Traversability Classification Using Super-voxel Method in Unstructured Terrain

Soohwan Song and Sungho Jo

Dept. of Computer Science, KAIST
291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Korea
dramanet30@kaist.ac.kr, shjo@kaist.ac.kr

Abstract. Estimating the traversability of terrain in an unstructured outdoor environment is one of the challenging issues in autonomous vehicles. When dealing with a large 3D point cloud, the computational cost of processing all of the individual points is very high. Thus voxelization methods are used extensively. In this paper, we propose a more fine-grained voxelization algorithm in the context of unstructured terrain classification. While the current shape of a voxel is a fixed-length cubic, we construct a flexible shape voxel which has spatial and geometrical properties. Furthermore, we propose a new shape histogram feature that represents the statistical characteristics of 3D points. The proposed method was tested using data obtained from unstructured outdoor environments for performance evaluation.

Keywords: unmanned vehicle, unstructured terrain, traversability classification, point cloud, voxel.

1 Introduction

Correctly classifying an outdoor environment into traversable and non-traversable regions is still a challenging task in autonomous vehicles. In particular, recognition in environments with unstructured terrain remains a difficult problem since the information obtained from sensors is highly inaccurate. In this paper we explore the traversability classification problem for sparse and unstructured terrain data. There are two important issues of concern.

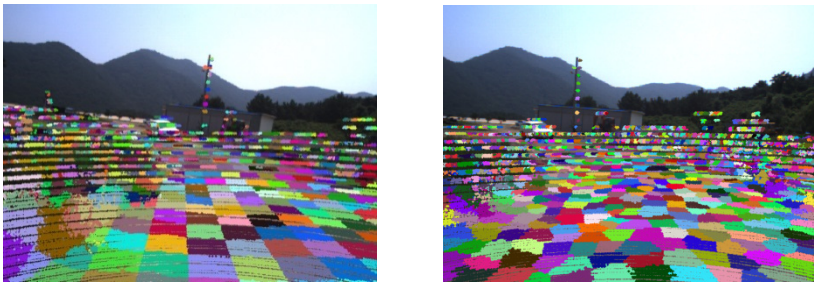
The first is to efficiently process the large amount of 3D point cloud data. For problems of traversability classification in 3D point clouds, *voxelization* methods are generally employed for efficiently processing large amounts of data [1] [2] [3]. Voxelization is a method to divide the 3D terrain into fixed-length cubes and extract their features (Fig. 1(a)). When a point cloud is converted into voxels, it is possible to lose the geometric information of the point cloud because of its fixed-length cubic shape. In order to overcome this problem, we employ a flexibly shaped *supervoxel* (Fig. 1(b)) which includes spatial and geometrical properties. The Voxel Cloud Connectivity Segmentation algorithm was recently proposed to generate supervoxels [4]. The algorithm clustered voxel-clouds which were generated from RGB-D images, and

J.-H. Kim et al. (eds.), *Robot Intelligence Technology and Applications 3,*
Advances in Intelligent Systems and Computing 345, DOI: 10.1007/978-3-319-16841-8_53

segmented only structured indoor scenes. Since this algorithm is unsuitable for sparse point clouds in outdoor environments, we modify it to be usable for sparse point clouds.

The second issue with respect to traversability classification is to define an appropriate feature which represents a local area well. There have been many studies on feature extraction for traversability classification. For point cloud classification, the *saliency features* [3] [5] have most commonly been used. The saliency features were used to capture the surface-ness, linear-ness, and scatter-ness of the local area. However these features could give inaccurate results when neighborhoods are not representative of the local geometry such as within sparse regions. In this paper, we propose a new histogram-type shape feature which works well in any environment. The new shape feature represents the statistical characteristics of 3D points with respect to shape and can be computed relatively fast. Furthermore, for more accurate classification, we extract a histogram-type color feature and combine the visual feature with the new shape feature.

The proposed supervoxel method and the new histogram-type features were tested using data obtained from unstructured outdoor environments for performance evaluation. Furthermore, this paper investigates whether the proposed methods can improve the performance of traversability classification.



(a) (b)

Fig. 1. Examples of (a) voxel and (b) supervoxel over-segmentation

2 Supervoxel

2.1 Point Features and Distance Measure

This section describes how to extract point features and measure the distance between them. Common approaches to extract a point feature is to directly use a surface normal [6] or compute a histogram feature which captures the variations of surface normals in a local patch [7] [8]. Since the surface normal is estimated by fitting a plane to some neighboring points, it will be greatly affected by density of neighboring points. So the normal is unsuitable for a point feature in sparse point clouds. Moreover, plane fitting methods become very computationally expensive when applied to millions of points. Thus we need to find other point features which do not consider the surface normal for quickly extracting the point features in sparse point clouds.

This paper extracts geometric information by analyzing local distributions of consecutive points. In [9], the authors used the *consecutive point information* (CPI) that can be obtained from a 2D LIDAR system. It used the angles between the y-axis and the line passing through two consecutive vertical points. In our work, we convert the point feature into a form that can be used with a 3D LIDAR system by adding information about consecutive horizontal points.

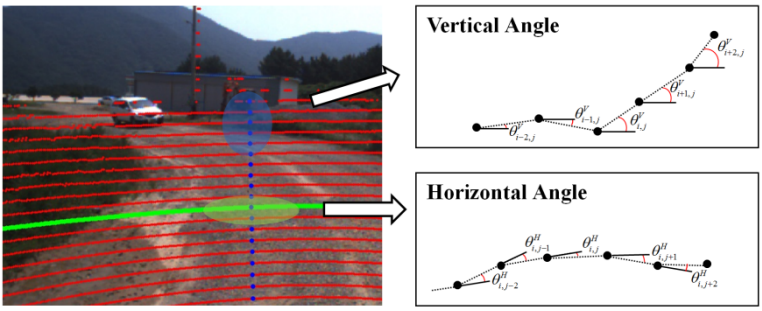


Fig. 2. Example of vertical and horizontal angels

In order to extract the CPI feature, we should define the order of consecutive points. The Cartesian coordinates of a 3D point cloud are converted to spherical coordinates (azimuth Θ and elevation ϕ) and they are discretized to predefined intervals. All points are sorted by azimuth and elevation, and represented as follows:

$$\begin{bmatrix} & \Theta_{1st} & \Theta_{2nd} & \Theta_{3rd} & \\ \phi_{1st} & \boldsymbol{p}_{1,1} & \boldsymbol{p}_{1,2} & \boldsymbol{p}_{1,3} & \\ \phi_{2nd} & \boldsymbol{p}_{2,1} & \boldsymbol{p}_{2,2} & \boldsymbol{p}_{2,3} & \dots \\ \phi_{3rd} & \boldsymbol{p}_{3,1} & \boldsymbol{p}_{3,2} & \boldsymbol{p}_{3,3} & \\ & & \dots & & \end{bmatrix}$$

Each $\boldsymbol{p}_{i,j}$ is a vector whose elements include the 3D Cartesian coordinates of ith elevation and jth azimuth point. The consecutive vertical point of a point $\boldsymbol{p}_{i,j}$ is $\boldsymbol{p}_{i+1,j}$ and consecutive horizontal point is $\boldsymbol{p}_{i,j+1}$. Let $\boldsymbol{v}_{i,j}^{V}$ be the vector passing through two consecutive vertical points, it can be computed as $\boldsymbol{v}_{i,j}^{V} = \boldsymbol{p}_{i+1,j} - \boldsymbol{p}_{i,j}$. Thus the angle $\theta_{i,j}^{V}$ between the x-y plain and $\boldsymbol{v}_{i,j}^{V}$ is computed as follows:

$$\theta_{i,j}^{V} = \sin^{-1}\left(\frac{\boldsymbol{v}_{i,j}^{V} \cdot \mathbf{z}}{|\boldsymbol{v}_{i,j}^{V}| \times |\mathbf{z}|}\right) \tag{1}$$

where $\mathbf{z}$ is a z-axis unit vector. The θ^{V} represents the inclination angle of the surface generated by consecutive vertical points. Small angles indicate a flat surface such as the ground and large angles indicate vertically oriented object surfaces. Furthermore, the surface of trees or bushes has a large angle variation.

Let $\boldsymbol{v}_{i,j}^{H}$ be the vector passing through two consecutive horizontal points, it can be computed as $\boldsymbol{v}_{i,j}^{H} = \boldsymbol{p}_{i,j+1} - \boldsymbol{p}_{i,j}$. The angle $\theta_{i,j}^{H}$ between two consecutive vectors $\boldsymbol{v}_{i,j-1}^{H}$ and $\boldsymbol{v}_{i,j}^{H}$, can be computed as follows:

$$\theta'^{H}_{i,j} = \cos^{-1}\left(\frac{\boldsymbol{v}_{i,j}^{H} \cdot \boldsymbol{v}_{i,j-1}^{H}}{|\boldsymbol{v}_{i,j}^{H}| \times |\boldsymbol{v}_{i,j-1}^{H}|}\right),\ \theta_{i,j}^{H} = \min\left(\theta'^{H}_{i,j}, \pi - \theta'^{H}_{i,j}\right) \tag{2}$$

The θ^{H} represents a discontinuity of a horizontal surface. The small angles indicate that the surface generated by consecutive horizontal points is flat such as the ground or the wall of a building. The large angles represent a discontinuous point, and the large angle variations indicate that the surface is scattered such as a tree or bush. Fig. 2 shows an example of vertical and horizontal angles.

In this paper, we use the mean and standard deviation of the neighboring angles within a window size as geometrical point features. The mean and standard deviation should be normalized to be within a range of [0, 1]. In order to normalize the mean and standard deviation, we divide the mean by $\pi/2$ and the standard deviation by $\mathbf{std}(0, \pi/2)$.

Let $\hat{\theta}^{V}$ and $\hat{\theta}^{H}$ be the normalized means of vertical and horizontal angles, and $\hat{\sigma}^{V}$ and $\hat{\sigma}^{H}$ be the normalized standard deviations of each angle, the point feature $\boldsymbol{f}_k$ used for the supervoxel over-segmentation task is defined like this:

$$\boldsymbol{f}_k = \left[x_k, y_k, z_k, \hat{\theta}_k^{V}, \hat{\theta}_k^{H}, \hat{\sigma}_k^{V}, \hat{\sigma}_k^{H}\right] \tag{3}$$

where *x*, *y*, and *z* are the Cartesian coordinates of a point.

In order to compute the point feature distances, it is necessary to normalize spatial distances by their respective maximum distances. We define the normalization constant as the distance between the center and maximally distant point in a cluster. Let R_{seed} be a seed resolution of initial cluster, we can normalize our spatial distance d^c by dividing by the normalization constant $\sqrt{3}R_{seed}$. Since point shape features are already normalized, it is not necessary to normalize the shape distance d^s. By applying a weight of point distance w_{point}, the distance of two point features $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ is computed as follows:

$$d^c = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2}$$

$$d^s = \sqrt{\left(\hat{\theta}_i^{V} - \hat{\theta}_j^{V}\right)^2 + \left(\hat{\theta}_i^{H} - \hat{\theta}_j^{H}\right)^2 + \left(\hat{\sigma}_i^{V} - \hat{\sigma}_j^{V}\right)^2 + \left(\hat{\sigma}_i^{H} - \hat{\sigma}_j^{H}\right)^2}$$

$$d = \sqrt{w_{point}\left(\frac{d^c}{\sqrt{3}R_{seed}}\right)^2 + \left(1 - w_{point}\right)(d^s)^2} \tag{4}$$

2.2 Supervoxel Segmentation

The supervoxel segmentation algorithm is summarized in Algorithm 1. The algorithm was originally motivated by previous work in [4]. The supervoxel segmentation algorithm in [4] used a dense depth image and it clustered the voxel-clouds. Therefore we modified the algorithm to be usable in sparse point clouds by clustering cloud points instead of voxel-clouds.

Algorithm 1 Supervoxel segmentation

1. Construct neighborhood graph $\boldsymbol{G}$.
2. Generate voxels V_k with resolution R_{seed}.
3. Initialize cluster center points P_k^{seed} to center of V_k.
4. Change the center points P_k^{seed} to the lowest gradient in radius R_{search} of each center.
5. Assign each P_i in V_k to each cluster C_k and calculate distances D_i^{min} between $P_i \in C_k$ and P_k^{seed}.
6. **for** each cluster C_k **do**
7. Extract the neighbor points P^* in radius $2 \times R_{seed}$ of P_k^{seed}.
8. Calculate distances D^* between P^* and P_k^{seed}.
9. Remove points whose distance D^* are higher than minimum distance D^{min} from P^*.
10. Find the connected points $\hat{P}^*$ of P^* from P_k^{seed} by using **BFS** of $\boldsymbol{G}$.
11. Assign the each points of $\hat{P}^*$ to cluster C_k and update D^{min} to its distance D^*.
12. **end for**

We begin by constructing a neighborhood graph $\boldsymbol{G}$, which ensures that disconnected points are not clustered in the same cluster. We obtained the neighborhood graph by connecting the six nearest neighbors of each point. Then, we divide the whole point cloud into a voxelized grid V_k with seed resolution R_{seed}. The seed points are initialized to the nearest point to the center of each voxel grid cell. The initial points are changed to the lowest gradient position in the search range R_{search} of each point. The i[th] point gradient $\boldsymbol{g}_i$ is computed as:

$$\boldsymbol{g}_i = \sum_{k \in \boldsymbol{p}_{adj}} \frac{|\boldsymbol{f}_i - \boldsymbol{f}_k|}{N_{adj}} \tag{5}$$

where $\boldsymbol{p}_{adj}$ and N_{adj} represent the neighboring points of the neighborhood graph and number of neighbor points, respectively.

We generate supervoxels by clustering points based on their distance between each seed point, P^{seed}. Each point is assigned to the nearest cluster center. First, the points within a same V_k are initialized to the same cluster C_k, and their distances to P_k^{seed} are stored in variable D^{min}. Then, the method iterates the following procedures for each cluster C_k. First we extract the neighbor points P^* in radius $2 \times R_{seed}$ of P_k^{seed} and calculate their distances, D^*, to P_k^{seed}. Then, the points whose distances D^* are lower than their D^{min} are assigned to cluster C_k and update D^{min}. In order

to ensure the connectivity of the points in C_k, only points connected to P_k^{seed} are selected. We find the connected points by traversing $\boldsymbol{G}$ using *breadth-first search* (BFS).

2.3 Features of Supervoxel

In this section, we define the features of each supervoxel cluster. The feature space consists of two different histogram feature sets, one with color characteristics and the other with geometric characteristics.

The first feature set $\boldsymbol{F^c}$ is a *LAB* color histogram derived from the color components in a supervoxel. $\boldsymbol{F^c}$ is produced by discretizing the colors in the supervoxel into a number of bins, and counting the number of points in each bin. Let F_b^c be a color histogram value of *b* index in a supervoxel cluster, it is computed as:

$$F_b^c = \frac{1}{N}\sum_{i=1}^{N} \mathbf{I}(\boldsymbol{f}_i^c \in \text{bin}(b)) \tag{6}$$

where $\boldsymbol{f}_i^c$ is a LAB color value of i$^{\text{th}}$ point, *N* is the number of points in a cluster and **I** is the indicator function. The color histogram $\boldsymbol{F^c}$ represents the probability mass function of the point color feature.

The second feature set $\boldsymbol{F^s}$ is a shape histogram of a supervoxel, which is derived from the point shape feature $\boldsymbol{f^s} = [\hat{\theta}^V, \hat{\theta}^H, \hat{\sigma}^V, \hat{\sigma}^H]$. Let F_b^s be a shape histogram value defined as:

$$F_b^s = \frac{1}{N}\sum_{i=1}^{N} \mathbf{I}(\boldsymbol{f}_i^s \in \text{bin}(b)) \tag{7}$$

Similar to $\boldsymbol{F^c}$, the shape histogram $\boldsymbol{F^s}$ represents the probability mass function of the point shape feature. As mentioned earlier, the performance of saliency features [3] [5] or other shape histogram features [7] [8] derived from surface normals are strongly influenced by the density of local points. However, our shape histogram feature is derived from CPI and works well in any environment. Furthermore, it can be computed relatively fast. We empirically obtained good results by dividing the feature values into three bins for angle means and two bins for angle standard deviations. Therefore, the number of shape histogram bins is $3^2 \times 2^2 = 36$.

3 Traversability Classification

In order to train the classification model, we clustered all of the supervoxels in the training dataset using *k-means clustering*, and modeled each cluster M_k by its mean value $\boldsymbol{F}_{M_k}$. Separate sets of cluster models were maintained for positive and negative examples. We adopted the χ^2 distance function as a similarity measure of clusters, defined as:

$$D(\boldsymbol{F},\boldsymbol{F}') = w_{voxel}\sum_{i=1}^{B^S}\frac{\left(Fr_i^S - F_i^S\right)^2}{Fr_i^S + F_i^S} + (1-w_{voxel})\sum_{j=1}^{B^C}\frac{\left(F'^C_j - F_j^C\right)^2}{F'^C_j + F_j^C} \quad (8)$$

where B^S and B^C represent the number of histogram bins, and w_{voxel} is a weight factor. The weight factor w_{voxel} allows us to control the relative contributions of the two components.

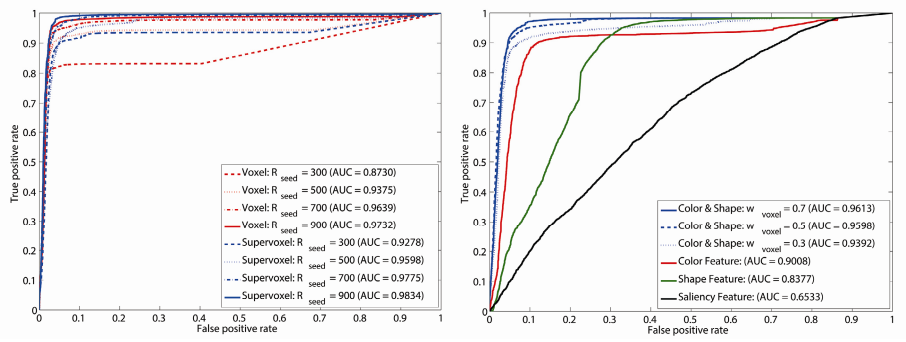
In order to find the traversable regions, the learned positive and negative models are compared to new input regions. If the similarity ratio of a new region is greater than a threshold λ, the region is classified as a traversable region. The classifier is defined as:

$$H(\boldsymbol{F}_k) = \mathbf{I}\left[\frac{D(\boldsymbol{F}_k,\boldsymbol{F}_{M_t})}{D(\boldsymbol{F}_k,\boldsymbol{F}_{M_{nt}})} \geq \lambda\right] \quad (9)$$

where M_{nt} is the non-traversable model closest to $\boldsymbol{F}_k$, and M_t is the closest traversable model [10] .

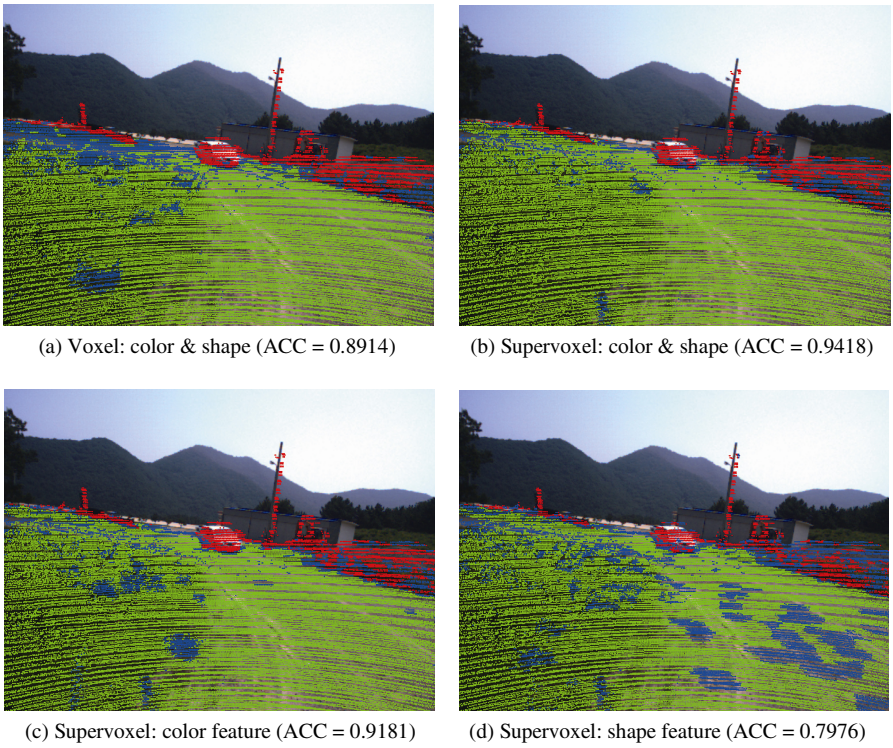
4 Experimental Results

In order to classify drivable regions, we obtained the point cloud data from a HDK-32E LiDAR sensor and 640x480 pixel images from a forward-facing camera. The environmental data captured was unstructured terrain which included foliage and dense vegetation over 40cm high. A relatively dense data frame was constructed by combining six consecutive overlapping data frames, and we extracted twenty consecutive dense data frames. We then manually labeled the dense frames for traversability classification. The last ten dense frames were used as a testing dataset and the preceding ten frames were used as the training dataset. For the supervoxel segmentation, we set the supervoxel parameters R_{search} and w_{point} to $0.5 \times R_{seed}$ and 0.5 respectively. Two hundred traversable and non-traversable models were trained for the classification task.



(a) (b)

Fig. 3. ROC curves showing the results of (a) the supervoxel and voxel methods respect to different R_{seed} and (b) supervoxel method under different feature sets.



(a) Voxel: color & shape (ACC = 0.8914)

(b) Supervoxel: color & shape (ACC = 0.9418)

(c) Supervoxel: color feature (ACC = 0.9181)

(d) Supervoxel: shape feature (ACC = 0.7976)

Fig. 4. Traversability classification results under different settings. The green and red points are correctly classified traversable and non-traversable regions respectively, and the blue points are incorrectly classified regions. ($R_{seed} = 500$, $w_{voxel} = 0.5$, and $\lambda = 1$).

To evaluate the performance of our proposed method, we divided the experiment into two subsections. First, the performance of the supervoxel method was compared to a general voxel method. Second, we compared the proposed shape feature against other state-of-the-art features.

The ROC curves of the voxel and supervoxel methods with respect to different voxel resolution R_{seed} are shown in Fig. 3(a). For general voxel classification, we used the same feature and classifier as that of the supervoxels ($w_{voxel} = 0.5$). As can be seen in Fig. 3(a), the supervoxel method always shows better performance than the voxel method for each value of R_{seed}. In particular, the supervoxel method improves approximately 5% at $R_{seed} = 300$. For the voxel method, the classification accuracy rapidly drops as the resolution R_{seed} decreases while in the case of the supervoxel method, the accuracy drop is modest. Since the voxel does not incorporate geometrical properties, it is a reasonable result that voxel under-performed with respect to the supervoxel.

In order to measure the performance of the proposed shape feature, it is compared to the color feature, saliency feature [3] and both color and shape features with respect to different weight factors w_{voxel}. Fig. 3(b) shows the ROC curves of the supervoxel method ($R_{seed} = 500$) with respect to different features. In this figure, accuracy of

the proposed shape feature is lower than that of the color feature. It indicates that the color is the most powerful feature for traversability classification. However, the major outcome of the experiment is that the performance of the color feature can be improved by more than 6% with the help of the shape feature. Since w_{voxel} represents degree of contribution of shape feature, the best performance at $w_{voxel} = 0.7$ indicates that the shape feature contributes more to the classification task than color feature. Furthermore, the shape feature performs better than the saliency feature. The saliency feature is unusable in sparse unstructured environments since it is influenced by the local point density, while our proposed feature works well in sparse environments.

Fig. 4 shows examples of traversability classification under different settings. As mentioned above, the performance of the voxelization method (Fig. 4(a)) is lower than the supervoxel method (Fig. 4(b)). Furthermore, we can verify that the combination of the shape feature (Fig. 4(d)) and color feature (Fig. 4(c)) offers excellent performance improvements. The results clearly indicate that the supervoxel method and its histogram-type features could improve both the performance and accuracy of traversability classification in unstructured terrains.

5 Conclusion

In this paper, the performance and accuracy of traversability classification in unstructured terrains is improved in a number of ways. The supervoxel method, a new, fine-grained voxelization method in sparse point clouds, is proposed for efficiently processing large amounts of point cloud data. Unlike traditional voxelization methods, it has geometrical properties and is usable for sparse point clouds. This paper proposes a new histogram-type shape feature that incorporates consecutive point information. The experimental results show that both the supervoxel method and the shape histogram features are successful and show better performance in most cases than other state-of-art methods.

Acknowledgements. This work was supported by the Technology Innovation Program, 10045252, Development of robot task intelligence technology that can perform task more than 80% in inexperience situation through autonomous knowledge acquisition and adaptational knowledge application, funded By the Ministry of Trade, industry & Energy (MOTIE, Korea).

References

1. Heckman, N., et al.: Potential negative obstacle detection by occlusion labeling. IEEE/RSJ. Int. Conf. Intell. Rob. & Syst (2007)
2. Stoyanov, T., et al.: Path planning in 3D environments using the normal distributions transform. IEEE/RSJ Int. Conf. Intell. Rob. & Syst (2010)

3. Bogil, S., Myungjin, C.: Traversable ground detection based on geometric-featured voxel map. In: Bogil, S., Myungjin, C. (eds.) IEEE. Korea-Japan Joint Workshop on Frontiers of Computer Vision, pp. 31–35 (2013)
4. Papon, J., et al.: Voxel cloud connectivity segmentation-supervoxels for point clouds. In: IEEE Conf. Computer Vision and Pattern Recognition, pp. 2027–2034 (2013)
5. Lalonde, J.F., Vandapel, N., Huber, D.F., Hebert, M.: Natural terrain classification using three-dimensional ladar data for ground robot mobility. Journal of Field Robotics 23(10), 839–861 (2006)
6. Rabbani, T., van den Heuvel, F., Vosselmann, G.: Segmentation of point clouds using smoothness constraint. In: Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, pp. 248–253 (2006)
7. Rusu, R., et al.: Learning informative point classes for the acquisition of object model maps. In: IEEE. Int. Conf. Control, Automation, Robotics and Vision, pp. 643–650 (2008), 2008
8. Rusu, R., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3D registration. In: IEEE Int. Conf. on Rob. & Autom (2009)
9. Yungeun, C., Seunguk, A., MyungJin, C.: Online urban object recognition in point clouds using consecutive point information for urban robotic missions. Robotics and Autonomous Systems (2014)
10. Dongshin, K., et al.: Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In: IEEE Int. Conf. on Rob. & Autom. (2006)