building3D Entry-Level (Tab. 1) our detector reaches the best Corner Recall (90.1) and F1 (93.2) and matches the best ACO (0.22). It trails only PBWR, which fits parametric roof typologies, on Corner Precision (96.5 against 97.0). On the larger Tallinn split our model remains competitive with BWFormer: it obtains the best Corner Recall (83.7 against 82.7) and an F1 of 87.8 against BWFormer's 88.4.

KeypointNet. On KeypointNet (Tab. 3) a single model trained jointly on all 16 categories surpasses the per-category KeypointDETR on 12 of the 16 categories in mIoU, with the largest gains on categories such as Cap (91.74 against 79.43), Bathtub (90.83 against 71.37), and Vessel (77.35 against 59.97). The Chamfer distances in Tab. 3b follow the same pattern. KeypointDETR remains ahead on a few categories, most notably Helmet, which we analyze below.

Effect of learned suppression. Table 4 isolates the contribution of the module by swapping only the post-processing stage, repeated on three backbones. On our backbone, learned suppression raises Corner F1 on Building3D Entry-Level from 90.70 (NMS) and 89.00 (DBSCAN) to 93.20, and on Tallinn from 85.10 and 83.60 to 87.80, while on KeypointNet it lifts mIoU from 84.25 (NMS) and 79.67 (DBSCAN) to 92.01. The same pattern holds when the module is applied to a PointNet++ or a DGCNN backbone, where it gives the best F1 and mIoU in every case. This indicates that the improvement holds across both structural and semantic keypoints, using different backbones.

Component ablation. The ablation in Tab. 2 builds the full model, starting from a PointNet++ base. On Building3D Entry-Level, the contrast GNN, the Point Transformer blocks, and the SA4 attention prior (which increases CP as spurious responses are suppressed, at the potential cost of a few weakly observed corners, lowering CR) together raise Corner F1 from 77.66 to 84.32, and the remaining refinements bring it to 88.55. Replacing DBSCAN with the learned suppression module, first restricted to rescoring, adds 3.95 F1 to reach 92.50; enabling relocation leaves F1 mostly unchanged (92.70) but improves ACO from 0.280 to 0.251, showing that relocation contributes to localization accuracy. On KeypointNet (Tab. 2b), moving from a Euclidean to a geodesic graph raises mIoU from 77.37 to 79.67; DBSCAN to greedy NMS lifts it to 84.25, and the learned suppression module reaches 92.01.

Failure analysis: Helmet. Helmet is our weakest category: Tab. 3 reports 52.36 mIoU and the largest Chamfer distance in the table (0.407), with prediction results in Fig. 1. This failure follows from the design of the pipeline: the detector consolidates candidates from local feature contrast and carries no category template, so it depends on landmarks that coincide with locally distinctive geometry. Helmet's landmarks are semantic conventions placed on a smooth shell with little such geometry. Cap, beside it in Fig. 1, has a similarly smooth global shape, but its landmarks sit on the brim, a sharp ridge, and the same trained model reaches 91.74 mIoU there. KeypointDETR recovers Helmet (85.05 mIoU) with the per-category keypoint set that our single model forgoes.

Localization is bounded by candidate generation. Our learned suppression module lowers the ACO over greedy NMS on every backbone (Tab. 4). DBSCAN keeps two localization wins, ACO on DGCNN Entry-Level (0.287 against 0.295) and CD on PointNet++ KeypointNet (0.111 against 0.115); both come from candidate generation rather than suppression: DBSCAN averages each cluster while NMS keeps a single vote, so the cluster mean can begin closer to the true corner.

5 Conclusion

We presented a 3D keypoint detector built around a learned suppression module that improves over heuristic NMS and DBSCAN post-processing by suppressing and relocating coarse candidates over several rounds. The same architecture surpasses KeypointDETR on 12 of 16 KeypointNet categories, attains the best Corner Recall and F1 on Building3D Entry-Level, and remains competitive with BWFormer on Tallinn; the module improves Corner F1 and mIoU across all three backbones, including DBSCAN and non-maximum suppression. Two limitations remain: candidate generation still relies on a vote threshold and greedy NMS, and the KeypointDETR comparison spans different training regimes. Learning candidate generation, training the backbone and the suppression module jointly, and extending to scene-level clouds are natural next steps.

References

1. Bardes, A., Ponce, J., LeCun, Y.: Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In: ICLR (2022)
2. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: European conference on computer vision. pp. 404–417. Springer (2006)
3. Cheng, S., Chen, X., He, X., Liu, Z., Bai, X.: PRA-Net: Point relation-aware network for 3d point cloud analysis. IEEE TIP **30**, 4436–4448 (2021)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 226–231. KDD'96, AAAI Press (1996)
5. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
6. Hosang, J., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4507–4515 (2017)
7. Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3588–3597 (2018)
8. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In: CVPR. pp. 11108–11117 (2020)
9. Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., Han, X., Chen, Y.W., Wu, J.: Unet 3+: A full-scale connected unet for medical image segmentation. In: ICASSP. pp. 1055–1059 (2020)
10. Huang, S., Wang, R., Guo, B., Yang, H.: Pbwr: Parametric-building-wireframe reconstruction from aerial lidar point clouds. In: CVPR. pp. 27778–27787 (2024)
11. Huang, S., Wang, R., Wang, X.: Buildingworld: A structured 3d building dataset for urban foundation models. In: AAAI. pp. 5085–5094 (2026)
12. Jin, H., Shen, Y., Lou, J., Zhou, K., Zheng, Y.: Keypointdetr: an end-to-end 3d keypoint detector. In: ECCV. pp. 374–390 (2024)
13. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1-2), 83–97 (1955)
14. Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., Jia, J.: Stratified transformer for 3d point cloud segmentation. In: CVPR. pp. 8500–8509 (2022)
15. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: International conference on machine learning. pp. 3744–3753. PMLR (2019)
16. Li, L., Song, N., Sun, F., Liu, X., Wang, R., Yao, J., Cao, S.: Point2roof: End-to-end 3d building roof modeling from airborne lidar point clouds. ISPRS Journal of Photogrammetry and Remote Sensing **193**, 17–28 (2022)
17. Li, Y., Si, S., Li, G., Hsieh, C.J., Bengio, S.: Learnable fourier features for multi-dimensional spatial positional encoding. Advances in Neural Information Processing Systems **34**, 15816–15829 (2021)
18. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. IEEE TPAMI **42**(2), 318–327 (2020)
19. Liu, K., Gao, Z., Lin, F., Chen, B.M.: Fg-net: A fast and accurate framework for large-scale lidar point cloud understanding. IEEE Transactions on Cybernetics **53**(1), 553–564 (2022)

20. Liu, Y., D'Aronco, S., Schindler, K., Wegner, J.D.: Pc2wf: 3d wireframe reconstruction from raw point clouds. arXiv preprint arXiv:2103.02766 (2021)
21. Liu, Y., Zhu, L., Ye, H., Huang, S., Gao, X., Zheng, X., Shen, S.: Bwformer: Building wireframe reconstruction from airborne lidar point cloud with transformer. In: CVPR. pp. 22215–22224 (2025)
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
23. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
24. Neubeck, A., Van Gool, L.: Efficient non-maximum suppression. In: 18th international conference on pattern recognition (ICPR'06). vol. 3, pp. 850–855. Ieee (2006)
25. Pang, Y., Wang, W., Tay, F.E., Liu, W., Tian, Y., Yuan, L.: Masked autoencoders for point cloud self-supervised learning. In: ECCV. pp. 604–621 (2022)
26. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9277–9286 (2019)
27. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. pp. 652–660 (2017)
28. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017)
29. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: 2011 International conference on computer vision. pp. 2564–2571. Ieee (2011)
30. Salti, S., Tombari, F., Spezialetti, R., Di Stefano, L.: Learning a descriptor-specific 3d keypoint detector. In: ICCV. pp. 2318–2326 (2015)
31. Shi, R., Xue, Z., You, Y., Lu, C.: Skeleton merger: An unsupervised aligned keypoint detector. In: CVPR. pp. 43–52 (2021)
32. Sipiran, I., Bustos, B.: Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. The Vis. Comput. **27**(11), 963–976 (2011)
33. Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. Advances in neural information processing systems **33**, 7537–7547 (2020)
34. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: ICCV. pp. 6411–6420 (2019)
35. Vapnik, V., Vashist, A.: A new learning paradigm: Learning using privileged information. Neural networks **22**(5-6), 544–557 (2009)
36. Wang, R., Huang, S., Yang, H.: Building3d: A urban-scale dataset and benchmarks for learning roof structures from point clouds. In: ICCV. pp. 20076–20086 (2023)
37. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM TOG **38**(5), 1–12 (2019)
38. Wei, G., Ma, L., Wang, C., Desrosiers, C., Zhou, Y.: Multi-task joint learning of 3d keypoint saliency and correspondence estimation. Computer-Aided Design **141**, 103105 (2021)
39. Wu, X., Lao, Y., Jiang, L., Liu, X., Zhao, H.: Point transformer v2: Grouped vector attention and partition-based pooling. NeurIPS **35**, 33330–33342 (2022)
40. Xu, M., Ding, R., Zhao, H., Qi, X.: Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In: CVPR. pp. 3173–3182 (2021)

41. You, Y., Liu, W., Ze, Y., Li, Y.L., Wang, W., Lu, C.: Ukpgan: A general self-supervised keypoint detector. In: CVPR. pp. 17042–17051 (2022)
42. You, Y., Lou, Y., Li, C., Cheng, Z., Li, L., Ma, L., Lu, C., Wang, W.: Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations. In: CVPR. pp. 13647–13656 (2020)
43. 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: CVPR. pp. 19313–19322 (2022)
44. 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. Advances in neural information processing systems **35**, 27061–27074 (2022)
45. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: ICCV. pp. 16259–16268 (2021)
46. Zhou, J., Wen, X., Ma, B., Liu, Y.S., Gao, Y., Fang, Y., Han, Z.: 3d-oae: Occlusion auto-encoders for self-supervised learning on point clouds. In: ICRA. pp. 15416–15423 (2024)