# FKAConv: Feature-Kernel Alignment for Point Cloud Convolution

## Supplementary material

Alexandre Boulch[1], Gilles Puy[1], and Renaud Marlet[1,2]

[1] valeo.ai, Paris, France
[2] LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France

We present complementary information about the FKAConv paper. In Section A, we detail the network architectures used for classification and semantic segmentation. In Section B, we briefly describe the datasets used for experiments. In Section C, we discuss the use of a learned normalization of the support point neighborhoods. Finally, in Section D, we provide more qualitative results on the semantic segmentation test datasets.

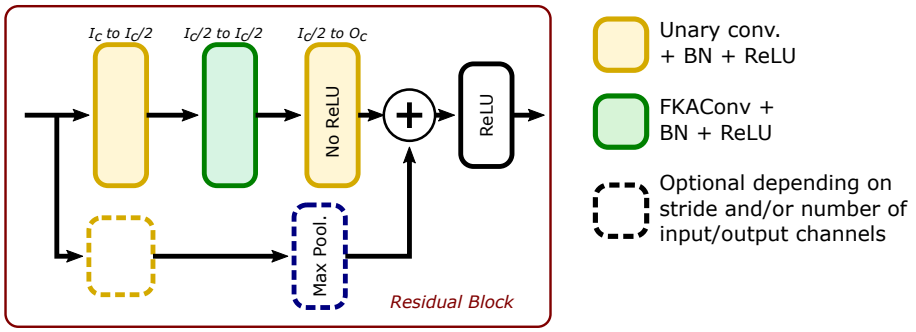
## A Network details

As mentioned in Section 5.1 of the paper, the networks we use in our experiments are based on the residual network architecture in KPConv [5], for which we replace the convolution by FKAConv and the neighborhood construction by our point sampling with space quantization. We detail here the main components.

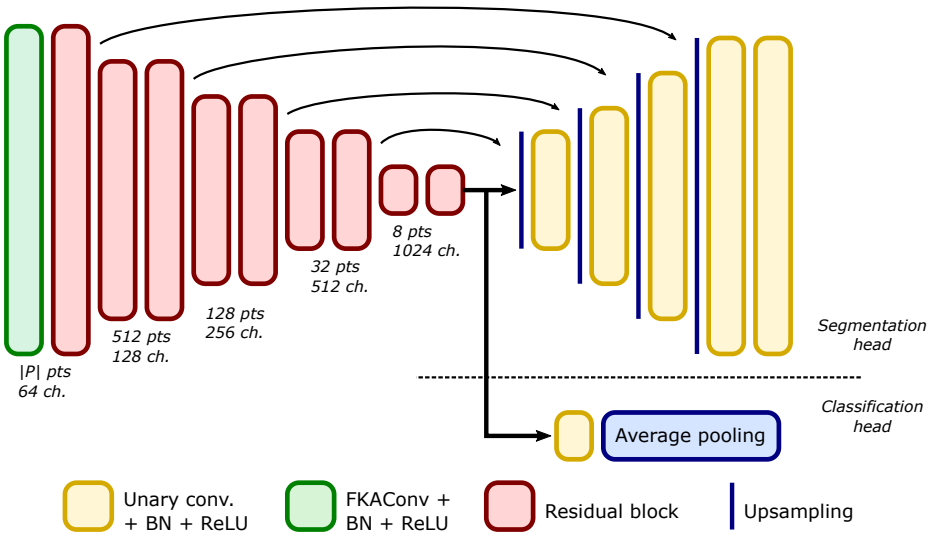
**Residual block.** The residual block in Fig. 1 (a) is the main module of our networks. This block is made of an FKAConv layer placed between two linear layers. The residual connection has one *optional* linear layer and one *optional* max-pooling layer. The *optional* linear layer is used only when the number of input channels is different from the number of output channels, and the *optional* max-pooling layer is used if the cardinality of the support points is different from the cardinality of the input points.

**Classification and segmentation networks.** The two networks for these tasks are presented in Fig. 1 (b). They share the same encoder structure, i.e., a FKAConv layer and 9 residual modules with a progressive reduction of the point cloud size.

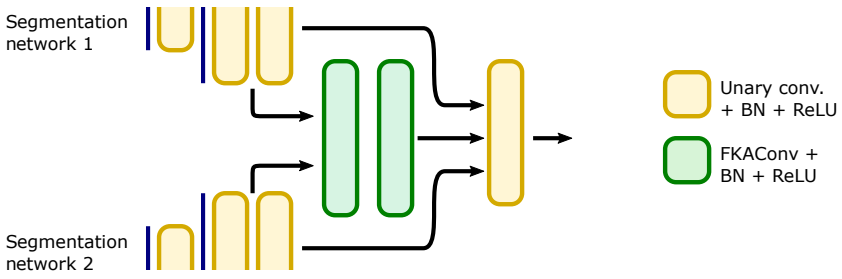
- The classification network has an extra point-wise fully-connected layer (or unary convolutional) with its output dimension equal to the number of classes. The final prediction is done by averaging the scores of the 8 final support points.
- The segmentation network has an encoder-decoder structure. The decoder is a stack of 5 unary layers with, nearest-neighbor up-sampling between each layer. We use skip connections from the encoder to the decoder: the




(a) Residual block. $I_c$ (resp. $O_c$) is the number of input (resp. output) channels.




(b) Classification and segmentation networks.




(c) Fusion module.

**Fig. 1.** Network architectures used for semantic segmentation and classification.

target points for up-sampling are the support points at the corresponding scale in the encoder, and the features from the encoder and the decoder are concatenated at each scale.

**Fusion network.** The fusion module presented in Fig. 1 (c) is identical to the module from the official repository of ConvPoint [2], except that it uses our proposed convolution. It is made of 3 layers: 2 FKAConv layers and 1 unary layer. The features from the penultimate layer of both segmentation networks are concatenated and given as input to the first FKAConv layer. The output of the second FKAConv layer is then concatenated with the predictions of the two segmentation networks and given to the unary layer.

**Parameters of the convolution.** In order to keep the setup simple, we use the same parameters for all FKAConv layers. The neighborhood size of the support points is fixed to 16 and we use 16 kernels. In the encoder, each odd residual block but the first reduces the number of support points from |P| to 512, 128, 32, and finally 8.

## B Datasets

We evaluate our convolution on three different tasks: object classification, part segmentation and semantic segmentation.

**Classification** is evaluated on ModelNet40 [6]. It contains 12,311 point clouds sampled from CAD models of 40 different categories. 9843 shapes are used for training, 2468 for testing.

**Part segmentation** is done on Shapenet [7]. It is composed of 16 object category, each category being annotated with 2 to 6 part labels. As a pre-processing, all models are first normalized to the unit sphere. In our implementation, the network has 50 outputs (one for each part) and the loss and scores are computed per object category.

**Semantic segmentation** is evaluated on 3 different indoor and outdoor datasets: S3DIS [1], NPM3D [4] and Semantic8 [3].

- S3DISS [1] is a subset of the 2D-3D-S dataset for semantic segmentation of building interiors. The data are acquired over 6 building floors with an RGBD camera. Each points is annotated with one of 13 labels: 12 semantic labels (floors, tables, chairs, etc.) and 1 label for a "clutter" class, mostly including office supplies. The evaluation is done using a 6-fold cross validation.
- NPM3D [4] is a lidar dataset for large-scale outdoor semantic segmentation. Points were acquired in 4 sites using a car equipped with lidar. 10 classes of urban entities are labeled, such as impervious surface, poll or pedestrian.

– Semantic8 [3] is the main dataset in the Semantic3D benchmark suite. It contains 30 lidar scenes, 15 for training and 15 for testing. Over 4 billion points are labeled with 8 classes such as building, vegetation and car, and a challenging class for scanning artifacts. The test set is particularly difficult as it covers several diverse scenes such as city streets, villages or old castles.

## C *K*-nearest neighbors search with learned neighborhood normalization

As described in Section 4 of the paper, we normalize the size of point neighborhoods. After recalling the principle of this normalization, we analyze here its effect empirically, for different datasets and for different layers in our networks.

To normalize the neighborhoods, we estimate an average neighborhood radius $r_t$ using an exponential average of the actual neighborhood radius $\hat{r}_t$ seen at training time in the successive batches indexed by $t$ (see Equation (1) below where $m$ is the momentum parameter). The point coordinates $(\mathbf{p}_i)_{1\leq i\leq k}$ of the $k$ nearest neighbors of a support point $\mathbf{q}$ are then centered and normalized as specified in Equation (2), yielding normalized points $(\hat{\mathbf{p}}_i)_{1\leq i\leq k}$:

$$r_t = \hat{r}_t * m + r_{t-1} * (1 - m), \tag{1}$$

$$\hat{\mathbf{p}}_i = (\mathbf{p}_i - \mathbf{q})/r_t. \tag{2}$$

We also proposed a gating mechanism to reduce, if needed, the negative effect of faraway points. Given the centered and normalized points $(\hat{\mathbf{p}}_i)_i$ computed above, the spatial gate weight $\mathbf{s} = (s_i)_i$ satisfies:

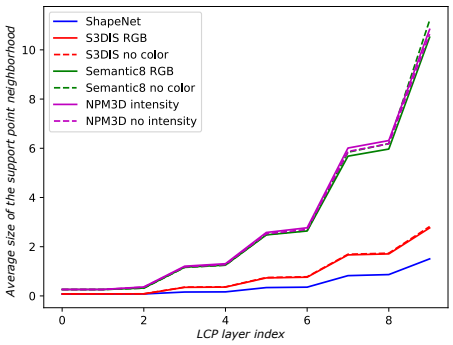
$$s_i = \sigma(\beta - \alpha||\hat{\mathbf{p}}_i||_2), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, and $\alpha$, $\beta$ are parameters to learn.

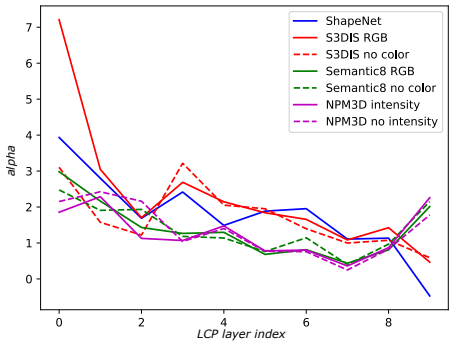
**Average neighborhood radius $r_t$.** First, we study the estimated neighborhood size $r_t$ at each layer of the encoder after training. This size is the averaged radius of the smallest sphere centered on the support point and encompassing the 16 nearest neighbors. We present in Fig. 2 (a) the evolution of this radius as a function of the layer's depth, for different datasets.

We observe that this radius is directly linked to the size of the bounding box of the input point cloud in the 3D space, i.e., to the size of the point cloud pillars (vertical infinite cylinders of diameter 8 meters for Semantic8 and NPM3D, and 2 meters for S3DIS) or to the size of the ShapeNet's CAD models.

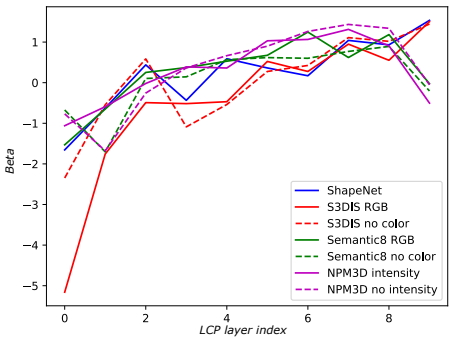
**Weighting function.** In Figs. 2 (b-c), we plot the values of parameters $\alpha$ and $\beta$ as a function of the depth of the FKAConv layers. First, we observe that these parameters, which are optimized with the network, take values that are different from the initial values that are set at the training initialization, i.e., $\alpha = 1$ and $\beta = 1$. Second, the values of $\alpha$ and $\beta$ are similar after training on



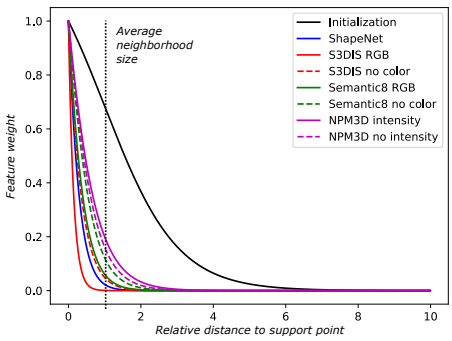

(a) Average neighborhood radius at each layer.




(b) $\alpha$ learned at each layer.

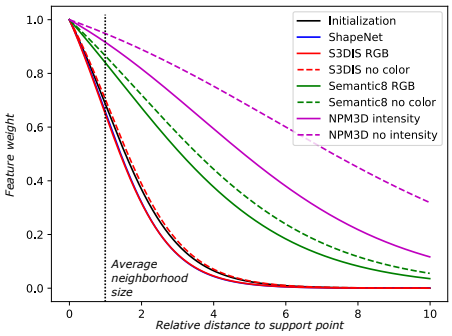



(c) $\beta$ learned at each layer.




Influence of the distance to support point on feature weight

(d) for the first FKAConv layer.




(e) for the $7^{\text{th}}$ residual block.

**Fig. 2.** Behavior of the learned neighborhood normalization and feature weighting across the segmentation network for various datasets.

Semantic8 and NPM3D. This could be expected as both datasets share common characteristics: outdoor urban scenes, same pillar size. Third, we observe the same global variations of the curves on all datasets: $\alpha$ tends to decrease with the depth while $\beta$ tends to increase. As a result, the transition of the gating function becomes wider in deeper layers, i.e., deeper layers tend to take into account more far away points.

We illustrate this phenomenon on Figs. 2 (d-e), where we represent the weighting function after optimization for the first and for the seventh FKAConv layer of the network. For comparison purpose, we normalize the curves by setting the weight to 1 at distance 0. The black curve is the initial function, before optimization.

At the first layer, only the neighboring points that are very close to the support point are taken into consideration to estimate matrix $\mathbf{A}$. We hypothesize that, in the absence of noise in data, a small neighborhood is sufficient to estimate local geometric features.

On the contrary, in the $7^{\text{th}}$ layer, all the neighboring points are taken into consideration to compute $\mathbf{A}$. At this stage, the number of support points is small and each point carries features that are discriminating for the task. The network considers all available information, including points that are faraway from the support points.

**Influence on performance.** To quantify the impact of the neighborhood normalization and gating mechanism, we trained a segmentation network using five different configurations.

- *Baseline (no normalization).* We do not normalize the neighborhood coordinates. The network sees different neighborhood sizes.
- *Baseline (normalization to the unit ball).* Each neighborhood is normalized into the unit sphere, regardless of its original size.
- *Learned normalization + fixed gating at $r$.* The radius used for normalization is estimated using the proposed exponential moving average. The gating mechanism is replaced by hard-thresholding: features of all points outside the ball with the learned radius $r$ are set to zero.
- *Learned normalization + fixed gating at $2r$.* Same as above but using a radius of $2r$ for hard-thresholding.
- *Our approach.* The radius used for normalization and the gating function are learned as proposed.

The results, reported in Table 1, show a slight improvement using our approach with respect to the baselines: the mean class intersection over union (mcIoU) increases by 0.2 point and the instance average intersection over union (mIoU) by 0.1 point. This gain may seem small, but it is significant on this dataset as the performance are close to saturation in the leaderboard.

We also observe that the learned gating is an important factor of the success of our approach. Fixed gating leads to a performance drop (see Table 1). This is

**Table 1.** Impact of the neighborhood normalization and gating on ShapeNet.

| Method | mcIoU | mIoU |
|---|---|---|
| Baseline (no normalization) | 84.3 | 85.3 |
| Baseline (normalization to unit ball) | 84.6 | 85.6 |
| Learned normalization + fixed gating | | |
| $s_i = 1$ if $d(\mathbf{p}, \mathbf{q}) < r_t$, and $s_i = 0$ otherwise. | 83.8 | 84.8 |
| $s_i = 1$ if $d(\mathbf{p}, \mathbf{q}) < 2r_t$, and $s_i = 0$ otherwise. | 84.0 | 85.2 |
| ($\mathbf{p}$ *denotes a point in the neighborhood of* $\mathbf{q}$, *and* $r_t$ *is the estimated radius in Equation (1).)* | | |
| Our method (learned normalization and gating) | **84.8** | **85.7** |

due to the fact that hard-thresholding suppresses too much information, particularly in the late stages of the network where the neighborhoods are less regular and more subject to size variation.

## D Qualitative results of segmentation

Finally, we provide more visual results illustrating our semantic segmentation predictions on datasets NPM3D (Fig. 3), Semantic8 (Fig. 4) and S3DIS (Fig. 5).

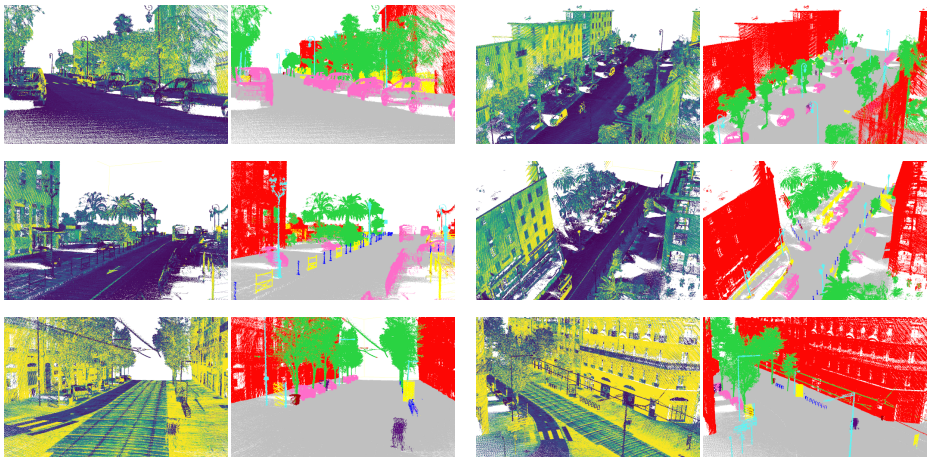


**Fig. 3.** Visual results of our predictions on the test scenes of the NPM3D dataset. The input data is colored according to lidar intensity, from blue (low intensity) to red (high intensity) through green, yellow and orange. Colored, predicted, semantic segments are immediately on the right to lidar images. The segmentation results are obtained with the fusion model (intensity + geometry).

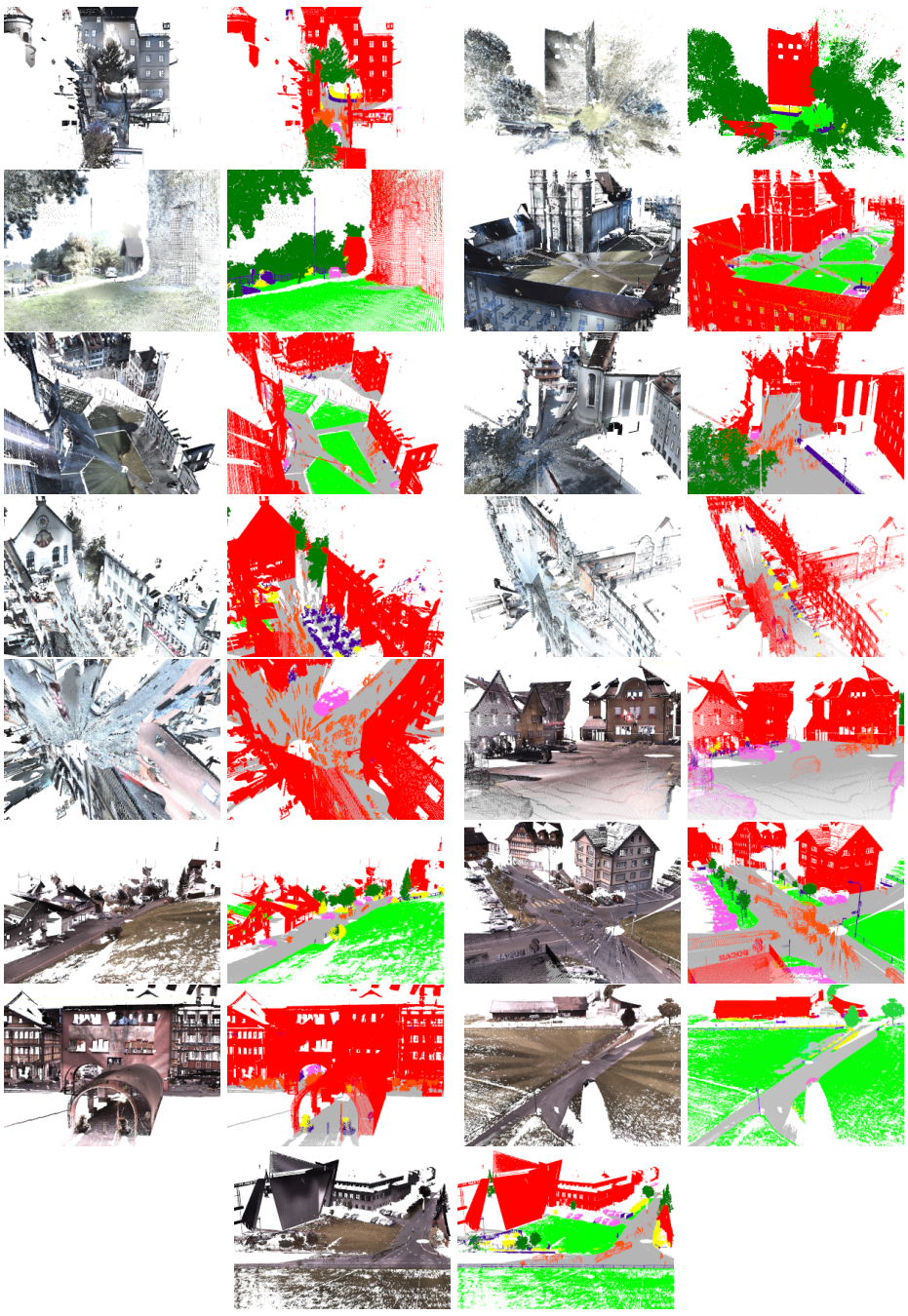


**Fig. 4.** Visual results of our predictions on the 15 test scenes of the Semantic8 dataset. The segmentation results are obtained with the fusion model (RGB + geometry).

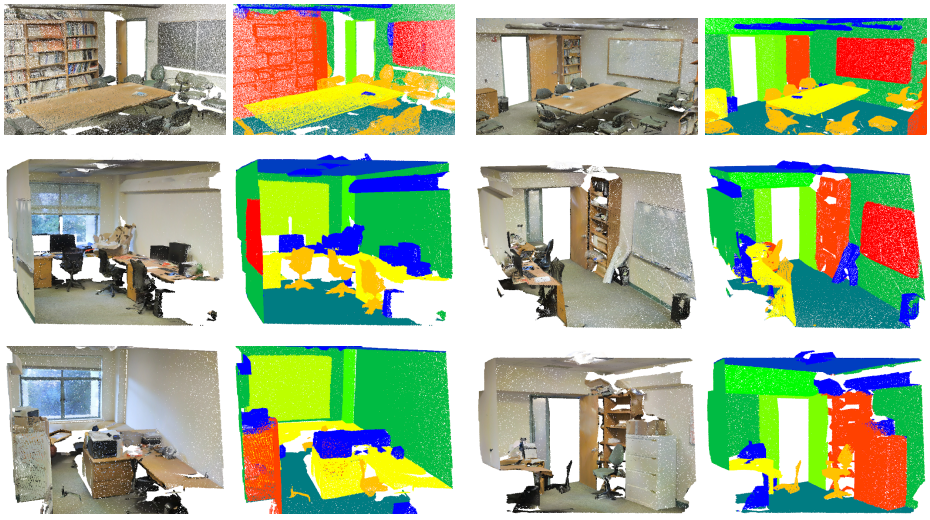


**Fig. 5.** Visual results of our predictions on the S3DIS dataset obtained with the fusion model (RGB + geometry).

## References


1. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3D semantic parsing of large-scale indoor spaces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1534–1543 (2016)
2. Boulch, A.: ConvPoint: Continuous convolutions for point cloud processing. Computers & Graphics **88**, 24 – 34 (2020)
3. Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K., Pollefeys, M.: Semantic3D.net: A new large-scale point cloud classification benchmark. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Annals). vol. IV-1-W1, pp. 91–98 (2017)
4. Roynard, X., Deschaud, J.E., Goulette, F.: Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. International Journal of Robotics Research (IJRR) **37**(6), 545–557 (2018)
5. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: KPConv: Flexible and deformable convolution for point clouds. In: IEEE International Conference on Computer Vision (ICCV) (2019)
6. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1912–1920 (2015)
7. Yi, L., Kim, V.G., Ceylan, D., Shen, I., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L., et al.: A scalable active framework for region annotation in 3D shape collections. ACM Transactions on Graphics (TOG) **35**(6), 210 (2016)