Spectral Graph Filter Learning for Point Labeling in Airborne LiDAR Data

Yinan Zhang, Eda Bayram
EPFL

***Abstract*–Semantic labeling is an important task for automated analysis of 3D point clouds produced via airborne laser scanning. It has been usually addressed by extracting pre-determined geometric features on 3D data[1], [2], or by training a classifier based on the features on the reduced dimension. In this study, we propose an algorithm for learning interpretable features for point-wise labeling of airborne LiDAR data. Our framework relies on spectral graph filter learning within a logistic regression scheme. We adopt a distributed implementation of graph filters, which leads to a scalable learning scheme and transferable features. Experiments show the potential of the proposed algorithm in distinguishing mostly confusing tree and roof points in a LiDAR scene.**

***Keywords*– 3D semantic labeling, Light Detection and Ranging (LiDAR), Graph Signal Processing**

I. Introduction

Thanks to the development of photogrammetry, geospatial data can be collected in an automatic fashion. Airborne Laser Scanning data provides quite dense and highly accurate, yet unorganized point cloud descriptions of the earth surface. The point-wise semantic labelling task of 3D Aerial LiDAR point clouds is still challenging due to the irregular point distribution.

Existing filtering algorithms usually fit the data on a 2-D grid, thus leading to inaccuracies due to resampling and interpolation[3]. Blomley et al. proposed a classification framework considering geometric features extracted from the local neighborhoods, yet some classes have similar geometric properties[2]. Graph convolutional neural networks [4], [5] can also be used for semantic segmentation in 3D point clouds. Despite high accuracy, this model lacks interpretability and the model parameters are not accountable.

Our target is to distinguish the points belonging to edges of a roof object or a tree object. Our method is based on spectral graph filtering where we adopt the graph signal representation model introduced in [6] to treat the irregular point cloud data produced by LiDAR. For this purpose, we embed the land view of the data, ie., (x,y) coordinates of the point cloud, into an unweighted and undirected graph structure that underlies the depth information, i.e., z coordinates of the point cloud, which we consider as the main signal regarding the nature of Airborne Laser Scanning process. Our main assumption is the following: the content of the signal existing on different classes relies on different graph frequency range. Since the depth change between roof points and neighboring ground points is very large, roof edge points are more likely to contain high-frequency signals, so are some tree points. Different from [6], which uses a priori determined graph filter, we aim at learning the filters to distinguish the points belonging to a certain class via logistic regression. We parametrize the filters via a set of polynomial coefficients in a distributed way on the graph structure, which permits a scalable learning scheme. The framework is shown in Fig. 1.

We use the airborne laser scanning data in Vaihingen provided by International Society for Photogrammetry and Remote Sensing (ISPRS) for 3D Semantic Labeling Contest[7]. The results show that filter learning can help distinguish roof edge points and tree points which are often confused by a randomly constructed high pass filter. The learned filters on a certain point cloud can be transfered to another new point cloud with a different number of points for the classification task. Our contribution is three-fold: (i) we present a filter learning framework for the graph signal representation model designed for Airborne LiDAR point clouds, (ii) we propose a polynomial graph filter design which yields a scalable learning scheme, (iii) the learned filter parameters are interpretable and transferable.

The rest of the paper is organized as follows. Section II introduces basics of graph signal filtering. Section III explains graph filter learning through logistic regression and the post-processing step. Section IV introduces ISPRS dataset and discuss experiment results. Lastly, we conclude in Section V. The code is available on github[1].

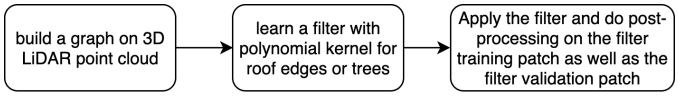


Fig. 1: Workflow

II. Graph Signal Representation of Airborne LiDAR Data

The section introduces the graph signal representation model for the 3D Airborne LiDAR data.

A. Graph Construction

We construct an undirected and unweighted graph $\mathscr{G}\{\mathscr{V},\mathscr{E}\}$, where $\mathscr{V}$ represents a set of vertices and $|\mathscr{V}| = N$,

[1]https://github.com/yinanzhangepfl/Filter-Learning-LiDAR

and $\mathscr{E}$ represents a set of edges. Points in the point cloud are assigned to nodes on the graph, which are connected by an edge within a k-nearest neighbors structure based on the Euclidean distance defined on their (x,y) coordinates. The number of nearest neighbors k in the graph representation should be set to an appropriate value so that the nodes in the parallel scanning lines shall be connected in order to capture depth change between roof points (marked in red) and other ones (marked in blue). Assume $\boldsymbol{A}$ is the adjacency matrix of the graph. The element $\boldsymbol{A}_{ij}$ is set to be 1 if there is an edge between node i and node j and 0 elsewhere. Fig. 2 shows a graph constructed for a patch of point cloud. The red points belong to the roof while the blue points are from other classes.

Figure 3 shows a scene of intersecting scanning strips. In the upper part, there exist two scanning directions thus leading to denser points, while in the lower part, there is only one scanning direction thus resulting in sparser points. This phenomenon of abrupt density change may cause a problem that we will detect thinner roof edges in the region of sparser point density. An unweighted graph has the ability to handle unequal point density distribution, because edge weights in the region of sparser point density are the same as those in the region of denser point density.

The graph laplacian matrix is defined as $Ł := \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is a diagonal degree matrix[8]. In order to maintain consistent filter outputs on point clouds whose graph representation may rely on different spectral ranges, and the numerical stability of the algorithm, we normalize the laplacian matrix by its largest eigenvalue, namely $Ł/\lambda_{max}$, and denote it by $Ł_{norm}$ (hereinafter written as Ł for simplicity). In this way, the eigenvalues of Ł are constrained to the range $[0, 1]$.

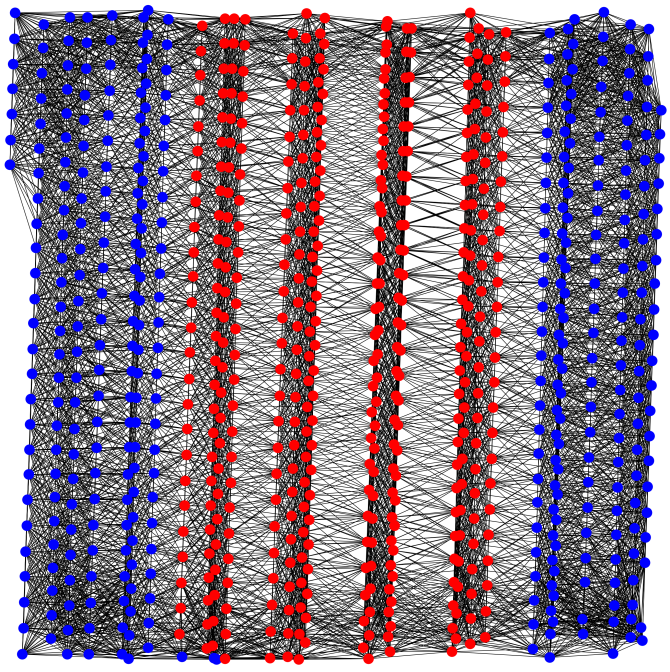


Fig. 2: Visualization of the graph structure

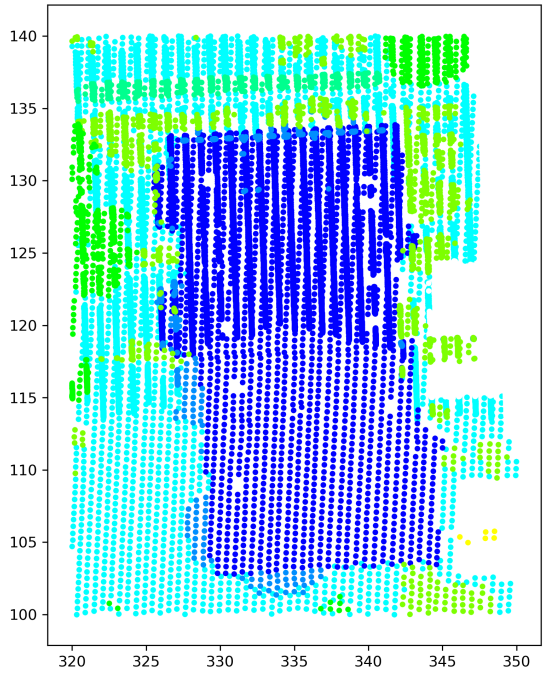


Fig. 3: Intersecting scanning stripes

[2]

B. Graph Signal

A vector $\boldsymbol{f}$ defined on the vertices of a graph is denoted as a graph signal[8], where the i-th element signifies the signal value of the i-th vertex. Here, the z coordinate of each point serves as the signal.

Let's take a look at eigendecomposition of the laplacian matrix Ł. Because Ł is a real symmetric matrix, it can be decomposed as $\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}$. The eigenvectors of Ł lie in the columns of $\boldsymbol{U}$. $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues of Ł lying on the diagonal. So we have:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \tag{1}$$

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_N \end{bmatrix} \tag{2}$$

The graph Fourier transform $\hat{\boldsymbol{f}}$ of the signal $\boldsymbol{f}$ in the time domain is further defined as: $\hat{\boldsymbol{f}} = \boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{f}$.

III. Graph Filter Learning via Logistic Regression

Similar to the classical signal processing, frequency filtering is an important concept for many graph representation learning tasks [9]. Filtering of a signal living on a graph structure can be realized via a function defined on the spectral domain of the graph, which is called a *graph kernel*. In the frequency domain, filtering is accomplished by the element-wise multiplication of the frequency representation of an input signal $\hat{\boldsymbol{f}}_{\boldsymbol{in}}$ with a graph kernel h, i.e., the component of the filter output, $\boldsymbol{f}_{out}$ at frequency λ is computed by $\hat{\boldsymbol{f}}_{\boldsymbol{out}}(\lambda) = \hat{\boldsymbol{f}}_{\boldsymbol{in}}(\lambda)h(\lambda)$.

Using the vertex domain representation of the signals, this can be simply written in the matrix-vector multiplication form as $\boldsymbol{f}_{out} = \boldsymbol{H}\boldsymbol{f}_{in}$, where the filtering operator $\boldsymbol{H}$ is calculated via applying the kernel h to the Laplacian matrix as $\boldsymbol{H} = h(Ł)$. We adopt a distributed implementation of the filtering operator [10], [11], where it is approximated via truncated polynomial expansion on Laplacian such as:

$$\boldsymbol{H} = \sum_{k=0}^{K} \boldsymbol{\alpha_k} Ł^k. \tag{3}$$

which provides K-hop localized filtering operation[4].

Thus, the filtered signal can be written as:

$$\begin{aligned} \boldsymbol{f}_{out} &= \boldsymbol{H}\boldsymbol{f}_{in} \\ &= \begin{bmatrix} \boldsymbol{f}_{in} & Ł\boldsymbol{f}_{in} & \dots & Ł^k\boldsymbol{f}_{in} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_0 \\ \vdots \\ \boldsymbol{\alpha}_K \end{bmatrix} \\ &= \boldsymbol{S}\boldsymbol{\alpha} \end{aligned} \tag{4}$$

where $\boldsymbol{\alpha}$ is a vector composed of polynomial coefficients to be learned, and $\boldsymbol{S}$ corresponds to an orthonormal basis of the Krylov subspace. Fortunately, we don't need to calculate the power of Ł, since $\boldsymbol{S}$ could be calculated by a sequential multiplication of Ł as $\boldsymbol{S} = \begin{bmatrix} \boldsymbol{f}_{in} & Ł\boldsymbol{f}_{in} & Ł(Ł\boldsymbol{f}_{in}) & \dots & Ł(Ł^{k-1}\boldsymbol{f}_{in}) \end{bmatrix}$. By this means, the computational cost is reduced a lot.

Through the principle of logistic regression, the cross entropy loss is adopted:

$$\mathscr{L} = \sum_{i=1}^{N} -\boldsymbol{y}_n \log(\sigma(\boldsymbol{f}_n - b)) - (1 - \boldsymbol{y}_n)\log(1 - \sigma(\boldsymbol{f}_n - b)) \tag{5}$$

where $\boldsymbol{f_n}$ is the n-th element of the filtered signal and $\sigma(\cdot)$ is the logistic function: $\sigma(z) = \frac{e^z}{1+e^z}$. The true binary label of the n-th element is $\boldsymbol{y}_n$, and this value can be either 1 (positive) or 0 (negative). When $\boldsymbol{f_n}$ is greater than or equal to the b, the n-th point is predicted as positive, else it is predicted as negative. We need to tune parameters $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_K$ to minimize the loss function.

IV. Experimental Results on ISPRS 3D semantic labelling dataset

A. ISPRS Dataset

The airborne laser scanning data from Vaihingen is composed of one training set and one test set. For both parts, information such as XYZ coordinates, reflectance and return count is given, but we only use spatial coordinates for the scope of this project. Points belong to one of these 9 classes: powerline, low vegetation, impervious surfaces, car, fence/hedge, roof, facade, shrub and tree. In the training set which we work on, reference information about the label of each point is given. The training set contains 753876 cloud points in total. Data points are randomly distributed in each category as shown below. However the train set exists duplicated points, with the proportion of 5.43% of the dataset. After deleting those points, there remains 712930 points. A small patch of the training scene is extracted and visualized in the 2D domain in figure 4

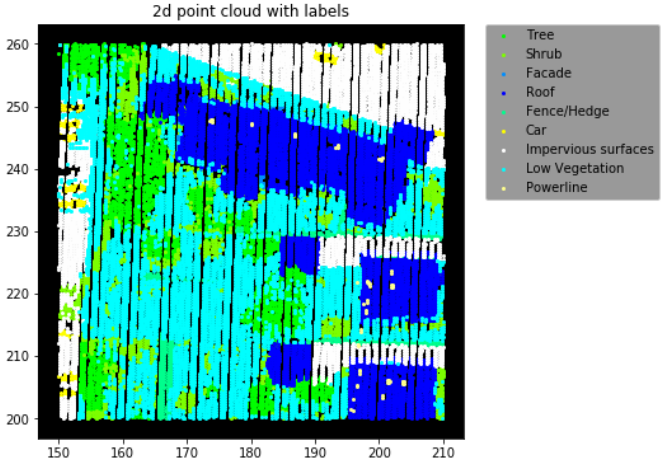


Fig. 4: 3D point cloud

label index	label name	# points
0	Powerline	546
1	Low Vegetation	180850
2	Impervious surfaces	193723
3	Car	4614
4	Fence/Hedge	12070
5	Roof	152045
6	Facade	27250
7	Shrub	47605
8	Tree	135173

TABLE I: Number of points in each class in the train set

B. Metrics for Performance Evaluation

We use confusion matrices to evaluate the constructed filters. A confusion matrix is a table made up of two rows and two columns that gives the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Type I error and type II error are calculated using formulas below:

$$TypeI\ error\ rate(false\ positive\ rate) = \frac{FP}{TP + FP} \tag{6}$$

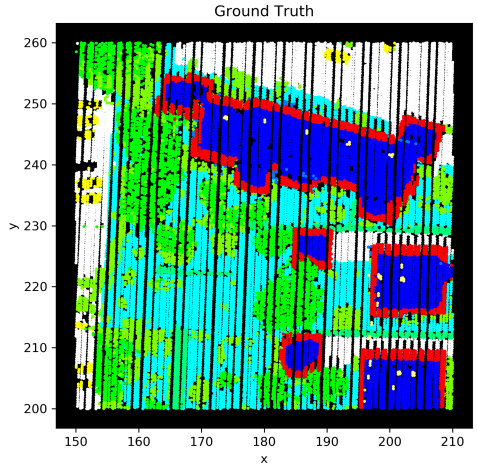
$$TypeII\ error\ rate(false\ negative\ rate) = \frac{FN}{TN + FN} \tag{7}$$

C. Ground Truth for Roof Edges

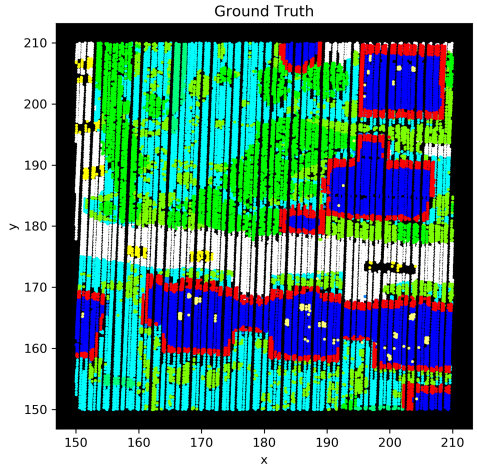
An undirected and unweighted graph whose nodes are connected by edges within a 30-nearest neighbors structure is used to prepare the ground truth data of roof edges. Signal is set to be 1 for roof points and 0 for others. Then, a filter with Mexican Hat kernel is applied to the graph signal. The filter

[2]check the legend in figure 4 to see the label represented by each color

response with respect to graph frequency is $\lambda\tau \exp(-\lambda\tau)$ where $\tau = 1.5$. Set the threshold to be 0.08, the filter output on two intersecting patches is shown in figure 5



(a) filter training patch



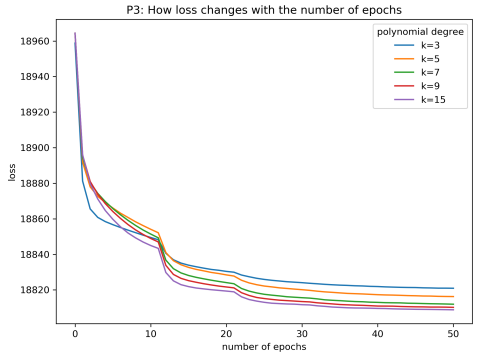
(b) filter validation patch

Fig. 5: ground truth for roof edges in two intersecting patches

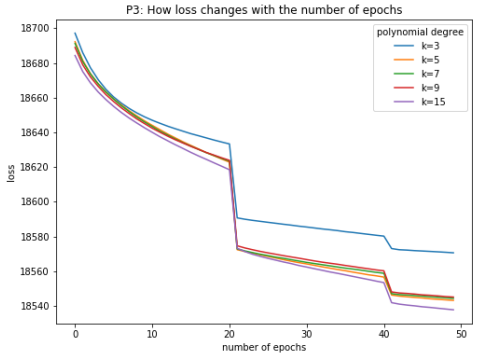
D. Filter Learning

For a start, build a graph as described in section II-A and connect a particular node to 30 nearest neighbors of it on the filter training patch as shown in figure 5a as well as the filter validation patch as shown in figure 5b. We train a filter for roof edges and a filter for trees on the filter training patch, then test their performance on the filter validation patch. In practice, $\boldsymbol{\alpha}_0$ is set to be 0 for the reasons mentioned below. On the one hand, including the term $\boldsymbol{\alpha}_0 \boldsymbol{f}_{in}$ in the expression of $\boldsymbol{f}_{out}$ does not help us learn depth changes between roof edges points (or tree points) and their neighboring points. On the other hand, the learned filter does not has transferability if we learn $\boldsymbol{\alpha}_0$. Starting from a Mexican Hat filter with $\tau = 1.5$, we adjust the number of polynomial terms k to be 3, 5, 7, 9 and 15 respectively and adopt adaptive learning rate during the training process. To find a good set of parameters, stochastic gradient descent method is implemented. When learning roof edges, the learning rate is set to be $1e-4$ initially, and after every 10 epochs, this values is reduced by half. When learning trees, the initial learning rate remains unchanged but is reduced by half every 20 epochs. For both roof edge and tree detection, choose $k = 15$.

The bias in the loss function defined in equation 5 is set to be 0.2. Figure 6 shows that training loss goes down during the filter learning process, and decaying learning rate can speed up the learning process when the training loss plateaus. The initial training error rate is 20.7 % and it decreases by nearly 4% after filter learning for roof edges. However, for trees, the training error rate even increases a little bit. This is because the graph frequency range of trees overlaps with that of roof bodies. Luckily, we could remove most of roof body points in the post-processing step. Figure 7 visualizes learned filters, from which we can see that a high pass filter is more suitable to recover roof edge points. This is because a large depth change between roof points and neighbouring points create high frequency features.

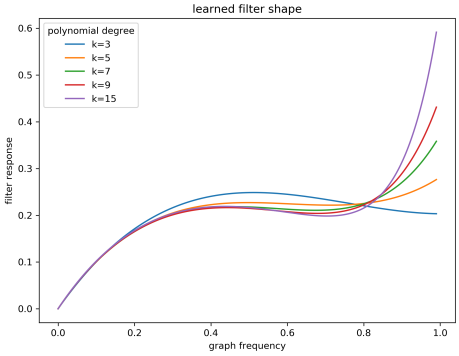


(a) when learning roof edges

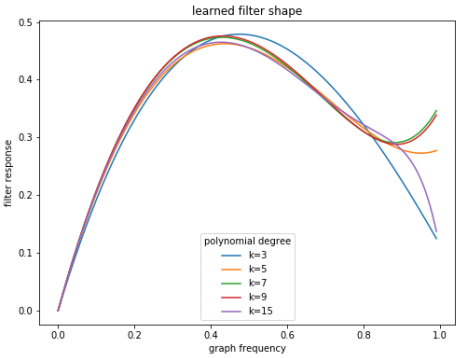


(b) when learning trees

Fig. 6: training loss vs. # of epochs with k = 3, 5, 7, 9, 15



(a) when learning roof edges



(b) when learning trees

Fig. 7: learned filter shape with k = 3, 5, 7, 9, 15

		Prediction Outcome		
		P	N	Total
Actual Value	P′	1042	1147	2189
	N′	1533	27517	29050
	Total	2575	28664	31039

(a) roof edge detection on filter training patch

		Prediction Outcome		
		P	N	Total
Actual Value	P′	1141	1267	2408
	N′	1446	28028	29474
	Total	2587	29295	31882

(b) roof edge detection on filter validation patch

		Prediction Outcome		
		P	N	Total
Actual Value	P′	2749	2759	5508
	N′	1537	24194	25731
	Total	4286	26953	31239

(c) tree detection on filter training patch

		Prediction Outcome		
		P	N	Total
Actual Value	P′	1479	2496	3975
	N′	2243	25664	27907
	Total	3722	28160	31882

(d) tree detection on filter validation patch

TABLE II: confusion matrix

	type I error	type II error
filter training patch	59.5%	4.2%
filter validation patch	55.9%	4.3%

(a) roof filter

	type I error	type II error
filter training patch	35.9%	10.2%
filter validation patch	60.3%	8.9%

(b) tree filter

TABLE III: evaluation of constructed filters on the training patch and the validation patch

E. Post-processing and Patch Alignment

The aim of post-processing is to improve the performance of a single filter. We will apply roof edge filter and tree filter generated from the filter training patch to the filter validation patch and then do the post-processing step to see if the results are comparable.

After acquiring reds points from a roof edge filter or a tree filter, We build graph representations on them (30nn normalized binary graph is used). Then we calculate $\text{Ł}Z = \boldsymbol{M}$, where Z is the graph signal composed of Z coordinate. The edge points of a roof are almost at the same level, but the neighboring tree points are more fluctuating. So for roof edge result, we keep those points that have smaller $|m| < 0.15$. For tree result, we keep points that have larger $|m| >= 0.15$.

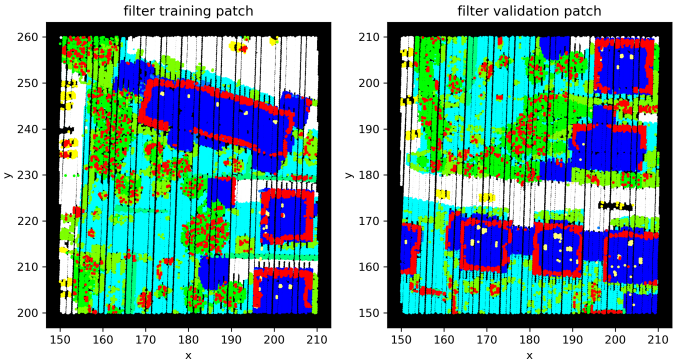
Table II shows that confusion matrices and table III shows type I and type II error rate when detecting roof edges and trees on different patches. Figure 8 visualizes the experiment results. For roof edges, results on the filter training patch and the filter validation patch are similar. This indicates that the constructed filter for roof edges has transferability. When we take a closer look at the results of the constructed tree filter, there is higher typeI error rate on the filter validation patch. This is because the frequency range of tree points is wider and also differs on different patches. The evidence is that if we only care about the error rates on the intersected part of these two patches, they are almost the same. Thus, the constructed roof edge filter is more powerful than the tree filter, since the frequency characteristics of roof edges are more distinguishable and consistent.

V. Conclusion

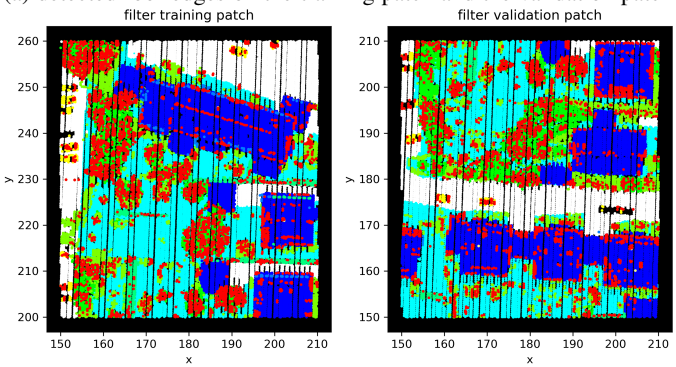
In this paper, we propose a scalable and interpretable learning scheme by constructing transferable graph filters for 3D point cloud labelling task. The post-processing step can help remove false positives and improve precision. It is found that the frequency features of roof edges are easier to capture than those of trees during the experiment. For further improvement, instead of training filters and then doing the post-processing step, we can complete both simultaneously to learn filter parameters based on the final result. Other models can also be adopted to interpret the filter kernel, such as Chebyshev polynomials.

References

[1] R. Blomley, B. Jutzi, and M. Weinmann, "3d semantic labeling of als point clouds by exploiting multi-scale, multi-type neighborhoods for feature extraction," 2016.



(a) detected roof edges on the training patch and the validation patch



(b) detected trees on the training patch and the validation patch

Fig. 8: detected points of interest

[2] R. Blomley and M. Weinmann, "Using multi-scale features for the 3d semantic labeling of airborne laser scanning data," *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 4, 2017.

[3] X. Meng, N. Currit, and K. Zhao, "Ground filtering algorithms for airborne lidar data: A review of critical issues," *Remote Sensing*, vol. 2, no. 3, pp. 833–860, 2010.

[4] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[6] E. Bayram, P. Frossard, E. Vural, and A. Alatan, "Analysis of airborne LiDAR point clouds with spectral graph filtering," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 8, pp. 1284–1288, 2018.

[7] ISPRS, "Isprs test project on 3d semantic labeling contest," http://www2.isprs.org/commissions/comm3/wg4/3d-semantic-labeling.html.

[8] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *arXiv preprint arXiv:1211.0053*, 2012.

[9] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[10] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via chebyshev polynomial approximation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 736–751, 2018.

[11] M. Contino, E. Isufi, and G. Leus, "Distributed edge-variant graph filters," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2017, pp. 1–5.

[12] G. Vosselman, "Slope based filtering of laser altimetry data," *International Archives of Photogrammetry and Remote Sensing*, vol. 33, no. B3/2; PART 3, pp. 935–942, 2000.

[13] J. Shan and S. Aparajithan, "Urban DEM generation from raw LiDAR data," *Photogrammetric Engineering & Remote Sensing*, vol. 71, no. 2, pp. 217–226, 2005.

[14] D. Wang, D. S. Yeung, and E. C. Tsang, "Weighted mahalanobis distance kernels for support vector machines," *IEEE Transactions on Neural Networks*, vol. 18, no. 5, pp. 1453–1462, 2007.

[15] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[16] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," *arXiv preprint arXiv:1901.10524*, 2019.

[17] D. Thanou and P. Frossard, "Distributed signal processing with graph spectral dictionaries," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015, pp. 1391–1398.

[18] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, "Reconstruction of graph signals through percolation from seeding nodes," *IEEE Transactions on Signal Processing*, vol. 64, no. 16, pp. 4363–4378, 2016.

[19] D. Thanou, D. I. Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Transactions on Signal Processing*, vol. 62, no. 15, pp. 3849–3862, 2014.

[20] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3d point cloud sequences," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, 2016.

[21] E. Bayram, E. Vural, and A. Alatan, "A graph signal filtering-based approach for detection of different edge types on airborne lidar data," in *Lidar Technologies, Techniques, and Measurements for Atmospheric Remote Sensing XIII*, vol. 10429. International Society for Optics and Photonics, 2017, p. 104290B.

[22] A. Cherqui, "Deep learning on graph for semantic segmentation of point ucloud," 2018. [Online]. Available: http://infoscience.epfl.ch/record/263797