Supplementary Material

A. Picasso Library

We summarize the major modules included in Picasso Library as Fig. A.1. All the novel operations introduced in this paper are colorized, while the previous point cloud based operations [32, 33] are left as blank. We also compare performance of the included convolutions for a very large input size, i.e. 65536 in Table A.1. For mesh-based convolutions, we use a batch of 16 meshes as input, whose vertex size each is 65536. For point based convolutions, we use only vertices of those meshes as input. Currently, 3D deep networks are widely taking data samples of <10000 points/vertices as input. For spatial graph convolutions, the graph construction based on neighborhood search takes significant amount of time, especially when the input size is large. For instance, see Table 10 of SPH3D-GCN [33]. We introduce mesh-based modules to take advantage of its geodesic connections and save graph construction time. It is worth noting that we estimate the runtime under Tensorflow 2.

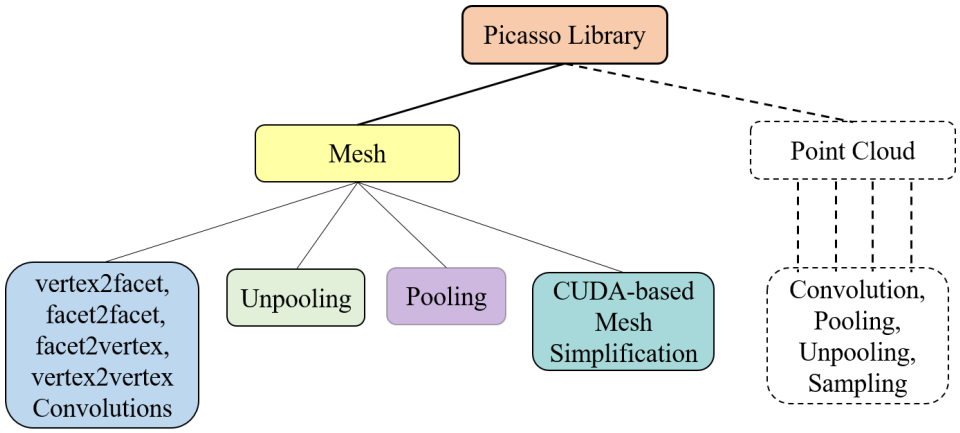

Figure A.1: Deep learning modules included in the Picasso Library.

Table A.1: Runtime comparison of different mesh-based convolutions and point-based convolutions. T represents the total number of filters in the kernel. C_{in} denotes the number of input feature channels, and λ is the multiplier of separable convolutions. Regardless of graph construction, the speed of mesh-based convolutions are similar to point-based convolutions. However, since point-based convolutions demand the graph to be pre-constructed before convolution, which takes significant amount of time when input size is large. For time concern, mesh-based convolutions are preferred as the input size grows.

Convolution Type	batch size	input vetex/point size	kernel configuration			runtime (ms)
			T	C_{in}	λ	
facet2facet	16	65536	3	6	8	125
vertex2facet	16	65536	3	6	8	89
facet2vertex	16	65536 $\xrightarrow{\text{strided}}$ 16384	12	48	2	195
hard SPH3D [33]	16	65536 $\xrightarrow{\text{strided}}$ 16384	$8 \times 2 \times 1 + 1$	48	2	118 + graph construction
fuzzy SPH3D [32]	16	65536 $\xrightarrow{\text{strided}}$ 16384	$8 \times 2 \times 1 + 1$	48	2	188 + graph construction

B. Quadric error computation

The plane function of an arbitrary facet can be denoted as $\mathbf{n}^\intercal \mathbf{x} + d = 0$, where $\mathbf{x} = [x, y, z]^\intercal$ is the point in 3D space, $\mathbf{n} = [n_x, n_y, n_z]^\intercal$ is the facet normal, and d is the intercept. The quadric Q of each facet is defined as $Q = (\mathbf{A}, \mathbf{b}, c) = (\mathbf{n}\mathbf{n}^\intercal, d\mathbf{n}, d^2)$. It associates a value named *quadric error* to an arbitrary point $\mathbf{x}$ in space , which is computed as

$$Q(\mathbf{x}) = \mathbf{x}^\intercal \mathbf{A} \mathbf{x} + 2\mathbf{b}^\intercal \mathbf{x} + c. \tag{B.1}$$

The quadric of each vertex $\mathbf{v}$ is an accumulation of the quadric of its adjacent facets $\mathcal{N}(\mathbf{v})$, represented as

$$Q_v = (\mathbf{A}_v, \mathbf{b}_v, c_v) = \left(\sum_{\mathbf{f}_i \in \mathcal{N}(\mathbf{v})} \mathbf{A}_i, \sum_{\mathbf{f}_i \in \mathcal{N}(\mathbf{v})} \mathbf{b}_i^\intercal, \sum_{\mathbf{f}_i \in \mathcal{N}(\mathbf{v})} c_i \right). \tag{B.2}$$

The quadric of each vertex cluster $\mathcal{C}$ to be contracted is an accumulation of the quadric of all the vertices in it, that is

$$Q_{\mathcal{C}} = (\mathbf{A}_{\mathcal{C}}, \mathbf{b}_{\mathcal{C}}, c_{\mathcal{C}}) = \left(\sum_{\mathbf{v} \in \mathcal{C}} \mathbf{A}_v, \sum_{\mathbf{v} \in \mathcal{C}} \mathbf{b}_v, \sum_{\mathbf{v} \in \mathcal{C}} c_v \right). \tag{B.3}$$

The *optimal vertex placement* of each cluster after contraction is ideally computed as

$$\bar{\mathbf{v}} = -\mathbf{A}_{\mathcal{C}}^{-1} \mathbf{b}_{\mathcal{C}}, \tag{B.4}$$

for which we determine if $\mathbf{A}_{\mathcal{C}}$ is a full-rank matrix by checking the reciprocal of its condition number. However, we still observe numerical instability in the decimated mesh. We therefore replace the computation of $\bar{\mathbf{v}}$ to be

$$\bar{\mathbf{v}} = \frac{1}{|\mathcal{C}|} \sum_{\mathbf{v} \in \mathcal{C}} \mathbf{v}. \tag{B.5}$$

Open3D [71] computes $\bar{\mathbf{v}}$ in the same way. This improves the final results noticeably. See Table B.1 for the performance comparison of PicassoNet ($l = 2$) on S3DIS Area 5 using Eq. (B.4) and Eq. (B.5).

Table B.1: Performance of PicassoNet ($l = 2$) on the fifth fold (Area 5) of S3DIS dataset under different computations of the optimal vertex placement $\bar{\mathbf{v}}$.

Methods	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PicassoNet ($l = 2$, B.4)	88.2	69.4	62.5	93.2	98.4	81.1	0.0	32.1	45.9	75.6	78.5	85.9	51.5	69.0	45.2	55.3
PicassoNet ($l = 2$, B.5)	88.7	69.5	63.1	94.8	98.4	81.4	0.0	30.4	53.7	71.2	76.8	87.5	48.1	69.9	51.2	57.2

C. Backward propagations of different convolutions

In the main paper, we present forward computations of the vertex2facet and facet2vertex convolutions in Eqs. (1), (2), (3) and Eqs. (4), (5), (6), respectively. In this section, we analyze their backward propagations in Eq. (C.1) and (C.2) correspondingly. The computations of facet2facet convolution are similar to those of vertex2facet convolution. We hence omit them here to avoid redundancy. We also provide the relate forward computations in Eq. (C.1) and C.2) as references. Please refer to the main paper for the notations. Finally, as the GMM-based fuzzy coefficients $\{\pi_{it}\}$ are readily computed with built-in Tensorflow functions, there is no need for us to analyze their backward propagations manually because Tensorflow is able to achieve it automatically.

$$\text{vertex2facet kernel:} \begin{cases} J = \frac{1}{K} \sum_k W_k I_k. \\ \frac{\partial J}{\partial I_s} = \frac{1}{K} \sum_k (\xi_{ks} W_k), s \in \{1, 2, 3\}. \\ \frac{\partial J}{\partial w_s} = \frac{1}{K} \sum_k (\xi_{ks} I_k), \ s \in \{1, 2, 3\}. \end{cases} \tag{C.1}$$

$$\text{facet2vertex kernel:} \begin{cases} I_v = \frac{1}{\mathcal{N}(\mathbf{v})} \sum_{\mathbf{f}_i \in \mathcal{N}(\mathbf{v})} \left(\sum_{t=1}^{T} \pi_{it} w_t \right) J_i. \\ \frac{\partial I_v}{\partial J_i} = \frac{1}{\mathcal{N}(\mathbf{v})} \left(\sum_{t=1}^{T} \pi_{it} w_t \right). \\ \frac{\partial I_v}{\partial w_t} = \frac{1}{\mathcal{N}(\mathbf{v})} \sum_{\mathbf{f}_i \in \mathcal{N}(\mathbf{v})} (\pi_{it} J_i). \end{cases} \tag{C.2}$$

D. 6-folds results on S3DIS

We report the 6 folds results of PicassoNet ($l = 2$) on S3DIS dataset in Table D.1. It can be noticed that the PicassoNet using only $l = 2$ layers of mesh convolutions in each block are already competitive to KPConv [59] and DCM-Net [52].

Table D.1: Performance of PicassoNet ($l = 2$) on the entire 6 folds of S3DIS. PicassoNet using only $l = 2$ layers of mesh convolutions in the convolution block are competitive to KPConv and DCM-Net.

	Methods	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
All 6 Folds	PointNet [43]	78.5	66.2	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	42.0	54.1	38.2	9.6	29.4	35.2
	Engelmann et al. [1a]	81.1	66.4	49.7	90.3	92.1	67.9	44.7	24.2	52.3	51.2	47.4	58.1	39.0	6.9	30.0	41.9
	SPG [30]	85.5	73.0	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
	PointCNN [35]	88.1	75.6	65.4	94.8	97.3	75.8	63.3	51.7	58.4	57.2	71.6	69.1	39.1	61.2	52.2	58.6
	SSP+SPG [29]	87.9	78.3	68.4	-	-	-	-	-	-	-	-	-	-	-	-	-
	DeepGCN [2a]	85.9	-	60.0	93.1	95.3	78.2	33.9	37.4	56.1	68.2	64.9	61.0	34.6	51.5	51.1	54.4
	KPConv [59]	-	79.1	70.6	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	69.3	74.9	61.3	60.3
	SPH3D-GCN [33]	88.6	77.9	68.9	93.3	96.2	81.9	58.6	55.9	55.9	71.7	72.1	82.4	48.5	64.5	54.8	60.4
	SegGCN [32]	87.8	77.1	68.5	92.5	97.6	78.9	44.6	58.2	53.7	67.3	74.6	83.9	68.0	65.7	46.8	58.5
	DCM-Net [52]	-	80.7	69.7	93.7	96.6	81.2	44.6	44.9	73.0	73.8	71.4	74.3	63.3	63.9	63.0	61.9
	PicassoNet (Prop. $l = 2$)	89.0	78.8	69.8	93.7	96.6	82.2	57.8	54.9	59.9	77.1	71.5	82.5	51.8	63.8	53.9	61.7

E. Results on ScanNet

Originally, we trained the PicassoNet using cropped meshes generated from the full-resolution meshes provided in ScanNet dataset [12], which produced mediocre results. We found that the reason of this phenomenon is caused by high-frequency signals in noisy areas as well [52]. We hence voxelize the full-resolution meshes in ScanNet [12] using a grid size of $4cm$, and re-conducted the experiment. The performance of PicassoNet ($l = 2$) on the *full* and *voxelized* validation set is reported in Table E.1. Similar results are expected on the test set of ScanNet.

Table E.1: 3D semantic segmentation performance of PicassoNet ($l = 2$) on the ScanNet validation set. 'full' refers to results on the full-resolution meshes, while '$4cm$' refers to the results on the voxelized meshes using a voxel size of $4cm$. Similar results are expected on the test set of ScanNet. For reference, we provide the results of DCM-Net and SPH3D-GCN on the validation set, as well as the results of popular approaches on the test set.

Method	mIoU	floor	wall	chair	sofa	table	door	cab	bed	desk	toil	sink	wind	pic	bkshf	curt	show	cntr	fridg	bath	other
DCM-Net (VC,rad/geo)	62.8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SPH3D-GCN (full)	60.0	93.9	76.6	86.4	76.7	69.0	40.1	55.6	68.4	55.3	85.3	58.8	51.0	6.8	42.2	57.2	64.8	56.2	38.9	79.5	36.1
SPH3D-GCN ($3cm$)	61.2	95.5	78.3	86.9	78.9	71.4	41.6	57.1	70.6	57.4	87.0	58.9	49.2	7.7	42.2	57.8	63.8	58.7	42.2	82.1	37.2
PicassoNet ($l = 2$, full)	62.5	94.4	78.0	86.5	75.3	69.4	47.4	54.1	68.8	56.6	88.4	62.6	50.6	18.2	54.4	63.4	64.8	54.1	41.6	80.4	40.3
PicassoNet ($l = 2$, $4cm$)	64.3	96.1	80.4	87.0	78.2	73.2	50.3	56.6	72.0	59.1	90.0	64.1	49.9	19.2	53.9	63.9	63.7	57.1	44.9	83.5	42.0
ScanNet [12]	30.6	78.6	43.7	52.4	34.8	30.0	18.9	31.1	36.6	34.2	46.0	31.8	18.2	10.2	50.1	0.2	15.2	21.1	24.5	20.3	14.5
PointNet++ [44]	33.9	67.7	52.3	36.0	34.6	23.2	26.1	25.6	47.8	27.8	54.8	36.4	25.2	11.7	45.8	24.7	14.5	25.0	21.2	58.4	18.3
SPLATNET$_{3D}$ [56]	39.3	92.7	69.9	65.6	51.0	38.3	19.7	31.1	51.1	32.8	59.3	27.1	26.7	0.0	60.6	40.5	24.9	24.5	0.1	47.2	22.7
Tangent-Conv [57]	43.8	91.8	63.3	64.5	56.2	42.7	27.9	36.9	64.6	28.2	61.9	48.7	35.2	14.7	47.4	25.8	29.4	35.3	28.3	43.7	29.8
PointCNN [35]	45.8	94.4	70.9	71.5	54.5	45.6	31.9	32.1	61.1	32.8	75.5	48.4	47.5	16.4	35.6	37.6	22.9	29.9	21.6	57.7	28.5
PointConv [65]	55.6	94.4	76.2	73.9	63.9	50.5	44.5	47.2	64.0	41.8	82.7	54.0	51.5	18.5	57.4	43.3	57.5	43.0	46.4	63.6	37.2
SPH3D-GCN [33]	61.0	93.5	77.3	79.2	70.5	54.9	50.7	53.2	77.2	57.0	85.9	60.2	53.4	4.6	48.9	64.3	70.2	40.4	51.0	85.8	41.4
KPConv [59]	68.4	93.5	81.9	81.4	78.5	61.4	59.4	64.7	75.8	60.5	88.2	69.0	63.2	18.1	78.4	77.2	80.5	47.3	58.7	84.7	45.0
SegGCN [32]	58.9	93.6	77.1	78.9	70.0	56.3	48.4	51.4	73.1	57.3	87.4	59.4	49.3	6.1	53.9	46.7	50.7	44.8	50.1	83.3	39.6
DCM-Net [52]	65.8	94.1	80.3	81.3	72.7	56.8	52.4	61.9	70.2	49.4	82.6	67.5	63.7	29.8	80.6	69.3	82.1	46.8	51.0	77.8	44.9

F. Mesh Simplification Efficiency

We summarize the statistics of vertex and facet sizes of the full-resolution meshes in ScanNet [12]. The minimum, maximum, average of (vertex, facet) number are $(9K, 16K)$, $(553K, 1063K)$, and $(150K, 286K)$. The standard deviations are $(81K, 155K)$. Similarly, we apply the proposed simplification algorithm to decimate all room samples in ScanNet into a mesh of 65536 vertices. The runtime comparison of our algorithm and QEM are shown in the *left* plot of Fig. F.1. We also

test the runtime of our algorithm while decimating the room meshes into different resolutions, for which we use identical configurations to those for S3DIS. The results are shown in the *right* plot of Fig. F.1. It can be noticed from the figure that our conclusions based on S3DIS dataset consistently hold for ScanNet.

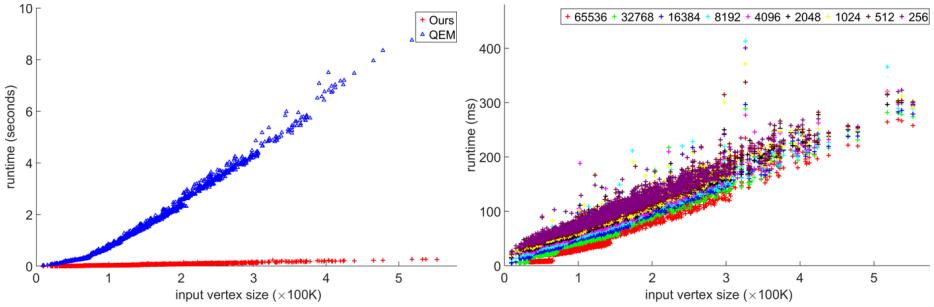

Figure F.1: The left figure compares the runtime of our mesh decimation method with QEM. The data is plotted by decimating every sample in ScanNet to a mesh of 65536 vertices by both methods. The right figure shows the time taken by our simplification algorithm to decimate all mesh samples in ScanNet to different vertex sizes, including 65536, 32768, 16384, 8192, 4096, 2048, 1024, 512, and 256.

References

[1a] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3D semantic segmentation of point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 716–724, 2017.

[2a] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs go as deep as CNNs? In Proceedings of the IEEE International Conference on Computer Vision, 2019.

Table F.1: Input representations of different methods on the S3DIS dataset.

Methods	Neural Element			Input Features	
	Voxel	Point	Super-Point	Geometric	Texture
SPG [30]	–	–	✓	[position, observation, geometrics]	$[r, g, b]$
SSP+SPG [29]	–	–	✓	[position, radiometry]	
SEGCloud [58]	✓	–	–	occupancy $\mathbf{1}_o$	
PointNet [43]	–	✓	–	$[x, y, z]$	
Tangent-Conv [57]	–	✓	–	[distance to tangent plane, height, normal]	
PointCNN [35]	–	✓	–	$[x, y, z,$ normal$]$	
GACNet [62]	–	✓	–	[height, eigenvalues]	
SPH3D-GCN [33]	–	✓	–	$[x, y, z]$	$[r, g, b]$
KPConv [59]	–	✓	–	$[1, x, y, z]$	
SegGCN [32]	–	✓	–	$[x, y, z]$	
DCM-Net [52]	–	✓	–	$[x, y, z,$ normal$]$	
PicassoNet (Proposed)	–	✓	–	$[x, y, z]$	$[r, g, b]$