Efficient Calculation of Multi-Scale Features for MMS Point Clouds

Keita Hiraoka[1], Genki Takahashi [2], Hiroshi Masuda [1]
[1] The University of Electro-Communications, Japan – keita.hiraoka@uec.ac.jp, h.masuda@uec.ac.jp
[2] Kokusai Kogyo Co., Ltd., Japan – genki_takahashi@kk-grp.jp

Commission II, WG II/2

Keywords: MMS, Point-Cloud, Labeling, Semantic Segmentation, Machine Learning, Multi-Scale Features

Abstract
Point clouds acquired by Mobile Mapping System (MMS) are useful for creating 3D maps that can be used for autonomous driving and infrastructure development. However, many applications require semantic labels to each point of the point clouds, and the manual labeling process is very time consuming and expensive. Therefore, there is a strong need to develop a method to automatically assigning semantic labels. For automatic labeling tasks, classification methods using multiscale features are effective because multiscale features include features of various scales of roadside objects. Multiscale features are calculated using points inside spheres of multiscale radii centered at each point in a point cloud. When calculating multiscale features that are useful for classifying MMS point clouds, it is necessary to calculate features using relatively large radii. However, when calculating multiscale features using wide range of neighbor points, existing methods, such as kd-tree, require unacceptably long computation time for neighbor search. In this paper, we propose a method to calculate multiscale features in practical time for semantic labeling of large-scale point clouds. In our method, an MMS point cloud is first divided into small spherical regions. Then, radius search using multiscale radii is performed, and multiscale features are calculated using those neighbor points. Our experimental results showed that our method achieved significantly faster computational performance than conventional methods, and multiscale features could be calculated from large-scale point clouds in practical time.

1. Introduction

Point clouds acquired by Mobile Mapping System (MMS) can be used for 3D map creation, infrastructure maintenance, and so on. In many applications, it is necessary to segment point clouds by adding a feature label to each point. Since it is very time-consuming to manually perform labelling tasks, there is a strong need to develop a method to automatically add feature labels to MMS point clouds.

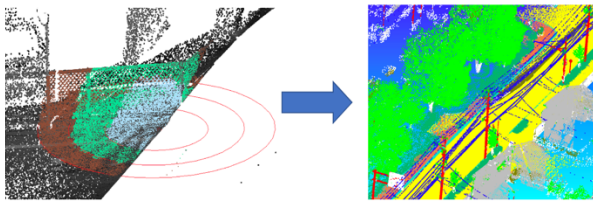
For automatic labeling tasks, classification methods using multiscale features are effective because multiscale features include features of various scales of roadside objects. Previous studies such as (Weinmann, 2017) have shown that multi-scale features are effective in automatically labeling point clouds with a high success rate. Multiscale features are calculated using points inside spheres with various radii centered at each point in a point cloud, as shown in Figure 1.

However, the calculation of multiscale features for large-scale point clouds often requires unacceptable computation time. This is because neighbor searches at various scales require a lot of computation time. Table 1 shows the computation time for multi-scale features of 4.8 million points when neighbor points were searched using FLANN (Muja and Lowe, 2009).

The kd-tree (Bentley, 1975) has been commonly used in neighbor search for point clouds. However, when the search radius is large, kd-tree is time-consuming because it traverses a large area of nodes in 3D space. The kd-tree is efficient when the search radius is small, but it is not efficient for large radii. Table 1 shows that as the radius increases by a factor of n, the computation time is approximately n squared. This is due to MMS point clouds tend to be distributed in planar regions.

In our experiments, as described in a later section, features computed using relatively large radii had a high contribution to the classification of MMS point clouds. However, in Table 1, it took more than 16 hours to classify only 4.8 million points using exact multi-scale features with radii of 1m, 2m, 3m and 5m. Since the state-of-the-art laser scanners for the MMS can measure more than 1 million points per second, the actual MMS point cloud contains far more points than this example. Therefore, it is difficult to obtain multi-scale features from actual MMS point clouds in practical time.

A straightforward way to address this problem is to down-sample point clouds instead of computing exact neighborhoods for each point. However, down-sampling may result in the loss of small-scale features. To avoid the loss of features, (Hackel, 2016) down-sampled point clouds at multiple scales with decreasing



(a) Neighbor points at multiple radii (b) Classified points
Figure 1. Classification using multi-scale features

Table 1. Calculation time for multi-scale features using FLANN

Radius	1m	2m	3m	5m	Total
Time	3035.7s	12322.2s	27498.2s	64236.7s	107092.8s
Rate	1	4.06	9.06	21.16	35.28

point density, and generated a separate search structure per scale level. In this method, approximate features are computed from down-sampled points rather than computing exact features from all points within a certain distance. Thus, computation time is greatly improved. However, it is not easy to determine the appropriate number of scales and their grid sizes for detecting various scale of features. It is difficult to predict in advance the contribution of features at each scale, especially in machine learning. Therefore, it is necessary to determine the scales experimentally.

Efficient neighbor search methods without down-sampling have also been proposed. (Nüchter, 2007) accelerated neighbor search by using cached kd-trees. This method can effectively detect neighbor points by using the history of node search in a kd-tree for previous point. However, it is difficult to dramatically reduce computation time with this method. As shown in Table 1, neighbor search for a large-scale point cloud takes an enormous amount of computation time. It is difficult to use multi-scale features in practical applications without dramatic improvement.

In this paper, we propose a new method to efficiently compute multi-scale features of point clouds and classify each point using machine learning. Unlike the method of (Hackel, 2016), our method computes multi-scale features using exact neighbor points without down-sampling the points. Therefore, our method does not suffer from the loss of small-scale features due to down-sampling. Our method achieves efficiency by computing accurate multi-scale features only for representative points in a two-step neighbor search, instead of computing the exact features for all points.

In the following section, we outline the proposed method. Section 3 describes our method for calculating multi-scale features, and Section 4 describes MMS-specific features. We describe experimental results in Section 5, and finally conclude our research in Section 6.

2. Overview

Our research goal is to classify each point in large-scale MMS point clouds in practical time. Roadside objects include traffic signals, utility poles, streetlights, guardrails, trees, and so on. The main task is to assign such a label to each point.

Figure 2 shows the overview of our method. In our method, each point is classified using multi-scale features and MMS-specific features. It is known that multi-scale features are effective for characterizing each point (Weinmann, 2017). The key issue is to effectively obtain neighbor points at multiple scales. In addition, for MMS point clouds, vehicle trajectories and scanlines can also be used to characterize each point. Since roadside objects are typically placed along roads, their labels are strongly related to their distances and heights from the vehicle trajectory. A scanline is defined as a series of points that are measured sequentially by the MMS. Since the length and curvature of each scanline reflect the shape and width of each object, they can be used as features for classification. In this paper, the features computed from the trajectory and scan lines are referred to as MMS-specific features. For classification, multi-scale features and MMS-specific features are concatenated to estimate the label of each point using the random forest (Breiman, 2001).

In our method, multi-scale features are computed using exact neighbor points without down-sampling. To speed up the calculation of multi-scale features, multi-scale neighborhoods are computed using two-step neighbor search, as shown in Figure 3.

In the first step, the entire point cloud is divided into small regions by radius search with a fixed small radius. This neighbor search is quite fast because it does not require traversing a wide range of nodes in the tree structure. In the second step, multi-scale neighborhoods are searched only using the representative point of each small region. This process is also fast because neighborhoods are searched from a relatively small number of representative points. Finally, the exact neighbor points are computed by considering the relationship between the radius of the small region and the radii at multi-scales.

3. Efficient Calculation of Multi-Scale Features

3.1 Dividing Point Cloud into Small Regions

Figure 4 shows the process of dividing a point cloud into small regions. First, an arbitrary point is selected from a point cloud, as shown in Figure 4(a). Then, neighbor points are searched within the distance r from the selected point (Figure 4(b)). In this

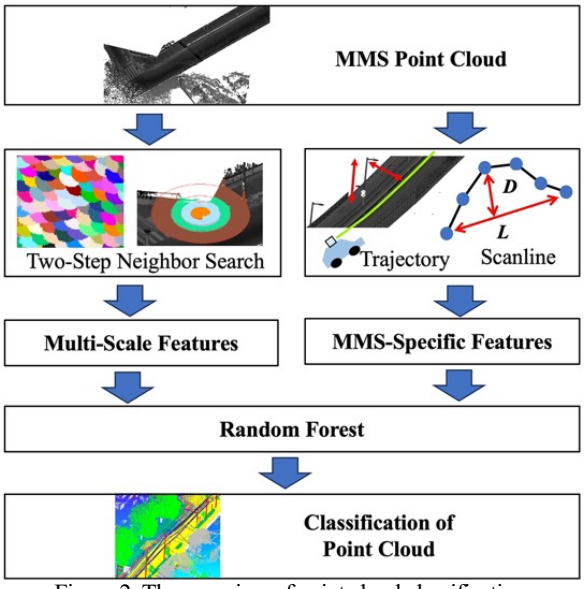


Figure 2. The overview of point cloud classification

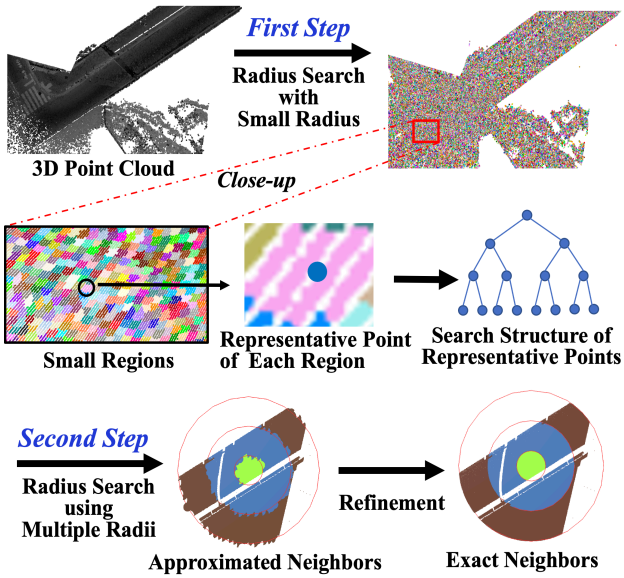


Figure 3. Overview of two-step neighbor search

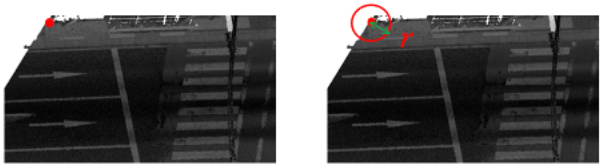
research, radius search is performed using FLANN (Muja and Lowe, 2009). As shown in Figure 4(c), the neighbor points are maintained as a small region, and a region number is assigned to each point in the small region. The next point is selected from points that do not have a region number, and a new small region is created from the neighbor points that are not included in any region (Figure 4(d)). A new region number is assigned to each point in the new small region. This process is repeated by selecting a point that do not have a region number (Figure 4(e). As a result, all points in the point cloud are subdivided into small regions, as shown in Figure 4(f).

In this research, the center point of radius search is called the representative point. Therefore, each small region has a representative point at the center of radius search. We note that some researchers call small regions with semantics as superpoints (Landrieu, 2018). However, in our research, a small region is simply a cluster of neighbor points, which are used for the second neighbor search. Therefore, we simply call a cluster of neighbor points as a small region.

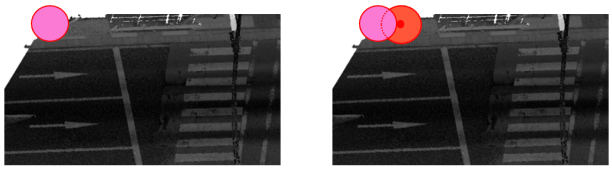
Table 2 shows the number of small regions and the calculation time when subdividing a point cloud with 4,787,033 points. The computation time for the subdivision into small regions is very short compared to the computation time shown in Table 1. Table 2 also shows the ratio of the number of points in the point cloud to the number of representative points.

Table 2. Number of regions subdivided from 4.8 million points.

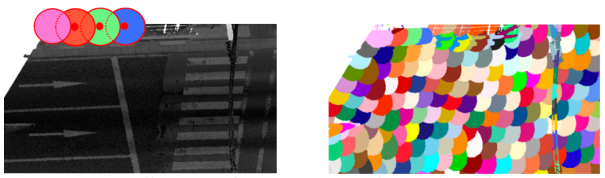
Radius r	10 cm	15 cm	20 cm	25 cm	30 cm
Regions	325,252	20,109	140,566	106,510	85,036
Rate	1/14.7	1/23.8	1/34.1	1/45.0	1/56.3
CPU Time	2.71 s	2.43 s	2.32 s	2.28 s	2.05 s



(a) Initial point (b) Radius search with r



(c) Initial small region (d) Next small region



(e) Repeat radius search (f) Small regions

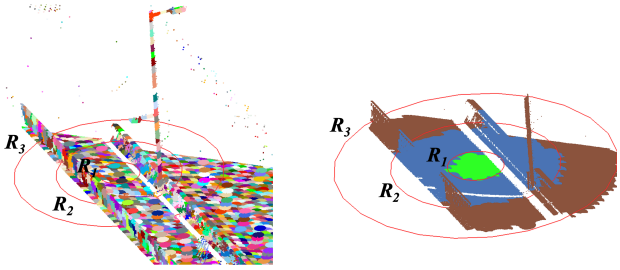
Figure 4. Dividing a point cloud into small regions

3.2 Neighbor Search using Representative Points

In the second step in Figure 3, radius search at multiple radii is performed only using representative points. Since the number of representative points is much smaller than the number of points in the point cloud, multi-scale radius search can be performed effectively.

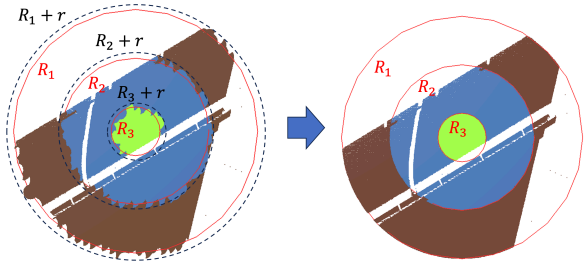
Let $\{R_i\}$ be the radii for multi-scale features. In this paper, R_i is assumed to be sufficiently large compared to the radius r of small regions. For each representative point, representative points within radius R_i are searched. Figure 5(a) shows radius search for representative points at multiple scales. The neighbor representative points are obtained for each of $\{R_i\}$ by radius search.

By collecting the points of small regions obtained as neighbourhoods, approximate neighbor points within radius R_i can be obtained, as shown in Figure 5(b). The obtained neighbor points exist within radius R_i with a deviation at most r. Thus, if r is sufficiently small compared to R_i, the neighbor points are a good approximation for radius search at radius R_i.



(a) Neighbor representative points (b) Approximated neighbors

Figure 5. Radius search at multi-scales (R_i =1m,3m,5m)



(a) Approximated neighbors (b) Exact neighbors

Figure 6. Refinement to obtain exact neighbor points.

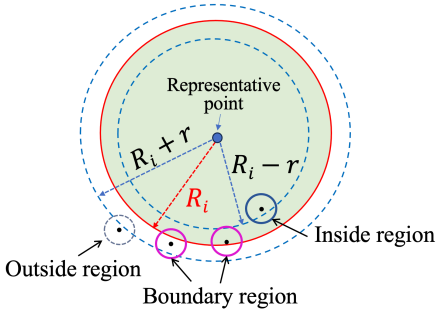


Figure 7. Criteria for refinement

3.3 Refinement for Exact Neighbor Points

In Figure 5(b), some points are inside the radius R_i but not included in the neighborhood, and some points are outside the radius R_i but included in the neighborhood. Such points can be detected efficiently and exact neighborhoods can be calculated, as shown in Figure 6.

For obtaining exact neighbor points, multi-scale radius searches are performed using the enlarged radii $\{R_i + r\}$. Let $\mathbf{p}_c$ be the center point of radius search. As described in 3.2, the approximated neighbor points can be obtained from the neighbor representative points calculated using an enlarged radius. Then, as shown in Figure 6(a), the approximated neighbor points include all of the exact neighbor points, because the points in each small region exist within radius r from the representative point of the small region.

Figure 7 shows three types of neighbor representative points. Let d be the distance between point $\mathbf{p}_c$ and a neighbor representative point. The following criteria can be used to determine if the points in a small region are included in the exact neighborhood.

(i) If $d > R_i + r$, no points in the small region are included in the exact neighbor points.
(ii) If $d \leq R_i - r$, all points in the small region are included in the exact neighbor points.
(iii) If $R_i + r \geq d > R_i - r$, points in the small region may be partly included in the exact neighbor points.

Only in case (iii), each point in the small region has to be checked whether the point is included in the exact neighbor points. This check can be made by checking whether the distance from $\mathbf{p}_c$ is less than R_i. Figure 6(b) shows the exact neighbor points calculated from points in Figure 6(a). Since the number of boundary regions is relatively small, the calculation of refinement can be done in a short time.

3.4 Multi-Scale Features

Table 3 shows feature values used in this research. Multi-scale features are calculated at each point using eigenvalues $\lambda_1, \lambda_2, \lambda_3$ $(\lambda_1 \geq \lambda_2 \geq \lambda_3)$. As shown in Table 3, various feature values can be calculated from eigenvalues (Weinmann, 2019). Eigenvalues are calculated using principal component analysis (PCA) from neighbor points at multiple scales.

Table 3. Multi-scale features

Linearity	$L_\lambda = (\lambda_1 - \lambda_2)/\lambda_1$
Planarity	$P_\lambda = (\lambda_2 - \lambda_3)/\lambda_1$
Sphericity	$S_\lambda = \lambda_3/\lambda_1$
Omnivariance	$O_\lambda = \sqrt[3]{\lambda_1 \cdot \lambda_2 \cdot \lambda_3}$
Anisotoropy	$A_\lambda = (\lambda_1 - \lambda_3)/\lambda_1$
Eigenentropy	$E_\lambda = -\sum_{i=1}^{3}(\lambda_i \ln \lambda_i)$
Sum of eigenvalues	$\sum_{\lambda,3D} = \lambda_1 + \lambda_2 + \lambda_3$
Change of curvature	$C_\lambda = \lambda_1/(\lambda_1 + \lambda_2 + \lambda_3)$
Verticality	$V = 1 - n_z$
Hight difference	$\Delta H_{3D} = z_{max} - z_{min}$
Height deviation	$\sigma_{H,3D} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})^2}$
Unit normal vector	(n_x, n_y, n_z)

When calculating features based on eigenvalues, the selection of neighbor distances greatly affects the labeling accuracy. By calculating features using various neighbor distances, features at multiple scales can be captured from a point cloud and the labeling accuracy is improved. In this research, we used 1m, 2m, 3m and 5m as the neighbor distances according to the sizes of typical roadside objects.

4. MMS-Specific Features

Since multi-scale features are calculated for the representative point of each small region, the same multi-scale features are added to all points in the small region. On the other hand, MMS-specific features are given for each point, independent of small regions.

MMS-specific features are calculated from the vehicle trajectory and scanlines. As shown in Figure 8, the MMS trajectory is used to calculate the height and horizontal distance at each point.

A scanline is a sequence of line segments consisting of consecutive measurement points, as shown in Figure 9. If the distance between two points is sufficiently small, the two points are considered to be sampled from the same object. Therefore, the length of a scanline is regarded as the width of an object. The curvature of a scanline is defined as the ratio of the distances L and D shown in Figure 9. The curvature is useful to distinguish between flat objects such as walls and curved objects such as poles.

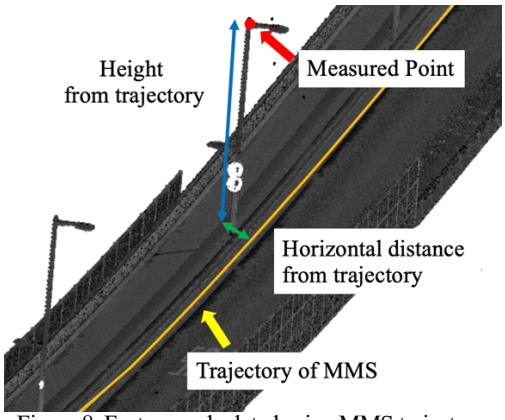


Figure 8. Features calculated using MMS trajectory

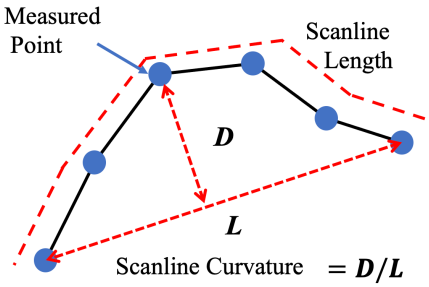


Figure 9. Length and curvature of scanline

5. Experiments

5.1 Calculation time

We evaluated calculation time for calculating multi-scale features using a common PC with a 3.70 GHz Intel Core i9 and 64GB RAM. The MMS was Trimble MX-8, and the laser scanner was RIEGL VQ-250. The small region radius r was set to 30cm, and the neighbor distances were 1m, 2m, 3m, and 5m. The point cloud used for this evaluation contained about 4.8 million points. We compared the calculation time for multi-scale features using the proposed method to the time directly using FLANN. The same code was used to calculate feature values from neighbor points.

The experimental result is shown in Table 4. Table 4 also shows the ratio of computation time using the two methods as the reduction rates. The computation time for all scales was greatly reduced, and the overall computation time was more than 1,000 times faster. In addition, the computation time increases significantly as the search radius increases in the conventional method, while the increase in computation time is relatively slow in the proposed method.

We also investigated the effect of the radius r of small regions by investigating the computation time of multiscale features. The results are shown in Table 5 and Figure 10. The results show that our method could greatly improve performance in all cases, but the computation time significantly increased when radius r was very small. However, roadside objects are relatively large and the scanline distances in MMS point clouds are typically about 5cm to 10 cm. Therefore, in our experiments, a radius of 30 cm for small regions was sufficient for classification of each point.

Table 4. Calculation time for multi-scale features

Radius R_i	1m	2m	3m	5m	Total
FLANN	3035.7s	12322.2s	27498.2s	64236.7s	107092.8s
Ours	2.2 s	7.5 s	13.8 s	29.5 s	53.0s
Reduction	1/1380	1/1643	1/1993	1/2178	1/2021

Table 5. Calculation time for different radii of small regions

Radius r	10 cm	20 cm	30 cm	40 cm	50 cm	60 cm
Total Time	408.7 s	112.9 s	53.0 s	31.1 s	21.1 s	15.2 s
Reduction	1/262	1/949	1/2021	1/3443	1/5075	1/7046

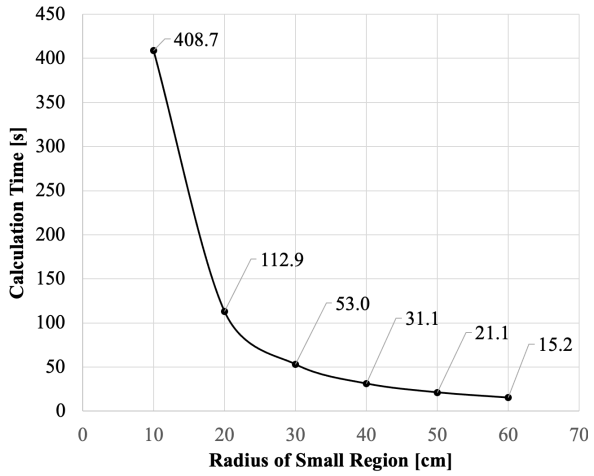
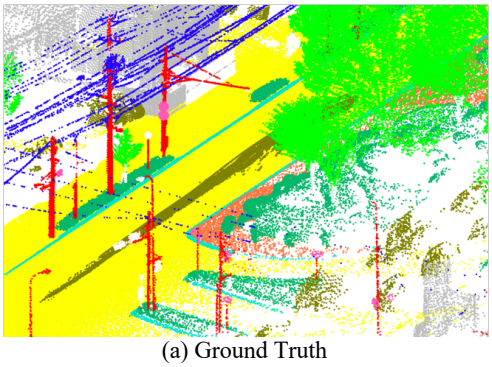


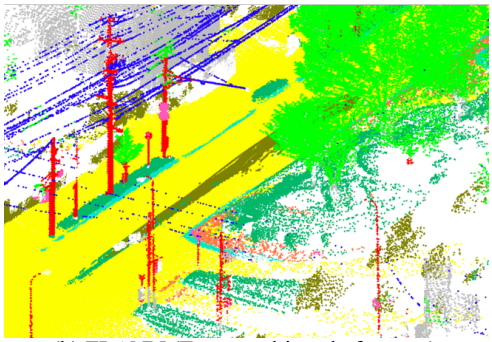
Figure 10. Relationship between the calculation time and the radius of small regions

Table 6. Classification results [%]

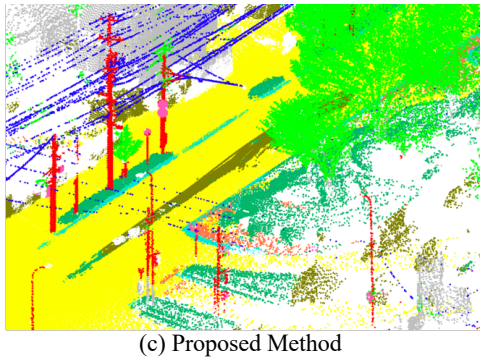
	FLANN+MMS	Proposed Method
Road	97.51	97.48
Grass	86.02	85.37
Shrub	78.97	79.86
Tree	94.34	94.45
Building	82.85	83.01
Car	75.76	77.94
Motorcycle	59.97	58.76
Electric Wire	83.97	84.52
Traffic Light	92.44	91.35
Traffic Sign	55.26	53.04
Destination Sign	61.54	58.09
Utility Pole	77.76	78.66
Curb	59.57	60.54
Guardrail	70.76	72.66
Trunk	50.1	52.07
(Average)	(75.1)	(75.2)



(a) Ground Truth



(b) FLANN (Exact multi-scale features)



(c) Proposed Method

Figure 11. Classified points

5.2 Classification Using Multi-Scale Features

We classified each point in the point cloud using the proposed method. The classification process is shown in Figure 2. In this process, multi-scale features and MMS-specific features are calculated for each point and the random forest estimates the label of the point from the concatenated features.

We defined 15 classes of objects for classification, as shown in Table 6. The point clouds used for classification were acquired in the same area as the one used for evaluating computation time. The total number of points was about 7.5 million and the ground-truth labels were manually assigned to the points by an operator. In the evaluation of classification, approximately 50% of points were used for training, 20% for validation, and 30% for testing.

Table 7. Contributions of features

Rank	Feature Values	Contribution	Type
1	Horizontal distance from trajectory	7.52%	MMS
2	Height	7.43%	MMS
3	Height from trajectory	6.74%	MMS
4	Scanline length	2.83%	MMS
5	Height deviation (1m)	2.80%	Multi-Scale
6	Verticality (5m)	2.75%	Multi-Scale
7	Hight difference (2m)	2.75%	Multi-Scale
8	Hight difference (3m)	2.59%	Multi-Scale
9	Intensity	2.49%	MMS
10	Hight difference (1m)	2.46%	Multi-Scale

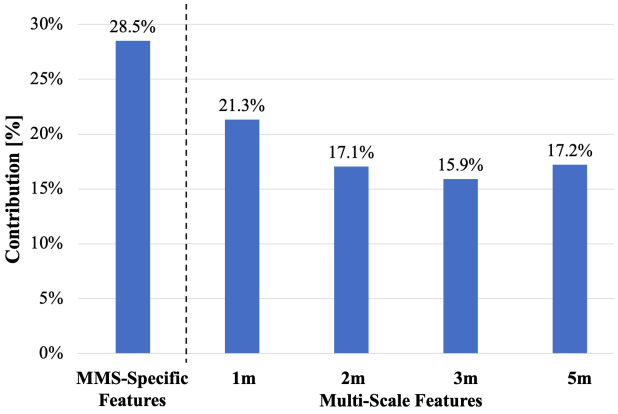


Figure 12. Comparison of contributions of multi-scale features

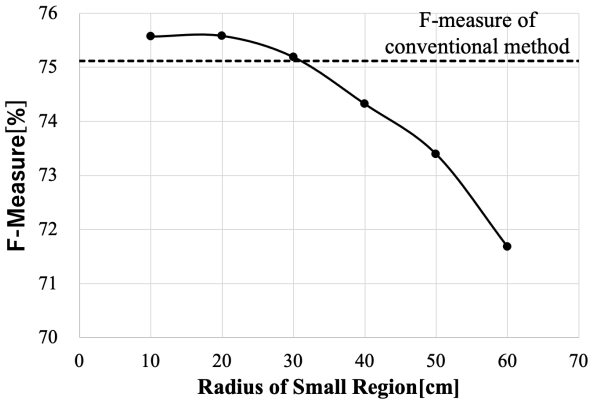


Figure 13. Relationship between the radius of small regions and classification accuracy

Table 6 shows the classification results as F-measures. The classification results of our method were compared with those obtained using the conventional FLANN. The results show that our method could achieve almost the same classification accuracy as the conventional method, although our method was significantly faster, as shown in Table 4.

Figure 11 visualizes classified points, in which points with the same class are drawn in the same color. These figures show that muti-scale features are very effective for point cloud classification, and our method provides visually equivalent results compared to those calculated with FLANN in very long computation time.

The random forest can provide the contribution that indicates how much each feature contributed to the classification. We investigated the contribution of each scale of features. Table 7 shows the top 10 contributions of features. The results indicate that MMS-specific features are particularly effective in classifying MMS point clouds and that both MMS-specific and multiscale features contribute to classification. Figure 12 shows the contribution of multi-scale features at each neighbor radius compared to MMS-specific features. Since multi-scale features at a wide range of scales contribute to classification to the same degree, it is worthwhile to compute multiscale features at many scales. Figure 12 also reveals that MMS-specific features contribute more effectively than multi-scale features at each scale.

We also investigated the effect of the small region radius r on classification accuracy. As shown in Figure 13, F-measures decreased as radius r was increased. In this figure, the F-measure of the conventional method (FLANN+MMS in Table 6) is shown as a dashed line. This result is due to the fact that multi-scale features are calculated only for the representative points of small regions. The results of Figures 10 and 13 suggest that the radius of the small regions should be selected based on the trade-off between computation speed and classification accuracy. In our experiments, $r = 30cm$ was selected because the F-measure was equivalent to that of the conventional method, as shown in Figure 13.

6. Conclusion

In this paper, we proposed a method to compute multi-scale features from large-scale MMS point clouds in a practical time. Our method computed the neighbor points in two steps using radius search. In the first step, we divided a point cloud into small regions, and in the second step, we searched neighbor point at the representative points of small regions. In our method, exact neighbor points without down-sampling could be obtained. In experiments, we confirmed that our method could compute multiscale features in practical time. We also confirmed that the multiscale features computed using our method could achieve good classification accuracy.

In future work, we would like to investigate methods for adaptively define the radii of small regions, because the computation time and classification accuracy are affected by the selection of the radius of small regions. In the current implementation, different MMS-specific features are assigned to each point in a small region while the same multiscale feature is

assigned to each point in a small region. The assignment of the same multiscale features may result in blurring of object boundaries. We would like to investigate methods to determine precise boundaries of objects for achieving high-quality classification of point clouds.

References

Breiman, L., 2001. Random Forests. Machine Learning, Vol.45, 5-32.

Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18(9), 509–517.

Hackel, T., Wegner, J. D., Schindler, K., 2016. Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences 3: 177–84.

Landrieu, L., Simonovsky, M., 2018. Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. Proceed- ings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 4558-4567.

Muja, M. Lowe, D. G., 2009. Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP.

Nüchter, A., K. Lingemann, J. Hertzberg., 2007. Cached K-D Tree Search for ICP Algorithms. *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)* 21–23.

Weinmann, M., Jutzi, B., Mallet, C., Weinmann, M., 2017. Geometric Features and Their Relevance for 3D Point Cloud Classification. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV- 1/W 1, 157-164.

Weinmann, M., 2019. Semantic Segmentation of Dense Point Clouds. 2nd International Workshop Point Cloud Processing.